Chapter 2
Order Independent Queues

A.E. Krzesinski

Abstract We present a class of queues which are quasi-reversible and therefore preserve product form distribution when connected in multinode networks. The essential feature leading to the quasi-reversibility of these queues is the fact that the total departure rate in any queue state is independent of the order of the customers in the queue. We call such queues Order Independent (OI) queues. A distinguishing feature of the OI class is that, among others, it includes the FCFS, processor sharing, infinite server and MSCCC queues but not the LCFS queue. We next examine OI queues where arrivals to the queue are lost when the number of customers in the queue equals an upper bound. We prove that such queues satisfy partial balance and we obtain the stationary distribution for the OI loss queue by normalising the stationary probabilities of the corresponding OI queue without losses. OI loss queues can be used to model systems with simultaneous resource possession with the option of queueing blocked customers. The OI loss queue thus extends previous loss models where customers are rejected when processing resources are not available. The OI loss class is next extended to include networks of queues which can be used to model systems with complex loss mechanisms. We finally present several applications of OI loss queues and OI loss networks.

2.1 Introduction

Much attention has been given to product form stationary distributions for queueing networks and associated Markov processes – see [17, 25, 30, 32] for a list of references. Most of the known processes which have a product form distribution are reversible or quasi-reversible. Quasi-reversibility was first presented by Muntz [24]

A.E. Krzesinski
Department of Mathematical Sciences, University of Stellenbosch, 7600 Stellenbosch, South Africa
e-mail: aek1@cs.sun.ac.za

R.J. Boucherie and N.M. van Dijk (eds.), *Queueing Networks: A Fundamental Approach*, International Series in Operations Research & Management Science 154,
DOI 10.1007/978-1-4419-6472-4_2, © Springer Science+Business Media, LLC 2011

and later developed into a general framework [17, 29]. In the product form Jackson networks [16], each node in isolation is a reversible Markov process. In the more general BCMP product form networks [3], nodes may not be reversible, but they are quasi-reversible.

However, product form is not restricted to systems of interconnected quasi-reversible nodes. For example, Pollett [27, 28] provided a framework for interconnecting a collection of reversible Markov processes in such a way that the resulting process has a product form invariant measure with respect to which the process is reversible, although individual nodes need not be quasi-reversible. Kelly [17] developed a general framework for interconnecting quasi-reversible processes and introduced a class of quasi-reversible symmetric queues that can be connected in a network with a product form distribution. Walrand [29, 30, 31] provided probabilistic arguments demonstrating how quasi-reversibility implies product form distribution. One of the most general frameworks for interconnecting quasi-reversible nodes [14] does not require the node to be customer preserving for each type of customer.

Other quasi-reversible queues are often variations of reversible queues [17, 30] and can be described as reversible queues by choosing an appropriate state space for the queue. There are other (unclassified) quasi-reversible queues [10, 17, 18, 30, 32] and processes such as the quasi-reversible Brownian process considered in [13], Ott's model analysed in [14] and quasi-reversible clustering processes [32].

This paper presents a class of quasi-reversible queues which in general are neither symmetric nor reversible. This class includes a large part of the class of symmetric queues, but not the whole class. In particular, the well known FCFS, Infinite-Server and Processor-Sharing queues are in the considered class which also contains the Multiserver Station with Concurrent Classes of Customers (MSCCC) [7, 11, 12] and the MultiServer centre with Hierarchical Concurrency Constraints (MSHCC) [21]. The quasi-reversibility for this class is not a result of the symmetry of the reversed process as is the case for symmetric queues. It arises from a symmetry property concerning the order of the customers in the queue, namely that the total departure rate in any queue state is independent of the order of the customers in the queue. We call such queues Order Independent (OI) queues.

In section 2.2 we define the OI queue using the notation presented in [17]. This allows us to simplify the comparison of the OI queue with other well known quasi-reversible queues and to emphasise the distinguishing features of the OI queue. The OI property is obtained by analysing the properties of so-called service rate functions which satisfy some special conditions. In section 2.2.3 we prove that if the instantaneous service rate of a multiclass queue is described by a set of such service rate functions, then the queue is quasi-reversible (in the appropriate state space). We derive a closed form expression for the stationary distribution of the queue.

In section 2.2.4 we show that an appropriate choice of relative service rate functions reduces the OI queue to the well known BCMP, MSCCC and MSHCC queues. Another OI queue which is a generalisation of example 3 from [18] is presented. Section 2.3 derives a computationally efficient recursive equation for the stationary distribution. Section 2.4 examines OI queues with losses. We prove that although OI loss queues are not quasi-reversible, their stationary distributions can be obtained

by normalising the stationary probabilities of the corresponding OI queue without losses. Finally, section 2.5 presents several applications of OI loss queues and OI loss networks.

Much of the work presented in this Chapter was done jointly with Sergei Berezner whom I wish to thank for his many valuable insights and discussions on the subject.

2.2 The OI Queue

Consider a queue serving customers of type c where $c \in \mathcal{C}$ and $\mathcal{C}$ is a finite set. Customers of type c arrive individually at the instants of a Poisson stream with rate λ_c. The customers, whether waiting or in service, form a queue in the order of their arrival. Arriving customers join the back of the queue and the front of the queue is identified with position 1. Each customer of type c presents a demand for service time which is exponentially distributed with mean $1/\mu_c$. All the random variables involved in the description of the queue are independent.

Let $\mathbf{C} = (c_n, \ldots, c_1)$ denote the state of a queue of length n where c_i denotes the type of the customer in position i, $i = 1, \ldots, n$. Let 0 denote the empty queue and $\mathcal{S} = \{0\} \cup \bigcup_{n=1}^{\infty} \mathcal{C}^n$ denote the state space of the queue where $\mathcal{C}^n$ is the $n-$fold product space of $\mathcal{C}$.

Let the total service effort in state $(c_n, \ldots, c_1)$ be supplied at the rate $\phi(c_n, \ldots, c_1)$. A portion $\gamma_i(c_n, \ldots, c_1)$ of the total service effort is directed at the customer in queue position i, $i = 1, \ldots, n$. Upon entering service a customer is served without interruption to completion. When the customer in queue position i completes service, the customer departs and the gap in the queue is closed by the obvious shift: the customers in positions $i+1, i+2, \ldots, n$ move to positions $i, i+1, \ldots, n-1$ respectively. We do not require that $\sum_{i=1}^{n} \gamma_i(c_n, \ldots, c_1) = 1$ so that a part of the service facility might be wasted. This is a distinguishing feature of an OI queue and this feature allows complex queueing disciplines such as MSCCC and MSHCC to be described as OI queues.

Note that the OI queue is completely described by the vector of types of customers. For some specific cases it may be possible to introduce auxiliary variables that describe additional characteristics of the queue and for such queues the concept of OI could be further extended. However, this will result in a cumbersome notation without extending the range of the models in the OI class.

We therefore restrict ourselves to queues that are completely described by a vector of types of customers. We also assume that the type membership of each customer does not change while it passes through the queue. This is a severe restriction since it prevents us from considering service consisting of a series of exponential stages and, as a result, general service distributions are not possible for OI queues. This restriction can be dropped for some specific models, for example, for PS or IS queues. But in general, service in stages does not lead to an OI queue and typical OI queues such as M/M/K, MSCCC and MSHCC do not permit service in stages.

2.2.1 The Definition of an OI Queue

We next present a general description of the OI queue and show how the OI property can be obtained by imposing certain conditions on the functions ϕ and γ.

First, we require that the (relative) proportion of the service effort supplied to the customer in queue position i depends only on the composition of the queue up to and including position i. This implies that when a server becomes free the queue is searched from the head to the tail for a customer which can be admitted into service.

Second, we require that in any state $(c_n,\ldots,c_1) \in \mathcal{S}$, the rate at which departures (service completions) occur is independent of the order of the customers in the queue and is thus the same for any state $(c_{\sigma(n)},\ldots,c_{\sigma(1)})$, where σ denotes any permutation of $(1,\ldots,n)$.

Last, we assume that in any state (except for the empty queue) there is a positive rate of service completion. This condition is required in order to ensure the irreducibility of the Markov chain. This restriction is usually satisfied in systems with exponential service that are fully described by a vector of customer types who do not change their type membership.

A formal description of the three conditions presented above can be given as follows. Consider the queue in state $(c_n,\ldots,c_1)$. The departure rate of the customer in queue position i is given by $\phi(c_n,\ldots,c_1)\mu_{c_i}\gamma_i(c_n,\ldots,c_1)$ and the total departure rate is given by the sum of these quantities over all the positions in the queue.

The queue is said to be an OI queue if, for all $(c_n,\ldots,c_1) \in \mathcal{S}$ and all $i = 1,\ldots,n$ the rates of service completion can be written as

$$\phi(c_n,\ldots,c_1)\mu_{c_i}\gamma_i(c_n,\ldots,c_1) = \mu(n)s_i(c_n,\ldots,c_1)$$

such that

(i) $s_i(c_n,\ldots,c_1) = s_i(c_i,\ldots,c_1)$ for any $1 \le i \le n$,
(ii) $k(c_n,\ldots,c_1) = \sum_{i=1}^{n} s_i(c_n,\ldots,c_1)$ is independent of permutations of $(c_n,\ldots,c_1)$, and
(iii) $\mu(n) > 0$ for $n > 0$ and $s_1(c) > 0$ for any $c \in \mathcal{C}$.

The function $s_i(\mathbf{C})$ regulates the rate at which service is given to the customer in position i in the queue relative to the other customers in the queue and the function $\mu(n)$ allows the service rate to depend upon the total number of customers in the queue.

2.2.2 The Implications of the OI Conditions

Condition (i) requires that the relative service rate of any customer in the queue depends only upon its own customer type and the customer type of each customer in front of it in the queue. This implies that it is only necessary to scan the queue from the front to determine the relative service rate given to any particular customer.

Condition (ii) is the distinguishing condition. It requires that the total departure rate due to customer service completions is independent of the order of the customers in the queue. Although this condition is restrictive, section 2.2.4 shows that condition (ii) is satisfied by several well known queues.

Condition (iii) is a necessary and sufficient condition for the Markov process describing the queue to be irreducible. This condition ensures that the empty state can always be reached from any other state due to departures. Because arrivals allow any state to be reached from the empty state, this is sufficient to ensure irreducibility. Note that because $S_i(\mathbf{C}) \geq 0$, a necessary and sufficient condition for $k(\mathbf{C}) > 0$ for any $\mathbf{C} \in \mathcal{S}$ is to require that $s_1(c) > 0$ for any $c \in \mathcal{C}$.

Theorem 2.1. *Conditions (i) and (ii) imply that the relative service rate given to a customer in the queue is independent of the order of the customers ahead of it in the queue.*

Proof. Let $(c_n,\ldots,c_1)$ be a queue state in $\mathcal{S}$ and let $(\sigma(1),\ldots,\sigma(n-1))$ denote a permutation of $(1,\ldots,n-1)$. Thus $(c_n,c_{\sigma(n-1)},\ldots,c_{\sigma(1)})$ is the queue state obtained when the first $n-1$ customers in the queue are rearranged according to the permutation $(\sigma(1),\ldots,\sigma(n-1))$. By definition

$$k(c_n,\ldots,c_1) = \sum_{i=1}^{n} s_i(c_n,\ldots,c_1) = s_n(c_n,\ldots,c_1) + k(c_{n-1},\ldots,c_1)$$

and likewise

$$k(c_n,c_{\sigma(n-1)},\ldots,c_{\sigma(1)}) = s_n(c_n,c_{\sigma(n-1)},\ldots,c_{\sigma(1)}) + k(c_{\sigma(n-1)},\ldots,c_{\sigma(1)}). \quad (2.1)$$

But condition (ii) requires that $k(c_n,c_{\sigma(n-1)},\ldots,c_{\sigma(1)}) = k(c_n,\ldots,c_1)$ and $k(c_{\sigma(n-1)},\ldots,c_{\sigma(1)}) = k(c_{n-1},\ldots,c_1)$. Substituting these two expressions into (2.1) yields $s_n(c_n,\ldots,c_1) = s_n(c_n,c_{\sigma(n-1)},\ldots,c_{\sigma(1)})$ which completes the proof. □

Note that the $s_i(c_i,\ldots,c_1)$ may be dependent on the permutations of $(c_i,\ldots,c_1)$, but not on permutations of $(c_{i-1},\ldots,c_1)$. However $k(c_n,\ldots,c_1)$ is not dependent on permutations of $(c_i,\ldots,c_1)$. For example, consider an OI queue with two customer types where only one customer from each type may be in service. Then, for example, $s_3(2,1,1) = 1$ whereas $s_3(1,2,1) = s_3(1,1,2) = 0$ (the last two instances of $s_3(c_3,c_2,c_1)$ illustrate theorem 2.1) and $k(2,1,1) = k(1,2,1) = k(1,1,2) = 2$.

