An approximation method for a non-preemptive multiserver queue with quasi-Poisson arrivals

Alexandre Brandwajn, Thomas Begin

An approximation method for a non-preemptive multiserver queue with quasi-Poisson arrivals

ALEXANDRE BRANDWAJN, Baskin School of Engineering, University of California Santa Cruz, USA
THOMAS BEGIN, Univ Lyon, ENS de Lyon, Université Claude Bernard Lyon 1, Inria, CNRS, France

We consider a non-preemptive multiserver queue with multiple priority classes. We assume distinct exponentially distributed service times and separate quasi-Poisson arrival processes with a predefined maximum number of requests that can be present in the system for each class. We present an approximation method to obtain the steady-state probabilities for the number of requests of each class in our system. In our method, the priority levels (classes) are solved "nearly separately", linked only by certain conditional probabilities determined approximately from the solution of other priority levels. Several numerical examples illustrate the accuracy of our approximate solution. The proposed approach significantly reduces the complexity of the problem while featuring generally good accuracy.

CCS Concepts: • **Mathematics of computing → Queueing theory**.

Additional Key Words and Phrases: Queueing, Multiserver, Non preemptive, Reduced complexity, Approximate solution

ACM Reference Format:
Alexandre Brandwajn and Thomas Begin. 2023. An approximation method for a non-preemptive multiserver queue with quasi-Poisson arrivals. *ACM Trans. Model. Perform. Eval. Comput. Syst.* , (January 2023), 22 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Multiserver queues with non-preemptive priority classes arise in a number of areas including computer I/O organization, wavelength-division multiplexing (WDM) in optical networks, call centers with calls of different priorities, as well as health care and industrial engineering [6, 9–13, 15, 16].

Given the complexity of analyzing a non-preemptive priority multiserver queue, there are relatively few papers devoted to the computation of its behavior. In 1966, Davis gave an explicit formula for the waiting time distribution in such a queue with Poisson arrivals and any number of priority classes with identical exponentially distributed service times [3]. In 1980, Williams proposed a solution to compute the steady-state probabilities of such a queue with two priority classes having the same service time distribution [20]. A few years later, Kella and Yechiali derived the Laplace transform of the waiting time in a M/M/c priority queue where all classes have the same mean service times [8]. In 1988, Gail et al. considered the case where there are two classes with exponentially distributed service times and different means [4]. While the authors were able

Authors' addresses: Alexandre Brandwajn, Baskin School of Engineering, University of California Santa Cruz, USA, alexb@soe.ucsc.edu; Thomas Begin, Univ Lyon, ENS de Lyon, Université Claude Bernard Lyon 1, Inria, CNRS, France, thomas.begin@univ-lyon1.fr.

2376-3639/2023/1-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

to obtain the generating functions associated with the equilibrium equations for any number of servers, the accuracy of their numerical evaluation deteriorates as the number of servers increases.

In 1990, Kao and Narayanan [7] formulated the problem of a non-preemptive priority multiserver queue with two classes as a quasi-birth-and-death process and proposed a solution to compute the steady-state probabilities. Their solution applies to problems in which the marginal distribution for the number of high priority customers in the system has a short tail. In their numerical results, they consider examples with up to 10 servers and up to 12 high priority customers. Three years later, Reddy, Nadarajan and Kandasamy [15] studied a non-preemptive priority multiserver queue with two classes. The lower-priority customers are served in batches while the higher priority customers are served singly. The authors use a matrix-geometric algorithmic approach to obtain the steady-state probabilities of the number of customers in the queue. In 1997, Wagner proposed a Laplace transform-based solution to compute the waiting times of customers in a non-preemptive priority multiserver finite capacity queue with multiple classes having the same service times [19]. In a separate work published the same year, Wagner [18] presented another solution, based on first passage time analysis and Little's formula, to obtain the mean values for an open queue with identical phase-type service time distributions for each priority class.

In 2002, using linear programming, Peköz introduced a lower bound on the mean waiting time for high-priority customers in a non-preemptive priority multiserver queue with two classes subject to an admission control [13]. One year later, Sleptchenko [16] derived an approximation with an arbitrarily small error to estimate the steady-state probabilities for a non-preemptive priority multiserver queue with two classes having exponentially distributed service times.

More recently, in 2008, Krishnamoorthy, Babu and Narayanan proposed an approach [11] to compute the solution of a non-preemptive priority multiserver queue in which customers arrive to the system according to a Markovian Arrival Process (MAP) and belong to one of two classes with different phase-type service time distributions. In their system, the authors assume that customers generate their priority at a constant rate while waiting in the system (self-generation of priorities). This latter assumption can be viewed as restrictive in cases where the proportion of customers of each class in the queue does not remain constant.

In 2021, Kim et al. obtained the generating function for the distribution of the number of customers [10] in a non-preemptive multiserver queue with two classes, Poisson arrivals, exponentially distributed service times (same mean for the two classes), and exponentially distributed server vacations.

In this paper, we consider a non-preemptive multiserver system with multiple priority classes. Requests of each class have their own exponentially distributed service times and arrive to the system according to separate quasi-Poisson processes with a predefined maximum number of requests of each class that can be present in the system. We propose an approximation method to obtain the steady-state probability distribution for the number of requests of each class in our system. The proposed approximate solution greatly reduces the complexity of the problem while featuring generally good accuracy, including when the number of servers increases.

This paper is organized as follows. In the next section, we describe the system considered and the proposed approximate solution. Section 3 is devoted to numerical examples illustrating the behavior of a multiserver non-preemptive priority queue with multiple priority classes, as well as the accuracy of the proposed approximation. Section 4 concludes our paper.

2 QUEUE CONSIDERED AND ITS APPROXIMATION

Consider the queueing system represented in Figure 1. There a total of C identical servers and L priority classes. Let n_ℓ be the current number of requests of class ℓ ($\ell = 1, \ldots, L$) in the system. Requests for this class arrive to the system according to a quasi-Poisson process with rate $\lambda_\ell(n_\ell)$.

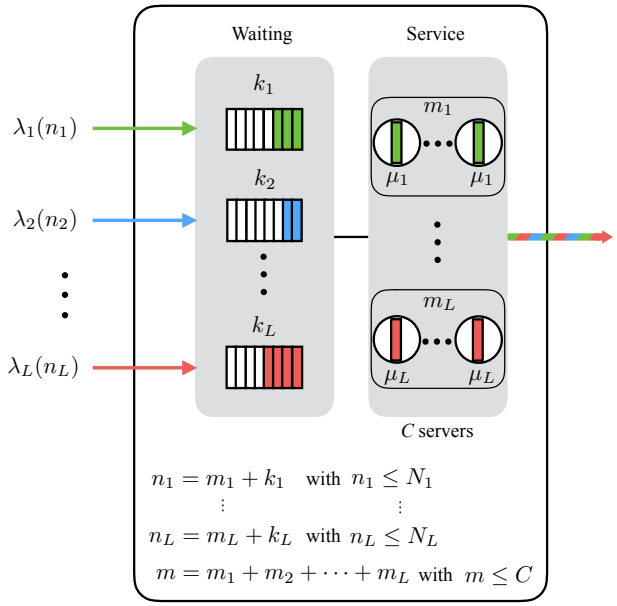


Fig. 1. The priority queueing system with L priority classes and C servers.

The number of requests of class ℓ that can be present in the system is limited to N_ℓ. This may be the result of the nature of the source of requests (e.g., N_ℓ discrete sources of requests for class ℓ) or it may be that requests arriving when the maximum number is reached are simply lost. Thus, our formulation of the problem allows us to represent sets of discrete memoryless sources, as well as Poisson arrival streams with separate finite capacity for each class.

For simplicity, we view each request class as having its own waiting line. When a server becomes available, a request of class ℓ can start its service only when there are no requests of higher priority $(1, \ldots, \ell - 1)$ waiting in their respective lines. The service times are assumed to be exponentially distributed with mean $1/\mu_\ell$ for class ℓ. Once started, the service of a request is not interrupted by arrivals of higher-priority requests.

Referring to class ℓ, we denote by m_ℓ the current number of requests in service and by k_ℓ the current number of requests waiting for server availability. We consider our multiserver priority system in steady state. The latter could be fully described by the numbers of requests of each class in service and waiting for service $(m_1, \ldots, m_L, k_1, \ldots, k_L)$. If we let $p(m_1, \ldots, m_L, k_1, \ldots, k_L)$ be the corresponding steady-state probability, one could derive the necessary balance equations for our system using standard techniques.

