Acta Informatica 26, 559–576 (1989)

Acta Informatica

Computational Algorithms for Networks of Queues with Rejection Blocking

I.F. Akyildiz [1] and Horst von Brand [2]

[1] School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA 30332, USA

[2] Department of Computer Science, Universidad Tecnica Federico Santa Maria, Valparaiso, Chile

Summary. Open, closed and mixed queueing networks with reversible routing, multiple job classes and rejection blocking are investigated. In rejection blocking networks blocking event occurs when upon completion of its service of a particular station's server, a job attempts to proceed to its next station. If, at that moment, its destination station is full, the job is rejected. The job goes back to the server of the source station and immediately receives a new service. This is repeated until the next station releases a job and a place becomes available. In the model jobs may change their class membership and general service time distributions depending on the job class are allowed. Two station types are considered: Either the scheduling discipline is symmetric, in which case the service time distributions are allowed to be general and dependent on the job class or the service time distributions at a station are all identical exponential distributions, in which case more general scheduling disciplines are allowed. An exact product form solution for equilibrium state probabilities is presented. Using the exact product form solution of the equilibrium state distribution, algorithms for computation of performance measures, such as mean number of jobs and throughputs, are derived. The complexity of the algorithms is discussed.

1. Introduction

Product form queueing networks (also referred to as BCMP [6] or Kelly [17, 18] or separable networks [24]) to which we will refer as *classical networks* in what follows, have proved invaluable in modeling a variety of computer systems, communication networks as well as manufacturing systems. They are flexible enough to represent adequately many of the features arising in such

Offprint requests to: I.F. Akyildiz

applications. They have not, however, been able to provide much insight into the phenomenon of blocking, because all algorithms for product form networks are based on the assumption that the stations have infinite buffer space.

In recent years there has been a growing interest in the analysis queueing networks with blocking. These are networks where the stations have finite capacities, hence blocking can occur if a station is full to its capacity. A job which wants to come to the full station must reside in its source station and block it until a place is available in the destination station. The interest in networks with blocking comes primarily from the realization that these models are useful in the study of the behavior of subsystems of computers and communication networks, in addition to detailed descriptions of several computer-related applications such as manufacturing systems. In recent years several investigators have published results on networks with different blocking policies [20]. In so-called "rejection blocking" networks the blocking event occurs when a job completing service at station i's server wants to join station j, whose capacity is full. The job is rejected by station j. That job goes back to station i's server and receives another round of service. This activity is repeated until station j releases a job, and a place becomes available. The "rejection blocking" type has been used to model systems such as communication networks, computer systems with limited multiprogramming, production lines and flexible manufacturing systems. Two variants of this blocking policy can be considered:

i) The destination of the job once it finishes service is fixed.

ii) The job selects a destination independently after each round of service according to the routing probabilities.

The definition of the second variant of rejection blocking makes deadlocks impossible if the network is irreducible (i.e., each station is reachable from every other station). As long as there is at least one free space in some station a job will eventually move into it, even if this takes a long sequence of trials. This makes a general analysis much simpler. No restrictions that make deadlocks impossible are needed and no special method to handle deadlocks has to be included in the model.

Note that for tandem and cyclic networks there is no difference between the variants of rejection blocking, since there is only one possible destination for each job. Same thing happens in a merge configuration where several stations feed a single station downstream. For networks with arbitrary topology the two variants of rejection blocking are different.

In this paper we investigate queueing networks with rejection blocking of variant ii). Once a job in class α finishes service in station i it determines, according to the routing probabilities $p_{i\alpha,j\beta}$, to which station j and class β it goes next. With a certain probability (that depends on the state of the destination station) the job will be rejected there. The rejected job returns to station i (in class α) to get another round of service, independent from the one it received before. When this new round of service is finished, the job again selects a destination station and class (independent from the ones selected before) and so on.

The rejection blocking policy was introduced by Caseau and Pujolle [8] who considered tandem networks only. They investigate various blocking poli-

cies and general service requirement distributions, with the aim of obtaining bounds on throughput. Pittel [21] shows that rejection blocking policies with multiple job classes and reversible routing have product form solutions if the rejection probabilities are of a particular class-dependent form. His work is restricted to exponential service requirement distributions and jobs are not allowed to change their class membership. Cohen [10] finds a product form solution for a cyclic rejection blocking network with two stations, multiple job classes and class-dependent service requirement distributions (not necessarily exponential) when the scheduling disciplines are processor sharing with load dependent service efforts. Hordijk and von Dijk [14] show that in special cases of queueing networks with rejection blocking there are product form solutions. They consider models with a single job class in which routing is reversible and models in which blocking is dominant, i.e. there are so many jobs in the system that no station can ever be empty. Balsamo and Iazeolla [4] find classical networks that share part of the state space with rejection blocking networks and show that their equilibrium state probabilities agree (up to a normalization constant) with those of the blocking network on the intersection of the state spaces. Akyildiz [1] gives computational algorithms for performance measures of the type of networks studied by [4, 14].

Akyildiz and von Brand [2] prove a duality among open and closed rejection blocking networks with one job class and exponential service time distributions. There is no restriction on the structure of the network. Using this result, they prove a product form solution for the case in which at most one station can be empty at a time. Van Dijk and Tijms [11] give a proof of insensitivity of the distribution of jobs (i.e. dependence only on the mean of the service requirement distribution) in a cyclic network with two stations, multiple job classes and symmetric scheduling disciplines. Tijms [27] presents this result in more accessible terms. Van Dijk and Akyildiz [12] study networks with mixed exponential and non-exponential parallel queues with interdependent service capacities and common pools. Under the invariance condition a product form solution is found for the steady state probabilities. It is also shown that the distribution has the insensitivity property.

The organization of this paper is as follows: In Sect. 2 we describe the model under investigation. Section 3 contains the exact product form solution for steady state probabilities. In Sect. 4 we give a general formula for throughput. Section 5 contains algorithms for the computation of throughput and mean queue lengths in open, closed and mixed queueing networks. In Sect. 6 we discuss the complexity of the algorithms. Finally, Sect. 7 concludes the paper.

2. Model Description

We consider queueing networks with N stations and C job classes. The description of the model is rather complex. We break it down into a description of the service time distributions, a definition of the scheduling disciplines and the probability that an arriving job is accepted, which we call the blocking function of the station.

