A Finite-Source Multiserver Queue with Preemptive Priorities

by

Alexandre Brandwajn

University of California
Santa Cruz

UCSC-CRL-96-19

Abstract

We consider systems with finite sources of requests, multiple servers, and an arbitrary number of preemptive priority classes. The service times and the source idle times are assumed to be exponentially distributed. We present an efficient approach based on marginal state description. The approach allows a recurrent analysis of priority levels where the usage of servers by higher priority requests is accounted for through server vanishing and reappearance rates.

We first consider systems with fixed priorities. The method is then extended to cases where sources can issue requests at several priority levels.

KEYWORDS: preemptive priority, multiple servers, finite source, exponential distribution, recurrent solution.

1. Introduction

Queueing systems with priority service occur frequently in computer systems and other applications. For open M/G/1 queues with priorities, the exact analytical solution is known and easy to evaluate numerically. However, in many applications, the requests for service may come from a finite and possibly relatively small number of sources, so that a Poisson arrival process cannot be used as an adequate representation. Also, often, service is provided by a pool of servers so that a single server model is not valid. This is the case with multiprocessor CPUs in computer systems, as well as in machine interference systems with a team of operatives to repair failing machines of different priorities.

Given the importance of the priority discipline, there is a large body of literature on this subject, and it is not the object of this paper to provide a comprehensive bibliography. Jaiswal and Thiruvengadam [6] gave a solution for a single server with priorities and finite source. This solution is generally considered too complex for practical applicability. Chandra and Sargent [4] proposed a numerical solution, based on embedded Markov chain analysis, to a single server with nonpreemptive fixed priorities. Kameda [7] considered a priority queue with finite sources and a single server. Causes of inaccuracies of several approximations, developed in particular for computer performance applications, were reviewed by Kaufman [8]. Veran [9] proposed a numeric method for the analysis of a single server queue with preemptive priorities. Bondi and Chuang [1], and Eager and Lipscomb [5] proposed additional approximations based on Mean Value Analysis. More recently, Wagner [10] studied multiple server queues with non preemptive priorities and finite source. Bunday, Khorram and Bokhari [3] solved the problem with preemptive priorities for two classes of requests and multiple servers in the case where the service discipline is Service in Random Order.

In this paper, we consider systems with finite sources of requests, multiple servers, and an arbitrary number of priority classes. The service times and the source idle times are assumed to follow negative exponential distributions. Our approach is based on marginal state description, and we first consider systems with fixed preemptive priorities. The method is then extended to cases where sources can issue requests at several priority levels.

2. A simple recurrent analysis

We start by considering the system depicted in Figure 1. The system consists of M servers shared by a set of c queues. Each queue corresponds to a class of users with a finite number of sources. For class i, i=1,...,c, we denote by N_i the number of sources, by $1/\lambda_i$ the mean time for an idle source to generate a request, and by $1/\mu_i$ the mean service time required for a request of this class. We assume that the request generation times and the intrinsic service times are exponentially distributed random variables, and that a source cannot generate a new request if it has an outstanding request in the system. We also assume that requests of different classes have different priorities. Class 1 is taken to be of highest priority, and the service discipline is assumed to be preemptive-resume across classes.

