Resource-Sharing Models with State-Dependent Arrivals of Batches

by

Gagan L. Choudhury,[1] *Kin K. Leung*[2] *and Ward Whitt*[3]

AT&T Bell Laboratories

October 7, 1994

[1] AT&T Bell Laboratories, Room 1L-238, Holmdel, NJ 07733-3030; email: gagan@buckaroo.att.com

[2] AT&T Bell Laboratories, Room 1K-211, Holmdel, NJ 07733-3030; email: kin@buckaroo.att.com

[3] AT&T Bell Laboratories, Room 2C-178, Murray Hill, NJ 07974-0636; email: wow@research.att.com

Abstract

We recently developed a new algorithm for calculating the blocking probability of each class in resource-sharing models with upper limit and guaranteed minimum sharing policies as well as the standard complete-sharing policy. These models may have multiple resources and multiple classes, with each class requiring multiple units from each resource. These models may also have state-dependent arrival and service rates. Our new algorithm is based on calculating normalization constants appearing in the product-form steady-state distributions by numerically inverting their generating functions. In the present paper we provide the basis for extending the algorithm to resource-sharing models with batch arrival processes. The batch sizes are mutually independent random variables with distributions depending on the class. We show that the steady-state distribution of the resource-sharing model has a product form for both complete-batch blocking and partial-batch blocking, and we derive the generating functions of the normalization constants for partial-batch blocking. We primarily focus on the Bernoulli-Poisson-Pascal (BPP) special case in which the batches have linear state-dependent arrival rates, which includes finite-source inputs and Poisson inputs for the batches as special cases. With batches, we require exponential service times, but when there are state-dependent arrivals of single customers (no batches), the service-time distributions can be general. By considering state-dependent arrivals for the batches, multiple resources and non-complete-sharing policies, our treatment extends recent results for resource-sharing models with batch arrivals by van Doorn and Panken (1993), Kaufman and Rege (1992) and Morrison (1993). Even for the batch models previously considered, our algorithm is faster than recursive algorithms previously developed for large models. We also provide a new derivation of the product-form steady-state distributions that helps explain why service-time insensitivity does not hold when there are batches.

Keywords and phrases: resource-sharing model, blocking probability, loss network, product-form steady-state distributions, normalization constant, partition function, numerical transform inversion, generating function, batch arrivals.

1. Introduction

Motivated largely by telecommunications applications, there recently has been considerable interest in multi-class multi-rate multi-resource generalizations of the classical Erlang loss model, often referred to as loss networks or resource-sharing models, e.g., see Kelly [12], Choudhury, Leung and Whitt [4,5] and references therein. The basic resource-sharing models are continuous-time Markov chains with very large state spaces, but because of their special structure they have product-form steady-state distributions, just like product-form closed queueing networks. Van Doorn and Panken [21] and Kaufman and Rege [11] recently showed that these resource-sharing models still have tractable product-form steady-state distributions when the arrival processes are extended from Poisson to batch-Poisson with general batch-size distributions, provided that the service-time distributions are exponential and that partial-batch blocking is used. (Van Doorn and Panken considered geometric batch-size distributions, while Kaufman and Rege considered general batch-size distributions.) As these authors point out, this result is of considerable interest because service requests do often arrive in batches and because batch-Poisson arrivals can be used to approximately represent arrival processes that are more ''peaked,'' ''bursty'' or variable than the ordinary Poisson process.

In this paper we provide a new derivation of this result, which helps to explain why it is true and helps to develop various extensions. In particular, we show that the resource-sharing model with batch-Poisson arrivals is equivalent to another model with ordinary Poisson arrivals, which can be shown to have a product-form distribution by standard methods. The equivalent model consists of several series of infinite-server (IS) queues in parallel, together with direct admission to downstream queues under certain conditions. In this equivalent model, we represent batches of customers in the original model by single ''macro'' customers. A macro customer enters the system when the batch enters and leaves only when the last member of the batch leaves. During the lifetime of a macro customer, the total number of ordinary (real) customers in it decreases. The macro customer is classified according to the number of ordinary customers in the batch remaining in the system. The service rate of a class-j macro customer with k ordinary customers remaining is k times the service rate of an ordinary customer. The exponential service-time distributions are critical for this representation, so that we clearly see why there is no service-time insensitivity in this resource-sharing model with batches.

We also separately consider the $M^X/G/\infty$ model (with batch-Poisson arrivals, i.i.d. service times having a general distribution and no capacity constraints) and show that the steady-state distribution depends on the service-time and batch-size distributions beyond their means, thus directly showing that service-time insensitivity does not hold. (Kaufman and Rege [11] give a different demonstration.) When the batch-size distribution is geometric and the service-time and distribution is exponential, the steady-state distribution is negative binomial. This negative binomial distribution coincides with the steady-state distribution in the IS queue with positive linear state-dependent arrival rate without batches considered by Delbrouck [7]. The connection between these two classes of resource-sharing models was noted by van Doorn and Panken [21] and Kaufman and Rege [11].

We consider both the partial-batch blocking considered in [11,21] and complete-batch blocking. (Our approach provides a basis for treating other forms of partial batch blocking as well, but we do not consider any others.) We show that both partial-batch blocking and complete-batch blocking lead to product-form steady-state distributions. We use our new representation to show that the resource-sharing model with partial-batch blocking and batch-Poisson arrivals has the same steady-state distribution as a standard resource-sharing model with ordinary Poisson arrivals (but different parameters). Thus, *previous algorithms* for resource-sharing models with ordinary Poisson arrivals can be applied to solve the resource-sharing model with partial-batch blocking and batch-Poisson arrivals.

We extend the results in [11,12] in three important directions. First, we allow multiple resources. Thus our resource-sharing models are generalizations of the loss networks in [12]. Second, we allow the batches to arrive in a *state-dependent* manner as well as according to a Poisson process. In particular, the arrival rate of class-j batches may depend on the number of class-j batches with customers still in the system. (We are able to treat dependence for the batches, but not for the customers themselves unless the batches are of size 1.) This includes finite-source inputs and positive-linear state-dependent inputs for the batches, i.e., the binomial and Pascal (negative binomial) cases in the binomial-Poisson-Pascal (BPP) trilogy considered by Delbrouck [7] and Dziong and Roberts [8]. We let the batches themselves be random, with a distribution that depends only on the class. These features provide more flexibility in arrival process modeling. For example, the binomial and Pascal cases represent arrival processes that are, respectively, less bursty and more bursty than Poisson.

Third, as in [4,5], we consider the *upper limit* (UL) and *guaranteed minimum* (GM) sharing policies as well as the standard *complete sharing* (CS) policy. The UL and GM bounds are very useful for providing protection against overloads and for providing different grades of service to different job classes. There are product-form steady-state distributions in these cases as well.

In addition to showing that the steady-state distributions have product-forms, we develop efficient numerical algorithms for computing the normalization constants and the blocking probabilities. Our approach here extends [4,5], where we considered single customers (no batches). We first derive expressions for the multidimensional generating function of the normalization constants in a compact form. (In the single-resource, complete-sharing, batch-Poisson case, the generating function agrees with that of Morrison [16].) Next we invert the generating function using the algorithm developed in Choudhury, Leung and Whitt [3,4,5] which is based on the Fourier-series method [1,2,6]. In Section 12 we show that the inversion algorithm is faster than previous recursions as well as new ones derived by us. The speed-difference is particularly impressive in the case of geometric batch-size distributions. Furthermore, our algorithm is applicable to many models for which recursive algorithms have not yet been developed.

In the single-resource, complete-sharing, batch-Poisson case, Morrison [16] has developed a uniform asymptotic approximation (UAA) that is fast and often remarkably accurate. Indeed, we used the algorithm developed here to provide exact numerical results to compare with UAA in [16]. However, as shown in [5], it is possible to construct large examples where the recursive algorithms breakdown and yet UAA is not sufficiently accurate. (Admittedly, these examples are somewhat pathological.)

Here is how the rest of this paper is organized. In Section 2 we start by considering a general series network with state-dependent arrival and service rates, and direct admission to downstream queues under certain conditions, and show that it has a product-form steady-state distribution. In Section 3 we extend the model in Section 2 to allow several of the series networks in parallel, with admission directly to downstream queues depending on the state of the entire network. By the argument in Section 2, this model also has a product-form steady-state distribution. (The product-form result extends to feedforward networks, but not more generally.) We consider these series-network models because the loss model can be shown to be equivalent to a special class of them, but these series-network models are of independent interest. They may also have other applications, e.g., to production line models.

In Section 4 we consider an alternative model, closer to the resource-sharing model, consisting of a single service facility but many classes of customers, with customers changing class and returning after completing service. Now, under certain conditions, upon arrival of one class, a customer of a different class may be admitted instead. For the case of a parallel-series network with infinite-server queues, the new model is equivalent to the parallel-series network in Section 3.

In Section 5 we consider the resource-sharing model of primary interest and apply the previous results to derive its product-form steady-state distribution. In Section 6 we derive the generating functions of the normalization constants with partial-batch blocking in the three BPP cases. In Section 7 we derive the generating functions of the normalization constants in the special case of geometric batch-size distributions. These generating functions have an especially compact form, which is important for producing a fast inversion algorithm. In Section 8 we show how to use the normalization constants to calculate blocking probabilities. We show that the blocking probabilities can be calculated by performing only two inversions. In Section 9 we give the generating functions for the upper limit (UL) and guaranteed minimum (GM) sharing policies.

In Section 10 we consider the $M^X/G/\infty$ queue and show that insensitivity does *not* hold, thus providing further insight into model behavior. In Section 11, following [5], which in turn follows Reiser and Kobayashi's [17] treatment of closed queueing networks, we show how the generating functions can be used to generate recursive algorithms for the normalization constants. In Section 12 we discuss the computational complexity of our algorithm and compare it to recursive algorithms. In Section 13 we discuss the scaling needed to control errors in the inversion algorithm. Finally, in Section 14 we give a few illustrative numerical examples.