2.1. Service Time Distributions

A job of class α requests service at station i distributed as $F_{i\alpha}$ with mean $1/\mu_{i\alpha}$. We will represent the service time distributions as mixtures of Erlang distributions of the following form [5]:

$$F_{i\alpha}=\sum_t g_{i\alpha;t} E_{t\nu_{i\alpha}} \tag{1}$$

where $E_{i\nu_{i\alpha}}$ is the Erlang distribution with t stages each with rate $\nu_{i\alpha}$. This means that with probability $g_{i\alpha;t}$ a job of class α arriving at station i will have to traverse t exponential stages, each of which has rate $\nu_{i\alpha}$. This clearly requires:

$$\sum_t g_{i\alpha;t}=1. \tag{2}$$

It also implies:

$$\frac{1}{\mu_{i\alpha}}=\sum_i g_{i\alpha;t}\frac{t}{\nu_{i\alpha}}. \tag{3}$$

By renewal theory the probability that at an arbitrary instant a job with service requirement distribution $F_{i\alpha}$ still has to traverse s stages is given by:

$$r_{i\alpha}(s)=\frac{\mu_{i\alpha}}{\nu_{i\alpha}}\sum_{t\geq s} g_{i\alpha;t}. \tag{4}$$

Note that:

$$r_{i\alpha}(1)=\frac{\mu_{i\alpha}}{\nu_{i\alpha}}. \tag{5}$$

2.2. Scheduling Disciplines

A scheduling discipline (f, ϕ, ψ) is defined by [11, 18, 27]:

$f(k)$: Total service effort when there are k jobs in the station.

$\phi(l, k)$: Fraction of the service effort destined to the job in position l when there are k jobs in the station (zero for l outside of $1\leq l\leq k$). This requires:

$$\sum_{1\leq l\leq k}\phi(l,k)=1 \qquad \forall k. \tag{6}$$

$\psi(l, k)$: Probability that an arriving job is placed in position l when there are k jobs in the station (zero for l outside of $1\leq l\leq k+1$). This requires:

$$\sum_{1\leq l\leq k+1}\psi(l,k)=1 \qquad \forall k. \tag{7}$$

The networks with general service time distributions considered in this paper are shown to have product form solutions under *symmetric* [18] (also called

station balancing [9]) scheduling disciplines. Formally, we call a scheduling discipline in the class defined above *symmetric* if

$$\psi(l, k) = \phi(l, k+1). \tag{8}$$

This framework clearly does not describe all possible scheduling disciplines, for example there is no way to give one job class priority over another. Scheduling disciplines that depend on the service requirement, like Shortest Job First cannot be described either. The class of scheduling disciplines that can be described is rich. Some examples are:

FCFS: First come, first served is described by $\phi(1, k) = 1$ and $\psi(k+1, k) = 1$.

LCFS: Last come, first served preemptive is described by $\phi(k, k) = 1$ and $\psi(k+1, k) = 1$.

PS: Processor sharing is described by $\phi(l, k) = 1/k$ and $\psi(k+1, k) = 1$.

RAND: Service in random order (Spirn [26]) is described by $\phi(1, k) = 1$ and $\psi(l, k) = 1/k$ for $l \geqq 2$.

Other scheduling disciplines like LBPS (Last batch processor sharing [19]) can also be described, but the functions ϕ and ψ become complex [9].

It should be noted that the description of a particular scheduling discipline is not unique. For example, the description for PS given above is *not* symmetric, but if we set $\psi(l, k) = 1/(k+1)$ the discipline becomes symmetric. The only difference between the two is that this alternative does not keep the jobs in their order of arrival. Of the remaining disciplines, FCFS and RAND are not symmetric, while LCFS is.

We assume that all stations in the network satisfy one of the following:

i) Have symmetric scheduling disciplines with general service time distributions that may depend on the job class. For scheduling disciplines in the class we consider the symmetry condition to be necessary and sufficient if the service time distributions are different for different job classes or are non-exponential.

ii) Have exponential service time distributions that do not depend on the job class. Here the scheduling discipline is arbitrary (non-symmetric) in the class of disciplines we consider.

We assume that a job selects a service requirement before starting to get service, i.e. when a job enters station i in class α it is assigned a number of stages of service according to the $g_{i\alpha;s}$. If a job in class α is in position l of station i and the number of jobs in station i is k_i, the rate at which that job advances to its next stage of service (or leaves the station if it is in its last stage of service) is $v_{i\alpha} f_i(k) \phi_i(l, k_i)$.

23. *Blocking Functions*

To complete the description of a single station, we need to define when an arriving job is rejected. Define a partition of the job classes, and denote the set of job classes that contains class α by $[\alpha]$. We write the probability that

a job of class α arriving at station i is accepted when there are a total of k_i jobs in it, of which $k_{i\alpha}$ are of class α and $k_{i[\alpha]}$ are of classes in the set that contains class α, as:

$$b_{i\alpha}(\mathbf{k}_i) = h_{i\alpha}(k_{i\alpha})\, h_{i[\alpha]}(k_{i[\alpha]})\, h_i(k_i). \tag{9}$$

Here $h_{i\alpha}$, $h_{i[\alpha]}$ and h_i are arbitrary functions.

2.4. *Job Routing*

The state of the network will be described by (ordered) N-tuples of station states. The state of station i is denoted by:

$$((\kappa_{i1}, \sigma_{i1}), (\kappa_{i2}, \sigma_{i2}), \ldots, (\kappa_{ik_i}, \sigma_{ik_i})). \tag{10}$$

Here k_i is the number of jobs in station i, κ_{il} is the class of the job in position l of station i and σ_{il} is the number of remaining stages of service for that job. We will denote the number of jobs of class α in station i by $k_{i\alpha}$. We will use $\mathbf{x}$ and $\mathbf{y}$ to denote arbitrary states of the network. We define the *occupancy* of the network as an N-tuple of strings of job classes, where the i-th string represents the classes of the jobs in station i in order. The *population* of the network gives the numbers of jobs of each class in each station. These are defined in the obvious ways for single stations. The occupancy of the network will be denoted by $\mathbf{n}$, and the population by $\mathbf{k}$. For single stations we will use $\mathbf{n}_i$ and $\mathbf{k}_i$.

The above description of a single station does not say what happens to a class α job that tries to leave station i to go to station j in class β but is rejected there. We specify that in that case the rejected job returns to station i in class α and is treated exactly like an arriving job, only that it cannot be rejected.

The structure of the network itself is fixed by:

$p_{i\alpha,j\beta}$: Routing probabilities. Probability that a job of class α that leaves station i tries to enter station j in class β.

$p_{0,j\beta}$: Probability that an exogenous job tries to enter station j in class β. We assume that new jobs arrive at a (fixed) rate γ to the network. The process that generates exogenous arrivals is assumed to be Poisson.

$p_{i\alpha,0}$: Probability that a job which finished service in station i in class α leaves the network.

One can define an equivalence relation on pairs (station, job class) by defining $(i, \alpha) \equiv (j, \beta)$ iff a job that starts in station i and class α can wind up in station j and class β after a series of transitions. We call each of the equivalence classes of this relation a *routing chain* or chain for short. Without loss of generality, we assume that the sets of job classes in different routing chains are disjoint. So we can identify a routing chain with a set of job classes.