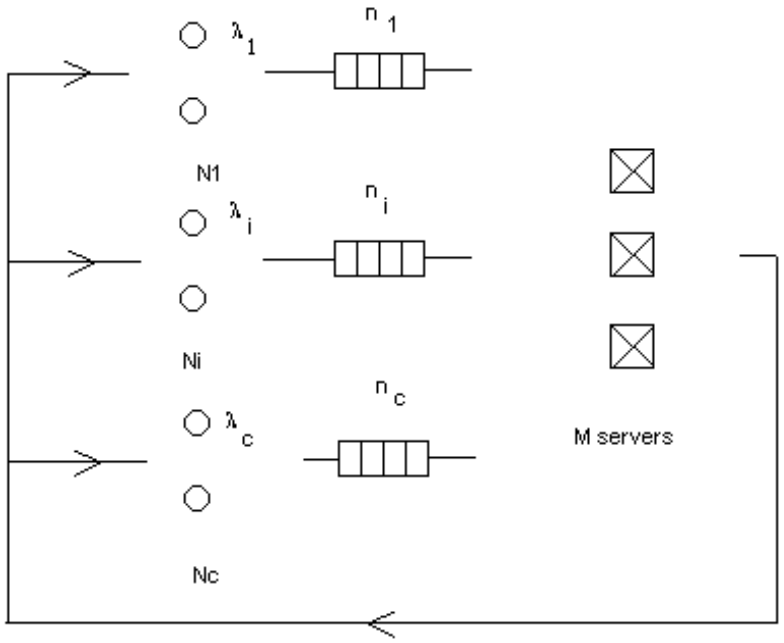

Figure 1: Finite source multiserver system with preemptive priorities

With preemptive-resume priority, requests of class i are only competing for servers with requests of classes 1, ...,i-1, as well as with other class i users. Denote by n_i the current number of class i requests, and by l_i (l_i = 0,...,M) the number of servers currently unavailable to class i, i.e, serving higher priority requests.

We consider the system in steady state, and use (n_i, l_i) as the state description for class i requests. We let $\alpha_i(n_i,l_i)$ be the rate at which servers become unavailable to class i given that there are n_i class i requests in the system and a total of l_i servers already unavailable, and by $\beta_i(n_i,l_i)$ the corresponding rate with which servers become available again, i.e., are done serving higher priority users.

We refer to $\alpha_i(n_i,l_i)$ and $\beta_i(n_i,l_i)$ as the server *vanishing* and r*eappearance rates*, respectively. If these rates were know, we could without difficulty write the balance equations for $p(n_i,l_i)$, the steady-state probability that there are n_i class i requests currently in the system, and that M-l_i servers are available at this priority level.

In order to obtain a recurrent solution for the priority system considered, we assume that the server vanishing and reappearance rates α_i and β_i do not depend on n_i, the number of class i requests, but are function a l_i alone: $\alpha_i(l_i)$, $\beta_i(l_i)$. Although this is an approximation, it certainly seems reasonable that lower priority requests should have little influence on arrivals and completions at higher priority levels.

As a result, we have the following balance equations for $p(n_i,l_i)$

$$[(N_i-n_i)\lambda_i + s(n_i,l_i)\mu_i + \alpha_i(l_i) + \beta_i(l_i)]p(n_i,l_i)$$

$$= (N_i-n_i+1)\lambda_i p(n_i-1,l_i) + s(n_i+1,l_i)\mu_i p(n_i+1,l_i) + \alpha_i(l_i-1)p(n_i,l_i-1) + \beta_i(l_i+1)p(n_i,l_i+1), \quad (1)$$

where

$$\begin{aligned} &s(n_i,l_i) = \min(n_i, M-l_i), \\ &\alpha_i(M) = 0, \\ &\beta_i(0) = 0, \end{aligned} \quad (2)$$

and terms for infeasible states are assumed to vanish.

Note that (1) corresponds to a simple two dimensional birth and death process with a state space of size $(N_i+1)(M+1)$. Several methods are available to solve such a system numerically for $p(n_i,l_i)$, e.g. [2] . It is clear also from the definition of the server vanishing and reappearance rates that the marginal probability that l_i servers are unavailable to class i requests can be expressed as

$$p(l_i) = \frac{1}{G} \prod_{j=1}^{l_i} \alpha_i(j-1)/\beta_i(j), \qquad l_i = 0, 1,\ldots \quad (3)$$

where G is a normalizing constant.