Unfortunately, the dimensionality of this system of balance equations grows very fast with the number of classes, the maximum number of users of each class and the number of servers. As shown in Figure 3, with 5 classes and just 5 servers, the number of states can quickly exceed a billion (see also Appendix). Therefore, we propose to consider the waiting line for each class "nearly separately". Specifically, we propose to describe class $\ell = 1, \ldots, L$ through the steady-state probabilities $p(m_1, \ldots, m_L, k_\ell)$, i.e., a full description of the state of the servers and the state of the waiting line only for the priority class considered. Let $m = \sum_{\ell=1}^{L} m_\ell$ be the total number of requests of all classes in service and recall that $n_\ell = m_\ell + k_\ell$ is the total number of class ℓ requests in the system (in service and in the waiting line). The balance equations for this state description are not

difficult to obtain, if a bit tedious. To derive these equations, we simply consider rate of flows out of and into each state $p(m_1, \ldots, m_L, k_\ell)$. As an example, for $m = C$ and $k_\ell > 0$, we have

$$\begin{aligned} p(m_1, \ldots, m_L, k_\ell) \Bigg[\lambda_\ell(n_\ell) + \sum_{i=1}^{\ell-1} m_i\mu_i \left[1 - P\{k_1 = 0, \ldots, k_{i-1} = 0, k_i > 0 | m_1, \ldots, m_L, k_\ell\}\right] \\ + \sum_{i=\ell}^{L} m_i\mu_i \Bigg] = p(m_1, \ldots, m_L, k_\ell - 1)\lambda_\ell(n_\ell - 1) \\ + p(m_1, \ldots, m_L, k_\ell + 1) m_\ell\mu_\ell . P\{k_1 = 0, \ldots, k_{\ell-1} = 0 | m_1, \ldots, m_L, k_\ell + 1\} \\ + \sum_{i=1:i\neq\ell}^{L} p(\ldots, m_i + 1, \ldots, m_\ell - 1, \ldots, k_\ell + 1)(m_i + 1)\mu_i \\ . P\{k_1 = 0, \ldots, k_{\ell-1} = 0 | (\ldots, m_i + 1, \ldots, m_\ell - 1, \ldots, k_\ell + 1)\} \\ + \sum_{j=1}^{L} \sum_{i=1:i\neq j}^{\ell-1} p(\ldots, m_i - 1, \ldots, m_j + 1, \ldots, k_\ell)(m_j + 1)\mu_j \\ . P\{k_1 = 0, \ldots, k_{i-1} = 0, k_i = 0 | \ldots, m_i - 1, \ldots, m_j + 1, \ldots, k_\ell\} \end{aligned} \tag{1}$$

$k_\ell = 1, \ldots, N_\ell; m_\ell = \max(0, C - \sum_{j\neq\ell} N_j), \ldots, \min(C, N_\ell - k_\ell); m_i = \max(0, C - \sum_{j\neq i} N_j), \ldots, \min(C - \sum_{j\neq i} m_j, N_i), i \neq \ell$.

Equation (1) and other balance equations for our state description are obtained by equating the rate of flow out of each state to the sum of rates of flow into the state considered. Note that, alternatively, the same equations could be derived by summing the full state equations for $p(m_1, \ldots, m_L, k_1, \ldots, k_L)$ over all k_i with $i \neq \ell$.

In equation (1), we let $\lambda_\ell(N_\ell) = 0$, and we assume that impossible terms simply vanish. Clearly, we must have $\sum_{m_1} \cdots \sum_{m_L} \sum_{k_\ell} p(m_1, \ldots, m_L, k_\ell) = 1$ for these probabilities to be normalized. The balance equations for the case when $k_\ell = 0$ can be derived in a similar manner and are given in the Appendix.

Any of a number of existing numerical solution approaches can be used to solve such a system of linear equations [17]. An alternative avenue is to use a semi-numerical approach akin to that in [2]. Let $p_\ell(k_\ell)$ be the steady-state probability that there are k_ℓ requests in the waiting line for class ℓ and denote by $p(m_1, \ldots, m_L | k_\ell)$ the conditional probability of the state of the servers given the number of class ℓ requests waiting in line. We have $p(m_1, \ldots, m_L, k_\ell) = p_\ell(k_\ell).p(m_1, \ldots, m_L | k_\ell)$ from the definition of conditional probability. $p_\ell(k_\ell)$ can be expressed as

$$p_\ell(k_\ell) = \prod_{i=1}^{k_\ell} \omega_\ell(i-1)/\nu_\ell(i), \tag{2}$$

where

$$\omega_\ell(k_\ell) = \sum_{(m_1, \ldots, m_L : m = C)} p(m_1, \ldots, m_L | k_\ell)\lambda_\ell(k_\ell + m_\ell), \quad \text{for } k_\ell = 0, \ldots, N_\ell - 1, \tag{3}$$

and

$$\begin{aligned} \nu_\ell(k_\ell) = \sum_{(m_1, \ldots, m_L)} p(m_1, \ldots, m_L | k_\ell) . P\{k_1 = 0, \ldots, k_{\ell-1} = 0 | m_1, \ldots, m_L, k_\ell\} \\ . (m_1\mu_1 + \cdots + m_L\mu_L), \end{aligned} \tag{4}$$

for $k_\ell = 1, \ldots, N_\ell$.

Formula (2) can be obtained directly by considering the rates with which k_ℓ increases and decreases. Alternatively, the same formula can be obtained by summing the equations for $p(m_1, \ldots, m_L, k_\ell)$ over all states $m_1, \ldots, m_L$. The solution of the resulting balance equations for $p_\ell(k_\ell)$ is readily seen to be given by formula (2).

With the help of these relationships in the balance equations for $p(m_1, \ldots, m_L, k_\ell)$, one can obtain the equations for the conditional probabilities $p(m_1, \ldots, m_L|k_\ell)$. The latter equations can then be solved using a simple iteration (see Appendix). The use of conditional probabilities is a matter of personal preference motivated by the potential for reduced numerical stability issues due to independent normalization of the equations for each value of k_ℓ.

We note that our balance equations for $p(m_1, \ldots, m_L, k_\ell)$ (as well as formula (4)) involve the joint conditional probabilities that the waiting lines of other priority classes are, or are not, empty given the current state of the priority class under consideration. For example, in equation (1), for $i < \ell$, $P\{k_1 = 0, \ldots, k_{i-1} = 0, k_i > 0|m_1, \ldots, m_L, k_\ell\}$ is the joint conditional probability that the waiting lines of priority classes 1 through $i-1$ are empty while that of class i is not, given that there are $m_1, \ldots, m_L$ requests in service and k_ℓ requests of class ℓ in the waiting line. If we knew these joint conditional probabilities exactly, the solution of equation (1) subject to the corresponding normalization condition would yield exact values for the steady-state probabilities $p(m_1, \ldots, m_L, k_\ell)$.

As an approximation, we propose to drop the dependence on the state of the waiting line for the class considered, i.e., k_ℓ, in these conditional probabilities, and to replace the joint conditional probabilities by the corresponding products of individual conditional probabilities. In other words, for any set of $J = 1, \ldots, L-1$ distinct indices $\{i_1, \ldots, i_J\}$ taken from the set $\{1, \ldots, \ell-1, \ell+1, \ldots, L\}$ we propose to use the following approximation:

$$P\{k_{i_1}, \ldots, k_{i_J}|m_1, \ldots, m_L, k_\ell\} \approx \prod_{j=1}^{J} P\{k_{i_j}|m_1, \ldots, m_L\}. \tag{5}$$

To the best of our knowledge, the proposed approximation is novel albeit similar in spirit to approximations used by other authors over the years to reduce the complexity of queueing models, e.g. [5, 14]. Note that one can readily obtain the individual conditional probability that there are k_i requests of class i waiting given the state of the servers $P\{k_i|m_1, \ldots, m_L\}$ from the probabilities $p(m_1, \ldots, m_L, k_i)$. Indeed, $P\{k_i|m_1, \ldots, m_L\} = p(m_1, \ldots, m_L, k_i)/\sum_{k_i} p(m_1, \ldots, m_L, k_i)$.

Thus, we propose to use the following fixed-point iterative procedure: starting from a reasonable set of initial values for the individual conditional probabilities $P\{k_\ell = 0|m_1, \ldots, m_L\}$ (in our implementation, we used $P\{k_\ell = 0|m_1, \ldots, m_L\} = 0$ for all $(m_1, \ldots, m_L)$ such that $\sum_{\ell=1}^{L} m_\ell = C$, as well as equally likely $p(m_1, \ldots, m_L|k_\ell)$ normalized for each k_ℓ, from which we can compute the marginal probabilities $p(k_\ell)$), we solve one by one the balance equations for priority classes $\ell = 1, \ldots, L$. Each solution produces the probabilities $p(m_1, \ldots, m_L, k_\ell)$ and hence the individual conditional probabilities $P\{k_\ell|m_1, \ldots, m_L\}$ for use in the solution of the balance equations of other priority levels. We iterate on this process until the values for the state probabilities of each priority level stabilize. Figure 2 summarizes the proposed approach. We don't have a theoretical proof that the proposed fixed-point iteration always converges to a unique fixed point. In practice, however, in the large number of examples we considered, the procedure always converged within just a few (or occasionally a few tens of) iterations.