The network is *closed for routing chain* Γ if $p_{0,j\beta}=0$ for all $(j,\beta)\in\Gamma$. The network is *open for routing chain* Γ if $p_{0,j\beta}$ is nonzero for at least one $(j,\beta)\in\Gamma$. The network is *closed* if it is closed for all routing chains. The network is *mixed* if it is closed for some routing chains and open for others. The network is *open* if it is open for all routing chains.

Let us now define $e_{i\alpha}$ as:

$$e_{i\alpha}=\gamma p_{0,i\alpha}+\sum_{r\kappa} e_{r\kappa}p_{r\kappa,i\alpha}. \tag{11}$$

There will be one such system of equations for each routing chain. Note that for closed chains this linear system is homogeneous. In that case, we take any particular non-trivial solution of the system as the $e_{i\alpha}$. In classical networks the $e_{i\alpha}$ are the throughputs of station i for jobs of class α if the network is open for the routing chain that contains job class α. If the network is closed for the routing chain that contains class α, they can be interpreted as relative throughputs. In the present case these quantities have no physical significance, since the routing of the jobs depends not only on the routing probabilities but also on blocking. We define them because they appear in the expression for the equilibrium state probabilities given below.

We furthermore assume that routing is *reversible* [18], i.e.

$$\begin{aligned} e_{i\alpha}p_{i\alpha,j\beta}&=e_{j\beta}p_{j\beta,i\alpha} \qquad \forall i,j,\alpha,\beta \\ \gamma p_{0,j\beta}&=e_{j\beta}p_{j\beta,0} \qquad \forall j,\beta. \end{aligned} \tag{12}$$

In a classical network reversible routing means that the flow of jobs from station i and class α to station j and class β is the same as the flow of jobs from station j and class β to station i and class α. Again, this interpretation is not applicable to blocking networks.

3. The Equilibrium State Distribution

For queueing networks as described above, Akyildiz and von Brand [3] prove that the equilibrium probability distribution of state $\mathbf{x}$ has the following exact product form:

$$\pi(\mathbf{x})=\frac{1}{G(\mathbf{K})}\cdot\prod_i\left[\prod_{1\le l\le k_i}\frac{h_i(l-1)}{f_i(l)}r_{i\kappa_{il}}(\sigma_{il})\prod_\Gamma\prod_{1\le l\le k_{i\Gamma}}h_{i\Gamma}(l-1)\prod_\alpha\prod_{1\le l\le k_{i\alpha}}\frac{e_{i\alpha}h_{i\alpha}(l-1)}{\mu_{i\alpha}}\right]. \tag{13}$$

Here i ranges over all stations, Γ ranges over all routing chains and α ranges over all job classes. $G(\mathbf{K})$ is the normalization constant, selected such that the equilibrium state probabilities add up to one.

The solution (13) can be verified by substituting it into the following global balance equations:

$$\pi(\mathbf{x}) = \sum_{y \in S} q(\mathbf{x}, \mathbf{y}) = \sum_{y \in S} q(\mathbf{y}, \mathbf{x})\, \pi(\mathbf{y}).$$

To simplify the task, it is convenient to use simpler (and more detailed) balance equations that add up to the global balance equations. In this way, if the proposed solution satisfies these simpler balance equations, it automatically satisfies the global balance equations. Such sets are called *local balance equations* [6], or also *job local balance equations* [14–16]. Details of the proof for the solution (13) can be found in [3].

We will frequently write Eq. (13) as

$$\pi(\mathbf{x}) = \frac{1}{G(\mathbf{K})} \prod_i \left[P_i(\mathbf{k}_i) \prod_{1 \le l \le k_i} r_{i\kappa_{il}}(\sigma_{il}) \right] \tag{14}$$

where P_i is defined as:

$$P_i(\mathbf{k}_i) = \prod_{1 \le l \le k_i} \frac{h_i(l-1)}{f_i(l)} \prod_r \prod_{1 \le l \le k_{ir}} h_{ir}(l-1) \prod_\alpha \prod_{1 \le i \le k_{i\alpha}} \frac{e_{i\alpha} h_{i\alpha}(l-1)}{\mu_{i\alpha}}. \tag{15}$$

The reason for using $\mathbf{k}_i$ in the above formula is that the factors $P_i(\mathbf{k}_i)$ do not depend on the order of the jobs in the station.

The distribution of populations (numbers of each class of job in each station) is obtained from the above after summing out over all possible σ_{il} and disregarding the order of the jobs in the station. This gives rise to a multinomial coefficient that we absorb:

$$\pi(\mathbf{k}) = \frac{1}{G(\mathbf{K})} \prod_i \left(\binom{k_i}{k_{i1}\, k_{i2} \ldots k_{iC}} P_i(\mathbf{k}_i) \right) \tag{16}$$

$$= \frac{1}{G(\mathbf{k})} \prod_i A_i(\mathbf{k}_i). \tag{17}$$

Here we defined:

$$A_i(\mathbf{k}_i) = \binom{k_i}{k_{i1}\, k_{i2} \ldots k_{iC}} P_i(\mathbf{k}_i). \tag{18}$$

For convenience in what follows, we define the function $\iota(P)$ by:

$$\iota(P) = \begin{cases} 1 & \text{if } P \text{ is true} \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

Furthermore, we define the value of a sum over an empty range as zero.

4. General Formula for Throughput

Using the equilibrium state distribution (13) general formulas for the throughputs can be derived from the following theorem.

Theorem 1. *In an open, mixed or closed queueing network with blocking as described above, the throughput of class α jobs at station i when the total population of the network is* $\mathbf{K}$, *is:*

$$\lambda_{i\alpha}(\mathbf{K})=\frac{e_{i\alpha}}{G(\mathbf{K})}\left[p_{i\alpha,0}\,H_{i\alpha,0}(\mathbf{K})+\sum_{j\beta}p_{i\alpha,j\beta}\,H_{i\alpha,j\beta}(\mathbf{K}-\mathbf{u}_{[\alpha]})\right] \tag{20}$$

where the functions H are defined by:

$$H_{i\alpha,0}(\mathbf{K})=\sum_{\mathbf{k}\in\mathbf{S}(\mathbf{K})}b_{i\alpha}(\mathbf{k}_i)\prod_s A_s(\mathbf{k}_s), \tag{21}$$

$$H_{i\alpha,j\beta}(\mathbf{K})=\sum_{\mathbf{k}\in\mathbf{S}(\mathbf{K})}b_{i\alpha}(\mathbf{k}_i)\,b_{j\beta}(\mathbf{k}_j)\prod_s A_s(\mathbf{k}_s). \tag{22}$$

By our convention about sums over empty ranges we have $H_{i\alpha,j\beta}(0)=H_{i\alpha,0}(0)=0$. In Eq. (20) $G(\mathbf{K})$ is the normalization constant for the network and the A_i are the functions defined by Eq. (18).