The following section demonstrates that the restrictions (i)–(iii) on $s_i(\mathbf{C})$ are sufficient to ensure that the OI queue is quasi-reversible at equilibrium and we find the stationary distribution (when it exists).

2.2.3 The Stationary Distribution

The OI queue can be modeled by a continuous–time, homogeneous Markov process $\mathbf{C}(t)$, $t \in \mathbf{R}^+$, $\mathbf{C}(t) \in \mathcal{S}$, where $\mathbf{C}(t)$ denotes the queue state at time t. The Markov

process is irreducible, since transitions due to arrivals allow any state to be reached from the empty state and condition (iii) ensures that the empty state can be reached from any state by the departure transitions.

For any $c \in \mathcal{C}$ and any $(c_n, \ldots, c_1) \in \mathcal{S}$ the transition rate due to type c arrivals is λ_c and the transition rate due to the departure of a type c customer in queue position $i+1$ where $1 \leq i \leq n$ is $\mu(n+1)s_i(c_n, \ldots, c_{i+1}, c, c_i, \ldots, c_1)$. The equilibrium equations for the queue are given by

$$\pi(0) \sum_{c \in \mathcal{C}} \lambda_c = \mu(1)k(c) \sum_{c \in \mathcal{C}} \pi(c) \tag{2.2}$$

and

$$\begin{aligned} &\pi(c_n, \ldots, c_1)\,(\lambda + \mu(n)k(c_n, \ldots, c_1)) \\ &= \sum_{c \in \mathcal{C}} \sum_{i=0}^{n} \mu(n+1)\pi(c_n, \ldots, c_{i+1}, c, c_i, \ldots, c_1)s_{i+1}(c_n, \ldots, c_{i+1}, c, c_i, \ldots, c_1) \\ &\quad + \quad \lambda_{c_n}\pi(c_{n-1}, \ldots, c_1) \end{aligned} \tag{2.3}$$

where $\lambda = \sum_{c \in \mathcal{C}} \lambda_c$.

The arrival flow to the system is a collection of independent Poisson flows each with rate λ_c with future arrivals being independent of the present state of the queue. If we can find a collection of positive numbers $\pi(c_n, \ldots, c_1)$ summing to unity and satisfying the equilibrium Eqs. (2.2) and (2.3) such that for all $(c_n, \ldots, c_1) \in \mathcal{S}$ and all $c \in \mathcal{C}$

$$\sum_{i=0}^{n} \frac{\pi(c_n, \ldots, c_{i+1}, c, c_i, \ldots, c_1)}{\pi(c_n, \ldots, c_1)} \mu(n+1)s_{i+1}(c, c_i, \ldots, c_1) = \beta_c \tag{2.4}$$

then the queue is quasi-reversible and this collection of numbers forms a stationary distribution of the queue [18]. Since customers in an OI queue preserve their type membership, $\beta_c = \lambda_c$. Thus (2.4) can be rewritten as

$$\sum_{i=0}^{n} \frac{\pi(c_n, \ldots c_{i+1}, c, c_i, \ldots, c_1)}{\pi(c_n, \ldots, c_1)} \mu(n+1)s_{i+1}(c, c_i, \ldots, c_1) = \lambda_c. \tag{2.5}$$

Note that condition (i) was applied in (2.4) to replace $s_{i+1}(c_n, \ldots, c_{i+1}, c, c_i \ldots, c_1)$ by $s_{i+1}(c, c_i, \ldots, c_1)$. Substituting (2.5) into (2.3) yields

$$\mu(n)k(c_n, \ldots, c_1)\pi(c_n, \ldots, c_1) = \lambda_{c_n}\pi(c_{n-1}, \ldots, c_1) \tag{2.6}$$

for all $(c_n, \ldots, c_1) \in \mathcal{S}$. From (2.6) we immediately obtain a proposed form of the stationary distribution. The result is stated and proved in the following theorem.

Theorem 2.2. *If the service rate functions $s_i(\cdot)$ conform to the conditions (i)–(iii) then for any $(c_n, \ldots, c_1) \in \mathcal{S}$ a collection of numbers*

$$\pi(c_n,\ldots,c_1)=\pi(0)\prod_{i=1}^{n}\frac{\lambda_{c_i}}{\mu(i)k(c_i,\ldots,c_1)} \tag{2.7}$$

where $\pi(0)$ is an arbitrary positive real number, is a solution to the equilibrium equations. The stationary distribution of the Markov chain exists if and only if

$$G=\sum_{(c_n,\ldots,c_1)\in\mathcal{S}}\prod_{i=1}^{n}\frac{\lambda_{c_i}}{\mu(i)k(c_i,\ldots,c_1)}<\infty$$

in which case the stationary distribution is given by (2.7) with $\pi(0)=1/G$ and the queue is quasi-reversible.

Proof. Equation (2.7) is clearly a solution to (2.6). We need to prove that (2.7) is also a solution to (2.5) and thus the solution to the equilibrium equations. We prove by induction on n that the equality (2.5) holds for all $(c_n,\ldots,c_1)\in\mathcal{S}$ and $c\in\mathcal{C}$.

Consider first the empty queue ($n=0$). Applying (2.7) to the right hand side of (2.5), and noting that $k(c)\equiv s_1(c)$, yields

$$\frac{\mu(1)\pi(c)s_1(c)}{\pi(0)}=\frac{\mu(1)\pi(0)s_1(c)}{\lambda_c\pi(0)\mu(1)k(c)}=\lambda_c$$

so that the base of the induction is proved.

Next assume that (2.7) satisfies (2.5) up to the $n-1$ value of the summation index and consider (2.5) for the value n of the summation index. The left hand side of (2.5) (lhs (2.5)) can be written as

$$\begin{aligned}\text{lhs }(2.5)&=\sum_{i=0}^{n}\frac{\pi(c_n,\ldots,c_{i+1},c,c_i,\ldots,c_1)}{\pi(c_n,\ldots,c_1)}\mu(n+1)s_{i+1}(c,c_i,\ldots,c_1)\\&=\sum_{i=0}^{n-1}\frac{\pi(c_n,\ldots,c_{i+1},c,c_i,\ldots,c_1)}{\pi(c_n,\ldots,c_1)}\mu(n+1)s_{i+1}(c,c_i,\ldots,c_1)\\&\quad+\frac{\pi(c,c_n,\ldots,c_1)}{\pi(c_n,\ldots,c_1)}\mu(n+1)s_{n+1}(c,c_n,\ldots,c_1).\end{aligned}$$

Application of (2.7) yields

$$\begin{aligned}\text{lhs }(2.5)&=\frac{\mu(n)k(c_n,\ldots,c_1)}{\mu(n+1)}\\&\quad\times\sum_{i=0}^{n-1}\frac{\pi(c_{n-1},\ldots,c_{i+1},c,c_i,\ldots,c_1)}{\pi(c_{n-1},\ldots,c_1)k(c_n,\ldots,c_{i+1},c,c_i,\ldots,c_1)}\mu(n+1)s_{i+1}(c,c_i,\ldots,c_1)\\&\quad+\frac{\lambda_c}{k(c,c_n,\ldots,c_1)}s_{n+1}(c,c_n,\ldots,c_1).\end{aligned}$$

Condition (ii) requires that $k(\mathbf{C})$ is independent of permutations of $\mathbf{C}$, hence we obtain $k(c_n,\ldots,c_{i+1},c,c_i,\ldots,c_1)=k(c,c_n,\ldots,c_1)$. Condition (iii) ensures that $k(c,c_n,\ldots,c_1)\mu(n+1)>0$. Thus

$$\text{lhs } (2.5) = \frac{k(c_n,\ldots,c_1)}{k(cc_n\ldots c_1)} \sum_{i=0}^{n-1} \frac{\pi(c_{n-1},\ldots,c_{i+1},c,c_i,\ldots,c_1)}{\pi(c_{n-1},\ldots,c_1)} \mu(n) s_{i+1}(c,c_i,\ldots,c_1) + \frac{\lambda_c s_{n+1}(c,c_n,\ldots,c_1)}{k(c,c_n,\ldots.c_1)}.$$

But by the induction assumption the sum on the right hand side of the above equation is equal to λ_c , which yields

$$\text{lhs } (2.5) = \lambda_c \frac{k(c_n,\ldots,c_1) + s_{n+1}(c,c_n,\ldots,c_1)}{k(c,c_n,\ldots,c_1)} = \lambda_c$$

which completes the induction. □

2.2.4 Models Covered by the OI Class

In this section we show that the OI class includes several of the BCMP queues and part of Kelly's class of symmetric queues. A distinguishing feature of the OI queues is that they include the MSCCC and the MSHCC queues.

2.2.4.1 BCMP Models in the OI Class

The M/M/K queue is an OI queue. Let $\mu(n) = 1$ and

$$s_i(c_n,\ldots,c_1) = \begin{cases} \mu & 1 \le i \le K \\ 0 & i > K \end{cases}$$

for $K \in \mathbf{Z}^+$. These functions $\mu(n)$ and $s_i(\cdot)$ conform to the conditions (i)–(iii) and this OI queue is equivalent to an M/M/K queue. All customer types must have the same average service rate else $k(\mathbf{C})$ would not be independent of permutations of $\mathbf{C}$. We will later show that the M/M/K queue is a special case of the MSCCC queue. This implies that the MSCCC queue could replace the M/M/K queue as a basic construction element (building block) for product form networks.

We next consider the Infinite Server (IS) queue. Let $\mu(n) = 1$ and $s_i(c_n,\ldots,c_1) = \mu_{c_i}$. These functions $\mu(n)$ and $s_i(\cdot)$ conform to the conditions (i)–(iii) and this OI queue is equivalent to an IS queue.

The choice of relative service rates for the Processor Sharing (PS) queue is obvious. Let $\mu(n) = 1/n$ and $s_i(c_n,\ldots,c_1) = \mu_{c_i}$. These functions $\mu(n)$ and $s_i(\cdot)$ conform to the conditions (i)–(iii) and the corresponding OI queue is equivalent to a PS queue. As was mentioned in section 2.2, it is possible to extend the OI framework to describe general service distributions for PS and IS queues. However, this extension complicates the proofs and does not lead to any fundamentally new models since in this case the OI conditions (i)–(iii) become very restrictive. In any event, typi-

cal OI queues such as M/M/K, MSCCC and MSHCC do not admit general service distributions.

Note that the PS and IS queues require all customers to be in service and that in these cases the service effort $k_c(\mathbf{C})$ directed at customers of type c given by

$$k_c(c_n,\ldots,c_1) = \sum_{i=1}^{n} s_i(c_n,\ldots,c_1)\,1(c_i = c)$$

is also independent of permutations of $\mathbf{C}$. In general, it is not necessary that all customers be in service at an OI queue with type dependent service rates. Service disciplines may exist for OI queues with type dependent service rates with a bound on the number of customers in service – all that is necessary for such a queue to be OI is that conditions (i)–(iii) hold.

Because of condition (i) the Last Come First Served (LCFS) queue cannot be modeled by an OI queue. The failure of the OI class to describe the LCFS queue is another argument proving the different nature of the OI class.

If the functions ϕ and γ defined in section 2.2 depend only on the total number of customers in the queue and if we allow the customers to join the queue in different positions then, under some strong symmetric assumptions (see [17] pages 72–73), the well known symmetric queues are obtained.

2.2.4.2 The MSCCC and MSHCC Queues

A distinguishing feature of the OI class is that it includes the MSCCC and MSHCC queues. In fact, the OI queues were found while investigating the MSCCC queue.

The MSCCC queue consists of K parallel identical exponential servers. The customers belong to a set of customer types. Customers of type c arrive individually at the instants of a Poisson stream with rate λ_c and present demands for service time which are exponentially distributed with mean $1/\mu$. The customers are queued for service in the order of their arrival. When a server becomes free, the queue is searched from the front looking for the first customer to admit into service subject to the following constraints: at most K customers can be in service and at most 1 customer of each type c can be in service. The queue can thus be described as FCFS subject to concurrency constraints.