Having obtained the probability $p(n_i,l_i)$ that the current number of class i requests is n_i and that l_i servers are busy with higher priority requests (classes 1,...,i-1), we can compute the server vanishing and reappearance rates for users of class i+1. Let $p(l_{i+1})$ be the steady state probability that l_{i+1} servers are unavailable to class i+1 requests. We have

$$p(l_{i+1}) = \begin{cases} \sum_{l_i=0}^{l_{i+1}} p(n_i = l_{i+1}-l_i, l_i), & l_{i+1}=0,\ldots,\min(M-1, N_i) \\ \sum_{l_i=0}^{M} \sum_{n_i=M-l_i}^{N_i} p(n_i,l_i), & l_{i+1}=M, \text{ if } N_i \geq M \end{cases} \quad (4)$$

The server vanishing rate is then given by

$$\alpha_{i+1}(l_{i+1}) = \sum_{l_i=0}^{l_{i+1}} p(n_i=l_{i+1}-l_i,l_i)[\alpha_i(l_i)+(N_i-n_i)\lambda_i]/\, p(l_{i+1}), \quad (5)$$

$$\text{for } l_{i+1} = 0,\ldots,\min(M-1, \sum_{j=1}^{i} N_j)$$

and the server reappearance rate can be expressed as

$$\beta_{i+1}(l_{i+1}) = \sum_{l_i=0}^{l_{i+1}} p(n_i=l_{i+1}-l_i,l_i)[\beta_i(l_i)+s(n_i,l_i)\mu_i]/p(l_{i+1}) \qquad (6)$$

$$\text{for } l_{i+1} = 1,\ldots,\min(M, \sum_{j=1}^{i} N_j)$$

(5) and (6) simply combine the unavailability of servers due to priority levels higher than i with the server usage by class i requests. Clearly, all servers are available to users at the highest priority level, so that we have $\alpha_1(l_1) = \beta_1(l_1) = 0$, for $\forall\ l_1$, and for requests of class 2 we have

$$\alpha_2(l_2) = (N_1-l_2)\, l_1, \quad l_2 = 0, \ldots, \min(M, N_1)-1 \qquad (7)$$

and

$$\beta_2(l_2) = \begin{cases} l_2\mu_1, & l_2 = 1, \ldots, \min(M-1, N_1), \\ M\mu_1 \text{ Prob}\{n_1=M \mid l_2=M\}, & l_2=M, \text{ if } N_1 \geq M. \end{cases} \qquad (8)$$

The conditional probability $\text{Prob}\{n_1=M \mid l_2=M\}$ accounts for the fact that the number of servers busy at level 1 decreases only when no class 1 requests are waiting in queue. This probability is readily computed from the solution of priority level 1.

Note that, starting with the highest priority level, we are thus able to solve our preemptive priority system recurrently, one level at a time. At each priority level, we solve the relatively simple system of equations (1) where the activity of higher priority levels is accounted for through the server vanishing and reappearance rates $\alpha_i(l_i)$ and $\beta_i(l_i)$. We have thus traded the solution of the whole priority system with a total of c classes for c solutions of much simpler birth and death processes, c-1 of which are two dimensional. The state description we use allows us to examine the marginal behavior of each priority level but does not provide full information about joint behavior of several priority levels.

In the next section, we present a few numerical examples which illustrate the accuracy of our approach.

3. Numerical examples

To obtain our recurrent solution, we introduced the assumption that the server vanishing and reappearance rates depend only on the number of servers busy serving higher priority customers and not on the number of lower priority requests. To see that this is in general only an approximation, consider again the server reappearance rates for customers of class 2. With the marginal state description chosen, for $l_2=M$, the corresponding rate is given by

$$\beta_2(n_2,l_2) = M\mu_1 \text{Prob}\{n_1=M \mid n_2, l_2=M\}. \qquad (9)$$

Clearly, the conditional probability that there are exactly M class 1 requests given that there are M servers busy with class 1 and n_2 class 2 requests may be influenced by the value of n_2. Note that the conditional probability in (9) can be expressed as

$$p(n_1=M,n_2) \,/ \sum_{n1\geq M} p(n_1,n_2) = p(n_1=M \mid n_2) \,/ \sum_{n1\geq M} p(n_1 \mid n_2), \qquad (10)$$

so that by dropping the dependency on n_2 we are assuming that the ratio of marginals $p(n_1=M) / \sum_{m \geq M} p(n_1)$ is close to the corresponding ratio of conditionals.