Proof. The throughput of jobs of class α at station i is the rate at which jobs in class α that finish service at station i are accepted at other stations in the network or leave it. This can be written

$$\lambda_{i\alpha}(\mathbf{K})=\sum_{\mathbf{x}}\pi(\mathbf{x})\sum_{1\le l\le k_i}\iota(\kappa_{il}=\alpha)\,\iota(\sigma_{il}=1)\,v_{i\alpha}\,\phi_i(l,k_i)\,f_i(k_i)\cdot\left[p_{i\alpha,0}+\sum_{j\beta}p_{i\alpha,j\beta}\,b_{j\beta}(\mathbf{k}_j)\right]. \tag{23}$$

Exchanging the order of summation so that the inner sums (over j and β) become outermost we get

$$\lambda_{i\alpha}(\mathbf{K})=\frac{1}{G(\mathbf{K})}\left[p_{i\alpha,0}\,X_{i\alpha,0}+\sum_{j\beta}p_{i\alpha,j\beta}\,X_{i\alpha,j\beta}\right]. \tag{24}$$

Here we defined:

$$X_{i\alpha,0}=G(\mathbf{K})\sum_{\mathbf{x}}\pi(\mathbf{x})\left[\sum_{1\le l\le k_i}\iota(\kappa_{il}=\alpha)\,\iota(\sigma_{il}=1)\,v_{i\alpha}\,\phi_i(l,k_i)\,f_i(k_i)\right], \tag{25}$$

$$X_{i\alpha,j\beta}=G(\mathbf{K})\sum_{\mathbf{x}}\pi(\mathbf{x})\left[\sum_{1\le l\le k_i}\iota(\kappa_{il}=\alpha)\,\iota(\sigma_{il}=1)\,v_{i\alpha}\,\phi_i(l,k_i)\,f_i(k_i)\,b_{j\beta}(\mathbf{k}_j)\right]. \tag{26}$$

The sum $X_{i\alpha,0}$ can be considered a special case of $X_{i\alpha,j\beta}$ in which $b_{j\beta}(\mathbf{k}_j)=1$. So we will concentrate on the other sum and recover $X_{i\alpha,0}$ afterwards using this fact.

First we substitute the equilibrium state probabilities (13) into the sum (26) and simplify. We can split the sum over the state of the network, x, into a sum over the state of station i, $\mathbf{x}_i$, and a sum over the rest of the state of the network. We use the functions P_i defined by (15)

$$X_{i\alpha,j\beta}=\sum_{\mathbf{x}_i}\sum_{\mathbf{x}:\mathbf{x}_i}\prod_{\substack{s\neq i\\ s\neq j}}[P_s(\mathbf{k}_s)\prod_{1\le l\le k_s}r_{s\kappa_{sl}}(\sigma_{sl})][P_j(\mathbf{k}_j)\prod_{1\le l\le k_j}r_{j\kappa_{jl}}(\sigma_{jl})]$$

$$\cdot b_{j\beta}(\mathbf{k}_j)\sum_{1\le l\le k_i}[P_i(\mathbf{k}_i)\prod_{1\le n\le k_i}r_{i\kappa_{in}}(\sigma_{in})]\,\iota(\kappa_{in}=\alpha)\,\iota(\sigma_{in}=1)\,v_{i\alpha}\,\phi_i(l,k_i)\,f_i(k_i)$$

$$=\sum_{\mathbf{x}_i}[v_{i\alpha}f_i(k_i)\sum_{1\le l\le k_i}[P_i(\mathbf{k}_i)\prod_{1\le n\le k_i}r_{i\kappa_{il}}(\sigma_{il})]\,\iota(\kappa_{il}=\alpha)\,\iota(\sigma_{il}=\alpha)\,\phi_i(l,k_i)]$$

$$\cdot\sum_{\mathbf{x}:\mathbf{x}_i}\prod_{\substack{s\neq i\\ s\neq j}}[P_s(\mathbf{k}_s)\prod_{1\le l\le k_s}r_{s\kappa_{sl}}(\sigma_{sl})][P_j(\mathbf{k}_j)\prod_{1\le l\le k_j}r_{j\kappa_{jl}}(\sigma_{jl})]\,b_{j\beta}(\mathbf{k}_j). \tag{27}$$

The states of station i and the rest of the network depend from each other, if at all, only through the restrictions of constant populations of closed chains. This means that we can split off the sum over the permutations of the jobs in station i and the sums over the remaining phases of service of all jobs in that station. Take a fixed population $\mathbf{k}_i$ in station i. First consider the sum S defined by:

$$S=\sum_{x_i:\mathbf{k}_i}\sum_{1\le l\le k_i}P_i(\mathbf{k}_i)\prod_{1\le n\le k_i}r_{i\kappa_{in}}(\sigma_{in})\,\iota(\kappa_{in}=\alpha)\,\iota(\alpha_{in}=1)\,v_{i\kappa_{in}}\,\phi_i(l,k_i)\,f_i(k_i). \tag{28}$$

Note that only terms with $\kappa_{in}=\alpha$ and $\sigma_{in}=1$ can survive in the inner sum of (28). Furthermore, of the products of the $r_{i\kappa_{in}}(\sigma_{in})$ all those that are *not* picked out by the ι functions sum up to one.

Using all these facts to simplify Eq. (28) and using the function P_i defined by (15), gives

$$S=P_i(\mathbf{k}_i)\,v_{i\alpha}\,r_{i\alpha}(1)\,f_i(k_i)\sum_{\mathbf{n}_i:\mathbf{k}_i}\sum_{1\le l\le k_i}\iota(\kappa_{il}=\alpha)\,\phi_i(l,k_i)$$

$$=P_i(\mathbf{k}_i)\,\mu_{i\alpha}\,f_i(k_i)\sum_{1\le l\le k_i}\phi_i(l,k_i)\sum_{\mathbf{n}_i:\mathbf{k}_i}\iota(\kappa_{il}=\alpha). \tag{29}$$

The innermost sum in expression (29) is a sum over all permutations of the jobs in station i. There are

$$\binom{k_i-1}{k_{i1}\ldots k_{i\alpha}-1\ldots k_{iC}} \tag{30}$$

permutations that have a job of class α at any particular position l. By definition, the $\phi_i(l,k_i)$ add up to one. As each permutation gives a sum with the same value, by (6), expression (29) simplifies to

$$\binom{k_i-1}{k_{i1}\ldots k_{i\alpha}-1\ldots k_{iC}}P_i(\mathbf{k}_i)\,\mu_{i\alpha}\,f_i(k_i). \tag{31}$$

By the form of the functions A_i defined by (18) we can then write

$$\binom{k_i-1}{k_{i1}\ldots k_{i\alpha}-1\ldots k_{iC}} P_i(\mathbf{k}_i)\,\mu_{i\alpha}\,f_i(k_i)=e_{i\alpha}\,A_i(\mathbf{k}_i-\mathbf{u}_\alpha)\,b_{i\alpha}(\mathbf{k}_i-\mathbf{u}_\alpha). \tag{32}$$