The MSCCC queue was first investigated while simulating shared memory multiprocessors [22]. The simulations conjectured, but could not prove, that the MSCCC queue had a product form solution. An analytic expression for the stationary distribution, which was first obtained by exploring the MSCCC state space using a symbolic mathematics computer package, was presented in [7]. The concurrency constraint was later extended [11] so that at most K customers can be in service and at most $B_c \geq 1$ customer of each type c can be in service. It was later shown [21] that a suitable choice of state descriptor yields a direct calculation of the stationary distribution. However, none of these analyses explained how the MSCCC queue is related to the product form queues.

The OI property presents a simple explanation of the MSCCC queue which is shown to be a quasi-reversible generalisation of the FCFS queue. Let $\mu(n) = 1$ and

$$s_i(c_n, \ldots, c_1) = \mu 1_i(c_n, \ldots, c_1)$$

where the indicator function

$$1_i(c_n, \ldots, c_1) = \begin{cases} 1 & \text{if the customer in position } i \text{ is in service in } (c_n, \ldots, c_1) \\ 0 & \text{otherwise} \end{cases}$$

Then

$$k(c_n, \ldots, c_1) = \mu(K \wedge \sum_{c \in \mathcal{C}} (M_c \wedge B_c))$$

where M_c is the number of type c customers in $\mathbf{C} = (c_n, \ldots, c_1)$ and $a \wedge b$ is the smaller of the two integers a and b. As required, $k(\mathbf{C})$ is independent of permutations of $\mathbf{C}$ and the functions $\mu(n)$ and $s_i(\cdot)$ conform to the conditions (i)–(iii). Furthermore, if not all K servers are busy, $k_c(\mathbf{C}) = M_c \wedge B_c$ is independent of permutations of $\mathbf{C}$.

Theorem 2.2 presents the stationary distribution of the MSCCC queue: the analysis is simpler than the theorems currently available [7, 11] and provides further insight into the behaviour of the MSCCC queue.

The constraints $(B_c)_{c \in \mathcal{C}}$ can be further divided into an hierarchical structure of concurrency constraints. Thus the set of customer types $\mathcal{C}$ is partitioned as $\{\mathcal{C}_r; r \in \mathcal{R}\}$ where $\mathcal{R}$ is a countable set and (with an abuse of notation) maximally $B_r > 0$ customers whose types are in $\mathcal{C}_r$ are allowed to be in service simultaneously where $r \in \mathcal{R}$. Next, for each $r \in \mathcal{R}$ the set $\mathcal{C}_r$ is partitioned as $\{\mathcal{C}_{rs}; s \in \mathcal{S}_r\}$ where $\mathcal{S}_r$ is a countable set and maximally $B_{rs} > 0$ customers whose types are in $\mathcal{C}_{rs}$ are allowed to be in service simultaneously where $s \in \mathcal{S}_r$. The $\mathcal{C}_{rs}$ can be further partitioned, with corresponding restrictions placed on the number of customers simultaneously in service, but we shall not go beyond the $\mathcal{C}_{rs}$ as this is a straightforward generalization. The queue discipline can thus be described as FCFS subject to concurrency constraints and the queue is correspondingly named the MultiServer centre with Hierarchical Concurrency Constraints (MSHCC) [21]. Then

$$k(c_n, \ldots, c_1) = \mu(K \wedge \sum_{r \in \mathcal{R}} (m_r \wedge B_r))$$

and

$$m_r = \sum_{s \in \mathcal{R}_r} (m_{rs} \wedge B_{rs})$$

where m_{rs} is the number of $\mathcal{C}_{rs}$ customers in $\mathbf{C} = (c_n, \ldots, c_1)$. The MSHCC queue conforms to the conditions (i)–(iii) which immediately provides us with the stationary distribution.

Another example of an OI queue is provided by example 3 from [18]. This queue consists of K servers with service rates $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_K$. The customers, which belong to a single customer type, are processed according to the following rule.

If j servers are busy then these j busy servers are the servers $1,\ldots,j$. If $j < K$ then an arriving customer will be served by server $j+1$, else the arrival joins the back of the queue. Let M denote the total number of customers in the queue. If a customer completes service at any one of the busy servers $i \in 1,\ldots,j$ where $1 \leq j \leq M \wedge K$ then the customer at the end of the queue in position M will be removed from processor j (only if $M < K$ in which case $M = j$) and go into service at server i (only if $i < K$). The queue is OI and the queue can be applied to model dynamic load balancing among asymmetric multiprocessors.

Summarising, the OI conditions (i)–(iii) do not leave much room for a wide range of applications. On the other hand, the fact that several well known quasi-reversible queues are OI queues speaks in favour of the OI discipline. Another valuable feature of OI queues which we prove in the next section is that they permit a standard normalising technique for calculating the stationary distributions of OI systems.

2.3 Numerical Techniques for the OI Queue

This section presents several techniques which are used in the numerical analysis of OI queues.

2.3.1 Aggregating the State Space

Equation (2.7) is too detailed to be of practical use when computing the performance measures of the OI queue. In order to reduce the complexity of these equations, we use the fact that $k(c_n,\ldots,c_1)$ is independent of the order of the customers in $(c_n,\ldots,c_1)$.

Define a mapping $A : \mathcal{S} \mapsto \mathbf{Z}^C$ such that for any $\mathbf{C} \in \mathcal{S}$

$$A(\mathbf{C}) = \mathbf{M}(\mathbf{C}) = (M_c(\mathbf{C}))_{c\in\mathcal{C}}$$

where M_c denotes the number of type c customers in $\mathbf{C}$. In the remainder of this section, where no confusion can arise we shall write $\mathbf{M} = \mathbf{M}(\mathbf{C})$. Let $A^{-1}(\mathbf{M})$ denote the set of elements in $\mathcal{S}$ which maps onto $\mathbf{M}$ under A. The mapping A allows us to define an aggregated state space

$$\mathcal{M} = \{\mathbf{M} = A(\mathbf{C}), \mathbf{C} \in \mathcal{S}\}$$

where (with an abuse of notation)

$$k(\mathbf{M}) = k(\mathbf{C})$$

for any $\mathbf{C} \in A^{-1}(\mathbf{M})$ and (with an abuse of notation)

$$\pi(\mathbf{M}) = \sum_{A(\mathbf{C})=\mathbf{M}} \pi(\mathbf{C})$$

for any $\mathbf{M} \in \mathcal{M}$. Because condition (ii) ensures that $k(\mathbf{C})$ is identical for all $\mathbf{C} \in A^{-1}(\mathbf{M})$, $k(\mathbf{M})$ is well defined and unique.

Let $\mathbf{1}_c$ denote a unit vector in the c-direction. From (2.7) it follows that

$$\mu(|\mathbf{M}|)k(\mathbf{M})\pi(\mathbf{M}) = \sum_{c\in\mathcal{C}} \lambda_c \pi(\mathbf{M}-\mathbf{1}_c) \tag{2.8}$$

with the convention that $\pi(\mathbf{M}) = 0$ if $\mathbf{M} \notin \mathcal{M}$.

The recursive (2.8) can be further simplified if the service effort $k_c(\mathbf{M})$ directed at customers of type c for any $\mathbf{M} \in \mathcal{M}$ can be determined. Although condition (ii) requires that $k(\mathbf{C})$ is independent of permutations of $\mathbf{C}$, this does not necessarily imply that $k_c(\mathbf{C})$ is independent of permutations of $\mathbf{C}$. However, in the domain where $k_c(\mathbf{C})$ is identical for all $\mathbf{C} \in A^{-1}(\mathbf{M})$, we can define

$$k_c(\mathbf{M}) = k_c(\mathbf{C})$$

for any $\mathbf{C} \in A^{-1}(\mathbf{M})$ in which case (2.8) reduces to

$$\mu(|\mathbf{M}|)k_c(\mathbf{M})\pi(\mathbf{M}) = \lambda_c \pi(\mathbf{M}-\mathbf{1}_c)$$

which provides a recursion for the efficient computation of the aggregated probabilities and the performance measures of the queue.

2.3.2 The Performance Measures: the MSCCC Queue

Unlike the BCMP queues, efficient recursions for the MSCCC queue apply only in a limited portion of the state space where $k(\mathbf{M}) < B$.

2.3.2.1 Invariant Measures over Special Sets

Let $\mathbf{M} = (M_1, \ldots, M_C)$ denote the state of the queue where M_c is the number of type c customers present in the queue. Let B_c denote the maximum number of type c customers in service and B the maximum number of customers in service in total. Define the sets

$$\begin{aligned}
\mathcal{M}(b) &= \{\, \mathbf{M} \in \mathcal{M} \mid k(\mathbf{M}) = b \,\} \\
\mathcal{M}(b,c) &= \{\, \mathbf{M} \in \mathcal{M}(b) \mid M_x = 0 \text{ if } x > c \,\} \\
\mathcal{M}(b,c,i) &= \begin{cases} \{\, \mathbf{M} \in \mathcal{M}(b,c) \mid M_c = i \,\} & \text{if } i < B_c \\ \{\, \mathbf{M} \in \mathcal{M}(b,c) \mid M_c \geq i \,\} & \text{if } i = B_c. \end{cases}
\end{aligned}$$

Define the functions

$$\begin{aligned} P(b) &= \Pr(\, b \text{ servers busy}\,) \\ P(b,c) &= \Pr(\, b \text{ servers busy and } M_x = 0 \text{ if } x > c\,) \\ P(b,c,i) &= \begin{cases} \Pr(\, b \text{ servers busy, } M_x = 0 \text{ if } x > c \text{ and } M_c = i\,) & \text{if } i < B_c \\ \Pr(\, b \text{ servers busy, } M_x = 0 \text{ if } x > c \text{ and } M_c \geq i\,) & \text{if } i = B_c. \end{cases} \end{aligned}$$

The first step in the calculation of $P(b)$ is to compute the $P(b,c,i)$. The function $k_c(\mathbf{M})$ defined in section 2.3.1 cannot be determined over the complete state space, but in the limited portion of the state space where $k(\mathbf{M}) < B$ where not all servers are busy, we know that $k_c(\mathbf{M}) = M_c \wedge B_c$ and $k(\mathbf{M} - \mathbf{1}_c) = k(\mathbf{M}) - 1$. Define $\rho_c = \lambda_c/\mu$. For any $c \in \mathcal{C}$ and $0 < b < B$ three cases arise

(1) $0 < i < b \wedge B_c$

$$\begin{aligned} P(b,c,i) &= \sum_{\mathbf{M} \in \mathcal{M}(b,c,i)} \pi(\mathbf{M}) \\ &= \sum_{\mathbf{M} \in \mathcal{M}(b,c,i)} \frac{\rho_c}{k_c(\mathbf{M})} \pi(\mathbf{M} - \mathbf{1}_c) \\ &= \frac{\rho_c}{i} \sum_{\mathbf{M} \in \mathcal{M}(b-1,c,i-1)} \pi(\mathbf{M}) \\ &= \frac{\rho_c}{i} P(b-1,c,i-1). \end{aligned} \tag{2.9}$$

(2) $i = B_c$ and $B_c \leq b \leq B$

$$\begin{aligned} P(b,c,B_c) &= \sum_{\mathbf{M} \in \mathcal{M}(b,c,B_c)} \pi(\mathbf{M}) \\ &= \sum_{\substack{\mathbf{M} \in \mathcal{M}(b,c) \\ M_c = B_c}} \pi(\mathbf{M}) + \sum_{\substack{\mathbf{M} \in \mathcal{M}(b,c) \\ M_c > B_c}} \pi(\mathbf{M}) \\ &= \sum_{\substack{\mathbf{M} \in \mathcal{M}(b,c) \\ M_c = B_c}} \frac{\rho_c}{B_c} \pi(\mathbf{M} - \mathbf{1}_c) + \sum_{\substack{\mathbf{M} \in \mathcal{M}(b,c) \\ M_c > B_c}} \frac{\rho_c}{B_c} \pi(\mathbf{M} - \mathbf{1}_c) \\ &= \frac{\rho_c}{B_c} \sum_{\mathbf{M} \in \mathcal{M}(b-1,c,B_c-1)} \pi(\mathbf{M}) + \frac{\rho_c}{B_c} \sum_{\mathbf{M} \in \mathcal{M}(b,c,B_c)} \pi(\mathbf{M}) \\ &= \frac{\rho_c}{B_c} P(b-1,c,B_c-1) + \frac{\rho_c}{B_c} P(b,c,B_c) \\ &= \frac{\rho_c}{B_c - \rho_c} P(b-1,c,B_c-1). \end{aligned} \tag{2.10}$$