To assess the accuracy of our approach, we use the numerical solution of the balance equations for examples with two priority levels, and discrete event simulation as comparison basis for examples with a larger number of priority classes. The reported confidence intervals are at 95% confidence level, and were obtained using the independent replications method with 10 runs of 55000 service completions each. We ran a large number of cases, comparing both the throughput (requests served per time unit) and the expected number of users in the system (queued and in service) for each class of requests. In the vast majority of cases, the results of our recurrent approach are very close to the exact numerical results or fall within the estimated confidence intervals, occasionally slightly outside.

A few examples illustrate the accuracy of our method. Under the heading method, we use e for exact numerical solution, s for simulation, and r for the recurrent approximate solution. The input data used is identified by a set number as follows:

set	number of classes	class	sources	mean interarrival time	mean service time
1	2				
		1	6	10	1.5
		2	18	5	3
2	2				
		1	6	10	1.5
		2	18	0.5	0.3
3	2				
		1	6	10	1.5
		2	20	0.5	0.3
4	2				
		1	6	10	1.5
		2	5	0.2	0.1
5	3				
		1	6	10	1.5
		2	5	0.2	0.1
		3	10	4	2
6	4				
		1	5	10	1
		2	5	0.2	0.05
		3	10	3	1
		4	10	1	0.1
7	5				
		1	3	10	1
		2	3	0.2	0.05
		3	4	3	1
		4	4	1	0.2

5	5	0.5	0.1

Table 1: results with 2 servers

set	class	method	throughput	expected number
1	1	e	0.516	0.841
		r	0.516	0.841
	2	e	0.409	15.956
		r	0.409	15.956
2	1	e	0.516	0.841
		r	0.516	0.841
	2	e	4.087	15.956
		r	4.087	15.956
3	1	e	0.516	0.841
		r	0.516	0.841
	2	e	4.087	17.956
		r	4.087	17.956
4	1	e	0.516	0.841
		r	0.516	0.841
	2	e	10.116	2.977
		r	10.117	2.977
7	1	s	0.274 +- 0.006	0.282+-0.007
		r	0.273	0.274
	2	s	11.510+- 0.039	0.705+-0.003
		r	11.500	0.700
	3	s	0.786+-0.007	1.627+-0.031
		r	0.809	1.573
	4	s	1.209+-0.024	2.789+-0.035
		r	1.440	2.560
	5	s	0.905+-0.023	4.542+-0.017
		r	0.532	4.734

Table 2: results with 3 servers

set	class	method	throughput	expected number
	1	e	0.521	0.787
		r	0.521	0.787
1				
	2	e	0.739	14.303
		r	0.739	14.303
	1	e	0.521	0.787
		r	0.521	0.787
3				
	2	e	7.394	16.303
		r	7.394	16.303
	1	s	0.524+-0.007	0.792+-0.015
		r	0.521	0.787
5	2	s	14.402+-0.067	2.119+-0.010
		r	14.409	2.118
	3	s	0.387+-0.007	8.452+-0.044
		r	0.389	8.445
	1	s	0.464+-0.011	0.452+-0.015
		r	0.455	0.455
	2	s	19.527+-0.035	1.105+-0.007
		r	19.488	1.102
6				
	3	s	1.506+-0.024	5.490+-0.105
		r	1.527	5.419
	4	s	0.492+-0.045	9.434+-0.069
		r	0.440	9.560

We observe that our method produces results which tend to fall within or slightly outside the estimated confidence intervals. This seems to be the typical behavior. Occasionally, however, the results obtained deviate more significantly from simulation results. This is illustrated by set 7 in Table 1. It has been our experience that this type of inaccuracy is more likely to occur when higher priority classes have longer average service time requirements than lower priority requests (similar to the observation in [8]) although this is not systematically so. Also, the errors don't appear to grow as the ratio of mean service times increases; on the contrary, they seem to peak and then vanish.