Now we turn to the rest of the expression for $X_{i\alpha,j\beta}$ in Eq. (27). We are summing this over all possible states of the rest of the network, given that the population in station i is given by $\mathbf{k}_i$. This includes, in particular, a sum over all possible remaining phases of service and over all possible permutations of the jobs in the other stations. The summation over all remaining phases of service makes the $r_{s\kappa_{s1}}(\sigma_{s\kappa_{s1}})$ disappear, and the sum over the permutations transforms $P_s(\mathbf{k}_s)$ into $A_s(\mathbf{k}_s)$:

$$X_{i\alpha,j\beta}=e_{i\alpha}\sum_{\mathbf{k}\in\mathbf{S}(\mathbf{K})} b_{i\alpha}(\mathbf{k}_i-\mathbf{u}_\alpha)\,A_i(\mathbf{k}_i-\mathbf{u}_\alpha)\,b_{j\beta}(\mathbf{k}_j)\,A_j(\mathbf{k}_j)\prod_{\substack{s\neq i\\ s\neq j}} A_s(\mathbf{k}_s). \tag{33}$$

The sum in Eq. (33) is simply a sum over the populations of the network with one less job in routing chain $[\alpha]$. To see this, start with the population space $\mathbf{S}(\mathbf{K}-\mathbf{u}_{[\alpha]})$. If we add a job in chain $[\alpha]$ to station i, leaving all other jobs undisturbed, we get almost the population state $\mathbf{S}(\mathbf{K})$, only those states that have $k_{i[\alpha]}=0$ are missing because of the fixed job in chain $[\alpha]$ in station i. But when we go the other way, i.e. from $\mathbf{S}(\mathbf{K})$ to $\mathbf{S}(\mathbf{K}-\mathbf{u}_{[\alpha]})$, the states with no class α jobs in station i disappear, since for $k_{i\alpha}=-1$ the functions P_i (and A_i) vanish.

We define a new set of functions:

$$H_{i\alpha,j\beta}(\mathbf{K})=\sum_{\mathbf{k}\in\mathbf{S}(\mathbf{K})} b_{i\alpha}(\mathbf{k}_i)\,b_{j\beta}(\mathbf{k}_j)\prod_s A_s(\mathbf{k}_s), \tag{34}$$

$$H_{i\alpha,0}(\mathbf{K})=\sum_{\mathbf{k}\in\mathbf{S}(\mathbf{K})} b_{i\alpha}(\mathbf{k}_i)\prod_s A_s(\mathbf{k}_s). \tag{35}$$

Equation (35) results from Eq. (34) by taking the environment 0 as a pair (station, class) that never rejects a job. Also, if there is a term $p_{i\alpha,0}H_{i\alpha,0}$, class α belongs to an open chain. But in that case, to delete a job of class α from the total population $\mathbf{K}$ makes no difference.

Pulling all results together, we get:

$$\lambda_{i\alpha}(\mathbf{K})=\frac{e_{i\alpha}}{G(\mathbf{K})}\left[p_{i\alpha,0}\,H_{i\alpha,0}(\mathbf{K})+\sum_{j\beta} p_{i\alpha,j\beta}\,H_{i\alpha,j\beta}(\mathbf{K}-\mathbf{u}_{[\alpha]})\right]. \tag{36}$$

Equation (36) is the result claimed in the theorem, Eq. (20). □

Remark. It is interesting that for non-blocking networks the functions H reduce to the normalization constant G. For α in a closed routing chain the formula for throughputs then reduces to Buzen's formula [7] and for α in an open chain to the identity $\lambda_{i\alpha}=e_{i\alpha}$.

Some elementary properties of the functions $H_{i\alpha,j\beta}$ are given by the following lemma.

Lemma 1. *The functions $H_{i\alpha,j\beta}$ are symmetric in $i\alpha$ and $j\beta$:*

$$H_{i\alpha,j\beta}(\mathbf{K}) = H_{j\beta,i\alpha}(\mathbf{K}). \tag{37}$$

If station j does not block jobs of class β (i.e. if $b_{j\beta}(\mathbf{k}_j) = 1$) then

$$H_{i\alpha,j\beta}(\mathbf{K}) = H_{i\alpha,0}(\mathbf{K}). \tag{38}$$

Proof. Equation (37) is immediate from the definition of the $H_{i\alpha,j\beta}$. Equation (38) follows by the observation made above about the relation between $X_{i\alpha,j\beta}$ and $X_{i\alpha,0}$. □

5. Computation of Performance Measures

The above results are general, but of a theoretical nature. Practical algorithms to compute performance measures are needed. This implies algorithms to compute the normalization constant $G(\mathbf{K})$, the mean populations and the throughputs. From the mean populations and the throughputs one gets mean sojourn times by Little's law [25]. It is convenient to treat the cases of open, closed and mixed networks separately since different methods are used for each type.

5.1. Open Networks

The result of Theorem 1 can be simplified for the particular case of open networks, as the following corollary shows.

Corollary 1. *For open networks, the throughputs are given by:*

$$\lambda_{i\alpha} = e_{i\alpha}\bar{b}_{i\alpha}\Big[p_{i\alpha,0} + \sum_{j\beta} p_{i\alpha,j\beta}\bar{b}_{j\beta}\Big]. \tag{39}$$

Here we have used bars to indicate expected values.

Proof. In [3] we proved that the populations of the stations are independent if the network is open. Moreover, the population distributions of the stations are exactly the same as for isolated stations of the same description and subject to independent Poisson streams of rates $e_{i\alpha}$ of jobs in class α. This is just what happens in classical open networks. If the network is open, the populations of all chains are infinite and so we can drop the parameter $\mathbf{K}$ of both G and $H_{i\alpha,j\beta}$. We confront sums of the form:

$$\begin{aligned} H_{i\alpha,j\beta} &= \sum_{\mathbf{k}} b_{i\alpha}(\mathbf{k}_i)\, A_i(\mathbf{k}_i)\, b_{j\beta}(\mathbf{k}_j)\, A_j(\mathbf{k}_j) \prod_{\substack{s\neq i\\ s\neq j}} A_s(\mathbf{k}_s) \\ &= G(\mathbf{K}) \prod_{\substack{s\neq i\\ s\neq j}} \sum_{\mathbf{k}_s} \pi_s(\mathbf{k}_s) \Big[\sum_{\mathbf{k}_i} b_{i\alpha}(\mathbf{k}_i)\,\pi_i(\mathbf{k}_i)\Big]\Big[\sum_{\mathbf{k}_j} b_{j\beta}(\mathbf{k}_j)\,\pi_j(\mathbf{k}_j)\Big] \\ &= G(\mathbf{K})\,\bar{b}_{i\alpha}\bar{b}_{j\beta}. \end{aligned} \tag{40}$$

The factoring of the sum into a product of sums is possible since the states of the stations are independent. The sums in brackets are nothing but the definitions of the mean values indicated in (40). The special terms for jobs that leave the network are explained in exactly the same way. □

5.2. *Closed Networks*

There is no simplification like the above for open networks in the case of closed and mixed networks. The functions $H_{i\alpha,j\beta}$ are convolution sums that are closely related to the function G that defines the normalization constant for the network.