2. A Series Network with Downstream Admission

In this section we consider a continuous-time Markov chain representing m state-dependent queues in series with a state-dependent external arrival process. As usual in product-form queueing networks, the service rate at a queue depends on the number of customers at that queue, while the arrival rate depends on the number of customers in the entire network. The arrivals all come to the first queue and join it if possible. However, in certain states the arrival is not admitted to the first queue, and is instead immediately placed in the j^{th} queue for some j, or is rejected altogether. After the customer assignment, the system operates as a standard series queueing network. Our purpose is to show that, under appropriate conditions on the admission rule, this modified series network has a product-form steady-state distribution, which is a truncation and renormalization of the familiar steady-state distribution of the standard open network in which all arrivals are assigned to the first queue. Our downstream admission can be regarded as a form of state-dependent routing, similar to previous forms considered by Jackson [10], Towsley [19], Krzesinski [13] and van Dijk [20], but it seems not to have been considered before.

Let $\mathbf{n} \equiv (n_1, \ldots, n_m)$ be the system state, where n_j is the number of customers in the j^{th} queue. Let $\mathbf{e}_j$ be the m-dimensional unit vector, with a 1 in the j^{th} place and 0's elsewhere. The state space S is a subset of $\mathbf{Z}_m^+$, the vector of nonnegative integer m-tuples. Let S be partitioned into $m+1$ subsets B_i, $1 \le i \le m+1$. In B_j arrivals are admitted to queue j, $1 \le j \le m$, and in B_{m+1} arrivals are blocked and lost. If $\mathbf{n} \in B_j$, then $\mathbf{n} + \mathbf{e}_j \in S$. We assume that $\mathbf{n} - \mathbf{e}_1 \in B_1$ if $\mathbf{n} \in S$ and $n_1 > 0$. We assume that $\mathbf{n} + \mathbf{e}_j \in S$ whenever $\mathbf{n} + \mathbf{e}_i \in S$ for $i < j$. (Customers who are admitted always have someplace to go.) We assume that if $\mathbf{n} \in B_j$, then $\mathbf{n} + \mathbf{e}_i \notin S$ for $i < j$. (Customers could not have been admitted to previous queues.) We assume that $\mathbf{n} - \mathbf{e}_j \in S$ whenever $\mathbf{n} \in S$ and $n_j > 0$, and that the origin is contained in B_1. We focus on the irreducible continuous-time Markov chain containing the origin. For simplicity, we assume that the state space S is finite.

Let $s(\mathbf{n}) = n_1 + \ldots + n_m$; let $\lambda(k)$ be the arrival rate when $s(\mathbf{n}) = k$, and let $\mu_i(k)$ be the service rate at queue i when $n_i = k$.

Theorem 2.1. *This series queue network with downstream admission as specified above has the proper product-form steady-state distribution*

$$\pi(\mathbf{n}) = f(\mathbf{n})/g(S) \ , \quad \mathbf{n} \in S \ , \tag{2.1}$$

where

$$f(\mathbf{n}) = \prod_{j=0}^{s(\mathbf{n})-1} \lambda(j) \prod_{i=1}^{m} \prod_{l=1}^{n_i} (1/\mu_i(l)) \tag{2.2}$$

and $g(S) = \sum_{\mathbf{n} \in S} f(\mathbf{n})$.

Proof. We show that partial balance is satisfied, i.e., for each state, the flow in due to arrivals at each queue equals the flow out due to departures from that queue. It is well known that such partial balance implies that π satisfies the global balance equations. Consider a state $\mathbf{n} \in S$ with $n_j > 0$ for $j > 1$. First, suppose that $\mathbf{n} + \mathbf{e}_{j-1} - \mathbf{e}_j$ is in S. This implies that $\mathbf{n} - \mathbf{e}_j \notin B_j$; if $\mathbf{n} - \mathbf{e}_j \in B_j$, then $\mathbf{n} - \mathbf{e}_j + \mathbf{e}_{j-1} \notin S$ by assumption, which is a contradiction. Thus, if $\mathbf{n} + \mathbf{e}_{j-1} - \mathbf{e}_j \in S$, then the only flow into state $\mathbf{n}$ at queue j is due to a departure from queue $j-1$. Then the flow in due to an arrival at queue j (from queue $j-1$) is $\pi(\mathbf{n} + \mathbf{e}_{j-1} - \mathbf{e}_j)\mu_{j-1}(n_{j-1} + 1)$, while the flow out due to a departure from queue j is $\pi(\mathbf{n})\mu_j(n_j)$. These are clearly equal under (2.1). Second, suppose that $\mathbf{n} + \mathbf{e}_{j-1} - \mathbf{e}_j \notin S$. Then necessarily $\mathbf{n} - \mathbf{e}_j \notin B_i$ for $i \le j-1$. Since $\mathbf{n} \in S$, $\mathbf{n} - \mathbf{e}_j \in S$ by assumption. Hence, $\mathbf{n} - \mathbf{e}_j \in B_j$. In this case the flow into state $\mathbf{n}$ at queue j is due entirely to an external arrival admitted directly to queue j. Then the flow is due to an arrival at queue j is $\pi(\mathbf{n} - \mathbf{e}_j)\lambda(s(\mathbf{n}) - 1)$, while the flow out due to a departure from queue j is again $\pi(\mathbf{n})\mu_j(n_j)$. These flows are also equal by (2.1). A similar argument holds for $j = 1$. By assumption, if $\mathbf{n} \in S$ and $n_1 > 0$, then $\mathbf{n} - \mathbf{e}_1 \in B_1$. ■

Note that when the batches arrive in an ordinary Poisson process so that $\lambda(j) = \lambda$ for all j in (2.2), the steady-state distribution in (2.2) factors into m terms; i.e., the steady-state distribution in (2.2) is the same as for m independent queues where each queue has its own Poisson arrival process. This representation is the basis for concluding that the resource-sharing model with batch-Poisson arrivals has the same steady-state distribution as another resource-sharing model with ordinary Poisson arrivals; see Section 5.

3. Parallel Series Networks with Downstream Admission

We now consider an extension of the model in Section 2 having r classes of customers and m_j different series networks in parallel for class j, $1 \le j \le r$. There are thus $\prod_{j=1}^{r} m_j$ separate series networks. We allow m_j to be infinite, which presents no problem. There is a state-dependent arrival process for each class. The arrival rate to series network k for class j is $\lambda_j(n)p_{jk}$ when the number of class j customers in the network is n, where $\Sigma_k^{m_j} p_{jk} = 1$. As before, there are state-dependent service rates. The service rate at queue i of network k of class j is $\mu_{jki}(n)$ when there are n customers at that queue. Let q_{jk} be the number of queues in series network k for class j. For the loss network to be considered in Section 5, $q_{jk} = k$ for all j and k.

Let $\mathbf{n} \equiv (n_{jki} : 1 \le j \le r,\ 1 \le k \le m_j,\ 1 \le i \le q_{jk})$ be the new network state and let $\mathbf{e}_{jki}$ be an associated unit vector. The interesting feature now is that the assignment of arrivals to queues depends on the entire system state $\mathbf{n}$.

For each of the series networks, we partition the state space as in Section 2. For series network k for class j, we have $q_{jk} + 1$ subsets. In B_{jki} arrivals to this network are admitted to queue i, $1 \le i \le q_{jk}$. In $B_{j,k,q_{jk}+1}$, arrivals to network k of class j are rejected altogether. Thus, if $\mathbf{n} \in B_{jki}$, then $\mathbf{n} + \mathbf{e}_{jki} \in S$. We assume that $\mathbf{n} - \mathbf{e}_{jk1} \in B_{jk1}$ whenever $\mathbf{n} \in S$ and $n_{jk1} > 0$. We assume that $\mathbf{n} + \mathbf{e}_{jkl} \in S$ whenever $\mathbf{n} + \mathbf{e}_{jki} \in S$ for $i < l$. (Again, customers in the network always have somewhere to go.) We assume that if $\mathbf{n} \in B_{jkl}$, then $\mathbf{n} + \mathbf{e}_{jki} \notin S$ for $i < l$. As before, we assume that $\mathbf{n} - \mathbf{e}_{jki} \in S$ whenever $\mathbf{n} \in S$ and $n_{jki} > 0$. We also assume that the origin is contained in S and that S is finite.

As a straightforward extension of Theorem 2.1, we get the following product-form result. Let $n_{jk} = \sum_{i=1}^{q_{jk}} n_{jki}$ and $n_j = \sum_{k=1}^{m_j} n_{jk}$.

Theorem 3.1. *This parallel-series network with downstream admission as specified above has proper product-form steady-state distribution*

$$\pi(\mathbf{n}) = f(\mathbf{n})/g(S)\ , \quad \mathbf{n} \in S\ , \tag{3.1}$$

where

$$f(\mathbf{n}) = \prod_{j=1}^{r} \left[\prod_{a=0}^{n_j - 1} \lambda_j(a) \right] \prod_{k=1}^{m_j} p_{jk}^{n_{jk}} \prod_{i=1}^{q_{jk}} (1/\prod_{l=1}^{n_{jki}} \mu_{jki}(l))\ , \tag{3.2}$$

and $g(S) = \sum_{\mathbf{n} \in S} f(\mathbf{n})$.

We remark that Theorem 3.1 extends to feedforward networks; i.e., we can have arrivals directly to downstream queues as well as splitting and joining. Then we must include the probability that the customer starting at queue i would ever reach queue j. If P_{ij} is the routing probability from queue i, then any arrival at i not admitted directly there tries queue j with probability P_{ij}. At each queue the customer is admitted if the state n allows an additional customer, where the admission rules are essentially as before.