In other words, we propose to replace a solution of a single system of balance equations for the steady-state probabilities $p(m_1, \ldots, m_L, k_1, \ldots, k_L)$ by a low number of solutions of L smaller systems of balance equations for the individual priority levels $p(m_1, \ldots, m_L, k_\ell)$, $\ell = 1, \ldots, L$. The

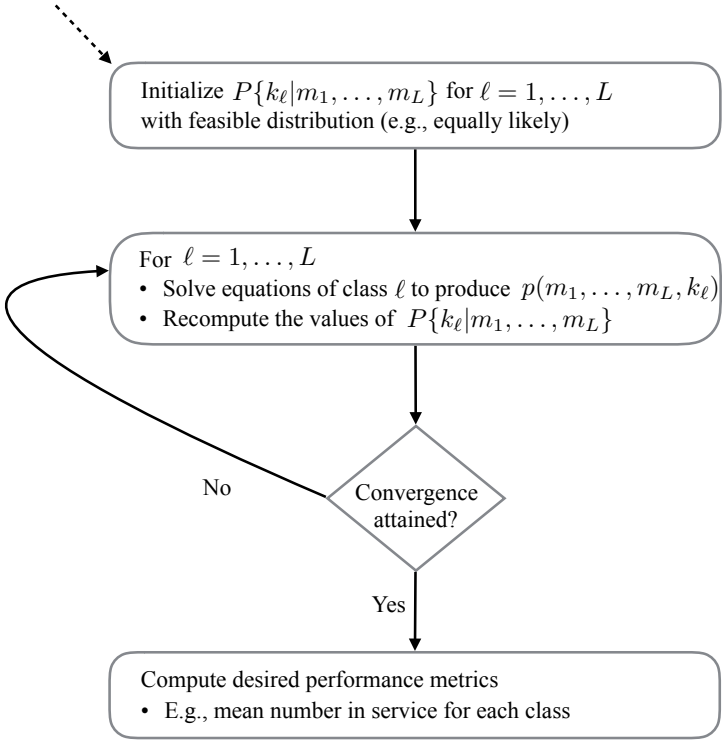


Fig. 2. Synoptic view of the approximate solution.

priority levels are solved "nearly separately" in the sense that the only link between the individual levels is through the conditional probabilities $P\{k_\ell|m_1,\ldots,m_L\}$, obtained from each individual level in the fixed-point iteration. Figure 3 compares the complexity (in terms of number of states explored) of the proposed approach with that of the single full state description (see also Appendix). In this example, all five priority classes have the same maximum number of requests in the system.

We observe that, as the maximum number of requests of each class in the system increases, the number of states in our approximation becomes several orders of magnitude smaller than the number of states in the full state description.

From the solution of priority class ℓ it is straightforward to obtain the mean number of requests of that class in the system as

$$\bar{n}_\ell = \sum_{m_1} \cdots \sum_{m_L} \sum_{k_\ell} (m_\ell + k_\ell) p(m_1,\ldots,m_L,k_\ell). \tag{6}$$

Similarly, the mean number of class ℓ requests in service can be obtained as

$$\bar{m}_\ell = \sum_{m_1} \cdots \sum_{m_L} \sum_{k_\ell} m_\ell p(m_1,\ldots,m_L,k_\ell). \tag{7}$$

The attained throughput of requests of class ℓ is then given by $\theta_\ell = \bar{m}_\ell \mu_\ell$ and the expected time a request spends in the system can be obtained from Little's formula [1] as $\bar{n}_\ell/\theta_\ell$. The overall expected time a request spends in the system (averaged over all classes) can be expressed as $\sum_{\ell=1}^{L} \bar{n}_\ell / \sum_{\ell=1}^{L} \theta_\ell$.

Table 1 summarizes the principal notation used in this paper.

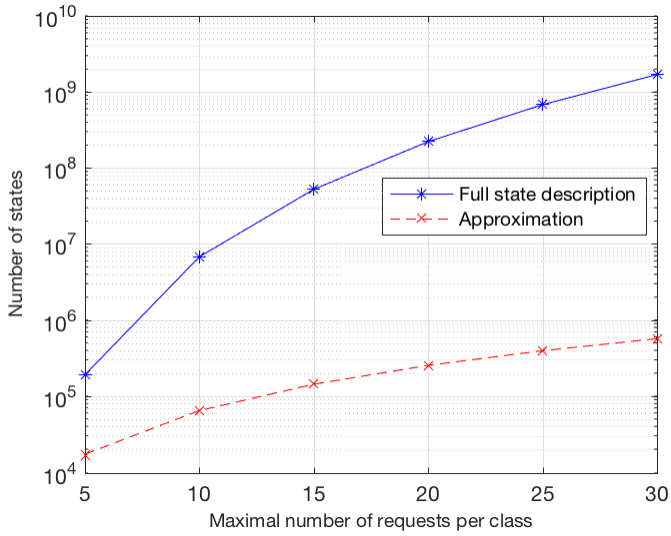


Fig. 3. Number of states explored for L=5 priority classes and C=5 servers.

Table 1. Principal notation used in the paper.

Symbol	Description
C	Number of servers
L	Number of priority classes
n_ℓ	Number of requests of class ℓ currently in the system
$\lambda_\ell(n_\ell)$	Arrival rate of requests of class ℓ given there are n_ℓ requests in the system
N_ℓ	Maximum number of requests in priority class ℓ
$1/\mu_\ell$	Mean service times for requests of priority class ℓ
m_ℓ	Number of requests of class ℓ currently in service
k_ℓ	Number of requests of class ℓ currently in waiting line
n_ℓ	Number of requests of class ℓ currently in system (service or waiting line)
m	Total number of requests of all classes in service
$p(m_1, \ldots, m_L, k_\ell)$	Steady-state probability of the state of the servers and the number in the waiting line for class ℓ
$P\{k_\ell \vert m_1, \ldots, m_L\}$	Conditional probability that there are k_ℓ requests of class ℓ in the waiting line given the state of the servers
$\bar{n}_\ell$	Mean number of requests of class ℓ in system
$\bar{m}_\ell$	Mean number of requests of class ℓ in service
θ_ℓ	Attained throughput for requests of class ℓ
α_ℓ	Unitary rate of request generation with $\lambda_\ell(n_\ell) = (N_\ell - n_\ell)\alpha_\ell$

In the next section, we present numerical examples to illustrate the behavior of such a multiserver queue with multiple non-preemptive priority classes, as well as the accuracy of the proposed approximation.

3 NUMERICAL EXAMPLES

In our numerical examples, we use two types of quasi-Poisson arrival processes. In the first one, the rate of the quasi-Poisson arrivals for classes $\ell = 1, \ldots, L$ is $\lambda_\ell(n_\ell) = (N_\ell - n_\ell)\alpha_\ell$. This corresponds to a set of N_ℓ discrete request sources for class ℓ, each source generating a new request with unitary rate $\alpha_\ell > 0$ when the given source is not currently waiting for the completion of a previous request. In the second type, arrivals to each class come from individual Poisson streams with possibly distinct intensities λ_ℓ, $\ell = 1, \ldots, L$. Arrivals that happen when the current number of requests for that class is N_ℓ are lost so that the maximum number of requests of class ℓ in the system is limited to N_ℓ. The majority of numerical results in this paper pertain to the mean number of servers used by each priority class and to the mean number of requests of each class in the system.

We use the results of discrete-event simulation to assess the accuracy of our approximation. We employ the independent replications method with 7 replications of 3,000,000 request completions each to estimate confidence intervals at 95% confidence level. The resulting confidence intervals tend to be very narrow so that we show only the mid-points of these intervals. Note that one could use a numerical solution of the balance equations for the full state description $p(m_1, \ldots, m_L, k_1, \ldots, k_L)$ for certain sets of parameters but we find it preferable to use a single method as the comparison basis in our accuracy studies. We did use the numerical solution of the full balance equations to ascertain the correctness of our simulation in the case of three priority classes.

3.1 Results with discrete sources of requests

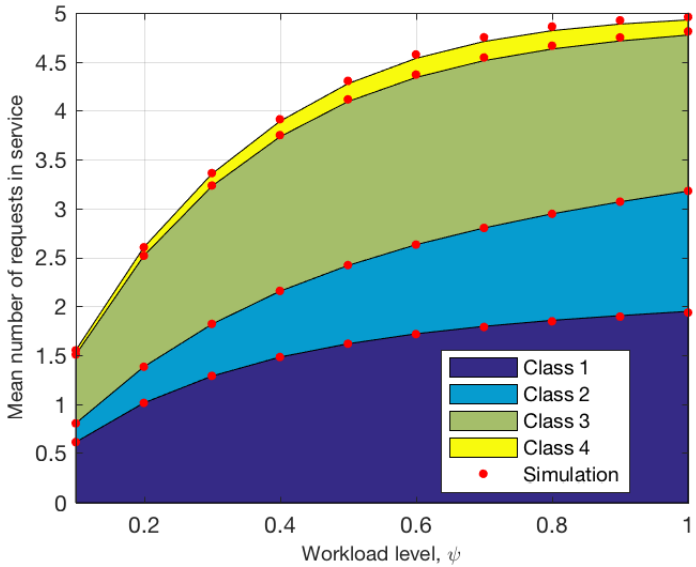
Figure 4 shows the results obtained for a queue with $L = 4$ priority classes and $C = 5$ servers for workload levels varying from low to high (all servers busy virtually all the time). We observe the close agreement between simulation results and those of our approximation both for the mean number of requests in service (Figure 4a) and for the mean number of requests in the system (Figure 4b). Note that here priority classes 1 (highest priority) and 3 dominate the server utilization. It is apparent that the high combined server utilization for classes 2 and 3 seems to flatten the growth of the server utilization for class 1.