If the occasional lower accuracy is of concern, it is possible to nearly eliminate this behavior by using a somewhat more involved state description. The resulting method is described in the next section.

4. Alternate method

The central idea is to consider priority classes in "adjacent" pairs while accounting for the influence of higher priority requests through server vanishing and reappearance rates, this time conditioned also on the current number of higher priority requests in the pair being considered.

Thus, as an example, for a system with 4 priority classes, we first solve for the joint behavior of classes 1 and 2. This yields the steady state probability $p(n_1,n_2)$ for the numbers of requests in the system. Then, we move on to the pair 2 and 3, and we use the state description (n_2,n_3,l_2) where l_2 denotes the number of servers unavailable to the pair, i.e. busy with class 1. Because we consider classes in consecutive pairs, we can now use server vanishing and reappearance rates that are a function of both n_2 and l_2,i.e. $\alpha_2(n_2,l_2)$, $\beta_2(n_2,l_2)$. We have

$$\alpha_2(n_2,l_2) = (N_1-l_2)\lambda_1, \quad l_2=0,\ldots,\min(M,N_1)-1, \quad n_2=0,\ldots,N_2 \tag{11}$$

$$\beta_2(n_2,l_2) = \begin{cases} l_2\mu_1, & l_2=1,\ldots,\min(M,N_1) \\ M\mu_1 \text{Prob}\{n_1=M \mid l_2=M, n_2\}, & l_2=M, \text{ if } N_1 \geq M, \quad \forall\, n_2 \end{cases} \tag{12}$$

The conditional probability in (12) is computed as

$$\text{Prob}\{n_1=M \mid l_2=M,n_2\} = p(n_1=M,n_2) \Big/ \sum_{n_i=M}^{N_1} p(n_1,n_2) \tag{13}$$

The solution of the corresponding balance equations yields the equilibrium distribution $p(n_2,n_3,l_2)$.

Finally, we consider the pair of priority classes 3 and 4. The state description used is (n_3, n_4,l_3), where l_3 is the number of servers unavailable to the pair. We denote by $\alpha_3(n_3,l_3)$ and $\beta_3(n_3,l_3)$ the server vanishing and reappearance rates, respectively, and we compute these rates using the following general formula for i=2, 3...

$$\alpha_{i+1}(n_{i+1},l_{i+1}) = \sum_{l_i=0}^{\min(l_{i+1},L_i)} p(n_i=l_{i+1}-l_i,n_{i+1},l_i)[\alpha_i(n_i,l_i)+(N_i-n_i)\lambda_i] \Big/ p(n_{i+1},l_{i+1}) \tag{14}$$

$$\text{for } l_{i+1} = 0,\ldots,\min(M, \sum_{j=1}^{i} N_j) - 1, \quad n_{i+1} = 0,\ldots,N_{i+1}$$

and

$$\beta_{i+1}(n_{i+1},l_{i+1}) = \sum_{l_i=0}^{\min(l_{i+1},L_i)} p(n_i=l_{i+1}-l_i,n_{i+1},l_i)[\beta_i(n_i,l_i)+s(n_i,l_i)\mu_i] / p(n_{i+1},l_{i+1}) \qquad (15)$$

for $l_{i+1} = 1,\ldots,\min(M, \sum_{j=1}^{i} N_j)$, $n_{i+1} = 0,\ldots,N_{i+1}$

where the probability $p(n_{i+1},l_{i+1})$ is obtained as

$$\sum_{l_i=0}^{\min(l_{i+1},L_i)} p(n_i=l_{i+1}-l_i,n_{i+1},l_i), \qquad \text{for } l_{i+1}=0,\ldots,\min(M-1, \sum_{j=1}^{i} N_j) \qquad (16)$$

$$\sum_{l_i=0}^{L_i} \sum_{n_i=M-l_i}^{N_i} p(n_i,n_{i+1},l_i), \qquad \text{if } \sum_{j=1}^{i} N_j \geq M$$

and $L_i = \min(M, \sum_{j=1}^{i-1} N_j)$ is the maximum number of servers unavailable to the priority class pair i, i+1.