Example 3.1. To see that the product-form result does *not* hold with downstream admission when feedback is allowed, it suffices to consider two queues in series where departures from queue 2 return to queue 1 with probability p and leave the network with probability $1-p$, $0 \le p < 1$. Let the external arrival rate be λ when there is less than or equal to 1 customer in the network, and 0 otherwise. Thus the state space is $S = \{(0,0), (0,1), (1,0), (0,2), (1,1), (2.0)\}$. Let the service rate in queue i be $k\mu_i$ when there are k customers in queue i. So far, this is a Jackson network with a product-form steady-state distribution. Now stipulate that an exogeneous arrival is admitted to queue 1 only in state (0,0). Let the exogeneous arrival be admitted to queue 2 in state (0,1) and rejected in state (1,0). It is not difficult to verify that the steady-state distribution has a product form if and only if $p = 0$. If $p = 0$, then the states (1,1) and (2,0) cannot be reached, but if $p > 0$, then they can. In particular, a product-form steady-state distribution can be written as $\pi(i,j) = Kx_i y_j$ for $x_0 = y_0 = 1$. By considering the flow in and out of (0,0), we see that $y_1 = \lambda/\mu_2(1-p)$. Then, by considering the flow in and out of (1,0), we see that $x_1(\mu_1 - \lambda) = \lambda(1-p)$, which produces a contradiction for $\mu_1 \le \lambda$. If

$\lambda < \mu_1$, then we still obtain a contradiction by considering the flow in and out of the remaining states. The two remaining variables x_2 and y_2 cannot be chosen to satisfy the remaining four equations.

4. An Alternative Single-Facility Model

For the case in which the networks in Sections 2 and 3 consist of all infinite-server (IS) queues, we can represent the network *equivalently* as a single IS queue with many classes. Then class (j,k,i) corresponds to the previous class-*j* size-*k* queue-*i*. The service rate for each class (j,k,i) customer is then μ_{jki}. Upon completing service, a class-(j,k,i) customer returns to the queue again as a class-$(j,k,i+1)$ customer, $1 \leq i \leq k-1$, while a class (j,k,k) customer leaves the system. The previous admission to a downstream queue corresponds now to an admission of a class (j,k,l) customer upon the arrival of a $(j,k,1)$ customer for $l > 1$ under certain circumstances. Thus, the previous steady-state distribution applies to this multi-class single-facility model as well.

We can make other versions of this multi-class single-facility model that have product-form as well. For example, we can allow the different classes to have different service requirements at the service facility if we use the processor-sharing discipline. Then again we have product form. However, the IS case is of special interest, because it provides a connection to the resource-sharing models, which we consider next.

5. Multiple Resources with Batch Arrivals

In this section we consider the resource-sharing model of primary interest. There are *p* resources with capacities K_i, $1 \leq i \leq p$, shared by *r* classes of customers. Class *j* arrives according to a state-dependent batch arrival process. The class-*j* batches have arrival rate $\lambda_j(n)$ when there are customers from *n* class-*j* batches still in the system being served. We assume that the batch sizes are mutually independent, with the batch-size distribution depending on the class *j*. The upper limit of the batch size for class *j* is m_j, which we allow to be infinite. The probability that a class-*j* batch contains *k* customers is p_{jk}.

Each class-*j* customer requires a_{ij} units of resource *i*, $1 \leq i \leq p$, where $a_{ij} = 0$ is allowed, but a_{ij} should be positive for at least one *i*. We assume that the capacities K_i and the resource requirements a_{ij} are integers, but that is not necessary for our product-form results. (We use the integer structure to construct generating functions in Section 6.) We consider two blocking policies: *partial-batch blocking* and *complete-batch blocking.* With partial-batch blocking, as many customers as possible from the batch are admitted. (Each customer must have all of its required resource units though.) With complete-batch blocking, if all customers cannot be admitted, then the entire batch is blocked. (Our approach also allows us to treat other forms of blocking, but we do not.) Each customer is treated independently once admitted. Each class-*j* customer holds all its resource units for an exponential holding time with mean μ_j^{-1}. In order to have a product-form steady-state distribution (when there are batches), it is important that this holding-time distribution be exponential, as we explain next. For any class whose batches are all of size 1, the service-time distribution can be general. Thus, our model includes as a special case the non-batch loss model with state-dependent arrival rates in Dziong and Roberts [8].

We represent these models (with the two forms of blocking) equivalently as networks of parallel series queues with single (non-batch) arrivals. We treat a class-*j* batch arrival of size *k* as a single ‘‘macro’’ customer who first goes to an infinite-server (IS) queue where he has exponential service time having mean $(k\mu_j)^{-1}$. We call a macro customer a size-*k* customer if upon arrival the batch is of size *k*. By a basic property of the exponential distribution, the time required for the first of the k customers to depart has an exponential distribution with mean $(k\mu_j)^{-1}$. (The exponential property is used critically here.) Thus, the macro customer corresponds to k customers for this exponential time with mean $(k\mu_j)^{-1}$. Then the macro customer moves to the next IS queue where it corresponds to the remaining $k-1$ customers. Due to the memoryless property of the exponential distribution, the macro customer then has an exponential service-time distribution time with mean $((k-1)\mu_j)^{-1}$. The macro customer continues this way through k IS queues in series and then departs from the system. (Note that the quantity q_{jk} in Section 3 equals k in this case.) A class-j size-k macro customer at queue i corresponds to $k + 1 - i$ customers from this original class-j batch of size k still remaining in the system.

We have a series network for all j and k such that $p_{jk} > 0$. The maximum value of k we need consider for class j is m_j, the upper limit of the batch size for class j, which can be infinite. Class-j macro customers corresponding to arriving batches of size k arrive at rate $\lambda_j(n) p_{jk}$ when there are n class-j macro customers in the system. Thus we have state-dependent arrivals, with the arrival rate depending on the number of class-j batches in the system, rather than the number of class-j customers.

It now remains to specify the system behavior with the two forms of blocking. With *complete-batch blocking,* an arrival of a class-j macro customer of size k is blocked if this customer cannot have all its requirements satisfied. Let $\mathbf{n} \equiv (n_{jkl} : 1 \le j \le r,$ $1 \le k \le m_j,\ 1 \le l \le k)$ be the state vector, where n_{jkl} is the number of class-j macro customers arriving as size k now at queue l in the series of k queues. Then a new class-j size-k macro customer arrival is blocked in state $\mathbf{n}$ if

$$ka_{ij} + \sum_{j=1}^{r} \sum_{k=1}^{m_j} \sum_{l=1}^{k} n_{jkl} a_{ij} (k+1-l) > K_i \quad \text{for at least one } i\ , \quad 1 \le i \le p\ . \tag{5.1}$$

Otherwise, the macro customer is admitted to queue 1 of series network (j,k); i.e., $\mathbf{n}$ becomes $\mathbf{n} + \mathbf{e}_{jk1}$.

On the other hand, with *partial-batch blocking,* the class-j size-k macro customer goes immediately to queue m, $1 \le m \le k$, if

$$a_{ij}(k-m+2) + \sum_{j=1}^{r} \sum_{k=1}^{m_j} \sum_{l=1}^{k} n_{jkl} a_{ij} (k+1-l) > K_i \quad \text{for } \textit{at least one } i\ , \tag{5.2}$$

while

$$a_{ij}(k-m+1) + \sum_{j=1}^{r} \sum_{k=1}^{m_j} \sum_{l=1}^{k} n_{jkl} a_{ij} (k+1-l) \le K_i \quad \text{for } \textit{all } i\ , \tag{5.3}$$

$1 \le i \le p$. This macro customer is blocked altogether if

$$a_{ij} + \sum_{j=1}^{r} \sum_{k=1}^{m_j} \sum_{l=1}^{k} n_{jkl} a_{ij} (k+1-l) > K_i \quad \text{for } \textit{at least one } i\ , \quad 1 \le i \le p\ . \tag{5.4}$$

The models with both forms of blocking can easily be seen to be special cases of the parallel-series network in Section 3. The service rate for each macro customer at IS queue l for class-j and size k is $(k-l+1)\mu_j$.

Let Q_j be the number of class-*j* ordinary customers in the system. From above,

$$Q_j \;=\; \sum_{k=1}^{m_j} \sum_{l=1}^{k} n_{jkl}(k-l+1) \;\;, \tag{5.5}$$

where n_{jkl} is the number of class-*j* macro customers originally of size-*k* currently at queue *l* in the series of *k* IS queues. The state space for partial-batch blocking is quite simple, namely,

$$S \equiv S_p \;=\; \left\{ \mathbf{n} : \sum_{j=1}^{r} \sum_{k=1}^{m_j} \sum_{l=1}^{k} n_{jkl}\, a_{ij}(k+1-l) \;\le\; K_i \;,\; 1 \le i \le p \right\} . \tag{5.6}$$

However, the state space for complete-batch blocking is more complicated. For example, suppose that there is only one class and one resource. Let the resource have capacity 6 and let all batches be of size 5. Let each customer require only one unit of the resource. Then the state space is a set of five-tuples. In particular *S* contains the null vector, all unit vectors and all vectors with sum two having a 1 in the fifth place. However, the vector $(0,0,0,2,0)$ is not in the state space, even though the sum of the units required is only 4, because the second class-*j* macro customer would necessarily have been rejected upon arrival.

To define the state space for complete-batch blocking, we introduce a *partial order* on the vectors $\mathbf{n}$. We say that $\mathbf{n} \le \mathbf{n}'$ if

$$\sum_{l=1}^{m} n_{jkl} \;\le\; \sum_{l=1}^{m} n'_{jkl} \quad \text{for all} \quad j,k,m \;. \tag{5.7}$$

If $\mathbf{n} \le \mathbf{n}'$, then $\mathbf{n}$ can be obtained starting from $\mathbf{n}'$ by departures only. Hence, For complete-batch blocking, we can represent the state space as

$$S \equiv S_C \;=\; \{ \mathbf{n} : \mathbf{n} \le \mathbf{n}' + \mathbf{e}_{jk1} \ \text{for } \textit{some } j,\, k \textit{ with } p_{jk} > 0 \quad \text{and for } \mathbf{n}' \textit{ with}$$

$$ka_{ij} \;+\; \sum_{j=1}^{r} \sum_{k=1}^{m_j} \sum_{l=1}^{k} n'_{jkl}\, a_{ij}(k+1-l) \;\le\; K_i \;,\; 1 \le i \le p \} \;. \tag{5.8}$$

Once we have determined the state spaces, the steady-state distribution can be obtained directly from Theorem 3.1.