(3) $i = 0$

$$\begin{aligned} P(b,c,0) &= \Pr\{\, b \text{ servers busy, } M_x = 0 \text{ if } x > c \text{ and } M_c = 0 \,\} \\ &= \Pr\{\, b \text{ servers busy, } M_x = 0 \text{ if } x > c-1 \,\} \\ &= P(b,c-1). \end{aligned} \tag{2.11}$$

The starting values of the recursion presented in (2.9), (2.10) and (2.11) are obtained from

$$P(0,c,0) = \begin{cases} 1 \text{ if } c = 0 \\ 0 \text{ otherwise} \end{cases} \tag{2.12}$$

for all $0 \le c \le C$. The next step is to compute the $P(b,c)$

$$P(b,c) = \sum_{i=0}^{b \wedge B_c} P(b,c,i) = P(b,c-1) + \sum_{i=1}^{b \wedge B_c} P(b,c,i) \tag{2.13}$$

where

$$\begin{aligned} P(b,0) &= \Pr\{\, b \text{ servers busy and } M_x = 0 \text{ if } x > 0 \,\} \\ &= \Pr\{\, b \text{ servers busy and } \mathbf{M} = \mathbf{0} \,\} \\ &= \begin{cases} 1 \text{ if } b = 0 \\ 0 \text{ otherwise.} \end{cases} \end{aligned}$$

The last step is to compute the $P(b)$

$$\begin{aligned} P(b) &= \Pr\{\, b \text{ servers busy} \,\} \\ &= \Pr\{\, b \text{ servers busy and } M_x = 0 \text{ if } x > C \,\} \\ &= P(b,C) \end{aligned}$$

which are multiples of $\pi(\mathbf{0})$. To calculate $\pi(\mathbf{0})$, note that when b customers are in service, the departure rate is $b\mu$. Thus $\rho = \sum_{b=0}^{B} bP(b) < B$ where $\rho = \sum_{c=1}^{C} \rho_c$ and $\sum_{b=0}^{B} P(b) = 1$ so that

$$B - \rho = \sum_{b=0}^{B-1} (B-b)P(b). \tag{2.14}$$

Let $\overline{P}(b)$ represent the values obtained by starting the algorithm with an arbitrary value for $P(0,0,0)$ in (2.12). Then

$$\pi(\mathbf{0}) = \frac{B-\rho}{\sum_{b=0}^{B-1}(B-b)\overline{P}(b)} \tag{2.15}$$

where $\overline{P}(0) = 1$. Thus for $0 \le b < B$

$$P(b) = \pi(\mathbf{0})\overline{P}(b) \tag{2.16}$$

and

$$P(B) = 1 - \sum_{b=0}^{B-1} P(b). \tag{2.17}$$

Equations (2.15) and (2.17) do not require the value $P(B)$ which is not available from the recursion presented in (2.9), (2.10) and (2.11).

Application of (2.9), (2.10) and (2.11) yields another recursion for $P(b,c)$ namely

$$\begin{aligned} P(b,c) &= \sum_{i=0}^{B_c} P(b,c,i) \\ &= \sum_{i=0}^{B_c-1} \frac{\rho_c^i}{i!} P(b-i,c-1) + \frac{\rho_c^{B_c}}{B_c!} \frac{B_c}{B_c-\rho_c} P(b-B_c,c-1) \end{aligned} \tag{2.18}$$

where $0 < c \leq C$ and $0 < b < B$ with the convention that $P(x,y) = 0$ if $x < 0$ or $y < 0$. The recursion (2.18) has the same computational complexity namely $O(CB^2)$ as (2.9), (2.10) and (2.11) but with O(B) as opposed to O(B^2) storage requirements. Note that (2.18) does not replace the recursions (2.9), (2.10) and (2.11) which are required in order to compute the expected number of type C customers in the system.

2.3.2.2 The Expected Queue Length

Let $L(b,C)$ denote the expected queue length of type C customers in $\mathcal{M}(b,C)$

$$\begin{aligned}
\frac{B_C}{\rho_C}L(b,C) &= \frac{B_C}{\rho_C}\sum_{\mathbf{M}\in\mathcal{M}(b,C)} M_C\pi(\mathbf{M}) \\
&= \frac{B_C}{\rho_C}\sum_{i=1}^{\infty}\sum_{\substack{\mathbf{M}\in\mathcal{M}(b,C)\\ M_C=i}} i\,\pi(\mathbf{M}) \\
&= B_C\sum_{i=1}^{B_C}\sum_{\substack{\mathbf{M}\in\mathcal{M}(b,C)\\ M_C=i}} \pi(\mathbf{M}-\mathbf{1}_C)+\sum_{i=B_C+1}^{\infty}\sum_{\substack{\mathbf{M}\in\mathcal{M}(b,C)\\ M_C=i}} i\,\pi(\mathbf{M}-\mathbf{1}_C) \\
&= B_C\sum_{i=1}^{B_C}\sum_{\substack{\mathbf{M}\in\mathcal{M}(b,C)\\ M_C=i}} \pi(\mathbf{M}-\mathbf{1}_C)+\sum_{i=B_C+1}^{\infty}\sum_{\substack{\mathbf{M}\in\mathcal{M}(b,C)\\ M_C=i}} \pi(\mathbf{M}-\mathbf{1}_C) \\
&\quad +\sum_{i=B_C+1}^{\infty}\sum_{\substack{\mathbf{M}\in\mathcal{M}(b,C)\\ M_C=i}} (i-1)\,\pi(\mathbf{M}-\mathbf{1}_C) \\
&= B_C\sum_{i=0}^{B_C-1}\sum_{\substack{\mathbf{M}\in\mathcal{M}(b-1,C)\\ M_C=i}} \pi(\mathbf{M})+\sum_{i=B_C}^{\infty}\sum_{\substack{\mathbf{M}\in\mathcal{M}(b,C)\\ M_C=i}} \pi(\mathbf{M}) \\
&\quad +\sum_{i=B_C}^{\infty}\sum_{\substack{\mathbf{M}\in\mathcal{M}(b,C)\\ M_C=i}} i\,\pi(\mathbf{M}) \\
&= B_C\sum_{i=0}^{B_C-1} P(b-1,C,i)+P(b,C,B_C) \\
&\quad +\sum_{i=0}^{\infty}\sum_{\substack{\mathbf{M}\in\mathcal{M}(b,C)\\ M_C=i}} i\,\pi(\mathbf{M})-\sum_{i=0}^{B_C-1}\sum_{\substack{\mathbf{M}\in\mathcal{M}(b,C)\\ M_C=i}} i\,\pi(\mathbf{M}) \\
&= B_C\sum_{i=0}^{B_C-1} P(b-1,C,i)+P(b,C,B_C)+L(b,C)-\sum_{i=0}^{B_C-1} iP(b,C,i)
\end{aligned}$$

which yields

$$\frac{(B_C-\rho_C)}{\rho_C}L(b,C) = P(b,C,B_C)+\sum_{i=0}^{B_C-1}\left(B_C P(b-1,C,i)-iP(b,C,i)\right) \quad (2.19)$$

The expected number $L(C)$ of type C customers in the queue is

$$L(C) = \sum_{b=0}^{B} L(b,C). \tag{2.20}$$

Equation (2.20) cannot be used directly to determine $L(C)$ because we do not have a value for $L(B,C)$. However, consider

$$\begin{aligned}
\sum_{b=0}^{B} bL(b,C) &= \sum_{b=0}^{B} b \sum_{\mathbf{M}\in\mathcal{M}(b)} M_C \pi(\mathbf{M}) \\
&= \sum_{b=0}^{B} \sum_{\mathbf{M}\in\mathcal{M}(b)} M_C k(\mathbf{M})\pi(\mathbf{M}) \\
&= \sum_{b=0}^{B} \sum_{\mathbf{M}\in\mathcal{M}(b)} M_C \sum_{c=1}^{C} \rho_c \pi(\mathbf{M}-\mathbf{1}_c) \\
&= \sum_{c=1}^{C} \rho_c \sum_{\mathbf{M}\in\mathcal{M}} M_C \pi(\mathbf{M}-\mathbf{1}_c) \\
&= \sum_{c=1}^{C-1} \rho_c \sum_{\mathbf{M}\in\mathcal{M}} M_C \pi(\mathbf{M}-\mathbf{1}_c) + \rho_C \sum_{\mathbf{M}\in\mathcal{M}} M_C \pi(\mathbf{M}-\mathbf{1}_C) \\
&= \sum_{c=1}^{C-1} \rho_c \sum_{\mathbf{M}\in\mathcal{M}} M_C \pi(\mathbf{M}-\mathbf{1}_c) \\
&\quad + \rho_C \sum_{\mathbf{M}\in\mathcal{M}} (M_C-1)\pi(\mathbf{M}-\mathbf{1}_C) + \rho_C \sum_{\mathbf{M}\in\mathcal{M}} \pi(\mathbf{M}-\mathbf{1}_C) \\
&= \sum_{c=1}^{C-1} \rho_c L(C) + \rho_C L(C) + \rho_C \\
&= \rho L(C) + \rho_C
\end{aligned}$$

so that

$$\rho L(C) = \sum_{b=0}^{B} bL(b,C) - \rho_C. \tag{2.21}$$

Combining (2.20) and (2.21) yields

$$(B-\rho)L(C) = \sum_{b=1}^{B-1} (B-b)L(b,C) + \rho_C. \tag{2.22}$$

Little's law is applied to obtain the expected time $W_C = L(C)/\lambda_C$ that a type C customer spends at the queue. Equations (2.9) through (2.22) can be incorporated into a Mean Value Analysis algorithm [7, 8, 11, 12] to compute the performance measures for a mixed multiclass network of BCMP and MSCCC queues.

Equation (2.22) yields the type C queue length. The performance measures for any type $c \in \mathcal{C}$ are obtained by reordering the set $\mathcal{C}$. An algorithm to compute the type C performance measures of the MSCCC queue is presented in the Appendix.

2.4 The OI Loss Queue

The class of OI queues can be extended to include queues with complex loss mechanisms. Such queues can be used to model blocking systems with simultaneous resource possession – see [26] for list of references on the application of such models.

Consider a queue serving customers which belong to C customer types. Let $\mathcal{C} = \{1, 2, \ldots, C\}$ denote the set of customer types. Customers of type c arrive individually to a queue of length n at the instants of a Poisson stream with rate $\lambda_c r(n)$, where the multiplier $r(n) > 0$ does not depend on c. Each customer of type c presents a demand for service which is exponentially distributed with mean $1/\mu$.

The customers, whether waiting or in service, form a queue in the order of their arrival. A Markov chain with vector states $\mathbf{C} = (c_n, \ldots, c_1)$ is used to describe the queue, where $c_i \in \mathcal{C}$, $i = 1, \ldots, n$ identifies the type of the customer in queue position i and n is the number of customers in the queue. The 0 state of the Markov chain describes the empty queue. Let $\mathcal{C}^n$ denote the $n-$fold product space of $\mathcal{C}$. Then $\mathcal{S} = 0 \cup \bigcup_{n=1}^{\infty} \mathcal{C}^n$ is the set on which the Markov chain is defined.

An arriving customer is either accepted to the queue, in which case it joins the back of the queue, or is rejected (lost) if certain limits on the numbers of customers of each type in the queue are exceeded. The rejection rule for arriving customers is defined as follows. Define a set $\widetilde{\mathcal{S}} \subset \mathcal{S}$ which satisfies the following properties

(a) if $(c_n, \ldots, c_1) \in \widetilde{\mathcal{S}}$ then $(c_{\sigma(n)}, \ldots, c_{\sigma(1)}) \in \widetilde{\mathcal{S}}$ for any permutation $\sigma(1), \ldots, \sigma(n)$ of $1, \ldots, n$ and
(b) if $(c_n, \ldots, c_1) \in \widetilde{\mathcal{S}}$ then $(c_{n-1}, \ldots, c_1) \in \widetilde{\mathcal{S}}$.

A type c customer arriving to a queue in the state $\mathbf{C}$ will be accepted if $(c, \mathbf{C}) \in \widetilde{\mathcal{S}}$ and rejected otherwise.