It is quite straightforward to derive the balance equations for the steady state probability $p(n_i,n_{i+1},l_i)$ that there are n_i and n_{i+1} requests of class i and i+1, respectively, and that li servers are unavailable to this pair. These equations are given in the Appendix. Their solution can be obtained by any of a number of methods.

With the state description chosen, the interference of class i on class i+1 is represented explicitly so that it seems best to use the results of the solution of the first pair for both classes 1 and 2, and retain the results for class i+1 for subsequent pairs (i=2,3,...). In practice, there seems to be very little difference in the marginal results for a class between the pairs i-1, i and i, i+1.

The added computational effort appears to pay off in that it reduces or practically eliminates the occasional lower accuracy observed with the simpler approach. This is illustrated in Tables 3 and 4 for the input sets exhibiting lower accuracy in Tables 1 and 2, respectively. For easier reference, we reproduce simulation results as well as those produced by the simpler recurrence. The results of the alternate method are identified by the letter a.

Table 3: alternate method, results with 2 servers

set	class	method	throughput	expected number
	1	s	0.274 +- 0.006	0.282+-0.007
		r	0.273	0.274
		a	0.273	0.274
	2	s	11.510+- 0.039	0.705+-0.003
		r	11.500	0.700
		a	11.500	0.700
	3	s	0.786+-0.007	1.627+-0.031
		r	0.809	1.573

		a	0.795	1.615
7				
	4	s	1.209+-0.024	2.789+-0.035
		r	1.440	2.560
		a	1.261	2.739
	5	s	0.905+-0.023	4.542+-0.017
		r	0.532	4.734
		a	0.920	4.540

Table 4: alternate method, results with 3 servers

set	class	method	throughput	expected number
	1	s	0.464+-0.011	0.452+-0.015
		r	0.455	0.455
		a	0.455	0.455
	2	s	19.527+-0.035	1.105+-0.007
		r	19.488	1.102
		a	19.488	1.103
6				
	3	s	1.506+-0.024	5.490+-0.105
		r	1.527	5.419
		a	1.515	5.455
	4	s	0.492+-0.045	9.434+-0.069
		r	0.440	9.560
		a	0.507	9.493

In the next section, we outline the extension of our approach to the case where customers of different classes are no longer each confined to a single priority level.

5. Classes spanning several priority levels

We now consider a generalization of the system of Figure 1 in that we dissociate customer classes and priority levels. Specifically, we assume that a new request generated by a class i (i=1,...c), source with probability q_{ik} requires service at priority level k, k=1,...,K, where K is the total number of priority levels. Clearly, we have $\sum_k q_{ik} = 1$ for all classes i. We also assume that the service times are exponentially distributed, and we denote by $1/\mu_{ik}$ the average service time required by class i request at priority level k.

The system described presents new difficulties: higher priority levels may now depend on lower priority levels in that the rate of request generation depends on the total number of requests of a given class currently in the system, i.e. possibly also on lower priority levels, and, also, the requests at any given priority level may now be a mixture of several distinct classes. We cast our discussion in the context of the recurrence of Section 2, and, as a simple approximation, we propose the following approach.