In Figure 5 we show the results for a system with $L = 4$ priority classes and $C = 8$ servers for 10 workloads levels. Again, our approximation tracks the simulation results very closely. In this example, class 2 dominates the server utilization to the point of limiting the growth of server utilization for the lowest priority class 4 which appears "squeezed" by higher priority classes at sufficiently high workload level. The mean number of class 4 users in the system exhibits corresponding accelerated growth.

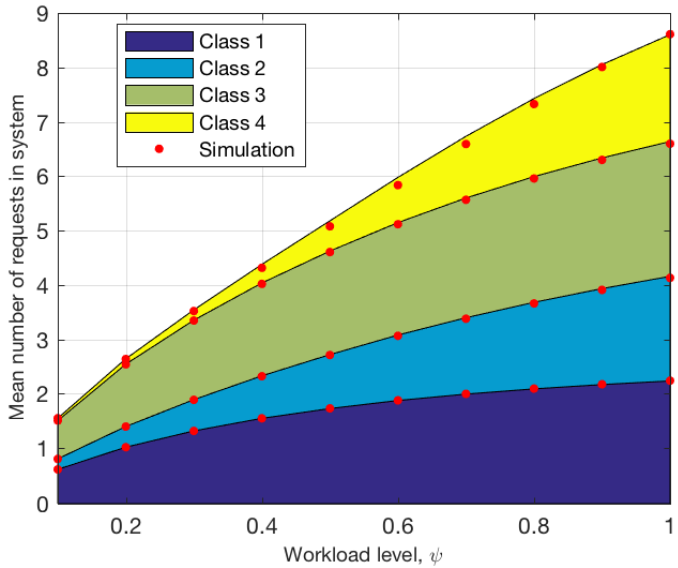
Figure 6 illustrates the results obtained for $L = 5$ priority classes and $C = 15$ servers over a range of workload levels. As in previous examples, we observe a good agreement between simulation and approximation results. Here, classes 1 and 4 dominate the system clearly "squeezing out" the lowest priority class 5 as the workload level increases beyond a certain point.

Overall, our examples illustrate the behavior of a non-preemptive priority multiserver queue for a range of class parameter values, including cases where different classes dominate the server utilization.

To gain a better understanding of the accuracy of the proposed approximation, we studied close to 500 examples randomly spanning a range of system parameters with a total of $L = 5$ priority classes. The number of discrete sources for a priority class varied randomly between 5 and 30. The number of servers in our examples varied randomly between 2 and 16. The mean service time for a

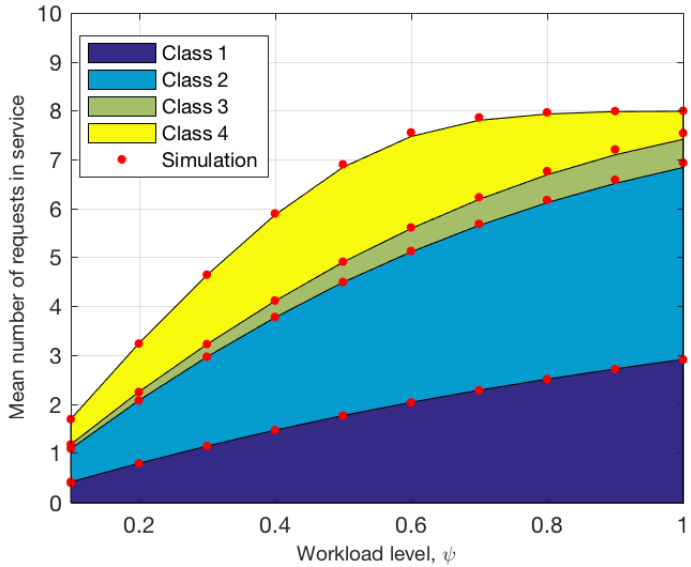


(a) Mean number of servers used by each class.

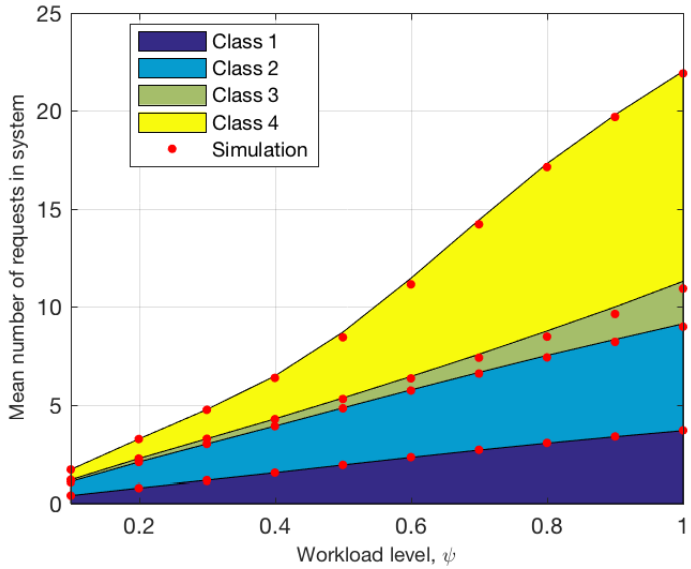


(b) Mean number of requests of each class in system.

Fig. 4. Queue with $L = 4$ priority classes and $C = 5$ servers, discrete sources with $N_\ell = \{3, 5, 3, 3\}$, $1/\mu_\ell = \{1.7, 1, 1.7, 0.3\}$ and $\alpha_\ell = \psi\{1.5, 0.4, 1.8, 0.5\}$ with $\psi = \{0.1, 0.2, \ldots, 1\}$,i.e., with ψ varying from 0.1 to 1 in steps of 0.1.

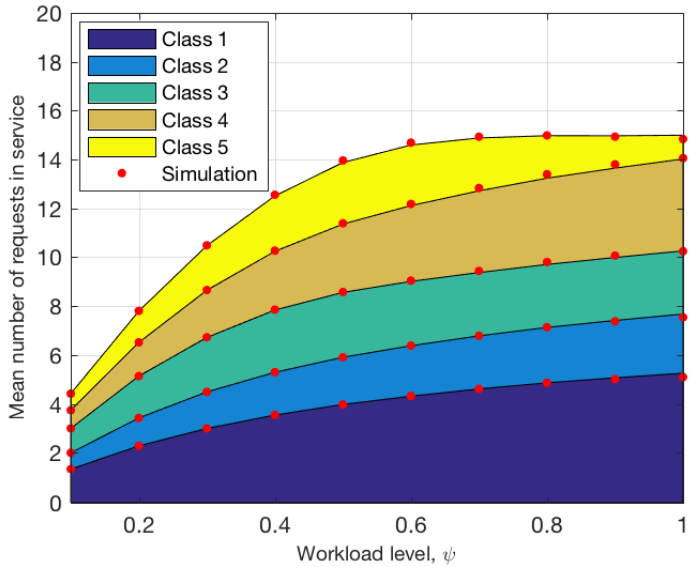


(a) Mean number of servers used by each class.

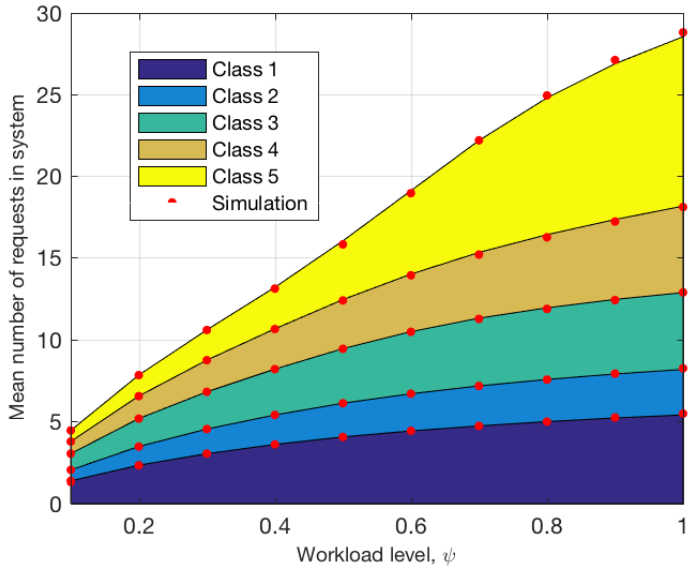


(b) Mean number of requests of each class in system.

Fig. 5. Queue with $L = 4$ priority classes and $C = 8$ servers, discrete sources with $N_\ell = \{12, 12, 6, 12\}$, $1/\mu_\ell = \{0.5, 1, 0.5, 0.3\}$ and $\alpha_\ell = \psi\{0.7, 0.6, 0.3, 1.5\}$ with $\psi = \{0.1, 0.2, \ldots, 1\}$,i.e., with ψ varying from 0.1 to 1 in steps of 0.1.