Condition (a) implies that acceptance does not depend on the order of the customers in the queue but only on the number of customers of each type present in the queue. To understand condition (b) let $A(\mathbf{C}) = \mathbf{n} = (n_c)_{c \in \mathcal{C}}$ where n_c denotes the number of type c customers in $\mathbf{C}$ and let $\widetilde{\mathcal{N}} = A(\widetilde{\mathcal{S}})$. Conditions (a) and (b) imply that the set $\widetilde{\mathcal{N}}$ is coordinately convex [26] so that $\mathbf{n} \in \widetilde{\mathcal{N}}$ implies $\mathbf{n} - \mathbf{1}_c \in \widetilde{\mathcal{N}}$ where $\mathbf{1}_c$ is a unit vector in the c^{th} direction.

The queue described above is said to be an OI loss queue on the set $\widetilde{\mathcal{S}}$ if for all $\mathbf{C} \in \widetilde{\mathcal{S}}$ and all $i = 1, \ldots, n$ the departure rate of the customer in queue position i can be written as $\mu(n) s_i(c_n, \ldots, c_1)$ where

(i) $s_i(c_n, \ldots, c_1) = s_i(c_i, \ldots, c_1)$ for any $1 \le i \le n$,
(ii) $k(c_n, \ldots, c_1) = \sum_{i=1}^{n} s_i(c_n, \ldots, c_1)$ is independent of permutations of $(c_n, \ldots, c_1)$ and
(iii) $\mu(n) > 0$ for $n > 0$ and $s_1(c) > 0$ for any $c \in \mathcal{C}$.

2.4.1 The Stationary Distribution

The OI loss queue can be modeled by a continuous-time, homogeneous Markov chain $X(t), t \in \mathbf{R}^+, X(t) \in \widetilde{\mathcal{S}}$, where $X(t) = \mathbf{C}$ denotes that the queue vector at time t is $\mathbf{C}$. The Markov chain is irreducible, since transitions due to arrivals allow any state to be reached from the empty state and condition (ii) ensures that the empty state can be reached from any state by the departure transitions.

For $\mathbf{C} \in \widetilde{\mathcal{S}}$ and $c \in \mathcal{C}$ we associate an indicator function $I_c(\mathbf{C})$ such that $I_c(\mathbf{C}) = 1$ if $(c, \mathbf{C}) \in \widetilde{\mathcal{S}}$, and $I_c(\mathbf{C}) = 0$ if $(c, \mathbf{C}) \notin \widetilde{\mathcal{S}}$. Then for any $c \in \mathcal{C}$ and any $\mathbf{C} = (c_n, \ldots, c_1) \in \widetilde{\mathcal{S}}$, the transition rate due to type c arrivals is $\lambda_c r(n) I_c(\mathbf{C})$ and the transition rate due to the departure of a type c_i customer in queue position i when there are n customers in queue is $\mu(n) s_i(c_i, \ldots, c_1)$.

The functions $r(n)$ and $I_c(\mathbf{C})$ parametrise the rejection rule. Thus $r(n) = 0$ applies blocking when the queue length reaches some threshold n and $I_c(\mathbf{C}) = 0$ implements a more refined form of blocking which depends on how many customers of the various types are present.

The equilibrium equations for the Markov chain are given by

$$\pi(0) \sum_{c \in \mathcal{C}} \lambda_c r(0) I_c(0) = \mu(1) k(c) \sum_{(c) \in \widetilde{\mathcal{S}}} \pi(c) \tag{2.23}$$

and

$$\begin{aligned} &\pi(\mathbf{C}) \left(\sum_{c \in \mathcal{C}} \lambda_c r(n) I_c(\mathbf{C}) + \mu(n) k(\mathbf{C}) \right) \\ &= \sum_{c \in \mathcal{C}} \sum_{i=0}^{n} \mu(n+1) \pi(c_n, \ldots, c_{i+1}, c, c_i, \ldots, c_1) s_{i+1}(c, c_i, \ldots, c_1) \\ &\quad + \lambda_{c_n} r(n-1) \pi(c_{n-1}, \ldots, c_1) \end{aligned} \tag{2.24}$$

for any state $\mathbf{C} = (c_n, \ldots, c_1) \in \widetilde{\mathcal{S}}$ with the convention that $\pi(\mathbf{C}) = 0$ if $\mathbf{C}$ is not in $\widetilde{\mathcal{S}}$.

In section 2.2 we showed that if $r(n) = 1$ for all n and $I_c(\mathbf{C}) = 1$ for all $c \in \mathcal{C}$ and $\mathbf{C} \in \widetilde{\mathcal{S}}$ (thus $\widetilde{\mathcal{S}} = \mathcal{S}$) then the solution to (2.23) and (2.24) can be found as a solution to the partial balance equations

$$\sum_{i=0}^{n} \frac{\pi(c_n, \ldots, c_{i+1} c c_i, \ldots, c_1)}{\pi(\mathbf{C})} \mu(n+1) s_{i+1}(c c_i, \ldots, c_1) = \lambda_c r(n) I_c(\mathbf{C}) \tag{2.25}$$

and

$$\mu(n) k(\mathbf{C}) \pi(\mathbf{C}) = \lambda_{c_n} r(n-1) \pi(c_{n-1}, \ldots, c_1) \tag{2.26}$$

for all $\mathbf{C} \in \widetilde{\mathcal{S}}$. This suggests the form of the stationary distribution for the OI loss queue which is stated in the following theorem.

Theorem 2.3. *If the service rate functions $s_i(\cdot)$ conform to the conditions (i) and (ii) on the set $\widetilde{\mathcal{S}}$ and $\widetilde{\mathcal{S}}$ conforms to conditions (a) and (b) then the stationary distribution of the Markov chain on $\widetilde{\mathcal{S}}$ associated with an OI loss queue is given by $\pi(0) = 1/G$ and*

$$\pi(c_n,\ldots,c_1) = \frac{1}{G}\prod_{i=1}^{n}\frac{\lambda_{c_i} r(i)}{\mu(i)k(c_i,\ldots,c_1)} \tag{2.27}$$

if and only if

$$G = 1 + \sum_{\substack{(c_n,\ldots,c_1)\in\widetilde{\mathcal{S}}\\ n>0}} \prod_{i=1}^{n}\frac{\lambda_{c_i} r(i)}{\mu(i)k(c_i,\ldots,c_1)} < \infty.$$

Further, the OI loss queue satisfies partial balance.

Proof. Equation (2.27) is clearly a solution to (2.26) on $\widetilde{\mathcal{S}}$. We need to show that (2.27) is also a solution to the partial balance equation (2.25) on $\widetilde{\mathcal{S}}$ and is thus a solution to the equilibrium equations. If the state $(c,\mathbf{C})$ does not belong to $\widetilde{\mathcal{S}}$ then from condition (b) $I_c(\mathbf{C}) = 0$ and from condition (a) any permutation of the state $(c,\mathbf{C})$ does not belong to $\widetilde{\mathcal{S}}$ implying that both sides of (2.25) are equal to zero and the equation is satisfied. If the state $(c,\mathbf{C})$ belongs to the set $\widetilde{\mathcal{S}}$ then $I_c(\mathbf{C}) = 1$ and (2.5) coincides with the form of (2.25) for an OI queue without losses [4] except for the arrival rate scaling factor $r(n)$ which does not affect the calculations and thus (2.27) is a solution to the equilibrium equations. □

Although most of the functions used in the definition of the OI loss queue are independent of the order of the customers in the queue, the queue order cannot be ignored. Indeed, let $\mathcal{N}$ be a set of non-negative integers and let $\mathcal{N}^C$ denote the $C-$fold product space of $\mathcal{N}$. Define a function A from $\widetilde{\mathcal{S}}$ into $\mathcal{N}^C$ such that $A(\mathbf{C}) = \mathbf{N} = (n_1,\ldots,n_C)$ whose c^{th} element n_c denotes the number of type c elements in the vector $\mathbf{C}$. Let $\widetilde{\mathcal{N}}$ denote the set of all vectors $\mathbf{N}\in\mathcal{N}^C$ for which there exists a vector $\mathbf{C}\in\widetilde{\mathcal{S}}$ such that $A(\mathbf{C}) = \mathbf{N}$. Consider the process $Y(t)$ on $\widetilde{\mathcal{N}}$ associated with the OI loss queue which records only the numbers of the customers of each type in the queue at time t. In general $Y(t)$ is not a Markov chain because the departure rates of the individual customer types $k_c(\mathbf{C}) = \sum_{i=1}^{n} s_i(c_i,\ldots,c_1)\,1(c_i = c)$ are not necessarily order independent.

If the $k_c(\mathbf{C})$ are order independent then $Y(t)$ is a Markov chain on the aggregated space $\widetilde{\mathcal{N}}$ and the following theorem holds.

Theorem 2.4. *Let $X(t)$ be a stationary Markov chain associated with an OI loss queue. If the service rate functions $k_c(\mathbf{C}), c\in\mathcal{C}$, are order independent on $\widetilde{\mathcal{S}}$ then the process $Y(t)$ is a reversible Markov chain on $\widetilde{\mathcal{N}}$ with a stationary distribution $\pi(\mathbf{N})$ which can be recursively calculated from the detailed balance equations*

$$\mu(n)k_c(\mathbf{N})\pi(\mathbf{N}) = \lambda_c r(n-1)\pi(\mathbf{N}-\mathbf{1}_c) \tag{2.28}$$

where $n_c > 0$, $n = n_1+\cdots+n_C$ and $k_c(\mathbf{N}) = k_c(\mathbf{C})$ for $\mathbf{N} = A(\mathbf{C})$.

Proof. $Y(t)$ is a Markov chain on the aggregated space since all the transitions due to customer arrivals and departures are well defined. Thus the stationary distribution $\pi(\mathbf{N})$ on $\widetilde{\mathcal{N}}$ is

$$\pi(\mathbf{N}) = \sum_{\mathbf{C}:\mathcal{A}(\mathbf{C})=\mathbf{N}} \pi(\mathbf{C}). \tag{2.29}$$

Equation (2.28) is obtained by summing (2.25) over all states $\mathbf{C}$ such that $A(\mathbf{C}) = \mathbf{N}$, taking the order independence of $k_c(\mathbf{C})$ into account. Equation (2.28) is the detailed balance equation for the Markov chain $Y(t)$ and is satisfied by the probabilities given by (2.29). Thus the Markov chain $Y(t)$ is reversible and its distribution can be recursively calculated from the detailed balance (2.28). □

Theorem 2.4 demonstrates that if the departure rates $k_c(\mathbf{C})$ of the individual customer types are order independent over the entire state space then the OI construction is not necessary to derive the equilibrium distribution and the queue can be examined via standard reversibility arguments as is shown for example in the model presented in section 2.5.7.

2.4.2 The Performance Measures: the MSCCC Loss Queue

Let $\mathbf{C} = (c_n, \ldots, c_1)$ denote the state of the MSCCC loss queue where $n \leq N$. The state space of the queue is $\mathcal{L} = \mathcal{S}_N = \{0\} \cup \bigcup_{n=1}^{N} \mathcal{C}^n$. Let $M_c(\mathbf{C})$ denote the number of type c customers in $\mathbf{C}$. Recall the mapping $\mathcal{A} : \mathcal{S}_N \mapsto \mathbf{Z}^C$ such that for any $\mathbf{C} \in \mathcal{S}_N$

$$\mathcal{A}(\mathbf{C}) = \mathbf{M}(\mathbf{C}) = (M_c(\mathbf{C}))_{c\in\mathcal{C}}$$

where $\mathbf{M}(\mathbf{C})$ is the counting vector of the state $\mathbf{C}$. Let $\mathcal{M}(N)$ denote the set of all counting vectors. As usual $|\mathbf{M}| = M_1 + \cdots + M_C$.

The stationary distribution $\pi_N(\mathbf{C})$ where $\mathbf{C} \in \mathcal{S}_N$ for the MSCCC loss queue will, on all states $\mathbf{C} \in \mathcal{S}_N$, coincide (up to a normalising constant) with the stationary distribution for the corresponding MSCCC queue with no losses. Define

$$\pi_N(\mathbf{M}) = \sum_{\substack{\mathbf{C}\in\mathcal{S}_N \\ \mathcal{A}(\mathbf{C})=\mathbf{M}}} \pi_N(\mathbf{C}).$$

2.4.2.1 Invariant Measures over Special Sets

Recall that $k(\mathbf{M})$ denotes the number of customers in service when the MSCCC is in a state with a counting vector $\mathbf{M}$. For any $c \in \mathcal{C}$ and non-negative integers n, b, i define the sets

$$\begin{aligned}
\mathcal{L}(n) &= \{\mathbf{M} \in \mathcal{L} \mid |\mathbf{M}| = n\} \\
\mathcal{L}(n,b) &= \{\mathbf{M} \in \mathcal{L}(n) \mid k(\mathbf{M}) = b\} \\
\mathcal{L}(n,b,c) &= \{\mathbf{M} \in \mathcal{L}(n,b) \mid M_x = 0 \text{ if } x > c\} \\
\mathcal{L}(n,b,c,i) &= \{\mathbf{M} \in \mathcal{L}(n,b,c) \mid M_c = i\}.
\end{aligned}$$

In the remainder of this section, where no confusion can arise we omit the subscript N which denotes the population constraint on the MSCCC loss queue.