Theorem 5.1. *The steady-state distribution of the vector* $\mathbf{n}$, *where* n_{jki} *is the number of class-j batches originally of size k with* $k-i+1$ *customers remaining to be served, has the product-form in Theorem 3.1, where* $p_{jk}(a) \equiv p_{jk}$ *(independent of a) and* $\mu_{jki}(l) \;=\; \mu_j(k-i+1)\,l$, *i.e.,*

$$f(\mathbf{n}) \;=\; \prod_{j=1}^{r} \left[\prod_{a=0}^{n_j-1} (\lambda_j(a)/\mu_j) \right] \prod_{k=1}^{m_j} p_{jk}^{n_{jk}} \prod_{i=1}^{k} (1/(k-i+1)^{n_{jki}}\, n_{jki}!) \tag{5.9}$$

and the state space S is as in (5.6) for partial-batch blocking and in (5.8) for complete-batch blocking.

Henceforth we restrict attention to partial-batch blocking. For partial-batch blocking, we can obtain a more compact representation by reducing the number of subscripts from three to two. All that matters is the class *j* and the number of remaining customers in each batch. Hence, now let n_{jl} be the number of class-j batches with l customers remaining to be served, i.e., let

$$n_{jl} = \sum_{k=l}^{m_j} n_{j,k,k-l+1} \ . \tag{5.10}$$

and let $\mathbf{n}$ be the associated state vector. Let $p_{jl}^c = \sum_{k=l}^{m_j} p_{jk}$. With this notation, we can use basic properties of the multinomial distribution to obtain the marginal steady-state distribution from (5.9).

Theorem 5.2. *For partial-batch blocking, the steady-state distribution of the vector* $\mathbf{n}$, *where* n_{jl} *is the number of class-j batches with l customers remaining to be served, has the product-form steady-state distribution*

$$\pi(\mathbf{n}) = g(S)^{-1} f(\mathbf{n}) \ , \quad \mathbf{n} \in S \ , \tag{5.11}$$

where

$$f(\mathbf{n}) = \prod_{j=1}^{r} \prod_{a=0}^{n_j - 1} \frac{\lambda_j(a)}{\mu_j} \prod_{l=1}^{m_j} \left[\frac{p_{jl}^c}{l} \right]^{n_{jl}} \Big/ n_{jl}! \tag{5.12}$$

with $g(S) = \sum_{\mathbf{n} \in S} f(\mathbf{n})$ *and*

$$S \equiv S_p = \left\{ \mathbf{n} : \sum_{j=1}^{r} \sum_{l=1}^{m_j} l a_{ij} n_{jl} \le K_i \ , \quad 1 \le i \le p \right\} . \tag{5.13}$$

In the case of Poisson arrivals for the batches, $\lambda_j(a)$ is independent of a, so that we can rewrite (5.12) as

$$f(\mathbf{n}) = \prod_{j=1}^{r} \left[\frac{\lambda_j}{\mu_j} \right]^{n_j} \prod_{l=1}^{m_j} \left[\frac{p_{jl}^c}{l} \right]^{n_{jl}} \Big/ n_{jl}! = \prod_{j=1}^{r} \prod_{l=1}^{m_j} \left[\frac{\lambda_j}{\mu_j} \frac{p_{jl}^c}{l} \right]^{n_{jl}} \Big/ n_{jl}! \tag{5.14}$$

Aside from the normalization, $f(\mathbf{n})$ in (5.14) can be represented as a truncated product of independent Poisson distributions. Thus, the steady-state probability distribution of the vector $(Q_1, \ldots, Q_r)$ where $Q_j \equiv \sum_{l=1}^{m_j} n_{jl}$ is the number of class-j customers can be written as a truncated and renormalized product of independent batch Poisson distributions. The mean of the class-j Poisson random variable is $(\lambda_j/\mu_j) \sum_{l=1}^{m_j} (p_{jl}^c/l)$ and the batch-size probability mass function is

$$\gamma_j(k) = (p_{jk}^c/k) / \sum_{l=1}^{m_j} (p_{jl}^c/l) \ , \quad 1 \le k \le m_j \ , \tag{5.15}$$

6. Generating Functions with Partial-Batch Blocking

We now focus on partial-batch blocking and construct the generating functions of the normalization constants $g(S_p) \equiv g(\mathbf{K})$ for $\mathbf{K} = (K_1, \ldots, K_p)$ using the compact representation in (5.10)– (5.13). At this point we are using the complete-sharing policy. By definition,

$$G(\mathbf{z}) = \sum_{K_1=0}^{\infty} \cdots \sum_{K_p=0}^{\infty} g(\mathbf{K}) z_1^{K_1} \ldots z_p^{K_p} \tag{6.1}$$

for a vector of complex variables $\mathbf{z} = (z_1, \ldots, z_p)$. As in [4,5], we obtain a more compact expression by changing the order of summation. For this purpose, let

$$\bar{K}_i = \sum_{j=1}^{r} \sum_{l=1}^{m_j} la_{ij} n_{jl} \ . \tag{6.2}$$

Then

$$G(\mathbf{z}) = \sum_{n_{11}=0}^{\infty} \cdots \sum_{n_{r,m_r}=0}^{\infty} \sum_{K_1=\bar{K}_1}^{\infty} \cdots \sum_{K_p=\bar{K}_p}^{\infty} \prod_{j=1}^{r} \prod_{a=0}^{n_j-1} \lambda_j(a) \prod_{l=1}^{m_j} \left[\frac{p_{jl}^c}{l\mu_j} \right]^{n_{jl}} \frac{1}{n_{jl}!} z_1^{K_1} \ldots z_p^{K_p}$$

$$= \frac{1}{\prod_{i=1}^{p} (1-z_i)} \sum_{n_{11}=0}^{\infty} \cdots \sum_{n_{r,m_r}=0}^{\infty} \prod_{j=1}^{r} \prod_{a=0}^{n_j-1} \lambda_j(a) \prod_{l=1}^{m_j} \left[\frac{p_{jl}^c}{l\mu_j} \right]^{n_{jl}} \frac{1}{n_{jl}!} \prod_{i=1}^{p} z_i^{la_{ij} n_{jl}}$$

$$= \frac{1}{\prod_{i=1}^{p} (1-z_i)} \prod_{j=1}^{r} \sum_{n_{j1}=0}^{\infty} \cdots \sum_{n_{j,m_j}=0}^{\infty} \prod_{a=0}^{n_j-1} \lambda_j(a) \prod_{l=1}^{m_j} \left[\frac{p_{jl}^c}{l\mu_j} \right]^{n_{jl}} \frac{1}{n_{jl}!} \prod_{i=1}^{p} z_i^{la_{ij} n_{jl}} \ . \tag{6.3}$$

From the final expression in (6.3), we see that the transform factors into r terms, one for each class. However, in general, the factors will have common z_i variables. When the state-dependent arrival rates have special structure, we can carry out the summation in (6.3).

First, for the special case in which $\lambda_j(k) = \lambda_j$ for all k and j (Poisson arrivals for all classes of batches), we get

$$G(\mathbf{z}) = \exp \left[\sum_{j=1}^{r} \left[\frac{\lambda_j}{\mu_j} \right] \sum_{l=1}^{m_j} \frac{p_{jl}^c}{l} \prod_{i=1}^{p} z_i^{la_{ij}} \right] \Big/ \prod_{i=1}^{p} (1-z_i) \ . \tag{6.4}$$

For the case of a single resource ($p = 1$), (6.4) agrees with (2.10) of Morrison [16]. Moreover, from (6.4) we can deduce that *the resource-sharing model with batch-Poisson arrivals has the same steady-state distribution as another resource sharing model with single (non-batch) arrivals.* In particular, (6.4) coincides with the generating function in (2.12) in [4] with p resources and $\sum_{j=1}^{r} m_j$ classes with ordinary Poisson arrivals; then class (j,l) has arrival rate $\lambda_j p_{jl}^c$, service rate $l\mu_j$ and requires la_{ij} resource units from resource i. This corresponds to the parallel-series network representation in Section 3 with each queue having its own arrival process and all departures leaving the system. The above is also clear from (5.14). In fact, (6.4) can be derived directly from (5.14) by treating (5.14) as a non-batch model and using (2.12) of [4].

For the special case in which $\lambda_j(k) = (N_j - k)\lambda_j'$ for all k and j (finite-source input for all classes of batches), we again can carry out the summation, and get

$$f(\mathbf{n}) = \prod_{j=1}^{r} \begin{bmatrix} N_j \\ n_j \end{bmatrix} \left[\frac{\lambda_j'}{\mu_j} \right]^{n_j} n_j! \prod_{l=1}^{m_j} \left[\frac{p_{jl}^c}{l} \right]^{n_{jl}} \frac{1}{n_{jl!}} \tag{6.5}$$

and

$$G(\mathbf{z}) = \left[\prod_{i=1}^{p} \frac{1}{1-z_i} \right] \prod_{j=1}^{r} \left[1 + \frac{\lambda_j'}{\mu_j} \sum_{l=1}^{m_j} \frac{p_{jl}^c}{l} \prod_{i=1}^{p} z_i^{la_{ij}} \right]^{N_j} . \tag{6.6}$$

We can rewrite (6.5) and (6.6) so that (6.5) is a product of multinomial probabilities if we divide by an appropriate constant. Then

$$f(\mathbf{n}) = \prod_{j=1}^{r} \begin{bmatrix} N_j \\ n_{j1}, \ldots, n_{j,m_j} \end{bmatrix} \prod_{l=1}^{m_j} q_{jl}^{n_{jl}} \tag{6.7}$$

and

$$G(\mathbf{z}) = \prod_{j=1}^{r} \left[1 - \sum_{l=1}^{m_j} q_{jl} + \sum_{l=1}^{m_j} q_{jl} \prod_{i=1}^{p} z_i^{la_{ij}} \right]^{N_j} \Big/ \prod_{i=1}^{p} (1-z_i) , \tag{6.8}$$

where $q_{jl} = x_{jl}/(1 + \sum_{k=1}^{m_j} x_{jk})$ with $x_{jl} = \lambda_j' p_{jl}^c / l\mu_j$. To get (6.7) and (6.8) from (6.5) and (6.6), we multiply by $\prod_{j=1}^{r} \prod_{l=1}^{m_j} (1-q_{jl})^{N_j}$.