(a) Mean number of servers used by each class.



(b) Mean number of requests of each class in system.

Fig. 6. Queue with $L = 5$ priority classes and $C = 15$ servers, discrete sources with $N_\ell = \{8, 4, 6, 10, 12\}$, $1/\mu_\ell = \{1, 0.2, 0.1, 0.8, 0.3\}$ and $\alpha_\ell = \psi\{2, 10, 20, 1, 2\}$ with $\psi = \{0.1, 0.2, \ldots, 1\}$,i.e., with ψ varying from 0.1 to 1 in steps of 0.1.

Table 2. Distribution of the relative errors (in percent) for the mean number of servers used with discrete sources.

Class	Mean	Median	<1%	<5%	<10%	<15%	≥15%
1	0.09	0.07	100.00	100.00	100.00	100.00	0.00
2	0.12	0.09	99.79	100.00	100.00	100.00	0.00
3	0.16	0.10	99.36	100.00	100.00	100.00	0.00
4	0.30	0.14	94.27	100.00	100.00	100.00	0.00
5	0.51	0.18	85.99	99.58	99.79	100.00	0.00
All	0.23	0.10	95.88	99.92	99.96	100.00	0.00

Table 3. Distribution of the relative errors (in percent) for the mean number of requests in system with discrete sources.

Class	Mean	Median	<1%	<5%	<10%	<15%	≥15%
1	0.20	0.12	97.88	100.00	100.00	100.00	0.00
2	0.72	0.23	80.25	97.88	100.00	100.00	0.00
3	1.85	0.43	64.54	89.38	94.27	99.15	0.85
4	3.39	1.06	48.20	79.41	90.66	95.33	4.67
5	5.78	1.79	41.40	67.73	79.83	86.84	13.16
All	2.39	0.36	66.45	86.88	92.95	96.26	3.74

class ($1/\mu_\ell$) was drawn randomly between 0.05 and 2. The unitary rates of discrete sources (α_ℓ) varied randomly to explore a range of server utilization. The number of classes in the study was between 3 and 5 with close to five hundreds samples for each number of classes. The results of these experiments are summarized in the form of relative errors compared to the values obtained from simulation in Tables 2 and 3 for the mean number of servers used and for the mean number of requests, respectively. We observe that there is some tendency for the accuracy to decrease for lower priority classes although this tendency does not appear to be uniform. Overall, i.e., averaged over all classes, for the mean number of requests in service, the mean relative error is below 0.25% while the corresponding median value is 0.1%. We observe that in over 99% of cases the relative error is below 10%. For the mean number in system, the mean relative error is below 2.5%, the median value 0.36% and the errors remain below 10% in over 92% of cases.

It appears that larger relative errors tend to occur when the servers are saturated, i.e., busy virtually all the time. This is illustrated in Table 4 which shows the relative errors for the mean number of requests in system for a set of per-server utilization level. We observe that while at per-server utilization level between 0.6 and 0.8, the mean relative error is less than 2%, it rises to almost 4.6% for utilization levels between 0.8 and 1.0. Also, the percentage of samples with relative errors over 15% increases from around 1.6% to over 5%.

We studied also the behavior of the method as the number of servers increases. In Table 5 we observe that the accuracy clearly improves as the number of servers increases. In particular, the mean relative error for all classes decreases from close to 3.5% with between 2 to 4 servers to less than 0.5% when the number of servers is between 8 and 10. Similarly, the percentage of samples in which the relative error is under 1% grows from some 46% with 2 to 4 servers to almost 89% when the number of servers reaches between 8 and 10. Note that Tables 4 and 5 show the mean errors for the mean numbers in system averaged over all priority classes.

Table 4. Distribution of the relative errors (in percent) across all classes for the mean number of requests in system with discrete sources for different utilization levels.

Per-server utilization	Mean	Median	<1%	<5%	<10%	<15%	≥15%
0 to 0.3	0.18	0.14	97.22	100.00	100.00	100.00	0.00
0.3 to 0.6	0.57	0.14	87.10	97.90	99.47	99.94	0.06
0.6 to 0.8	1.80	0.42	66.81	89.56	95.85	98.43	1.57
0.8 to 1.0	4.58	1.96	36.23	68.12	85.02	94.69	5.31

Table 5. Distribution of the relative errors (in percent) across all classes for the mean number of requests in system with discrete sources for different numbers of servers.

Number of servers	Mean	Median	<1%	<5%	<10%	<15%	≥15%
2 to 4	3.36	1.22	46.26	78.00	90.02	96.26	3.74
5 to 7	1.11	0.33	73.12	95.10	98.92	99.88	0.12
8 to 10	0.47	0.16	88.65	98.72	99.86	100.00	0.00
11 to 13	0.30	0.12	94.24	99.30	100.00	100.00	0.00
14 to 16	0.16	0.09	98.32	100.00	100.00	100.00	0.00

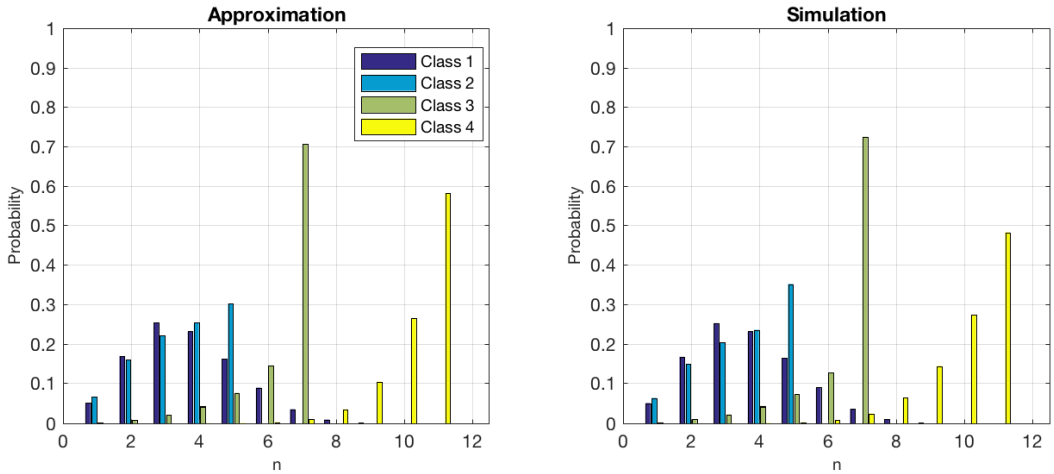


Fig. 7. Probability of the number of requests in system for $L = 4$ priority classes, $C = 3$ servers, discrete sources with $N_\ell = \{8, 4, 6, 10\}$,$\alpha_\ell = \{0.4, 2, 4, 0.2\}$ and $1/\mu_\ell = \{1, 0.2, 0.1, 0.8\}$.

So far, we have considered the mean number of servers busy and the mean number of requests in the system in comparisons between simulation and our approximation. Figure 7 shows an example of the probability distribution for the number of requests in the system obtained using the proposed approximation. The agreement in general shape and values of the distribution appears reasonably good.

Table 6. Distribution of the relative errors (in percent) for the mean number of servers used with Poisson arrival streams.

Class	Mean	Median	<1%	<5%	<10%	<15%	≥15%
1	0.07	0.05	100.00	100.00	100.00	100.00	0.00
2	0.09	0.05	99.80	100.00	100.00	100.00	0.00
3	0.17	0.08	97.78	100.00	100.00	100.00	0.00
4	0.37	0.10	88.51	99.80	100.00	100.00	0.00
5	0.89	0.15	76.41	95.56	99.19	100.00	0.00
All	0.32	0.07	92.50	99.07	99.84	100.00	0.00

Table 7. Distribution of the relative errors (in percent) for the mean number of requests in system with Poisson arrival streams.

Class	Mean	Median	<1%	<5%	<10%	<15%	≥15%
1	0.21	0.10	96.37	100.00	100.00	100.00	0.00
2	0.66	0.29	78.23	99.60	100.00	100.00	0.00
3	1.73	0.98	50.20	92.34	99.19	100.00	0.00
4	3.35	2.14	35.08	72.58	93.95	98.99	1.01
5	5.90	3.80	23.99	58.67	80.24	90.73	9.27
All	2.37	0.64	56.77	84.64	94.68	97.94	2.06

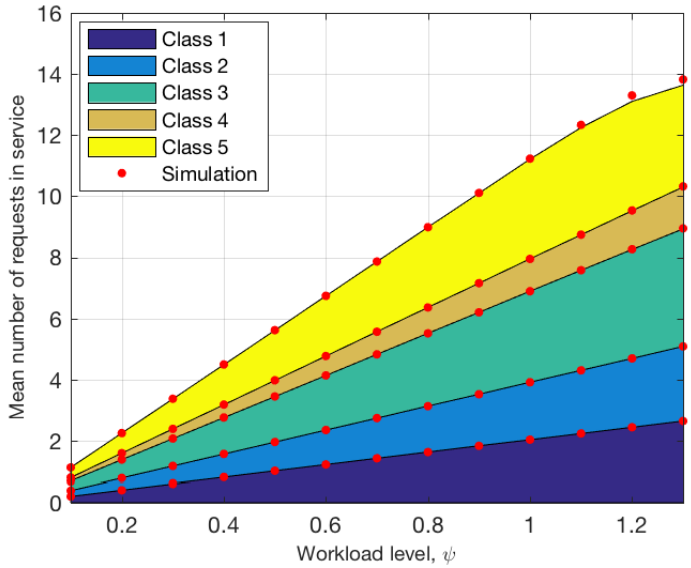
3.2 Results with Poisson arrival streams

Figure 8 shows an example of results obtained with individual Poisson arrival streams for each priority class, L=5 priority classes and C=14 servers. In this example the top 3 classes and the lowest priority class dominate the utilization of the servers. We observe a very good agreement between approximation and discrete-simulation results for the mean number of servers used by each priority class. For the mean numbers of requests in system, the approximation results happen to be a bit less accurate for the last two priority classes. This is visible in particular for class 4 "sandwiched" between two priority classes with significantly higher server utilizations.