Let $Q(n)$ denote the probability that n customers are present at the MSCCC.

Let $Q(n,b)$ denote the probability that n customers are present and b servers are busy.

Let $Q(n,b,c)$ denote the probability that n customers are present, b servers are busy and no customers of types higher than c are present.

Let $Q(n,b,c,i)$ denote the probability that n customers are present, b servers are busy, i customers of type c are present and no customers of types higher than c are present.

The first step in the calculation of $Q(n)$ is to compute the $Q(n,b,c,i)$. In the domain $k(\mathbf{M}) < B$ where not all servers are busy, the number of type c customers in service is given by $k_c(\mathbf{M}) = M_c \wedge B_c$. For any $c \in \mathcal{C}$ and $0 < b < B$ two cases arise.

(1) $0 < i \leq n \leq N$

$$\begin{aligned}
Q(n,b,c,i) &= \sum_{\mathbf{M} \in \mathcal{L}(n,b,c,i)} Q(\mathbf{M}) \\
&= \sum_{\mathbf{M} \in \mathcal{L}(n,b,c,i)} \frac{\rho_c}{i \wedge B_c} Q(\mathbf{M} - \mathbf{1}_c) \\
&= \begin{cases} \dfrac{\rho_c}{i} Q(n-1,b-1,c,i-1) & 0 < i \leq B_c \\ \dfrac{\rho_c}{B_c} Q(n-1,b,c,i-1) & i > B_c \end{cases}
\end{aligned} \tag{2.30}$$

with the convention that $Q(n,b,c,i) = 0$ if the condition $0 < (i \wedge B_c) \leq b < B$ is violated.

(2) $i = 0$ and $0 \leq n \leq N$

$$\begin{aligned}
Q(n,b,c,0) &= \Pr(n \text{ customers present, } b \text{ servers busy, } M_x = 0 \text{ if } x \geq c) \\
&= \Pr(n \text{ customers present, } b \text{ servers busy, } M_x = 0 \text{ if } x > c-1) \\
&= Q(n,b,c-1).
\end{aligned} \tag{2.31}$$

The starting values for the recursion presented in (2.30) and (2.31) are given by

$$Q(n,0,c,0) = \begin{cases} 1 & n = c = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.32}$$

for all $0 \leq n \leq N$ and $0 \leq c \leq C$. The next step is to compute the $Q(n,b,c)$

$$Q(n,b,c) = \sum_{i=0}^{n} Q(n,b,c,i) = Q(n,b,c-1) + \sum_{i=1}^{n} Q(n,b,c,i)$$

where

$$Q(n,0,c) = \begin{cases} 1 & n = c = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.33}$$

The last step is to compute the $Q(n,b)$

$$\begin{aligned} Q(n,b) &= \Pr(n \text{ customers present and } b \text{ servers busy}) \\ &= \Pr(n \text{ customers present, } b \text{ servers busy and } M_x = 0 \text{ if } x > C) \\ &= Q(n,b,C) \end{aligned}$$

which yields

$$Q(N) = \sum_{b=0}^{B} Q(N,b). \tag{2.34}$$

Equation (2.34) cannot be used to compute $Q(N)$ since $Q(N,B)$ is not known. However, since the arrival rate of accepted traffic to the queue (the accepted traffic is the offered traffic minus the lost traffic) equals the departure rate from the queue

$$\rho(1 - Q(N)) = \sum_{b=0}^{B} bQ(b) = \sum_{b=0}^{B} b \sum_{n=b}^{N} Q(n,b) = \sum_{b=0}^{B} bR(N,b)$$

so that

$$BR(N,B) = \rho(1 - Q(N)) - \sum_{b=0}^{B-1} bR(N,b).$$

Adding $B\sum_{b=0}^{B-1} R(N,b)$ to both sides of the above equation yields

$$B = \rho(1 - Q(N)) + \sum_{b=0}^{B-1} (B-b)R(N,b) \tag{2.35}$$

so that

$$\rho Q(N) = \rho - B + \sum_{b=0}^{B-1} (B-b)R(N,b)$$

where $R(N,0) = \pi(\mathbf{0})$ yields an expression for $Q(N)$ in terms of the previously computed values of $Q(N,b)$ where $0 \le b < B$.

The $Q(N,b)$ are multiples of $\pi(\mathbf{0})$. To calculate $\pi(\mathbf{0})$, note that if $\overline{Q}(N)$ and $\overline{Q}(N,b)$ represent the values obtained by starting the algorithm with an arbitrary value of $Q(0,0,0,0)$ in (2.32), then

$$\rho(1 - \pi(\mathbf{0})\overline{Q}(N)) = \pi(\mathbf{0}) \sum_{b=0}^{B} b\overline{R}(N,b).$$

Subtracting $B\pi(\mathbf{0})\sum_{b=0}^{B}\overline{R}(N,b) = B$ from both sides of the above equation yields

$$B - \rho(1 - \pi(\mathbf{0})\overline{Q}(N)) = \pi(\mathbf{0})\sum_{b=0}^{B-1}(B-b)\overline{R}(N,b)$$

so that

$$\pi(\mathbf{0}) = \frac{B+\rho}{\sum_{b=0}^{B-1}(B-b)\overline{R}(N,b) - \overline{Q}(N)} \tag{2.36}$$

where $\overline{R}(N,0) = 1$ yields an expression for $\pi(\mathbf{0})$ in terms of the previously computed values of $Q(N,b)$ where $1 \leq b < B$. Then

$$Q(N,b) = \begin{cases} \pi(\mathbf{0})\overline{Q}(N,b) & 0 \leq b < B \\ \\ \pi(\mathbf{0})Q(N) - \sum_{b=0}^{B-1} Q(N,b) & b = B. \end{cases}$$

Note that $\lim_{N\to\infty} Q(N) = 0$ and $\lim_{N\to\infty} R(N,b) = P(b)$ so that in the limit where no losses occur, (2.35) and (2.36) reduce to (2.14) and (2.15).

Another recursion with the same computational complexity as the recursion presented in (2.30) and (2.31) but with $O(B^2)$ as opposed to $O(B^3)$ storage requirements is

$$Q(n,b,c) = \sum_{i=0}^{B_c-1}\frac{\rho_c^i}{i!}Q(n-i,b-i,c-1) + \frac{\rho_c^{B_c}}{B_c!}Q(n-B_c,b-B_c,c-1)F(n,c)$$

where $0 < c \leq C$ and $0 < b < B$ and

$$F(n,c) = \begin{cases} \dfrac{1-(\rho_c/B_c)^{n-B_c+1}}{1-\rho_c/B_c} & \rho_c \neq B_c \\ \\ n - B_c + 1 & \rho_c = B_c. \end{cases}$$

2.4.2.2 The Expected Queue Length

Let $L(n,b,C)$ denote the average number of type C customers at the MSCCC when $b < B$ servers are busy and n customers of all types are present. For $0 < b < B$

$$L(n,b,C) = \sum_{i=1}^{n} iQ(n,b,C,i)$$

where the $Q(n,b,c,i)$ are given in (2.30). Given the relatively small number n of terms in the above summation, it is not necessary as was in the case of the MSCCC without losses – see section 2.3.2.2 – to compute an analytic expression for the sum.

Let $L(n,C)$ denote the average number of type C customers at the MSCCC when there are n customers at the MSCCC

$$L(n,C) = \sum_{b=1}^{B} L(n,b,C).$$

The values of the $L(n,B,C)$ are not known. Consider therefore the following identity

$$BL(n,C) = \sum_{b=1}^{B} (B+b-b)L(n,b,C) = \sum_{b=1}^{B} bL(n,b,C) + \sum_{b=1}^{B-1} (B-b)L(n,b,C).$$

Since $k(\mathbf{M}) = b$ we have

$$\begin{aligned}
\sum_{b=1}^{B} bL(n,b,C) &= \sum_{b=1}^{B} b \sum_{\mathbf{M}\in\mathcal{L}(n,b)} M_C Q(\mathbf{M}) \\
&= \sum_{b=1}^{B} b \sum_{\mathbf{M}\in\mathcal{L}(n,b)} M_C \sum_{x=1}^{C} \frac{\rho_x}{k(\mathbf{M})} Q(\mathbf{M}-\mathbf{1}_x) \\
&= \sum_{x=1}^{C-1} \rho_x \sum_{b=1}^{B} \sum_{\mathbf{M}\in\mathcal{L}(n,b)} M_C Q(\mathbf{M}-\mathbf{1}_x) + \rho_C \sum_{b=1}^{B} \sum_{\mathbf{M}\in\mathcal{L}(n,b)} M_C Q(\mathbf{M}-\mathbf{1}_C) \\
&= \sum_{x=1}^{C-1} \rho_x \sum_{\mathbf{M}\in\mathcal{L}(n-1)} M_C Q(\mathbf{M}) + \rho_C \sum_{\mathbf{M}\in\mathcal{L}(n-1)} (M_C+1) Q(\mathbf{M}) \\
&= \rho L(n-1,C) + \rho_C Q(n-1)
\end{aligned}$$

where $\rho = \sum_{c=1}^{C} \rho_c$ so that

$$BL(n,C) = \rho L(n-1,C) + \rho_C Q(n-1)) + \sum_{b=1}^{B-1} (B-b)L(n,b,C).$$

The last term $L(N,C)$ of this recursion is the expected number of type C customers in the MSCCC loss queue. Summing the values of $L(N,c)$ over all $c \in \mathcal{C}$ yields the expected queue length – which result can be obtained at earlier stage from $\sum_{n=1}^{N} nQ(n)$.

2.4.3 OI Loss Networks

Although we have presented the OI system as a single queue, OI systems can also be used to model networks of queues.

Let $\mathbf{C}(t)$ be a continuous time, homogeneous Markov process denoting the state of an OI queue at time t on the state space $\widetilde{\mathcal{S}}$ with the stationary distribution $\pi(\mathbf{C})$. Let $\widetilde{\mathcal{S}}'$ be a subset of $\widetilde{\mathcal{S}}$ satisfying conditions (a) and (b). Then by definition $\mathbf{C}(t)$ is an OI loss queue on the set $\widetilde{\mathcal{S}}'$ with a stationary distribution given by $\pi(\mathbf{C})/G$ where G is the corresponding normalising constant. This truncation property of the OI queue can be extended to a network of independent OI queues as follows.

Let $\mathbf{C}^k(t) = (c^k_{n_k}(t), \ldots, c^k_1(t))$ where $k = 1, \ldots, K$ denote a collection of independent OI queues on the corresponding states $\widetilde{\mathcal{S}}^k$ with the stationary distribution $\pi^k(\mathbf{C}^k)$. Consider the queue $\mathbf{C}(t) = (\mathbf{C}^1(t), \ldots \nu C^K(t))$ on the state $\widetilde{\mathcal{S}} = \widetilde{\mathcal{S}}^1 \times \widetilde{\mathcal{S}}^2 \times \cdots \times \widetilde{\mathcal{S}}^K$ with stationary distribution $\pi(\mathbf{C}) = \pi^1(\mathbf{C}^1) \ldots \pi^K(\mathbf{C}^K)$. Let $\widetilde{\mathcal{S}}'$ be a subset of $\widetilde{\mathcal{S}}$ such that

(i) $\mathbf{C} \in \widetilde{\mathcal{S}}'$ implies $(\mathbf{C}^1_\sigma, \ldots, \mathbf{C}^K_\sigma) \in \widetilde{\mathcal{S}}'$ for any permutations $\mathbf{C}^k_\sigma$ of $\mathbf{C}^k$.
(ii) $A(\widetilde{\mathcal{S}}') = \widetilde{\mathcal{N}}'$ is coordinately convex

then [5] the queue $\mathbf{C}(t)$ obeys the truncation property so that the stationary distribution of $\mathbf{C}(t)$ on the set $\widetilde{\mathcal{S}}'$ is given by $\pi(\mathbf{C})/G$ where G is the corresponding normalising constant.