Certain mixed networks can be handled like closed networks, as the following discussion shows. The same idea can be applied to open networks, but for the case of open networks the method outlined in Sect. 5.1 is much simpler.

If all stations of a network have finite capacities, we get exactly the same balance equations if we add a station with enough capacity to the network and define the routing probabilities appropriately. For simplicity, call the new station 0. We also need a new job class that will be used only in station 0. Call the new job class 0. Formally, we define station 0 as a symmetric station type with:

$$f_0(k_0)=\begin{cases}0 & \text{if } k_0=0\\ 1 & \text{otherwise,}\end{cases} \tag{41}$$

$$\mu_{00}=\gamma. \tag{42}$$

The routing probabilities are slightly redefined:

$$p'_{i\alpha,j\beta}=\begin{cases}p_{i\alpha,j\beta} & \text{if } i,j\neq 0\\ p_{0,j\beta} & \text{if } i=0,\quad \alpha=0 \text{ and } j\neq 0\\ p_{i\alpha,0} & \text{if } i\neq 0,\quad j=0 \text{ and } \beta=0\\ 0 & \text{otherwise.}\end{cases} \tag{43}$$

The above routing probabilities were selected so that only jobs of class 0 can enter station 0 (all job classes go to class 0 when leaving the original network) and so that the total rate at which jobs try to enter the original network is still γ. We still have to define the number of jobs in the new routing chain [0]. The number of jobs has to be such that all stations in the original network are full when the new station 0 is empty. If this is satisfied, the new network is clearly equivalent to the original network, but it is a closed network.

We now turn to the practical aspects of computing the desired performance measures. The computation of the mean populations of the stations is straightforward for a closed network, almost exactly the same methods that would be used to compute them in a classical network with load-dependent service efforts can be used. The $H_{i\alpha,j\beta}$ can be computed from scratch by a convolution algorithm. We define the following arrays:

a_i: Array with elements $A_i(k_i)$.

$\mathbf{k}_{i\alpha}$: Array with elements $k_{i\alpha} A_i(\mathbf{k}_i)$.

$\mathbf{b}_{i\alpha}$: Array with elements $b_{i\alpha}(\mathbf{k}_i) A_i(\mathbf{k}_i)$.

We write $*$ for the convolution operation. If we understand that the multiplication of arrays means their convolution, we can write for the (array of) normalization constants of the network:

$$\mathbf{G} = \prod_t \mathbf{a}_t. \tag{44}$$

For the (array of) mean populations we have:

$$\bar{\mathbf{k}}_{i\alpha} = \mathbf{k}_{i\alpha} * \prod_{t \neq i} \mathbf{a}_t \tag{45}$$

and for the (array of) $H_{i\alpha, j\beta}$:

$$\mathbf{H}_{i\alpha, j\beta} = \mathbf{b}_{i\alpha} * \mathbf{b}_{j\beta} * \prod_{\substack{t \neq i \\ t \neq j}} \mathbf{a}_t. \tag{46}$$

When $j = 0$, Eq. (46) simplifies to:

$$\mathbf{H}_{i\alpha, 0} = \mathbf{b}_i * \prod_{t \neq i} \mathbf{a}_t. \tag{47}$$

Equations (44) to (47) provide a straightforward method to compute the $G(\mathbf{K})$ and $H_{i\alpha, j\beta}(\mathbf{K})$, and so the throughputs in the network. The problem with this approach is the large amount of computation required. Another, less onerous, approach is to compute the array $\mathbf{G}$ and to compute the other needed quantities from it by deconvolution and convolution operations.

We denote the inverse of the convolution operation by $\div$. From Eqs. (44) to (47) we can write:

$$\bar{\mathbf{k}}_{i\alpha} = (\mathbf{G} \div \mathbf{a}_i) * \mathbf{k}_{i\alpha}, \tag{48}$$

$$\mathbf{H}_{i\alpha, 0} = (\mathbf{G} \div \mathbf{a}_i) * \mathbf{b}_{i\alpha}, \tag{49}$$

$$\mathbf{H}_{i\alpha, j\beta} = (\mathbf{H}_{i\alpha, 0} \div \mathbf{a}_j) * \mathbf{b}_{j\beta}. \tag{50}$$

Using Eqs. (44) and (48) to (50) provides a faster way to compute the needed values and the throughputs in the network. A similar method has been used to obtain performance measures in classical networks [25].

5.3. *Mixed Networks*

In the general case of mixed networks complications arise. At least some of the sums in Eqs. (21) and (22) are infinite, and they do not fall apart as is the case for open networks in Corollary 1. To compute $G(\mathbf{K})$ and $H_{i\alpha, j\beta}(\mathbf{K})$ an approach similar to the one described by Sauer and Chandy [25] can be used.

We will need to distinguish the populations of open and closed chains for the network and for individual stations. The populations in closed chains will

be written $\mathbf{k}_c$ and $\mathbf{k}_{ic}$ for the network and station i, respecti ... or open chains we write $\mathbf{k}_0$ and $\mathbf{k}_{i0}$. So we get

$$\begin{aligned} G(\mathbf{K}) &= \sum_{\mathbf{k}} \prod_{t} A_t(\mathbf{k}_t) \\ &= \sum_{\mathbf{k}_c} \sum_{\mathbf{k}_0} \prod_{t} A_t(\mathbf{k}_{tc}, \mathbf{k}_{t0}) \\ &= \sum_{\mathbf{k}_c} \prod_{t} \sum_{\mathbf{k}_{t0}} A_t(\mathbf{k}_{tc}, \mathbf{k}_{t0}). \end{aligned} \tag{51}$$

Here we have used the fact that the open chain populations ... stations are independent. This allows us to write the sum of products as a product of sums, like it was done in Eq. (40). If we define

$$\tilde{A}_t(\mathbf{k}_{tc}) = \sum_{\mathbf{k}_{t0}} A_t(\mathbf{k}_{tc}, \mathbf{k}_{t0}) \tag{52}$$

we can write the normalization constant of the network as:

$$G(\mathbf{K}) = \sum_{\mathbf{k}_c} \prod_{t} \tilde{A}_t(\mathbf{k}_{tc}). \tag{53}$$