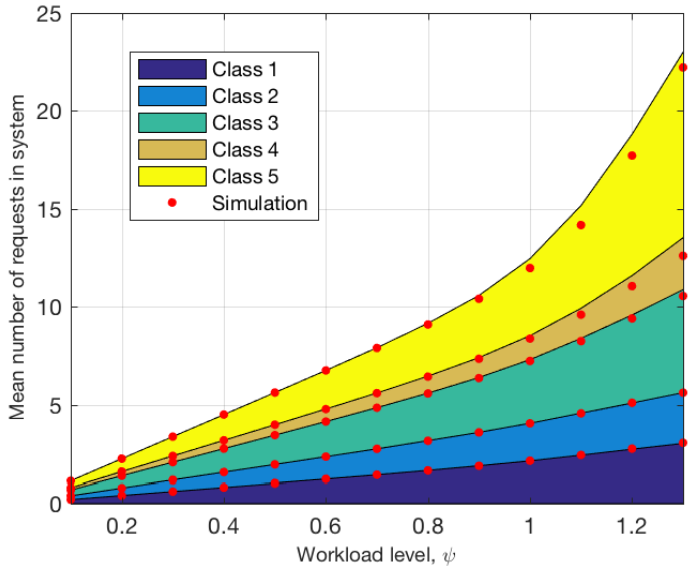
In Table 6 we summarize the results obtained for the mean number of servers used by each class with a total of $L = 5$ priority classes. The number of servers varies randomly between 2 and 16. As was the case for discrete request sources, the mean service time for a class ($1/\mu_\ell$) was drawn randomly between 0.05 and 2, and the intensities of the Poisson streams (λ_ℓ) varied randomly so as to explore a range of server utilization. The number of priority classes in our study was between 3 and 5 with close to five hundred samples for each number of classes. The accuracy for the mean number of servers used is clearly good: the mean error is below 1%. Table 7 shows corresponding results for the relative errors in the mean numbers of requests in the system. We observe, as was the case with discrete request sources, that the accuracy has a tendency to degrade for lower priority classes. Even so, the mean relative error for the mean number in system for class 5 in our case remains under 6%, while the corresponding median is below 4%.

Again, it has been our experience that the accuracy tends to degrade as the utilization level increases. This is illustrated in Table 8. The mean error for the mean number of requests in system increases from 0.45% when the per-server utilization is between 0.3 and 0.6 to almost 2.7% when the utilization level reaches between 0.8 and 1.0.

On the other hand, as shown in Table 9, the accuracy of the method tends to improve as the number of servers increases. The percentage of samples in which the error is below 1% increases



(a) Mean number of servers used by each class.



(b) Mean number of requests of each class in system.

Fig. 8. Queue with $L = 5$ priority classes, $C = 14$ servers, Poisson arrival streams with $N_\ell = \{14, 14, 14, 14, 14\}$, $1/\mu_\ell = \{0.4892, 1.8447, 0.6680, 0.7300, 1.1586\}$ and $\lambda_\ell = \psi\{4.1523, 1.0179, 4.4616, 1.4495, 2.8338\}$ with $\psi = \{0.1, 0.2, \ldots, 1.3\}$,i.e., with ψ varying from 0.1 to 1.3 in steps of 0.1.

Table 8. Distribution of the relative errors (in percent) across all classes for the mean number of requests in system with Poisson arrival streams for different utilization levels.

Per-server utilization	Mean	Median	<1%	<5%	<10%	<15%	≥15%
0 to 0.3	0.17	0.07	97.22	100.00	100.00	100.00	0.00
0.3 to 0.6	0.45	0.13	86.94	99.27	100.00	100.00	0.00
0.6 to 0.8	1.40	0.53	61.18	93.77	99.06	99.80	0.20
0.8 to 1.0	2.66	1.53	40.00	83.10	96.28	99.22	0.78

Table 9. Distribution of the relative errors (in percent) across all classes for the mean number of requests in system with Poisson arrival streams for different numbers of servers.

Number of servers	Mean	Median	<1%	<5%	<10%	<15%	≥15%
2 to 4	1.54	0.84	54.56	93.89	99.46	99.89	0.11
5 to 7	1.52	0.59	59.44	92.32	98.87	99.77	0.23
8 to 10	1.28	0.35	67.91	93.68	98.64	99.75	0.25
11 to 13	0.93	0.17	78.47	95.38	99.15	100.00	0.00
14 to 16	0.84	0.13	81.02	96.00	98.75	99.50	0.50

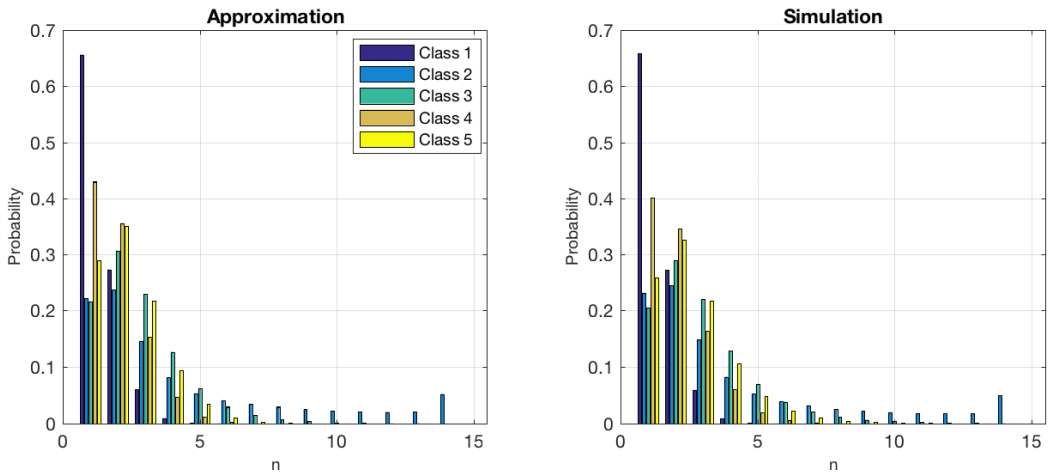


Fig. 9. Probability of the number of requests in system for $L = 5$ priority classes, $C = 6$ servers, Poisson arrival streams with $N_\ell = \{13, 13, 14, 14, 15\}$,$\lambda_\ell = \{0.4, 20, 1.33, 0.2, 0.2\}$ and $1/\mu_\ell = \{1, 0.05, 1, 4, 6\}$.

from less than 55% with between 2 and 4 servers to almost 68% with between 8 and 10 servers. The mean error decreases as the number of servers increases although it appears to remain under 2% in all cases anyway. To interpret these results properly, note that Tables 8 and 9 show the mean errors for mean numbers in system averaged over all priority classes.

As a final example, we show in Figure 9 the probability distribution for the number of requests per class obtained with our approximation and Poisson arrival streams for each class. Although not perfectly, the general shape of the distributions agrees reasonably well with simulation results.

Table 10. Distribution of the relative errors (in percent) for the mean number of requests in system with discrete sources dismissing samples with $C = 2$ or 3 servers.

Class	Mean	Median	<1%	<5%	<10%	<15%	≥15%
1	0.17	0.11	98.75	100.00	100.00	100.00	0.00
2	0.40	0.18	89.75	100.00	100.00	100.00	0.00
3	0.93	0.32	73.50	97.25	99.50	100.00	0.00
4	1.88	0.72	55.50	89.50	98.00	99.75	0.25
5	3.49	1.08	48.50	77.00	89.50	95.00	5.00
All	1.37	0.27	73.20	92.75	97.40	98.95	1.05

3.3 Additional discussion

Overall, the accuracy of the method appears to increase as the number of servers increases. This may not be surprising given the nature of our approximation since, intuitively, with larger numbers of servers, one would expect the priority levels to behave more independently. It is therefore likely that a significant fraction of examples in which the relative errors for the mean numbers of requests in the system exceed 10% or 15% are attributable to systems with 2 or 3 servers. Indeed, we show in Table 10 the distribution of relative errors for the mean numbers of requests in the system with discrete sources when the number of servers is between C =4 and C =16. These data points are the same as the ones used in Table 3 except that points corresponding to 2 or 3 servers were removed. Clearly, the accuracy of the method appears improved. In particular, for class 5, the percentage of relative errors below 10% is now close to 90% vs. close to 80%, and the percentage of relative errors under 15% increases from about 87% to 95%.