Several applications of OI queues and OI networks are presented in the following section.

2.5 OI Applications

2.5.1 Multiported Memory

Consider [7, 8, 15] a computer system consisting of N processors accessing K memory modules via a partitioned multiple bus system. Each of the G groups of B buses gives access to a subset of K/G memory modules. Each memory module k is B_k–ported so that maximally B_k processors can access memory module k simultaneously. The system is modelled as a closed queueing network consisting of an IS centre representing the processors and G MSCCC centres, each representing one group of B buses and the associated memory modules. Each MSCCC centre consists of B servers which represent the B buses in its group. The N customers in the network belong to K classes.

A processor service interval followed by a data transfer to/from memory module k is modelled as a customer departing from the processor service centre, and moving to the g^{th} MSCCC centre where group g contains an access path to memory module k. The customer changes class to class k and queues for service at the MSCCC centre. The class k customer enters into service if one of the B servers is free (a bus is available) and if at most B_k class k customers are in service (at most B_k processors can access memory module k simultaneously).

2.5.2 A Messaging Card

In some distributed architectures such as telephone switching exchanges, the messaging function between a high level peripheral (decentralised call processor) and a lower level one (line or trunk controller) is performed on the high level side by

a specialised processor (the messaging card) which controls simplex channels to the lower level peripherals. The processor time is partitioned into B fixed time slots, each of which is allocated to a process whose function is to send outgoing messages. The exchange forwards messages to K destinations, each of which is reachable on any of B_k channels. When a buffer is queued for transmission it has to wait for a process to be available to service the request and for an outgoing channel to be free. The messaging card can be modelled by a MSCCC centre consisting of B servers (the transmission processes) serving customers belonging to K classes, each with its own concurrency limit B_k.

2.5.3 Multilayer Window Flow Control

Consider a set of K application which share a common data link. At the data link layer maximally B transmitted packets may remain unacknowledged. For each application k maximally B_k packets may remain unacknowledged. The data link can be modelled [12] as a MSCCC centre consisting of B servers with an average service time equal to the packet transfer time averaged over all the packets. The customer arrivals represent packet transmission requests. A class k packet is transmitted if a server is available (data link flow control) and if at most $B_k - 1$ class k packets are in service (application flow control).

2.5.4 Machine Scheduling Model

Jobs of type $k \in \{1, \ldots, C\}$ arrive according to a Poisson process with intensity λ_k and are processed by machines of type k in a FCFS order. There are B_k machines of type k which are operated by a common pool of B machine operators. If the job processing times are exponentially distributed and are job type independent then this model is solved by the MSCCC centre. If the job processing times are exponentially distributed but are job type dependent, then in the two special cases $B = 1$ and $B = B_1 + \cdots + B_C$ the stationary probabilities are a sum of product forms [1].

2.5.5 Blocked Calls Cleared

In the remainder of this section we apply OI queues to model circuit-switched networks where some blocked calls are queued for connection when the requested circuits become available. This call queueing mechanism differs from the admission control generally used in circuit switched networks where blocked calls are lost – see [19, 20] for an extensive literature on the subject of circuit-switched loss networks.

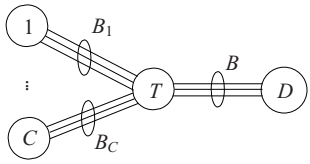

Fig. 2.1: A circuit-switched network

Consider figure 2.1 which represents (part of) a circuit-switched network. There are C source switches. Each switch c is connected by an access link consisting of B_c circuits to a tandem switch T which switches the incoming calls among B outgoing circuits to a destination switch D where $B < B_1 + \cdots + B_C$. Calls arrive at switch c according to a Poisson process with rate λ_c. Call holding times are exponentially distributed with unit mean.

A call from switch c is connected if a circuit from c to T and a circuit from T to D is available, else the call is lost. This model is OI although standard methods can also be used [23] to obtain an analytic solution. Such loss models were considered in a more general framework [2, 19] and there are many articles on this topic for example [9].

2.5.6 Blocked Calls Queued

Rather than clearing blocked calls, the blocked calls can be held for a short period while waiting for the required circuits to become free. This will significantly decrease the loss probability and increase the circuit utilisation at the expense of introducing a small connection delay.

Call queueing is implemented by storing the signalling information for each call in a buffer at the transit switch T until the call is completed. When a call completes, the queue is scanned from the front looking for the first call that can be connected. If the call (say it originated at switch c) cannot be connected because all B_c circuits in link c are busy, the next call in the queue is considered. Connection requests are thus attempted on a FCFS basis.

The MSCCC queue provides an exact analytic solution to this model. Calls which arrive and find N calls in the system (queued or in service) are lost. A call from switch c will be connected to the destination switch D if a circuit in link c is available (less than B_c customers of type c are in service) and a circuit from T to D is available (less than B customers of all types are in service). The limit N models call holding: up to $N - \min(B_1, \ldots, B_C)$ blocked calls can be queued. When $N = B$ we obtain Mitra's model [23].

A feature of this model is that if the network is overloaded by class c calls then the queue will be filled with class c calls and arrivals of other classes will be rejected. This deficiency is addressed in the next section.

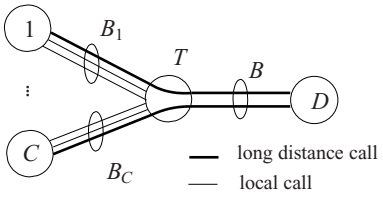

Fig. 2.2: Local and long distance calls

2.5.7 Blocked Calls Queued with Source Rejection

Calls which arrive and find N calls in the system (queued or in service) are lost. A call from switch c which finds all B_c access circuits busy is lost – this is termed source rejection. Calls which seize an access circuit and find all B outgoing circuits busy are queued in a finite buffer. When an outgoing circuit becomes free, the queued calls are searched in FCFS order for the next call to be connected. Thus maximally N customers of all types and maximally B_c customers of type $c \in \mathcal{C}$ are admitted to the system and up to $N - \min(B_1, \ldots, B_C)$ blocked calls can be queued.

For all $\mathbf{n} \in \widetilde{\mathcal{N}}$ the un-normalised aggregated stationary probabilities for above model are given by

$$\pi(\mathbf{n}) = g(n) \prod_{c=1}^{C} \frac{\rho_c^{n_c}}{n_c!}$$

where

$$g(n) = \begin{cases} 1 & 0 \le n \le B \\ \dfrac{n!}{B! B^{n-B}} & B < n \le N \end{cases}$$

so that $Bg(n) = ng(n-1)$ for $B < n \le N$. This model is an OI queue although standard methods can also be used [6] to obtain an analytic solution.

A further OI specialisation of this model is obtained by partitioning the set of customer types into $\{1, \ldots, J\}$ and $\{J+1, \ldots, C\}$. Source rejection is applied to the types $\{J+1, \ldots, C\}$ but is not applied to the types $\{1, \ldots, J\}$ so that all customers of types $\{J+1, \ldots, C\}$ are in service whereas customers of types $\{1, \ldots, J\}$ may be queued.

2.5.8 Local and Long Distance Calls

Consider the network presented in figure 2.2. A local call from switch c requires one circuit from switch c to the tandem switch T. A long distance call from switch c requires one circuit from switch c to switch T and one circuit from T to the destination switch D. Blocked local calls are lost. Blocked long distance calls are queued in a finite buffer of size $N - \min(B_1, \ldots, C)$.

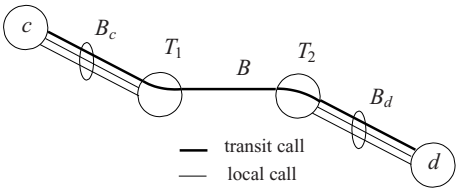

Fig. 2.3: Local and transit calls

Let $\mathcal{C}_+ \subseteq \mathcal{C}$ denote the set of long distance call types and let $n_+ = \sum_{j\in\mathcal{C}_+} n_j$ denote the number of long distance calls, both in service and queued. The unnormalised aggregated stationary probability for the queue is given by [5]

$$P(\mathbf{n}) = \begin{cases} \displaystyle\prod_{j\in\mathcal{C}} \frac{\rho_j^{n_j}}{n_j!} & 0 \le n_+ \le B \\ \\ \displaystyle\frac{n_+!}{B!B^{n_+-B}} \prod_{j\in\mathcal{C}} \frac{\rho_j^{n_j}}{n_j!} & B < n_+ \le N \end{cases}$$

where $\rho_j = \lambda_j/\mu_j$. Note that $P(\mathbf{n})$ can be viewed as the product of two "independent" queues: an $M/M/B/B$ queue serving the local calls and an $M/M/B/N$ queue serving the long distance calls. The processing of the local and long distance calls is not independent.

2.5.9 Local and Transit Calls

Consider the network presented in figure 2.3. A local call from switch c requires one circuit from link c. A local call from switch d requires one circuit from link d. Blocked local calls are lost. A transit call from switch c to switch d requires one circuit from link c, one circuit from link d and a circuit from switch T_1 to switch T_2. A transit call is lost if it is blocked on link c or on link d. If the transit call is not blocked on the local links it seizes the local circuits and requests a circuit from T_1 to T_2. If the circuit is available it is connected. If the circuit is not available and if there are less than N transit calls in progress (queued and in service) on the link from T_1 to T_2 then the transit call is queued, else it it is lost.

2.5.10 Hierarchical Tree Networks

Several networks as presented in figure 2.1 are connected to form the network shown in figure 2.4. A call from switch c is lost if it is blocked on any link connecting the source switch c to the switch T. A finite queue is used to buffer calls that are blocked on the final link from T to D. This model is a variant of the Multiserver Centre with Hierarchical Classes of Customers [21].

2.5.11 Local and External Networks

Consider the network presented in figure 2.5. The network consists of edge switches marked E and and interior switches marked I in a core network and access switches marked A in an access network. The switches are connected by multi-circuit links. The topology of the network is arbitrary and we assume that the switches in the access network are not directly connected to each other, although this assumption is not necessary.

Local calls are routed among the switches in the core network. Blocked local calls are lost. Consider an outgoing call from the network via an edge switch E to an access switch A. An outgoing call is lost if it is blocked on its local route. If the outgoing call is not blocked on its local route then it seizes the circuits in its local route and requests a circuit in the outgoing link from switch E to switch A. If an outgoing circuit is available the call is connected. If an outgoing circuit is not available and if there are less than N outgoing calls in progress (queued and in service) on the outgoing link then the outgoing call is queued, else it it is lost. Note that the local circuits acquired by outgoing calls are held while the calls are queued for connection, and that each outgoing link carries traffic in the outgoing direction only.

Call queueing is accomplished by storing the signalling information for the blocked call in a buffer at the edge switch E. The (signalling units for the) calls

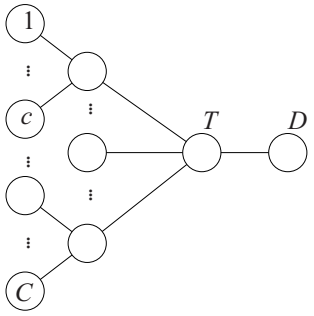

Fig. 2.4: Hierarchical tree network

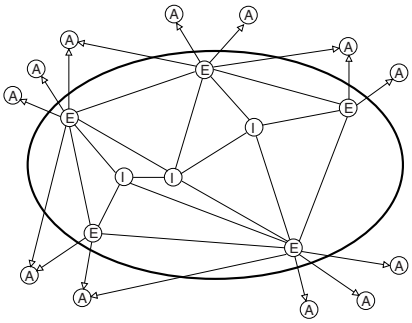

Fig. 2.5: Network with outgoing traffic

are stored in FCFS order. When a call completes and releases a circuit in the outgoing link to the access switch A the first call in the queue will be connected to the destination A.