Turning to the Pascal part of the BPP trilogy, we now consider positive linear state-dependent arrivals. Now let $\lambda_j(k) = \alpha_j + \beta_j k$ for positive parameters α_j and β_j. This leads to the negative multinomial or multivariate negative binomial distribution. (With infinite state space, we would need to assume that $\beta_j < \mu_j$ in order to have a proper steady-state distribution, but we can allow $\beta_j > \mu_j$ because we have a finite state space. This will reduce the radius of convergence of the generating function of the normalization constant.) In particular, (6.3) becomes

$$G(\mathbf{z}) = \left[\prod_{i=1}^{p} \frac{1}{1-z_i} \right] \sum_{n_{11}=0}^{\infty} \cdots \sum_{n_{r,m_r}=0}^{\infty} \prod_{j=1}^{r} \begin{bmatrix} r_j + n_j - 1 \\ r_j - 1,\ n_{j1}, \ldots, n_{r,m_r} \end{bmatrix} \times$$

$$\left[\frac{\beta_j}{\mu_j} \right]^{n_j} \left[\prod_{l=1}^{m_j} \frac{p_{jl}^c}{l} \prod_{i=1}^{p} z_i^{la_{ij}} \right]^{n_{jl}}$$

$$= \prod_{i=1}^{p} (1-z_i)^{-1} \prod_{j=1}^{r} \left[1 - \sum_{l=1}^{m_j} \frac{\beta_j p_{jl}^c}{\mu_j l} \prod_{i=1}^{p} z_i^{la_{ij}} \right]^{-r_j} . \tag{6.9}$$

for $r_j = \alpha_j / \beta_j$.

We remark that (6.6) can be regarded as a special case of (6.9) in which we allow β_j to be negative. Then $\alpha_j = N_j \lambda_j'$, $\beta_j = -\lambda_j'$ and $r_j = \alpha_j / \beta_j = -N_j$. However, the model with negative β_j does not make sense (leads to negative arrival rates) unless $\lambda_j (k) \equiv \alpha_j + \beta_j k$ equals 0 for some k, which occurs in the finite-source case.

We also can consider a mixed case in which for the batches some classes have finite-source inputs, others have Poisson inputs, and the remaining have positive linear state-dependent inputs. Instead of (6.4), (6.6) or (6.9), the generating function then has factors from each of these forms.

Finally, we note that when m_j is large (e.g., infinite), we can truncate the finite sums in (6.4), (6.6) and (6.9), because without loss of generality we can regard $p_{jl}^c = 0$ for l strictly greater than the maximum number of class-j customers that the system can support.

7. Geometric Batch-Size Distributions

Formulas (6.4), (6.6) and (6.9) are in the natural form when we directly specify p_{jk} as a finite probability mass function. However, simplifications can occur if we consider parametric batch-size distributions. When we treat parametric batch-size distributions, we often will want to allow infinite support; i.e., we allow $m_j = \infty$.

From (6.4), (6.6) and (6.9), we see that interest centers on the generating function

$$Q_j (z) \equiv \sum_{l=1}^{\infty} \frac{p_{jl}^c z^l}{l} \ . \tag{7.1}$$

Hence, the key is to express $Q_j (z)$ in closed form. We now show that this is easy to do when the pmf p_{jk} has a geometric tail.

Theorem 7.1. *Suppose that* $p_{jk} = A \gamma_j^{k-1}$, *for* $k \geq m$. *Then*

$$p_{jl}^c = \begin{cases} A(1-\gamma_j)^{-1} \gamma_j^{l-1} \ , & l \geq m \\ A(1-\gamma_j)^{-1} \gamma_j^{m-1} + p_{jl}' \ , & 1 \leq l \leq m-1 \end{cases} \tag{7.2}$$

for $p_{jl}' = \sum_{k=l}^{m-1} p_{jk}$, $1 \leq l \leq m-1$, *and*

$$Q_j (z) = \frac{A}{\gamma_j (1-\gamma_j)} \sum_{l=1}^{\infty} \frac{(\gamma_j z)^l}{l} + \sum_{l=1}^{m-1} \frac{p_{jl}' z^l}{l} + \frac{A}{\gamma_j (1-\gamma_j)} \sum_{l=1}^{m-1} (\gamma_j^m - \gamma_j^l) \frac{z^l}{l}$$

$$= \frac{-A \ln (1-\gamma_j z)}{\gamma_j (1-\gamma_j)} + \sum_{l=1}^{m-1} \frac{p_{jl}' z^l}{l} + \frac{A}{\gamma_j (1-\gamma_j)} \sum_{l=1}^{m-1} (\gamma_j^m - \gamma_j^l) z^l \ . \tag{7.3}$$

Of course, Formula (7.3) is of no help when the cutoff point m in (7.2) is greater than we can achieve with direct truncation of p_{jl}^c for large l. In the special case of a pure geometric distribution, $m = 1$ and $A = (1-\gamma_j)$ so that (7.3) becomes

$$Q_j(z) = -\ln(1-\gamma_j z)/\gamma_j \ . \tag{7.4}$$

More generally, we can express $Q_j(z)$ in terms of the generating function of p_{jk}, i.e.,

$$P_j(z) \equiv \sum_{k=1}^{\infty} p_{jk} z^k \ . \tag{7.5}$$

The following result was also derived in Section 2 of Morrison [16].

Theorem 7.2. *The generating function $Q_j(z)$ in (7.1) can be represented as*

$$Q_j(z) = \int_0^z [(1-P_j(u))/(1-u)]\, du \ . \tag{7.6}$$

Proof. The integrand $(1-P_j(z))/(1-z)$ is well known to be the generating function of $p^c_{j,k+1}$; see p. 265 of Feller [9]. Then

$$\int_0^z \sum_{k=0}^{\infty} p^c_{j,k+1} u^k du = \sum_{k=0}^{\infty} \frac{p^c_{j,k+1} z^{k+1}}{k+1} = Q_j(z) \ . \quad \blacksquare$$

We can easily deduce (7.4) from Theorem 7.2 too. We now apply (7.4) with (6.4), (6.6) and (6.9) to yield closed-form formulas in the three BPP cases when the batch-size distributions are geometric. This yields the generating function for the multi-resource version of the model considered in [11,21] in the Poisson case.

Corollary 1. *For the multi-class resource-sharing model with Poisson arrivals of batches for each class, if $p_{jk} = (1-\gamma_j)\gamma_j^{k-1}$, $k \geq 1$, for all j, then (6.4) becomes*

$$G(\mathbf{z}) = \prod_{i=1}^{p} (1-z_i)^{-1} \prod_{j=1}^{r} (1-\gamma_j \prod_{i=1}^{p} z_i^{a_{ij}})^{-(\lambda_j/\mu_j\gamma_j)} \ . \tag{7.7}$$

As noted by van Doorn and Panken [21] and Kaufman and Rege [11], (7.7) has the same form as the Pascal case with single arrivals ($m_j = 1$) in (6.9), with different parameters.

Corollary 2. *For the multi-class resource-sharing model with finite-source input for batches, if $p_{jk} = (1-\gamma_j)\gamma_j^{k-1}$, $k \geq 1$, for all j, then (6.6) becomes*

$$G(\mathbf{z}) = \prod_{i=1}^{p} (1-z_i)^{-1} \prod_{j=1}^{r} (1 - \frac{\lambda'_j}{\mu_j\gamma_j} \ln(1-\gamma_j \prod_{i=1}^{p} z_i^{a_{ij}}))^{N_j} \ . \tag{7.8}$$

Corollary 3. *For the multi-class resource-sharing model with Pascal input for batches, if $p_{jk} = (1-\gamma_j)\gamma_j^{k-1}$, $k \geq 1$, for all j, then (6.9) becomes*

$$G(\mathbf{z}) = \prod_{i=1}^{p} (1-z_i)^{-1} \prod_{j=1}^{r} [1 + \frac{\beta_j}{\mu_j\gamma_j} \ln(1-\gamma_j \prod_{i=1}^{p} z_i^{a_{ij}})]^{-r_j} \ . \tag{7.9}$$

For the inversions algorithms, Corollaries 1–3 are significant because they produce algorithms just as fast as for the non-batch case. In contrast, for a general batch distribution, computing $G(z)$ would require $\sum_{j=1}^{r} m_j$ terms to compute.

8. Blocking Probabilities

It is important to distinguish between *call blocking* and *time blocking.* Call blocking refers to the blocking experienced by arrivals (which depends on the state at arrival epochs), while time congestion refers to the blocking that would take place at an arbitrary time if there were an arrival at that time (as in the virtual waiting time). With Poisson arrivals, the two probability distributions agree, but not more generally; see [15]. Since the steady-state distribution π refers to an arbitrary time, blocking probabilities computed directly from it are time blocking, but we show how to treat both time blocking and call blocking.

For interpretation of the results with batches, it is also useful to identify and calculate two separate components of the blocking probabilities: First, there is the *conditional blocking probability* for a class-j customer given that this customer is the k^{th} customer in his batch and, second, there is the probability that an arbitrary customer is the k^{th} customer in his batch. (This customer is blocked if and only if $k-1$ or fewer from his batch can be admitted assuming partial-batch blocking.) By the familiar length-biasing argument, the probability that an arbitrary class-j customer is the k^{th} customer in his batch is

$$\nu_{jk} \equiv p_{jk}^c / \sum_{k=1}^{m_j} p_{jk}^c \ . \tag{8.1}$$

The possible values of k are the positive integers up to the maximum possible batch size, m_j. The probability distribution in (8.1) is proper if the pmf p_{jk} has a finite mean. We assume that this is the case.

Given that a class-j customer is k^{th} in his batch, the conditional probability that this customer would not be admitted at an arbitrary time (time blocking) is easily seen to be

$$B_{jk}^t = 1 - \frac{g(\mathbf{K} - k\mathbf{a}_j)}{g(\mathbf{K})} \ , \tag{8.2}$$

where $\mathbf{a}_j \equiv (a_{1j}, \ldots, a_{pj})$ is the requirements vector for class i, just as in (2.11) of [4].