Consider priority level k. The throughput of class i requests at this level can be expressed as

$$\theta_{ik} = \lambda_i q_{ik}(N_i - n_{ik} - n_{ix}) \qquad (17)$$

where n_{ik} and n_{ix} denote the expected number of class i requests at level k, and at other priority levels, respectively. We treat class i at level k as an "independent" class of users with a fictitious rate of request generation φ_{ik} determined by from the relationship

$$\varphi_{ik}(N_i - n_{ik}) = \theta_{ik} \qquad (18)$$

Thus, during the analysis of each priority level, we treat the request classes as tied to the given priority level. Clearly, since n_{ix} generally is not known when dealing with level k, and n_{ik} and n_{ix} depend on each other, a fixed point iteration seems the approach of choice. Denote by n the current total number of requests at level k, and by l the number of servers unavailable to level k. Denote also by $n_{ik}(n)$ the conditional expected number of class I requests at level k given, and let $m_{ik}(n,l)$ be the conditional expected number of servers busy with class i requests at level k given n and l. To deal with multiple classes of requests at a single priority level, we use the following simple approximation.

$$n_{ik}(n) \approx n\, f_{ik}(n), \quad n=1,\ldots \qquad (19)$$

$$m_{ik}(n,l) \approx s(n,l) f_{ik}(n), \quad n=1,\ldots \qquad (20)$$

where

$$f_{ik}(n) = [N_i - n_{ik}(n-1)]\sigma_{ik} \Big/ \sum_j [N_j - n_{jk}(n-1)]\sigma_{jk}, \text{ for } n=1,..,\ k=1,\ldots K \qquad (21)$$

and

$$\sigma_{jk} = \varphi_{ik} / \mu_{ik}. \qquad (22)$$

Note that, starting with $n_{ik}(0) = 0$, for all i and k, (19) and (21) allow us to obtain $n_{ik}(n)$ recurrently. Hence, referring to the state description (n,l), we can express the rate of arrivals of requests to level k as

$$\lambda_k(n) = \sum_j [N_j - n_{jk}(n-1)]\, \varphi_{jk}, \qquad n=0,1\ldots \qquad (23)$$

and the state dependent rate of service at level k as

$$\mu_k(n,l) = \sum_j m_{jk}(n,l)\mu_{jk}, \qquad n=1,2,\ldots \qquad (24)$$

These state dependent arrival and service rates allow us to proceed with the solution of priority levels in a manner analogous to the case where there is only one request class at each level. Clearly, knowing the probability distribution for (n,l) at level k and the conditional averages $n_{ik}(n)$ it is straightforward to obtain the averages n_{ik} .

With a First Come First Served discipline at a priority level, the relationships defined through (19), (20) and (21) can be expected to be best satisfied when the mean service times of request classes at the given priority level ($1/\mu_{ik}$) are not very different. Note that this is the case in a number of applications where service at a given priority level is synonymous with a cap on service requirements. The recurrence defined by (19) and (21) may be subject to numerical loss of accuracy so that in practice care must be taken to ensure that none of the $n_{jk}(n)$ is allowed to exceed N_j.

With these caveats, the general accuracy of the proposed approach to classes spanning several priority levels is good. As an example, Table 5 shows the results obtained with 3 request classes corresponding to parameter set 5 except that class spans several priority levels. In case 1 and 2, we have q_{11}=0.5, q_{12}=0.4, q_{13}=0.1 with 4 and 3 servers, respectively. Case 3 pertains to 3 servers and q_{11}=q_{12}=0.5. The simpler recurrent solution of Section 1 was used in all three cases. Typically, the number of iterations needed to achieve stable values of throughputs with classes on multiple priority levels is on the order of 10.

Table 5: results with a class on several priority levels

case	class	method	throughput	expected number
	1	s	0.496+-0.006	1.048+-0.030
		r	0.494	1.056
1	2	s	16.261+-0.038	1.746+-0.006
		r	16.310	1.738
	3	s	0.817+-0.007	6.739+-0.065
		r	0.808	6.767
	1	s	0.450+-0.007	1.501+-0.048
		r	0.448	1.519
2	2	s	15.091+-0.070	1.983+-0.013
		r	15.276	1.945
	3	s	0.411+-0.009	8.335+-0.048
		r	0.400	8.400
	1	s	0.524+-0.006	0.810+-0.013
		r	0.503	0.969
3	2	s	14.465+-0.063	2.108+-0.010
		r	14.898	2.020
	3	s	0.394+-0.007	8.401+-0.043
		r	0.378	8.489

Cases 1 and 2 show what seems to be the typical accuracy of the approach with only small percentage errors. Case 3 illustrates occasional larger errors: some 5% on throughputs, and 20% on average number of requests in the system. Note that some part of this deviation may be due to the use of the simpler single level recurrence.