The functions $H_{i\alpha, j\beta}$ can be handled in the same way:

$$\begin{aligned} H_{i\alpha, j\beta}(\mathbf{K}) &= \sum_{\mathbf{k}} b_{i\alpha}(\mathbf{k}_i)\, A_i(\mathbf{k}_i)\, b_{j\beta}(\mathbf{k}_j)\, A_j(\mathbf{k}_j) \prod_{\substack{t \neq i \\ t \neq j}} A_t(\mathbf{k}_t) \\ &= \sum_{\mathbf{k}_c} \sum_{\mathbf{k}_0} [b_{i\alpha}(\mathbf{k}_{ic}, \mathbf{k}_{i0})\, A_i(\mathbf{k}_{ic}, \mathbf{k}_{i0})\, b_{j\beta}(\mathbf{k}_{jc}, \mathbf{k}_{j0})\, A_j(\mathbf{k}_{jc}, \mathbf{k}_{j0}) \prod_{\substack{t \neq i \\ t \neq j}} A_t(\mathbf{k}_{tc}, \mathbf{k}_{t0})] \\ &= \sum_{\mathbf{k}_c} [\sum_{\mathbf{k}_{i0}} b_i\alpha(\mathbf{k}_{ic}, \mathbf{k}_{i0})\, A_i(\mathbf{k}_{ic}, \mathbf{k}_{i0})] [\sum_{\mathbf{k}_{i0}} b_{j\beta}(\mathbf{k}_{jc}, \mathbf{k}_{j0})\, A_j(\mathbf{k}_{jc}, \mathbf{k}_{j0})] \\ &\quad \cdot \prod_{\substack{t \neq i \\ t \neq j}} [\sum_{\mathbf{k}_{t0}} A_t(\mathbf{k}_{tc}, \mathbf{k}_{t0})]. \end{aligned} \tag{54}$$

Defining $\tilde{b}_{i\alpha}(\mathbf{k}_{ic})$ by:

$$\tilde{b}_{i\alpha}(\mathbf{k}_{ic})\, \tilde{A}_i(\mathbf{k}_{ic}) = \sum_{\mathbf{k}_{i0}} b_{i\alpha}(\mathbf{k}_{ic}, \mathbf{k}_{i0})\, A_i(\mathbf{k}_{ic}, \mathbf{k}_{i0}) \tag{55}$$

we can write (54) as

$$H_{i\alpha, j\beta}(\mathbf{K}) = \sum_{\mathbf{k}_c} \tilde{b}_{i\alpha}(\mathbf{k}_{ic})\, \tilde{b}_{j\beta}(\mathbf{k}_{jc}) \prod_{t} \tilde{A}_t(\mathbf{k}_{tc}). \tag{56}$$

For the case $j = 0$ we have

$$H_{i\alpha, 0}(\mathbf{K}) = \sum_{\mathbf{k}_c} \tilde{b}_{i\alpha}(\mathbf{k}_{ic}) \prod_{t} \tilde{A}_t(\mathbf{k}_{tc}). \tag{57}$$

Equations (53), (56) and (57) are convolution sums. Note the similarity of Eq. (56) and (57) to Eqs. (46) and (47). We define the following arrays

$\tilde{\mathbf{a}}_i$: Array with elements $\tilde{A}_i(\mathbf{k}_{ic})$.

$\tilde{\mathbf{k}}_{i\alpha}$: Array with elements

$$\sum_{\mathbf{k}_{i0}} k_{i\alpha} A_i(\mathbf{k}_{ic}, \mathbf{k}_{i0}) \tag{58}$$

Expression (58) simplifies to $k_{i\alpha} \tilde{A}_i(\mathbf{k}_{ic})$ when α is in a closed chain.

$\tilde{\mathbf{b}}_{i\alpha}$: Array with elements $\tilde{b}_{i\alpha}(\mathbf{k}_{ic}) \tilde{A}_i(\mathbf{k}_{ic})$.

Note the similarity of these definitions with the corresponding definitions in Sect. (5.2). We can write

$$\mathbf{G} = \prod_t \tilde{\mathbf{a}}_t, \tag{59}$$

$$\bar{\mathbf{k}}_{i\alpha} = \tilde{\mathbf{k}}_{i\alpha} * \prod_{t \neq i} \tilde{\mathbf{a}}_t, \tag{60}$$

$$\mathbf{H}_{i\alpha, j\beta} = \tilde{\mathbf{b}}_{i\alpha} * \tilde{\mathbf{b}}_{j\beta} * \prod_{\substack{t \neq i \\ t \neq j}} \tilde{\mathbf{a}}_t, \tag{61}$$

$$\mathbf{H}_{i\alpha, 0} = \tilde{\mathbf{b}}_{i\alpha} * \prod_{t \neq i} \tilde{\mathbf{a}}_t. \tag{62}$$

Using the same idea as in Sect. (5.2), we get the following formulas

$$\bar{\mathbf{k}}_{i\alpha} = (\mathbf{G} \div \tilde{\mathbf{a}}_i) * \tilde{\mathbf{k}}_{i\alpha}, \tag{63}$$

$$\mathbf{H}_{i\alpha, 0} = (\mathbf{G} \div \tilde{\mathbf{a}}_i) * \tilde{\mathbf{b}}_{i\alpha}, \tag{64}$$

$$\mathbf{H}_{i\alpha, j\beta} = (\mathbf{H}_{i\alpha, 0} \div \tilde{\mathbf{a}}_j) * \tilde{\mathbf{b}}_{j\beta}. \tag{65}$$

Equations (52), (55), (58), (59) and (63) to (65) describe the algorithm. The infinite sums in Eqs. (52), (55) and (58) are needed for each possible population of the closed chains in the station. Much work can be saved by using the product form of A_i given by Eqs. (15) and (18). All the dependence on $\mathbf{k}_{ic}$, except for the total number of jobs in the closed chains, can be factored out of the sum. It is then enough to compute an infinite sum for each *total* population of the closed chains in the station. This observation also shows that the closed network defined by the quantities with tildes is a closed network of the type investigated here.

6. Complexity of the Algorithms

As can be seen from Theorem 1, the complexity of the computation of the throughputs depends on the topology of the network. This contrasts with the case of classical networks, where the computation of the throughputs does not depend on the number of connections between the stations. The case of an open network reduces essentially to handle N isolated stations by Corollary 1. Each station gives rise to a (possibly infinite) sum. By the results of Sect. 5.3,

the case of a mixed network reduces to the case of a closed network after evaluating the (possibly infinite) sums of Eqs. (52), (55) and (58). No general rules can be given for the sums that arise in these cases. In favorable cases (a single server station that does not block, for example), the sums can be evaluated analytically. In unfavorable cases (general load-dependence of the service effort and/or general blocking functions) the sums will have to be evaluated numerically. Only in the case of closed networks we can give the complexities of the algorithms with certainity.

A $H_{i\alpha, j\beta}$ must be computed for each pair $(i\alpha, j\beta)$ for which $p_{i\alpha, j\beta} \neq 0$. By Lemma 1 we need to compute only half of the above values by symmetry. Call the number of $H_{i\alpha, j\beta}$ that have to be computed E. Note that E could be very large, up to $\frac{1}{2} C \cdot N \cdot (N-1)$ if every station is connected to all the other stations in each class and there are no class changes.