In our discussion of the accuracy we have concentrated on the mean number of servers used and on the mean number of requests in the system for each priority class. By Little's formula [1], the mean response time for each class can be computed as the ratio of the mean number of requests in the system to the attained request throughput. Since the accuracy of the mean number of servers used (proportional to the throughput) is generally very good, it follows that the accuracy of the mean response time per class is on the same order as that of the mean number of request in the system for the given class.

The next section concludes our paper.

4 CONCLUSIONS

We have presented an approximation method to obtain the steady-state probabilities of the number of requests in service and waiting in line for each priority class in a non-preemptive multiserver queue. The service times for each class are assumed to be exponentially distributed with potentially different mean values. For each class, requests arrive to the system according to separate quasi-Poisson processes with a predefined maximum number of requests in the system.

The proposed method allows us to solve each priority class "nearly separately", whereby the solutions of the different classes are only linked by the conditional probabilities of the numbers of requests of each class in the waiting line given the state of the system. The latter quantities undergird our approximation and are determined through a fixed-point iteration. In its principle, the method can accommodate an arbitrary number of priority classes although in practice its accuracy degrades somewhat with the number of classes. The computational complexity of the solution of each priority class grows only moderately as the number of classes increases (see Appendix).

In our numerical examples, we have considered quasi-Poisson arrivals resulting from a set of discrete memoryless sources, as well as individual Poisson streams for each priority class. To

illustrate the performance of a non-preemptive priority multiserver queue under varying load conditions, we have included results for 4 and 5 priority classes with discrete request sources and 5 priority classes in the case of individual Poisson streams. A study of the behavior of the proposed approximation using close to 1500 examples with between 3 and 5 priority classes and up to 16 servers indicates that its overall accuracy is generally good with the mean relative errors for the mean number of requests of a priority class in the system and the mean number of servers busy below 3% and 1%, respectively.

ACKNOWLEDGMENTS

The authors wish to thank the anonymous referees for their thorough and constructive review of an earlier version of this paper.

REFERENCES

[1] Gunter Bolch, Stefan Greiner, Hermann De Meer, and Kishor S Trivedi. 2006. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications.* John Wiley & Sons.
[2] Alexandre Brandwajn and Thomas Begin. 2016. Breaking the dimensionality curse in multi-server queues. *Computers & Operations Research* 73 (2016), 141–149.
[3] Richard H Davis. 1966. Waiting-time distribution of a multi-server, priority queuing system. *Operations Research* 14, 1 (1966), 133–136.
[4] HR Gail, SL Hantler, and BA Taylor. 1988. Analysis of a non-preemptive priority multiserver queue. *Advances in Applied Probability* 20, 4 (1988), 852–879.
[5] Mor Harchol-Balter, Takayuki Osogami, Alan Scheller-Wolf, and Adam Wierman. 2005. Multi-server queueing systems with multiple priority classes. *Queueing Systems* 51 (2005), 331–360.
[6] Xiaohong Huang and Maode Ma. 2011. A performance model for differentiated service over single-hop passive star coupled WDM optical networks. *Journal of Network and Computer Applications* 34, 1 (2011), 183–193.
[7] Edward PC Kao and Kumar S Narayanan. 1990. Computing steady-state probabilities of a nonpreemptive priority multiserver queue. *ORSA Journal on Computing* 2, 3 (1990), 211–218.
[8] Offer Kella and Uri Yechiali. 1985. Waiting times in the non-preemptive priority M/M/c queue. *Stochastic Models* 1, 2 (1985), 257–262.
[9] Robert F Kern. 2009. IBM System z & DS8000 Technology Synergy. *IBM ATS Americas Disk Storage* (2009), 1–25.
[10] Bara Kim, Jeongsim Kim, and Ole Bueker. 2021. Non-preemptive priority M/M/m queue with serversâĂŹ vacations. *Computers & Industrial Engineering* 160 (2021), 107390.
[11] A Krishnamoorthy, S Babu, and Viswanath C Narayanan. 2008. MAP/(PH/PH)/c queue with self-generation of priorities and non-preemptive service. *Stochastic Analysis and Applications* 26, 6 (2008), 1250–1266.
[12] Kristina Kunert, Magnus Jonsson, Annette Böhm, and Tomas Nordström. 2017. Providing efficient support for real-time guarantees in a fibre-optic AWG-based network for embedded systems. *Optical Switching and Networking* 24 (2017), 47–56.
[13] Erol A Peköz. 2002. Optimal policies for multi-server non-preemptive priority queues. *Queueing systems* 42, 1 (2002), 91–101.
[14] Harry G Perros, Yves Dallery, and Guy Pujolle. 1992. Analysis of a queueing network model with class dependent window flow control. In *[Proceedings] IEEE INFOCOM'92: The Conference on Computer Communications.* IEEE, 968–977.
[15] GV Krishna Reddy, R Nadarajan, and PR Kandasamy. 1993. A nonpreemptive priority multiserver queueing system with general bulk service and heterogeneous arrivals. *Computers & operations research* 20, 4 (1993), 447–453.
[16] Andrei Vasilievich Sleptchenko. 2003. *Multi-class, multi-server queues with non-preemptive priorities.* Eurandom.
[17] William J Stewart. 2021. *Numerical solution of Markov chains.* CRC press.
[18] Dietmar Wagner. 1997. Analysis of mean values of a multi-server model with non-preemptive priorities and non-renewal input. *Stochastic Models* 13, 1 (1997), 67–84.
[19] Dietmar Wagner. 1997. Waiting times of a finite-capacity multi-server model with non-preemptive priorities. *European Journal of Operational Research* 102, 1 (1997), 227–241.
[20] TM Williams. 1980. Nonpreemptive multi-server priority queues. *Journal of the Operational Research Society* 31 (1980), 1105–1107.

A BALANCE EQUATIONS FOR CLASS ℓ WHEN $m = 0, \ldots, C-1 \quad (k_\ell = 0)$

These equations can be derived directly by considering the flow of transitions into and out of each state $p(m_1, \ldots, m_L, k_\ell)$. One can also obtain these same equations from the balance equations for $p(m_1, \ldots, m_L, k_1, \ldots, k_L)$ by summing them over all k_i with $i \neq \ell$.

$$\begin{aligned} p(m_1, \ldots, m_L, k_\ell = 0) \left[\sum_{i=1}^{L} (\lambda_i(m_i) + m_i \mu_i) \right] \\ = \sum_i p(m_1, \ldots, m_i - 1, \ldots, m_L, k_\ell = 0) \lambda_i(m_i - 1) \\ + \sum_i p(m_1, \ldots, m_i + 1, \ldots, m_L, k_\ell = 0)(m_i + 1)\mu_i \\ .P\{k_1 = 0, \ldots, k_{\ell-1} = 0, k_{\ell+1} = 0, \ldots, k_L = 0 | m_1, \ldots, m_i + 1, \ldots, m_L, k_\ell = 0\} \end{aligned} \tag{8}$$

B BALANCE EQUATIONS FOR CLASS ℓ WHEN $m = C$ AND $k_\ell = 0$

$$\begin{aligned} p(m_1, \ldots, m_L, k_\ell = 0) \\ . \left[\lambda_\ell(m_\ell) + \sum_{i=1:i\neq\ell}^{L} m_i \mu_i \left[1 - P\{k_1 = 0, \ldots, k_{i-1} = 0, k_i > 0 | m_1, \ldots, m_L, k_\ell = 0\} \right] + m_\ell \mu_\ell \right] \\ = \sum_{i=1}^{L} p(m_1, \ldots, m_i - 1, \ldots, m_L, k_\ell = 0) \lambda_i(m_i - 1) \\ + p(m_1, \ldots, m_L, k_\ell = 1) m_\ell \mu_\ell . P\{k_1 = 0, \ldots, k_{\ell-1} = 0 | m_1, \ldots, m_L, k_\ell = 1\} \\ + \sum_{i=1:i\neq\ell}^{L} p(\ldots, m_i + 1, \ldots, m_\ell - 1, \ldots, k_\ell = 1)(m_i + 1)\mu_i \\ .P\{k_1 = 0, \ldots, k_\ell = 0 | \ldots, m_i + 1, \ldots, m_\ell - 1, \ldots, k_\ell = 1\} \\ + \sum_{j=1}^{L} \sum_{i=1:i\neq j}^{L} p(\ldots, m_i - 1, \ldots, m_j + 1, \ldots, k_\ell = 0)(m_j + 1)\mu_i \\ .P\{k_1 = 0, \ldots, k_{i-1} = 0, k_i > 0 | \ldots, m_i - 1, \ldots, m_j + 1, \ldots, k_\ell = 0\} \end{aligned} \tag{9}$$

C POSSIBLE ITERATIVE SOLUTION OF BALANCE EQUATIONS FOR A PRIORITY CLASS

As mentioned in Section 2, there are several possible methods to solve the balance equations for a single priority class (equations (1), (8), (9)). One of them is to use a semi-numerical approach based on the relationship $p(m_1, \ldots, m_L, k_\ell) = p_\ell(k_\ell) p(m_1, \ldots, m_L | k_\ell)$ and on the form of $p_\ell(k_\ell)$ given by formula (2), where $p_\ell(k_\ell)$ is the steady-state probability that there are k_ℓ requests in the waiting line for class ℓ.