Let γ denote a fixed route (an ordered sequence of links) connecting an originating switch I in the network via an edge switch E to an access switch A. Let $\mathcal{R}$ denote the set of interior routes. Let n_γ denote the number of calls in route $\gamma \in \mathcal{R}$ and let $\mathbf{n} = (n_\gamma)$ be a vector of numbers of calls. Let n_k where $k \in \mathcal{K}$ denote the number of calls (queued or in service) in progress on outgoing link k where $\mathcal{K}$ is the set of access links, and $\mathcal{R} \cap \mathcal{K} = \emptyset$. Then the un-normalised aggregated stationary probabilities for the network are given by [5]

$$P(\mathbf{n}) = \prod_{k \in \mathcal{K}} g_k(n_k) \prod_{\gamma \in \mathcal{R}} \frac{\rho_\gamma^{n_\gamma}}{n_\gamma!}$$

and

$$g_k(n_k) = \begin{cases} 1 & 0 \le n_k \le B_k \\ \dfrac{n_k!}{B_k! B_k^{n_k - B_k}} & B_k < n_k \le N_k \end{cases}$$

2.5.12 Transit Calls among Networks

Consider the system presented in figure 2.6. Each network consists of switches connected by multi-circuit links. Some of the switches are connected to gateway switches which route transit traffic in both directions between the networks. The

topology of the networks is arbitrary. We assume one pair of gateway switches but there may be many such pairs.

Local calls are routed among switches within their originating networks. Blocked local calls are lost. Consider a transit call from network 1 to network 2. A transit call is lost if it is blocked on its local route in network 1 from its originating switch to the gateway switch G or if it is blocked on its local route in network 2 from the gateway switch G to its destination switch (assuming a signalling link separate from the bearer link). If the transit call is not blocked on its local routes then it seizes the circuits in its local routes and requests a circuit on the transit link between the networks. If a transit circuit is available the call is connected. If a transit circuit is not available and if there are less than N transit calls in progress (queued and in service) on the transit link then the transit call is queued, else it it is lost.

The transit link carries two-way traffic and an identical call queueing mechanism is applied to calls in both directions. Each gateway has a buffer to store the signalling units for blocked calls. Both gateway switches are aware via the signalling link of the arrival order of blocked calls in both directions. The buffers are managed as a single logical queue whose entries are ordered according to the times of arrival of the blocked calls. The call resources are partitioned into two classes (local routes and transit routes) and the resource classes are ordered such that all calls first request their local routes and then their transit route – deadlock therefore cannot occur.

All of the models presented in sections 2.5.1 through 2.5.12 can be modelled as OI queues. Their stationary distributions are therefore immediately available. Efficient algorithms with space-time complexity $\mathrm{O}(B^2C)$ exist for the calculation of the blocking probabilities for models 2.5.5, 2.5.6 and 2.5.7. The complexity of model 2.5.10 is $\mathrm{O}(N^\ell)$ where ℓ is the depth of the tree. Models 2.5.9, 2.5.11 and 2.5.12 have the same complexity as the corresponding loss networks where blocked calls are lost.

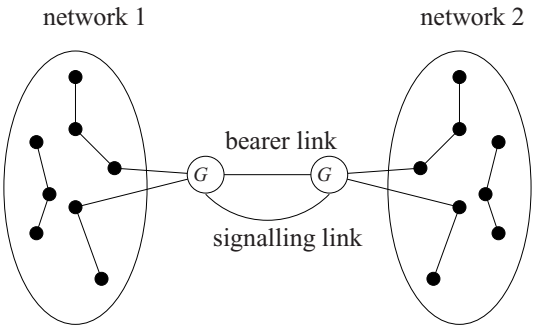

Fig. 2.6: Networks with transit traffic

Glossary

B the number of servers
B_c the number of type c customers that can be in service simultaneously
$\mathcal{C}$ the set of customer types
$\mathbf{C}$ the queue state $(c_n,\ldots,c_1)$
c_i the type of the customer in queue position i
$\gamma_i(\mathbf{C})$ the portion of the total service effort directed at the customer in queue position i
λ_c the Poisson arrival rate of customers of type c
μ_c the Poisson service rate of customers of type c
$\pi(\mathbf{C})$ the equilibrium probability
$\phi(\mathbf{C})$ the rate at which the total service effort in state $\mathbf{C}$ is supplied
$\mathcal{S}$ the state space of the queue
$s_i(\mathbf{C})$ the rate at which service is given to the customer in queue position i in the queue relative to the other customers in the queue
$a \wedge b$ the smaller of the two integers a and b
σ a permutation of $(1,\ldots,n)$

2.6 An Algorithm to Compute the Performance Measures of the MSCCC

The values of the $P(b,c,i)$ are stored in the elements of the array $P[b,i]$. The index c is suppressed to save storage. The algorithm is applied C times to compute the performance measures of all the types. After each application the customer types are relabelled.

// allocate and initialise the variables
double $P[0:B-1,0:\max(B_1,\ldots,B_C)],L[0:B-1]$
P[0,0] = 1
for $b = 1$ to $B-1$ **do**
  $P[b,0] = 0$
end for
// compute the un-normalised probabilities $P(b,c,i)$
for $c = 1$ to C **do**
  for $b = 1$ to $B-1$ **do**
    for $i = 1$ to $\min(b, B_c - 1)$ **do**
      $P[b,i] = \rho_c * P[b-1,i-1]/i$ // eq. (2.9)
    end for
    if $b \geq B_c$ **then**
      $P[b,B_c] = \rho_c * P[b-1,B_c-1]/(B_c - \rho_c)$ // eq. (2.10)
    end if
  end for
  for $b = 1$ to $B-1$ **do**

17: **for** $i = 1$ to $\min(b, B_c)$ **do**
18: $P[b,0] = P[b,0] + P[b,i]$ // eq. (2.13)
19: **end for**
20: **end for**
21: **end for**
// normalise the probabilities $P(b,c,i)$
22: $S = 0$
23: **for** $b = 0$ to $B-1$ **do**
24: $S = S + (B-b) * P[b,0]$ // eq. (2.15)
25: **end for**
26: $G = (B-\rho)/S$
27: **for** $b = 0$ to $B-1$ **do**
28: **for** $i = 1$ to $\min(b, B_c - 1)$ **do**
29: $P[b,i] = P[b,i] * G$ // eq. (2.16)
30: **end for**
31: **end for**
// compute the type C queue length $L(C)$
32: $L = \rho_C$
33: **for** $b = 1$ to $B-1$ **do**
34: $L[b] = P[b, B_C]$
35: **for** $i = 0$ to $\min(b, B_C - 1)$ **do**
36: $L[b] = L[b] + B_C * P[b-1,i] - i * P[b,i]$ // eq. (2.19)
37: **end for**
38: $L = L + (B-b) * L[b]$ // eq. (2.22)
39: **end for**
40: $L = L * \rho_C / (B_C - \rho_C) / (B-\rho)$ // eq. (2.22)

References

1. I. Adan, J. Visschers and J. Wessels. Sum of product forms solutions to MSCCC queues with job type dependent processing times. Memorandum COSOR 98-19, Eindhoven University of Technology, The Netherlands (1998).
2. J.M. Akinpelu, The Overload Performance of Engineered Networks With Nonhierarchical and Hierarchical Routing, International Teletraffic Congress, Vol 10 (1983) 3.2.4.1 – 3.2.4.7.
3. F. Baskett, K.M. Chandy, R.R. Muntz and J. Palacios, Open, Closed and Mixed Networks of Queues with Different Classes of Customers, Journal of the ACM Vol 22 No 2 (1975) 249 – 260.
4. S.A. Berezner and A.E. Krzesinski, Quasi-reversible multiclass queues with order independent departure rates. Queueing Systems, 19 (1995) 345 – 359.
5. S.A. Berezner and A.E. Krzesinski, Order Independent Loss Queues. Queueing Systems, 23 (1996) 331 – 335.
6. S.A. Berezner and A.E. Krzesinski, Call Queueing in Circuit-Switched Networks. Telecommunication Systems Vol 6 (1996) 147 – 160.
7. J.-Y. Le Boudec, A BCMP Extension to Multiserver Stations with Concurrent Classes of Customers. In: *Proc. 1986 ACM Sigmetrics Conference, Performance Evaluation Review* Vol 14 No 1 (1986) 78 – 91.

8. J.-Y. Le Boudec, The MULTIBUS Algorithm. Performance Evaluation Vol 8 No 1 (Feb 1988) 1 – 18.
9. D.Y. Burman, J.P. Lehoczky and Y. Lim, Insensitivity of blocking Probabilities in a Circuit-Switching Network, Journal of Applied Probability, Vol 21 (1984) 850 – 859.
10. X. Chao and M. Pinedo, On Generalized Networks of Queues with Positive and Negative Arrivals, Prob. Eng. Inf. Sci, Vol 7 (1993) 301 – 334.
11. S. Crosby and A.E. Krzesinski, Product Form Solutions for Multiserver Centers with Concurrent Classes of Customers, Performance Evaluation, Vol 11 No 4 (1990) 265 – 281.
12. S. Crosby, A.E. Krzesinski and J.-Y. Le Boudec, A MSCCC Model of Multilayer Window Flow Control. In: *Fifth International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, G Balbo and G. Serazzi, Eds. (Elsevier 1992).
13. J.M. Harrison and R.J. Williams, On the Quasi-Reversibility of a Multiclass Brownian Station, Annals of Probability Vol 18 (1990) 1249 – 1268.
14. W. Henderson, C.E.M. Pearce, P.K. Pollett, P.G. Taylor, Connecting Internally Balanced Quasi-Reversible Markov Processes, Advances in Applied Probability Vol 24 (1992) 934 – 959.
15. M. Hofri and Y. Kogan, Exact and Asymptotic Analysis of Large Multiple-Bus Multiprocessor Systems, Proc. Performance '90, Elsevier Science Publishers B.V. (North Holland) (1990) 373–389.
16. J.R. Jackson, Networks of Waiting Lines, Operation Research Vol 5 (1957) 518 – 521.
17. F.P. Kelly, Reversibility and Stochastic Networks, John Wiley and Sons (1979) ISBN 0-471-27601-4.
18. F.P. Kelly, Networks of Quasi-reversible Nodes, Applied Probability – Computer Science, the Interface: Proceedings of the ORSA–TIMS Boca Raton Symposium, Ed. R. Disney, Birkhauser Boston, Cambridge, Ma (1981).
19. F.P. Kelly, Blocking probabilities in Large Circuit-switching Networks, Advances in Applied Probability Vol 18 (1986) 473 – 505.
20. F. P. Kelly, Loss Networks, Annals of Applied Probability, Vol. 1, No. 3, (1991) 473 – 505.
21. A.E. Krzesinski and R. Schassberger, The Multiserver Center with Hierarchical Concurrency Constraints, Prob. Eng. Inf. Sci, Vol 6 (1992) 147 – 156.
22. M.A. Marsan, G. Balbo, G. Chiola and S. Donatelli, On the Product Form Solution of a Class of Multiple-Bus Multiprocessor System Models, Journal of Systems and Software Vol 1 No 2 (1986) 117 – 124.
23. D. Mitra, Asymptotic Analysis and Computational Methods for a Class of Simple Circuit-Switched Networks with Blocking, Advances in Applied Probability, Vol 19 (1987) 219 – 239.
24. R.R. Muntz, Poisson Departure Processes and Queueing Networks, IBM Research Report RC4145, IBM Thomas J. Watson Research Center, Yorktown Heights, New York (1972).
25. R.D. Nelson, The Mathematics of Product Form Queueing Networks, ACM Computing Surveys, Vol 25 (1993) 339 – 369.
26. E. Pinsky and A. Conway, Exact computation of blocking probabilities in state-dependent multi-facility blocking models, in *IFIP WG 7.3 International Conference on the Performance of Distributed Systems and Integrated Communication Networks*, T. Hasegawa, H. Takagi and Y. Takahashi eds., Kyoto, Japan 1991.
27. P.K. Pollett, Connecting Reversible Markov Processes, Advances in Applied Probability, Vol 18 (1986) 880 – 909.
28. P.K. Pollett, Preserving Partial Balance in Continuous-time Markov Chains, Advances in Applied Probability Vol 19 (1987) 431 – 453.
29. J. Walrand and P. Varaiya, Interconnections of Markov Chains and Quasi-Reversible Queueing Networks, Stochastic Processes and Applications Vol 10 (1980) 209 – 219.
30. J. Walrand, An Introduction to Queueing Networks, Prentice-Hall (1988), ISBN 0-13-493818-6.
31. J. Walrand, A Probabilistic Look at Networks of Quasireversible Queues, IEEE Trans. Inf. Theory IT-29 (1983) 825 – 831.
32. P. Whittle, Systems in Stochastic Equilibrium, John Wiley and Sons (1986) ISBN 0-471-90877-8.

http://www.springer.com/978-1-4419-6471-7

Queueing Networks
A Fundamental Approach
Boucherie, R.J.; van Dijk, N.M. (Eds.)
2011, XXIV, 800 p., Hardcover
ISBN: 978-1-4419-6471-7