Hence, the overall unconditional blocking probability for a class j arrival at an arbitrary time (time blocking) is

$$B_j^t = \sum_{k=1}^{m_j} \nu_{jk} B_{jk}^t = 1 - \sum_{k=1}^{m_j} \nu_{jk} \frac{g(\mathbf{K} - k\mathbf{a}_j)}{g(\mathbf{K})} \ . \tag{8.3}$$

As noted above, if the class-j batches arrive in a Poisson process, then (8.2) and (8.3) also yield the call blocking, but not more generally. However, the call blocking always can be obtained by calculating the time blocking in a modified model, as we indicated in Section 4 of [5]. In particular, let B_{jk} be the conditional blocking probability for class j given that the class-j customer is k^{th} in his batch, and let B_j be the unconditional blocking probability (call blocking). Let $\mathbf{A} \equiv (a_{ij})$ be the requirements matrix. In general,

$$B_{jk} \;=\; 1 \;-\; \frac{\sum\limits_{\mathbf{n}:\mathbf{An}\,\le\,\mathbf{K}-k\mathbf{a}_j} \lambda_j(n_j)\,\pi(\mathbf{n})}{\sum\limits_{\mathbf{n}:\mathbf{An}\,\le\,\mathbf{K}} \lambda_j(n_j)\,\pi(\mathbf{n})} \;=\; 1 \;-\; \frac{\sum\limits_{\mathbf{n}:\mathbf{An}\,\le\,\mathbf{K}-k\mathbf{a}_j} \lambda_j(n_j)\,f(\mathbf{n})}{\sum\limits_{\mathbf{n}:\mathbf{An}\,\le\,\mathbf{K}} \lambda_j(n_j)\,f(\mathbf{n})} \tag{8.4}$$

and, paralleling (8.3),

$$B_j \;=\; \sum_{k=1}^{m_j} \nu_{jk} B_{jk} \;. \tag{8.5}$$

However, we can rewrite $\lambda_j(n_j)\,f(\mathbf{n})$ as $\lambda_j(0)\,\overline{f}(\mathbf{n})$ and thus (8.4) as

$$B_{jk} \;=\; 1 \;-\; \frac{\sum\limits_{\mathbf{n}:\mathbf{An}\,\le\,\mathbf{K}-k\mathbf{a}_j} \overline{f}(\mathbf{n})}{\sum\limits_{\mathbf{n}:\mathbf{An}\,\le\,\mathbf{K}} \overline{f}(\mathbf{n})} \;=\; 1 - \frac{\overline{g}(\mathbf{K}-k\mathbf{a}_j)}{\overline{g}(\mathbf{K})} \;\;, \tag{8.6}$$

where $\overline{f}(\mathbf{n})$ is the analog of $f(\mathbf{n})$ with $\lambda_j(m)$ replaced by $\overline{\lambda}_j(m) \equiv \lambda_j(m+1)$, and $\overline{g}(\mathbf{K})$ is the analog of $g(\mathbf{K})$ with $f(\mathbf{n})$ replaced by $\overline{f}(\mathbf{n})$. We summarize this result as follows.

Theorem 8.1. *The class-j blocking probabilities B_{jk} in (8.4) and B_j in (8.5) coincide with the time-blocking quantities B_{jk}^t in (8.2) and B_j^t in (8.3) for the modified model in which the class-j batch arrival-rate function is changed from $\lambda_j(n)$ to $\overline{\lambda}_j(n) \equiv \lambda_j(n+1)$.*

For the special case in which $\lambda_j(n) \;=\; \alpha_j \;+\; \beta_j n$,

$$\overline{\lambda}_j(n) \;=\; \lambda_j(n+1) \;=\; \alpha_j \;+\; \beta_j(n+1) \;=\; (\alpha_j \;+\; \beta_j) \;+\; \beta_j n \;, \tag{8.7}$$

so that the modified model is a model of the same general form. For the BPP model (without batches), this approach to computing call congestion was pointed out by Dziong and Roberts [8], p. 273. Note that $\overline{\lambda}_j$ coincides with λ_j when there are Poisson arrivals and $\overline{\lambda}_j$ reduces to the arrival rate with one less class-*j* source when class *j* has a finite source input, agreeing with well known properties.

From (8.2)–(8.6), we see that B_{jk}^t and B_{jk} can be computed by calculating just two normalization constant values, while B_j^t and B_j can be calculated from $m_j + 1$ normalization constant values. As indicated in Section 5.2 of [4], when $\mathbf{K}$ is large and m_j is small, the computation of $g(\mathbf{K}-k\mathbf{a}_j)$ by numerical inversion for many k and j can be made much more efficient by shared computation. However, if m_j is large (e.g., infinite), then this approach would not be possible since $\mathbf{K}-k\mathbf{a}_j$ will vary greatly.

However, it is also possible to calculate the overall blocking probabilities B_j^t by performing only *two* inversions. To see this, let $V_j(z) \;\equiv\; \sum_{k=1}^{m_j} \nu_{jk} z^k$ and

$$h_j(\mathbf{K}) \;=\; \sum_{k=1}^{m_j} \nu_{jk}\, g(\mathbf{K}-k\mathbf{a}_j) \;\;, \tag{8.8}$$

where $g(\mathbf{K}) \;=\; 0$ if $K_i \;<\; 0$ for any i, and note that

$$H_j(\mathbf{z}) \equiv \sum_{K_1=0}^{\infty} \ldots \sum_{K_p=0}^{\infty} h_j(\mathbf{K}) z_1^{K_1} \ldots z_p^{K_p}$$

$$= \sum_{k=1}^{m_j} \nu_{jk} z_1^{ka_{1j}} \ldots z_p^{ka_{pj}} \sum_{K_1=ka_{1j}}^{\infty} \ldots \sum_{K_p=ka_{pj}}^{\infty} g(\mathbf{K}-k\mathbf{a}_j) z^{K_1-ka_{1j}} \ldots z_p^{K_p-ka_{pj}}$$

$$= V_j\Big(\prod_{i=1}^{p} z_i^{a_{ij}}\Big) G(\mathbf{z}) \ . \tag{8.9}$$

Hence,

$$B_j = 1 - h_j(\mathbf{K})/g(\mathbf{K}) \ , \tag{8.10}$$

where $h_j(\mathbf{K})$ is the coefficient of $z_1^{K_1} \ldots z_p^{K_p}$ of the generating function in (8.9).

For an arbitrary batch-size distribution with finite support, the computational complexity in computing $v_j(\prod_{i=1}^{p} z_i^{a_{ij}})$ is $O(m_j)$, which is the same as that for computing $G(\mathbf{z})$ as is clear from (6.4), (6.6) and (6.9). So the computational complexity in evaluating $h_j(K)$ and $g(K)$ are of the same order. With geometric batch-size distributions, we get a simple closed form for $V_j(\prod_{i=1}^{p} z_i^{a_{ij}})$ as shown below. If $p_{jk} = (1-\gamma_j)\gamma_j^{k-1}$ for $k \geq 1$, then $\nu_{jk} = (1-\gamma_j)\gamma_j^{k-1}$, $k \geq 1$, and

$$V_j\Big(\prod_{i=1}^{p} z_i^{a_{ij}}\Big) = \frac{(1-\gamma_j) \prod_{i=1}^{p} z_i^{a_{ij}}}{1-\gamma_j \prod_{i=1}^{p} z_i^{a_{ij}}} \ . \tag{8.11}$$

To understand performance, we believe it is useful to calculate both ν_{jk} and B_{jk} as well as $\nu_{jk} B_{jk}$, $1 \leq k \leq \eta_j$, and the overall class-*j* blocking probability B_j. However, if we are only interested in the overall blocking probability B_j, then it is possible to get it more directly, as noted above. As noted by Kaufman and Rege [11] and Morrison [16], in the case of Poisson arrivals of class-*j* batches, we can apply Little's law to obtain

$$E(Q_j) = \lambda_j \gamma_j (1-B_j)/\mu_j \ , \tag{8.12}$$

where Q_j is the steady-state number of class-*j* customers and γ_j is the mean class-*j* batch size. Indeed, this argument shows that calculating the blocking probability B_j is equivalent to calculating the mean EQ_j whenever the overall arrival rate is known.

9. Partial-Batch Blocking with Other Sharing Policies

The approach we have taken makes it possible to immediately obtain the steady-state distribution for the upper limit (UL) and guaranteed minimum (GM) sharing policies with each of the three BPP form of batch arrivals. (See [4,5] for background on UL and GM). To illustrate, we give the generating functions for Poisson arrivals of batches. The other two BPP cases can be treated similarly.

For the UL policy, we introduce upper limits M_j for class j. The state space for UL generalizes (5.13) the way (2.14) of [4] generalizes (2.10) of [4], namely,

$$S_{UL}(\mathbf{K,M}) = \{\mathbf{n}: \sum_{j=1}^{r} \sum_{l=1}^{m_j} l a_{ij} n_{jl} \le K_i \ , \ \ 1 \le i \le p \ ; \ \sum_{l=1}^{m_j} l n_{jl} \le M_j \ , \ \ 1 \le j \le r\} \ . \quad (9.1)$$

Let the generating function depend on $\mathbf{y} \equiv (y_1, \ldots, y_r)$ as well as $\mathbf{z} = (z_1, \ldots, z_p)$, as in [4,5]. For UL, the generating function is

$$G(\mathbf{z},\mathbf{y}) \equiv \sum_{K_1=0}^{\infty} \cdots \sum_{K_p=0}^{\infty} \sum_{M_1=0}^{\infty} \cdots \sum_{M_r=0}^{\infty} g(K,M) z_1^{K_1} \ldots z_p^{K_p} y_1^{M_1}, \ldots y_r^{M_r}$$

$$= \exp\left[\sum_{j=1}^{r} \frac{\lambda_j}{\mu_j} y_j \sum_{l=1}^{m_j} \frac{p_{jl}^c}{l} \prod_{i=1}^{p} z_i^{l a_{ij}}\right] \Big/ \prod_{i=1}^{p} (1-z_i) \prod_{j=1}^{r} (1-y_j) \ . \quad (9.2)$$