We do a total of $(N-1)$ convolutions to compute G. To compute the needed $\mathbf{H}_{i\alpha, 0}$ we compute at most E deconvolutions and E convolutions. To get the needed $\mathbf{H}_{i\alpha, j\beta}$ from the $\mathbf{H}_{i\alpha, 0}$ we compute E deconvolutions and E convolutions. This gives a total of $O(N+4E)$ convolutions and deconvolutions. Each convolution/deconvolution involves $O(RK^R)$ operations, if we assume that there are R routing chains and that there are K jobs in each chain. The total number of operations to compute the throughputs is then $O(RK^R(N+4E))$. This contrasts with the convolution algorithm for classical networks [22, 23] where the total number of operations to compute throughputs is $O(NRK^R)$, since only G is needed. To compute the mean populations in a station one executes a convolution and a deconvolution for each class. This totals $O(CNRK^R)$ operations. The operation count for the mean populations is the same as in the classical case, since the algorithm is essentially the same.

A copy of the array G is needed all the time in the above, and a scratch space of the same size is needed to hold the values of the $\mathbf{H}_{i\alpha, 0}$ when computing the $\mathbf{H}_{i\alpha, j\beta}$. The space complexity of the algorithm is then $O(K^R)$.

7. Conclusions

We have derived general formulas for throughput in some queueing networks with blocking. The formulas, while significantly more complex than the classical formulas [7] are still similar to them. In particular, general mixed networks can be treated in the same way as in classical networks. From the general formulas we construct algorithms to compute performance measures. The algorithms are more complex than their counterparts for classical networks, but they should be useful in practice. As is the case with classical networks, the algorithms proposed will become infeasible for large networks with many classes and/or large populations. So, an interesting area for future research would be to find exact or approximate algorithms that take less time or space.

References

1. Akyildiz, I.F.: Exact Analysis of Queueing Networks with Rejection Blocking. Proc. Int. Conf. "Queueing Networks with Finite Capacities", pp. 24–36. Amsterdam: North-Holland 1988

2. Akyildiz, I.F., von Brand, H.: Dual and Selfdual Networks of Queues with Rejection Blocking. Comput. J. (To appear)
3. Akyildiz, I.F., von Brand, H.: Exact Solutions for Open, Closed and Mixed Queueing Networks with Rejection Blocking. J. Theor. Comput. Sci. (1989) (To appear)
4. Balsamo, S., Iazeolla, G.: Some Equivalence Properties for Queueing Networks with and without Blocking. Proc. Performance 83 Conference, pp. 351–360. Amsterdam: North Holland 1983
5. Barbour, A.: Networks of Queues and the Method of Stages. Adv. Appl. Probab. **8**, 584–591 (1976)
6. Baskett, F., Chandy, K.M., Muntz, R.R., Palacios, F.G.: Open, Closed and Mixed Networks of Queues with Different Classes of Customers, J. A.C.M. **22**, 248–260 (1975)
7. Buzen, J.P.: Queueing Network Models of Multiprogramming. Ph.D. Thesis, Div. Eng. Appl. Sci., Harvard Univ., Cambridge, Mass., Aug. 1971
8. Caseau, P., Pujolle, G.: Throughput Capacity of a Sequence of Queues with Blocking due to Finite Waiting Room. IEEE Trans. Software Eng. **SE-5**, 631–642 (1979)
9. Chandy, K.M., Martin, A.J.: A Characterization of Product Form Queueing Networks. J. A.C.M. **30**, 286–299 (1983)
10. Cohen, J.W.: The Multiple Phase Service Network with Generalized Processor Sharing. Acta Inf. **12**, 245–284 (1979)
11. van Dijk, N.M., Tijms, H.: Insensitivity in Two-Node Blocking Models with Applications. Proc. Teletraffic Anal. Comput. Perform. Evaluation, pp. 329–340. Amsterdam: North-Holland 1986
12. van Dijk, N.M., Akyildiz, I.F.: Networks of Mixed Processor Sharing Parallel Queues and Common Pools. Technical Report, Georgia Tech., GIT-ICS-88-022, June 1988
13. Gordon, W.J., Newell, G.F.: Cyclic Queueing Systems with Restricted Queues. Oper. Res. **15**, 266–277 (1967)
14. Hordijk, A., van Dijk, N.M.: Networks of Queues with Blocking. Proc. Performance '81, pp. 51–65. Amsterdam: North Holland 1981
15. Hordijk, A., van Dijk, N.M.: Adjoint Processes, Job-Local-Balance and Insensitivity of Stochastic Networks. Bull. 44th Session Int. Stat. Inst. **50**, 776–788 (1982)
16. Hordijk, A., van Dijk, N.M.: Networks of Queues. Part I: Job-Local-Balance and the Adjoint Process. Part II: General Routing and Service Characteristics. Proc. Int. Conf. Modeling Comput. Syst. Vol. 60, pp. 158–205. Berlin Heidelberg New York: Springer 1983
17. Kelly, F.P.: Networks of Queues with Customers of Different Types. J. Appl. Probab. **12**, 542–554 (1975)
18. Kelly, F.P.: Reversibility and Stochastic Networks. New York: Wiley 1979
19. Noetzel, A.S.: A Generalized Queueing Discipline for Product Form Network Solutions. J. ACM **26**, 779–793 (1979)
20. Onvural, R.O., Perros, H.G.: On Equivalencies of Blocking Mechanisms in Queueing Networks with Blocking. Oper. Res. Lett. **5**, 293–297 (1986)
21. Pittel, B.: Closed Exponential Networks of Queues with Saturation. The Jackson-Type Stationary Distribution and its Asymptotic Analysis. Math. Oper. Res. 357–378 (1979)
22. Reiser, M., Kobayashi, H.: Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms. IBM J. Res. Dev. **19**, 283–294 (1975)
23. Reiser, M., Kobayashi, H.: Horner's Rule for the Evaluation of General Closed Queueing Networks. Commun. ACM **18**, 592–593 (1975)
24. Samelson, C.H., Bulgren, W.G.: A Note on Product Form Solution for Queueing Networks with Poisson Arrivals and General Service Time Distributions with Finite Means. J. ACM **29**, 830–840 (1982)
25. Sauer, C.H., Chandy, K.M.: Computer Systems Performance Modeling. Englewood Cliffs, N.J.: Prentice Hall 1981
26. Spirn, J.: Queueing Networks with Random Selection for Service. IEEE Trans. Software Eng. **SE-5**, 287–289 (1979)
27. Tijms, H.J.: Stochastic Modeling and Analysis: A Computational Approach. Chichester: John Wiley 1986

Received September 23, 1987/February 27, 1989