6. Conclusion

We have presented an approach to the solution of finite source multiserver priority queues under memoryless distributional assumptions. The approach analyzes priority levels in isolation, and the interference of higher priority levels is represented though server vanishing and reappearance rates. The accuracy of this method is generally good although occasionally larger errors can be observed. It is possible to eliminate or reduce these errors through the use of a computationally somewhat more costly variant of the method where priority levels are treated in pairs. Finally, we have considered the case where sources of a given class may issue requests at several priority levels. . We propose a simple fixed point approximation which appears to yield relatively accurate results. The extension of such a fixed point iteration to non-preemptive multiserver priority systems appears possible.

7. Bibliography

1. Bondi, A. B., Chuang Y.-M.: A New MVA-Based Approximation for Closed Queueing Networks with a Preemptive Priority Server, Performance Evaluation 8, 195-221 (1988).

2. Brandwajn, A: An Iterative Solution of Two-Dimensional Birth and Death Processes, Operations Research 27, 595-605 (1979).

3. Bunday, B.D., Khorram, E., Bokhari, H.M.: The G/M/r machine interference model with a mixture of priority and ordinary machines, in Proc. Third International Workshop on Queueing Networks with Finite Capacity, July 1995, Ilkley, UK.

4. Chandra, M. J., Sargent, R.G.: A numerical method to obtain the equilibrium results for the multiple finite source priority queueing model, Management Science 29, 1298-1308 (1983).

5. Eager, D., Lipscomb, J.N.: The AMVA Priority Approximation, Performance Evaluation 9, 173-193 (1988).

6. Jaiswal, N.K. and Thiruvengadam: Finite source priority queues, SIAM Journal of Applied Mathematics 15, 1273-1293 (1967).

7. Kameda, H: A finite-source queue with different customers, Journal of the ACM 29, 478-491 (1982).

8. Kaufman, J.S.: Approximation Methods for Networks of Queues with Priorities, Performance Evaluation 4, 183-198 (1984).

9. Veran, M: Exact analysis of a priority queue with finite source, in Proc. Int. Seminar on Modelling and Performance Evaluation Methodology, Paris, Springer Verlag, 371-390, 1985.

10. Wagner, D. Steady State Analysis of a Finite Capacity Multi-Server Model with Nonpreemptive Priorities and Nonrenewal Input, in Proc. Third International Workshop on Queueing Networks with Finite Capacity, July 1995, Ilkley, UK.

Appendix

Balance Equations for $p(n_i,n_{i+1},l_i)$

$$p(n_i,n_{i+1},l_i)[(N_i-n_i)\lambda_i + (N_{i+1}-n_{i+1})\lambda_{i+1} + \alpha_i(n_i,l_i) + \beta_i(n_i,l_i) + s(n_i,l_i)\mu_i]$$

$$= p(n_i-1,n_{i+1},l_i)(N_i-n_i+1)\lambda_i + p(n_i,n_{i+1}-1,l_i)\,(N_{i+1}-n_{i+1}+1)\lambda_{i+1}$$

$$+ p(n_i+1,n_{i+1},l_i)s(n_i+1,l_i)\mu_i + p(n_i,n_{i+1}+1,l_i)\; s(n_{i+1}+1,n_i+l_i)\mu_{i+1}$$

$$+ p(n_i,n_{i+1},l_i-1)\alpha_i(n_i,l_i-1) + p(n_i,n_{i+1},l_i+1)\;\beta_i(n_i,l_i+1)$$

where unfeasible terms are assumed to vanish.