Similarly, we can obtain a closed-form expression for the GM policy. As in [4] we assume that the requirement matrix A has a special form. In particular $a_{ij} = b_j$ or 0 for all i and j. Let $\delta_{ij} = 1$ if $a_{ij} > 0$ and $\delta_{ij} = 0$ otherwise. Paralleling (2.18) of [4], the state space for GM is

$$S_{GM}(\mathbf{K,M}) = \{\mathbf{n}: \sum_{j=1}^{r} \sum_{l=1}^{m_j} \max\{l a_{ij} n_{jl} \ , \ \delta_{ij} M_j\} \le K_i \ , \ \ 1 \le i \le p\} \ . \quad (9.3)$$

Then the generating function is

$$G(\mathbf{z},\mathbf{y}) = \prod_{i=1}^{p} \left[\frac{1}{1-z_i}\right] \times$$

$$\prod_{j=1}^{r} \left\{ \frac{\exp\left[\frac{\lambda_j}{\mu_j} \sum_{l=1}^{m_j} \frac{p_{jl}^c}{l} \prod_{i=1}^{p} z_i^{l b_j \delta_{ij}} - y_j \exp\left[\frac{\lambda_j}{\mu_j} \sum_{l=1}^{m_j} \frac{p_{jl}^c}{l} y_j^{b_j} \prod_{i=1}^{p} z_i^{\delta_{ij} l b_j}\right]\right.}{1-y_j} \right.$$

$$\left. + \ y_j \frac{\prod_{i=1}^{p} z_i^{\delta_{ij}} \exp\left[\frac{\lambda_j y_j}{\mu_j} \sum_{l=1}^{m_j} \frac{p_{jl}^c}{l} \prod_{i=1}^{p} z_i^{\delta_{ij} l b_j}\right]}{1-y_j \prod_{i=1}^{p} z_i^{\delta_{ij}}} \right\} \ . \quad (9.4)$$

As in [5], we can also treat the combined UL and GM sharing policy.

10. The Infinite-Server Queue with Batch Poisson Arrivals

A special case of the model we have considered is the model with *no* capacity limits, i.e., $K_i = \infty$ for all i. Then Theorem 5.2 holds with the state spaces $S_P = S_C$ being the set of nonnegative vectors $\mathbf{n} \equiv (n_{jl})$ with $l \le m_j$, $1 \le j \le r$. In the case of Poisson arrivals for the batches, the entire model reduces to r independent $M^X/M/\infty$ queues, one for each class, where M^X denotes a batch Poisson arrival process.

In order to understand the need for exponential service times in the general resource-sharing model with batch arrivals, it is thus natural to consider the $M^X/G/\infty$ model, which has i.i.d. service times with a general service-time distribution. The question is whether insensitivity holds: Does the steady-state distribution of the number of busy servers depend on the service-time distribution only through its mean? Even if this is not always true, could this be true for some special batch-size distributions?

Let Q be the steady-state number of busy servers in the $M^X/G/\infty$ model. Then, from Shanbhag [18] or p. 677 of Liu, Kashyap and Templeton [14],

$$Ez^Q = \exp\{-\lambda \int_0^\infty [1-P(1-(1-z)H^c(x))]\,dx\}\ , \tag{10.1}$$

where λ is the Poisson rate, $P(z) \equiv \sum_{k=1}^{\infty} p_k z^k$ is the batch-size generating function, $H(x)$ is the service-time cdf and $H^c(x) \equiv 1-H(x)$. Moreover, the mean and variance are

$$EQ = \lambda P'(1) \int_0^\infty H^c(y)\,dy \quad \text{and} \quad \mathrm{Var}\ Q = EQ + \lambda P''(1) \int_0^\infty H^c(y)^2\,dy\ . \tag{10.2}$$

Since $P'(1)$ is the mean batch size and $\int_0^\infty H^c(y)\,dy$ is the mean service time, the steady-state mean EQ depends on the batch-size distribution and the service-time cdf H only through their means. However, from (10.2) we see that the variance Var Q does not. Provided that $P''(1) \ne 0$, Var Q depends on H beyond its mean. Since $P''(1) = 0$ if and only if $p_1 = 1$, Var Q depends on H beyond its mean for all nondegenerate batch-size distributions. (We recover the well known insensitivity in the $M/G/\infty$ model when there are no batches.) Similarly, no matter what service-time distribution is used, Var Q depends on the batch-size distribution beyond its mean.

We close this section by noting that with a geometric batch-size distribution, i.e., $p_k = (1-p)p^{k-1}$, $k \ge 1$,

$$P(z) = (1-p)z/(1-pz) \tag{10.3}$$

and

$$Ez^Q = \exp(-\lambda \int_0^\infty [(1-z)H^c(x)/((1-p) + p(1-z)H^c(x))\,dx)\ . \tag{10.4}$$

In the case of exponential service times, (10.4) simplifies because

$$\frac{e^{-\mu x}}{A + Be^{-\mu x}} = \frac{-1}{\mu B}\ \frac{d}{dx} \ln\,(A + Be^{-\mu x})\ .$$

Then

$$Ez^{Q} \;=\; [(1-p)/(1-pz)]^{\lambda/\mu p} \;, \tag{10.5}$$

which is the generating function of the negative binomial distribution, consistent with the geometric batch case in (7.7).

11. Recursive Algorithms from the Generating Functions

For actually calculating the normalization constants, we propose the numerical inversion algorithm in [3–5]. However, sometimes recursive algorithms are also attractive. In Section 11 of [5] we showed that the generating functions can be used to derive recursions for the normalization constants. This method for obtaining recursions was first proposed by Reiser and Kobayashi [17] for closed queueing networks, but it has not been widely used since. As noted in [17], the starting point for the recursions is the fact that the coefficients of a generating function that is a product of two generating functions is the convolution of the coefficients from the two component generating functions. This remains true with vectors using multidimensional convolution; see Lemma 11.1 of [5].

It is straightforward to apply convolution to our generating functions because they are expressed directly as products of generating functions, although in general the computational complexity is quite high because the number of factors involving $\mathbf{z}$ is $r + 1$. (Note that $\prod_{i=1}^{p} (1-z_i)^{-1}$ can be regarded as a single factor.)

As indicated in Section 11 of [5], once we remove the factor $\prod_{i=1}^{p} (1-z_i)^{-1}$, the coefficients of the remaining generating function are often easy to determine.

We now complement [5] by treating the other two BPP cases in (6.6) and (6.9) and the Poisson case with geometric batches in (7.7). The following theorem applies to each of the r factors.

Theorem 11.1. *If*

$$G(\mathbf{z}) \;=\; (1 \;+\; \rho \sum_{l=1}^{m_j} b_l \prod_{i=1}^{p} z_i^{l a_i})^{c} \tag{11.1}$$

for constants c, ρ and b_l and nonnegative integers l and a_i, as in (6.6) and (6.9), then

$$g(\mathbf{K}) \;=\; \rho \sum_{l=1}^{m_j} b_l g(\mathbf{K}-l\mathbf{a}) \;+\; c\rho a_i K_i^{-1} \sum_{l=1}^{m_j} b_l l g(\mathbf{K}-l\mathbf{a}) \;,\; 1 \le i \le p \;, \tag{11.2}$$

for $\mathbf{a} \;=\; (a_1 , \ldots , a_p)$.

Proof. Let $G^{(i)}(\mathbf{z})$ be the partial derivative of $G(\mathbf{z})$ with respect to z_i. Differentiate the generating function with respect to z_i and get

$$G^{(i)}(\mathbf{z}) \;=\; r(1 \;+\; \rho \sum_{l=1}^{m_j} b_l \prod_{i=1}^{p} z_i^{l a_i})^{(c-1)} \rho \sum_{l=1}^{m_j} b_l l a_i \prod_{i=1}^{p} z_i^{l a_i} z_i^{-1} \;,$$

so that

$$G^{(i)}(\mathbf{z})(1+\rho \sum_{l=1}^{m_j} b_l \prod_{i=1}^{p} z_i^{la_i}) = c\rho G(\mathbf{z}) \sum_{l=1}^{m_j} b_l la_i \prod_{i=1}^{p} z_i^{la_i} z_i^{-1} . \tag{11.13}$$

On the other hand, note that

$$G^{(i)}(\mathbf{z}) = \sum_{K_1=0}^{\infty} \cdots \sum_{K_p=0}^{\infty} K_i g(\mathbf{K}) z_1^{K_1} \ldots z_p^{K_p} z_i^{-1} . \tag{11.4}$$

Match the coefficients of $z_1^{K_1} \ldots z_i^{K_i - 1} \ldots z_p^{K_p}$ in the two representations of $G^{(i)}(\mathbf{z})$, (11.3) and (11.4). ■

12. Computational Complexity and Comparison with Previous Algorithms

For simplicity, we assume that $m_j = m$ for all j. Comparing the generating function expressions with those in the non-batch case in [5], we observe that computation of each generating function value requires m times as many operations as in the non-batch case. Hence, the overall computational complexity is m times as much as in the non-batch case. However, in the special case of geometric batch-size distributions, the factor m is not there, so that the computational complexity is the same as in the non-batch case.

Let C_{CS} and C_{UG} represent the computational complexity with the complete-sharing policy and the combined upper-limit and guaranteed-minimum policy, respectively. Then, following Section 9 of [5], we get the following expressions: With general batch-size distribution and either Poisson or BPP arrivals,

$$C_{CS} = O(\overline{r} m \overline{K}^{\overline{p}}) \tag{12.1}$$

and

$$C_{UG} = O(\overline{r} m M \overline{K}^{\overline{p}}) , \tag{12.2}$$

where M is the upper limit parameter, assumed for simplicity to be the same for all classes. With judicious truncation in the inversion formula, $\overline{K} \le K$ and $\overline{K} = O(\sqrt{K})$ for large K. With special structure allowing dimension reduction, $\overline{p} < p$. With multiplicity (i.e., many classes having the same parameter), $\overline{r} < r$.