As an example, using a superscript to denote the iteration number, based on the balance equation (8), we have for $m = 0, \ldots, C-1$

$$
\begin{aligned}
p^{\tau}(m_1,\ldots,m_L|k_\ell=0)\left[\sum_{i=1}^{L}(\lambda_i(m_i)+m_i\mu_i)\right] \\
=\sum_i p^{\tau}(m_1,\ldots,m_i-1,\ldots,m_L|k_\ell=0)\lambda_i(m_i-1) \\
+\sum_i p^{\tau-1}(m_1,\ldots,m_i+1,\ldots,m_L|k_\ell=0)(m_i+1)\mu_i \\
.P\{k_1=0,\ldots,k_{\ell-1}=0,k_{\ell+1}=0,\ldots,k_L=0|m_1,\ldots,m_i+1,\ldots,m_L,k_\ell=0\}
\end{aligned}
\tag{10}
$$

Similarly, we can transform the other balance equations into equations for $p(m_1,\ldots,m_L|k_\ell)$ for $m=C$ and other values of k_ℓ. At each iteration, the probabilities $p(m_1,\ldots,m_L|k_\ell)$ have to be normalized for each value of $k_\ell=0,1,\ldots$ The iteration continues until a convergence criterion (e.g., $\max_{(m_1,\ldots,m_\ell,\ldots,m_L),k_\ell=0,\ldots,N_\ell}(|1-p^{\tau}(m_1,\ldots,m_L|k_\ell)/p^{\tau-1}(m_1,\ldots,m_L|k_\ell)|)<\epsilon$, where ϵ is the desired convergence stringency) is reached. This is summarized in Algorithm 1.

Algorithm 1 Possible iterative solution of a single priority class

1: Select the initial distribution $p^0(m_1,\ldots,m_\ell,\ldots,m_L|k_\ell)$ for $k_\ell=0,\ldots,N_\ell$ such that $\sum_{m_1,\ldots,m_L}p^0(m_1,\ldots,m_\ell,\ldots,m_L|k_\ell)=1$.
2: **for** $\tau=1,2,\ldots,\tau_{\max}$ **do**
3: **for** each $k_\ell=0,1,\ldots,N_\ell$ **do**
4: Compute new values for $p^{\tau}(m_1,\ldots,m_\ell,\ldots,m_L|k_\ell)$ using equations (10) and analogous equations obtained from (1) and (9).
5: Normalize the newly computed values of $p^{\tau}(m_1,\ldots,m_\ell,\ldots,m_L|k_\ell)$ so that $\sum_{m_1,\ldots,m_L}p^{\tau}(m_1,\ldots,m_\ell,\ldots,m_L|k_\ell)=1$.
6: **end for**
7: **if** $\max_{(m_1,\ldots,m_L),k_\ell=0,\ldots,N_\ell}|1-p^{\tau}(m_1,\ldots,m_L|k_\ell)/p^{\tau-1}(m_1,\ldots,m_L|k_\ell)|<\epsilon$ **then**
8: Go to Line 11
9: **end if**
10: **end for**
11: Compute $p_\ell(n_\ell)$, $\bar{n}_\ell$ and $\bar{m}_\ell$ using formulas (2) (with (3) and (4)), (6) and (7). Compute also $P\{k_\ell=0|m_1,\ldots,m_L\}=p(m_1,\ldots,m_L|k_\ell=0).p(k_\ell=0)/\sum_{k_\ell=0}^{N_\ell}p(k_\ell).p(m_1,\ldots,m_L|k_\ell)$ for use in the solution of other priority classes.

In our numerical studies, we used $\epsilon=10^{-7}$ and $\tau_{\max}=10^4$.

D COMPARISON OF THE NUMBER OF STATES BETWEEN EXACT AND APPROXIMATE SOLUTIONS

As mentioned earlier, the proposed approximation results in a significantly reduced number of states compared to the exact solution using the full state description $p(m_1,\ldots,m_L,k_1,\ldots,k_L)$. Figure 10 shows the ratio of the numbers of states in both approaches as a function of the maximum number of requests per class (in the case where all priority classes have the same maximum number of requests). This ratio in our case is on the order of 10^3. In Figure 11 we examine how the numbers of states explored in both approaches change as the number of classes L increases. It is clear that the numbers of states in our approximation grows very moderately compared to the increase of the number of states in the exact solution.

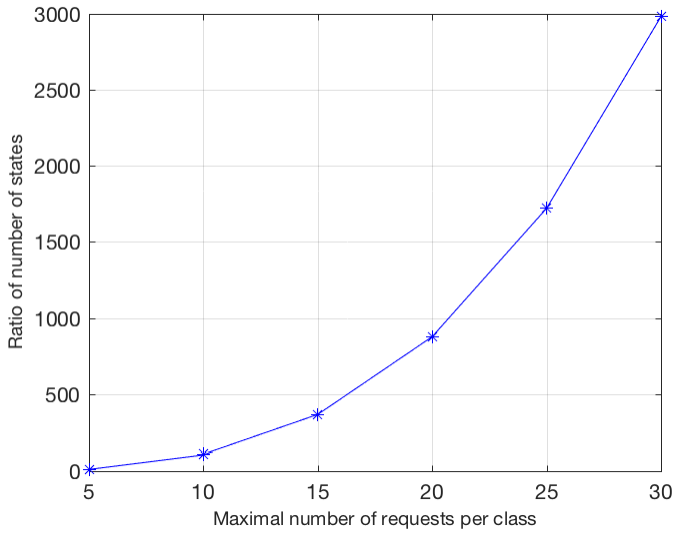


Fig. 10. Ratio of the number of states of the exact solution to our approximation as a function of the maximal number of requests per class with C=5 servers and L=5 classes.

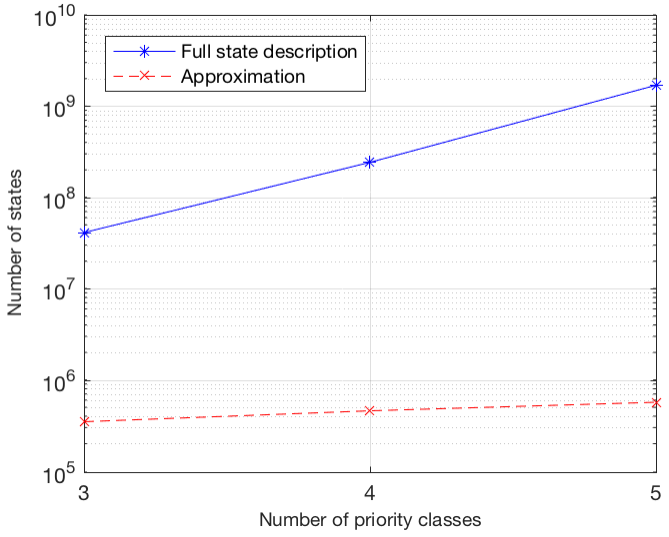


Fig. 11. Number of states explored with C=5 servers and N_ℓ=30 for all priority classes as a function of the number of priority classes.

We used the following formulas to compute the number of states explored in the exact solution and in our approximation:

For the full state description $(m_1, \ldots, m_L, k_1, \ldots, k_L)$:

$$\sum_{m_1=0}^{C-1} \sum_{m_2=0}^{C-1-m_1} \cdots \sum_{m_L=0}^{C-1-m_1 \cdots - m_{L-1}} 1 + \sum_{m_1=0}^{C} \sum_{m_2=0}^{C-m_1} \cdots \sum_{m_L-1=0}^{C-m_1 \cdots - m_{L-2}} (1+N_1-m_1)(1+N_2-m_2)\ldots(1+N_L-m_L^*) \tag{11}$$

with $m_L^* = C - (m_1 + \cdots + m_{L-1})$.

For the state description used in the approximation, for class $\ell = 1, \ldots, L$,

$$\sum_{m_1=0}^{C-1} \sum_{m_2=0}^{C-1-m_1} \cdots \sum_{m_L=0}^{C-1-m_1 \cdots - m_{L-1}} 1 + \sum_{m_1=0}^{C} \sum_{m_2=0}^{C-m_1} \cdots \sum_{m_L-1=0}^{C-m_1 \cdots - m_{L-2}} (1 + N_\ell - m_\ell^*) \tag{12}$$

with $m_L^* = C - m_1 \cdots - m_{L-1}$ when $\ell = L$ and $m_\ell^* = m_\ell$ otherwise.