With geometric batch-size distribution and either Poisson or BPP arrivals,

$$C_{CS} = (O(\overline{r} \overline{K}^{\overline{p}}) \tag{12.3}$$

and

$$C_{UG} = O(\overline{r} M \overline{K}^{\overline{p}}) . \tag{12.4}$$

The previous recursive algorithms in [11,16,21] are for the special case of $p = 1$, Poisson arrivals, and complete sharing policy. The computational complexities are as follows: With a general batch-size distribution,

$$C_{CS} = O(rmK) \tag{12.5}$$

and, with a geometric batch-size distribution,

$$C_{CS} = O(rK^2) \ . \tag{12.6}$$

Comparing (12.5) and (12.1) with $\bar{p} = 1$, we again see that inversion is faster for large K since in that case $\bar{K} = O(\sqrt{K})$. Comparing (12.6) and (12.3) with $\bar{p} = 1$ we see that inversion is much faster for large K.

Finally, we can also compare the inversion against the new recursive algorithm obtained by combining Theorem 11.1 with Section 11 of [5]. For complete sharing and BPP arrivals, the computational complexity for a general batch-size distribution is

$$C_{CS} = O(rmK^{2p}) \ . \tag{12.7}$$

Comparing (12.7) with (12.1), we again see that the inversion is much faster than the recursion.

13. Scaling for the Inversion Algorithm

The inversion algorithm is as in [3,4,5]. As mentioned there, an important ingredient is scaling. We now discuss scaling in the special case of $p = 1$. In [4,5] we explain how to extend the scaling from $p = 1$ to general p.

The generating functions in (6.4), (6.6) and (6.9) have the following special structure:

$$G(z) = (1-z)^{-1} \prod_{j=1}^{r} G_j(z) \ , \tag{13.1}$$

where $G_j(z)$ differs from case to case. We invert a scaled generating function

$$\bar{G}(z) = \alpha_0 G(\alpha z) \ , \tag{13.2}$$

where α_0 and α are chosen to control the inversion error and avoid numerical underflow or overflow. Based on the scaling considered in many cases in [4,5]. We developed a heuristic scaling in Section 7.3 of [5]. We use this scaling here as well. We let α be the largest number in the interval $(0,1]$ such that

$$\sum_{j=1}^{r} \frac{zG_j'(z)}{G_j(z)} \Big|_{z = \alpha} \le K \ , \tag{13.2}$$

where $G_j'(z) = \frac{d}{dz} G_j(z)$. Also

$$\alpha_0^{-1} = \prod_{j=1}^{r} G_j(\alpha) \ . \tag{13.3}$$

To illustrate, we show what the scaling becomes for the special case of Poisson arrivals and geometric batch sizes. From (7.7), we see that

$$G_j(z) = (1-\gamma_j z)^{-(\lambda_j/\mu_j\gamma_j)} \ , \tag{13.4}$$

so that

$$G_j'(z) = (\lambda_j/\mu_j)(1-\gamma_j z)^{-(\lambda_j/\mu_j\gamma_j)-1} . \tag{13.5}$$

Hence, (13.2) becomes

$$\sum_{j=1}^{r} \frac{\alpha\lambda_j/\mu_j}{1-\alpha\gamma_j} \le K . \tag{13.6}$$

From (13.3) we get,

$$\alpha_0 = \prod_{j=1}^{r} (1-\gamma_j\alpha)^{(\lambda_j/\mu_j\gamma_j)} . \tag{13.7}$$

14. Numerical Examples

We consider only a single resource. We could have considered multiple resources as in [4,5], but not too many unless we can exploit dimension reduction. Dimension reduction applies with batch arrivals just as before.

Our first example has two classes. Each class-1 arrival requires 1 resource unit, while each class-2 arrival requires 12 resource units. Both classes have Poisson arrivals of batches with geometric batch-size distributions. The batch-size parameters are $\gamma_1 = 0.2$ and $\gamma_2 = 0.5$; see (7.4). The remaining parameters and some computational results are displayed in Table 1. The offered load is $\rho_j = \lambda_j/\mu_j$. The generating function is given by (7.7) with $p = 1$ and $r = 2$.

capacity K, number of resource units	offered loads		1st member blocking		arbitrary member blocking (class 1 only)	
	ρ_1	ρ_2	class 1	class 2	inversion	asymptotic
60	10	2	.02515320	.25556360	.03137566	.02582764
600	100	20	.00733499	.08839576	.00917112	.00919358
6000	1000	200	.00259070	.03103518	.00323814	.00323897
60,000	10,000	2000	.00103268	.01236362	.00129072	—
600,000	100,000	20,000	.00056141	.00672104	.00070169	—

Table 1. Numerical results for the first example with two classes, each having Poisson arrival of batches and geometric batch-size distributions.

Computational results are shown for the first member of a batch for both classes, which coincides with time blocking since the batch arrival processes are Poisson. Computational results are also shown for the blocking probability of an arbitrary class-1 arrival as well, and this is compared to the uniform asymptotic approximation (UAA) developed by Morrison [16].

We consider five different resource capacities, ranging from fairly small ($K = 60$) to large ($K = 600,000$). For the first two rows (cases), comparison was made with the recursive algorithms of [11] and [21] and the results match to all displayed digits. For the other rows, the recursions are too slow, and thus were not done. Our algorithm with truncation works in seconds even for the last row.

For the arbitrary class-1 arrival, comparisons with UAA confirmed the accuracy of the inversion algorithm in third row. The UAA is also very fast, but it had a numerical overflow problem in the last two rows (which could be addressed with more careful implementation). As usual [4,5,6], we also confirm our numerical accuracy by running the algorithm with two different values of the inversion parameter l. Here we used $l = 1$ and $l = 2$.

Our second example is a modification of the first example in which the two batch arrival processes are made Pascal arrival processes with peakedness parameter $z = 2$ instead of Poisson processes (with peakedness $z = 1$); see Section 3 of [5]. There are no previous algorithms covering this example. We verify our accuracy in this case only by doing the inversion twice, once with $l = 1$ and once with $l = 2$. (We could also have used the new recursion derived in Section 11, but we did not.) The model parameters and the computational results are displayed in Table 2. As with the first example, the inversion algorithm runs in seconds for all cases.

The blocking probabilities in Table 2 are clearly higher than they are for the corresponding cases in Table 1, reflecting the fact that the variability (as measured by the peakedness) has increased.

capacity K, number of resource units	offered loads		1st member blocking		arbitrary member blocking	
	ρ_1	ρ_2	class 1	class 2	class 1	class 2
60	10	2	.01882023	.29498320	.02359333	.50702647
600	100	20	.01199065	.14662554	.01495709	.26248041
6000	1000	200	.01022163	.11733099	.01274590	.21077849
60,000	10,000	2000	.00996266	.11335132	.01242253	.20369979
600,000	100,000	20,000	.00993494	.11292966	.01238792	.20294905

Table 2. Numerical results for the second example with two classes, each having a Pascal arrival processes with peakedness parameter $z_j = 2$ and a geometric batch-size distribution.

References

[1] J. Abate and W. Whitt, The Fourier-Series Method for Inverting Transforms of Probability Distributions. *Queueing Systems* 10 (1992) 5-88.

[2] J. Abate and W. Whitt, Numerical Inversion of Probability Generating Functions. *Oper. Res. Letters* 12 (1992) 245-251.

[3] G. L. Choudhury, K. K. Leung and W. Whitt, Calculating Normalization Constants of Closed Queueing Networks by Numerically Inverting Their Generating Functions. *J. ACM*, to appear.

[4] G. L. Choudhury, K. K. Leung and W. Whitt, An Algorithm to Compute Blocking Probabilities in Multi-Rate Multi-Class Multi-Resource Loss Models. *Adv. Appl. Prob.* 27 (1995) to appear.

[5] G. L. Choudhury, K. K. Leung and W. Whitt, An Inversion Algorithm To Calculate Blocking Probabilities in Loss Networks with State-Dependent Rates, submitted for publication.

[6] G. L. Choudhury, D. M. Lucantoni and W. Whitt, Multidimensional Transform Inversion with Applications to the Transient M/G/1 Queue, *Ann. Appl. Prob.* 4 (1994) 719-740.

[7] L. E. N. Delbrouck, On the Steady-State Distribution in a Service Facility with Different Peakedness Factors and Capacity Requirements. *IEEE Trans. Commun.*, COM-31 (1983) 1209-1211.

[8] Z. Dziong and J. W. Roberts, Congestion Probabilities in a Circuit-Switched Integrated Services Network. *Perf. Eval.* 7 (1987) 267-284.

[9] W. Feller, *An Introduction to Probability Theory and its Applications*, Vol. I, third edition, Wiley, New York.

[10] J. R. Jackson, Jobshop-like Queueing Systems. *Management Science* 10 (1963) 131-142.

[11] J. S. Kaufman and K. M. Rege, Blocking in a Shared Resource Environment with Batched Poisson Arrival Processes. AT&T Bell Laboratories, Holmdel, NJ, 1992.

[12] F. P. Kelly, Loss Networks. *Ann. Appl. Prob.* 1 (1991) 319-378.

[13] A. E. Krzesinski, Multiclass Queueing Networks with State-Dependent Routing. *Perf. Eval.* 7 (1987) 125-143.

[14] L. Liu, B. R. K. Kashyap and J. G. C. Templeton, On the $\mathrm{GI}^{\mathrm{X}}/\mathrm{G}/\infty$ System. *J. Appl. Prob.* 27 (1990) 671-683.

[15] B. Melamed and W. Whitt, On Arrivals That See Time Averages. *Operations Res.* 38 (1990) 156-172.

[16] J. A. Morrison, Blocking Probabilities for Multiple Class Batched Poisson Arrivals to a Shared Resource. *Perf. Eval.*, to appear.

[17] M. Reiser and H. Kobayashi, Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms. *IBM J. Res. Dev.* 19 (1975) 283-294.

[18] D. N. Shanbhag, On Infinite Server Queues with Batch Arrivals. *J. Appl. Prob.* 3 (1966) 274-279.

[19] D. Towsley, Queueing Network Models with State-Dependent Routing. *J. ACM* 27 (1986) 323-337.

[20] N. M. van Dijk, *Queueing Networks and Product Forms,* Wiley, New York, 1993.

[21] E. A. van Doorn and F. J. M. Panken, Blocking Probabilities in a Loss System with Arrivals in Geometrically Distributed Batches and Heterogeneous Service Requirements. *ACM/IEEE Trans. Networking,* 1 (1993) 664-667.

CONTENTS