Analyzing priority queues with 3 classes using tree-like processes

B. Van Houdt and C. Blondia

April 4, 2006

Abstract

In this paper we demonstrate how tree-like processes can be used to analyze a general class of priority queues with three service classes, creating a new methodology to study priority queues. The key result is that the operation of a 3-class priority queue can be mimicked by means of an alternate system that is composed of a single stack and queue. The evolution of this alternate system is reduced to a tree-like Markov process, the solution of which is realized through matrix analytic methods. The main performance measures, i.e., the queue length distributions and loss rates, are obtained from the steady state of the tree-like process through a censoring argument. The strength of our approach is demonstrated via a series of numerical examples.

1 Introduction

In this paper we introduce a new methodology to analyze priority queues with three classes of service. The key feature of our approach lies in reformulating the traditional three queue problem into a combined queue and stack problem. We show that the behavior of the reformulated problem can be captured by a Markov chain with a tree structured state space [16, 20]. This Markov chain—that neither fits within the GI/M/1 or M/G/1 paradigm—is reduced to a binary tree-like process [3, 17], which is a special case of a tree structured QBD Markov chain [19]. The queue length distributions and loss rate, which are the main performance measures captured by this methodology, are obtained by applying a censoring argument on the tree-like process.

We model the incoming jobs as a Markov arrival process with marked jobs, i.e., the MMAP[3] process [5, 4]. The MMAP[3] is a very general set of arrival processes that allows correlated inter-arrival times and correlation between the classes of consecutive jobs. Moreover, the input traffic does *not* need to consist of three independent streams (one for each priority class). Furthermore, we allow that the amount of time needed to execute a job depends on its class, while the processing time of consecutive jobs of the same class is independent and identically distributed (iid). Thus, in general, one cannot simply lump the

priority one and two traffic into a single class as this would create correlation into the service times of the lumped job class.

We will demonstrate our approach on a discrete-time preemptive priority queueing system without batch arrivals. The approach can be adapted such that it also applies to the continuous time setting or to nonpreemptive systems. In order to identify the key features of our new methodology, we restrict ourselves to the case where the service time of a class-i job is geometric with a mean $1/p_i$. Phase-type distributed service times can also be incorporated into the model. The methodology is exact if the waiting rooms for the priority 1 and 3 jobs are both finite while the waiting room for the priority 2 jobs has an infinite capacity. The infinite nature of the priority 2 queue does not jeopardize the practical relevance of our model as priority queues are mainly effective if sufficient jobs have the lowest priority. Therefore, hardly any priority 1 or 2 jobs will leave the system without receiving full service; hence, whether the priority 2 queue is infinite or finite and sufficiently large makes little difference.

The study of priority queues has a long history that started in the 1950s (according to Miller [9]) and a plethora of applications in both communication and manufacturing systems has benefited from its development. Starting from the elementary M/M/1 priority queue, the complexity and generality of the models under study has grown ever since. Various, more recent studies have focused on priority queues with Markovian input traffic and phase-type (or general) service requirements [1, 2, 7, 13, 14, 15].

In [1, 2] Alfa *et al.* studied priority queues with $C \geq 2$ classes, MMAP[C] input and phase-type services by setting up a QBD Markov chain [6] that generalizes Miller's result [8]. Although the queues considered in [1, 2] are infinite, a similar approach can be used to solve the system when some of the queues are finite. In case of 3 service classes and a finite capacity queue of size K for each of the class-1 and class-3 jobs, the resulting QBD has size $O(K^2)$ block matrices (the structure of which can be exploited to compute the rate matrix R). Our approach has the advantage that the blocks characterizing the binary tree-like process are of size $O(K)$ only. The arrival process in [1, 2] is however somewhat more general as one job per service class may arrive at each time instant, as opposed to one job in total at each instant in our model.

Takine and Hasegawa [14] analyzed the preemptive priority queue with $C \geq 2$ service classes, C independent MAP arrival streams and state-dependent service times, using the workload process of the queue. They generalize the model presented by Machihara in [7] that relied on the diagonalization of some matrices. In our setting we do not require independent MAP arrivals streams and focus on the queue length distributions, while the approach in [14] can deal with more general service requirements and is especially effective to compute the waiting time distributions. Takine [13, 15] also studied the nonpreemptive priority MAP/G/1 queue and derived various formulas for the generating function of the queue length and Laplace-Stieltjes transform of the

waiting time for each job class.

The remainder of the paper is organized as follows. The stack and queue model reformulation of the 3-class priority queue is given Section 2. Section 4 presents a Markov chain that captures the evolution of this model for the specific queueing system described in Section 3. In Section 5 we show how to reduce this Markov chain to a tree-like process, while Section 6 discusses the computation of its steady-state probabilities and of the performance measures of interest. We demonstrate our approach via a set of numerical examples in Section 7.

2 The 3-class priority queue and a combined stack and queue problem

Consider a preemptive priority queueing system with three service classes consisting of a single server and three waiting rooms, one for each service class. Assume that class-1 jobs have the highest priority, followed by the class-2 jobs and finally the class-3 jobs. The processing times of consecutive jobs belonging to the same class are iid. Let the capacity of both[1] the class-1 and class-3 waiting rooms be finite and of size $K > 0$, while the class-2 waiting room is considered infinite in size. Jobs that find their corresponding waiting rooms full, leave the system without being executed. A class-3 job that is pushed out of the service facility will (re)occupy the leading position in the class-3 waiting room, possibly pushing out the youngest waiting class-3 job. We refer to this traditional model as the three queue model (3Q). We now replace the class-2 and class-3 waiting rooms by a single, infinite size stack, to obtain a new model termed the stack & queue (S&Q) model. We will show that the state of the 3Q model is uniquely identified by the current state of the S&Q model.

The evolution of the S&Q model goes as follows. An example to further clarify the relationship between the two systems is depicted in Figure 1. During the description of the $S\&Q$ model, we will refer to this figure and indicate the time instants at which such an event occurs. As long as there is at least one class-1 job in the system (meaning a class-1 job is being processed), we use the queue to store priority 1 jobs (time 1–3, 6–9 and 15), while all arriving class-2 or class-3 jobs are simply pushed on the stack (time 2, 3 and 8). If there are no other class-1 jobs in the system when a priority 1 job arrives, it will occupy the server. The possible class-2 or class-3 job that was in service will be pushed on the stack[2] together with the entire contents of the queue, making the queue available again for class-1 jobs (time 1,6 and 15). Hence, as far as the class-1

[1]There is no need to select the same capacity $K > 0$ for the class-1 and class-3 waiting rooms. It only simplifies the presentation of the model.

[2]The reason for first pushing the possible class-2 job on the stack (before the contents of the queue) will become apparent further on.

jobs are concerned there is no difference between the 3Q model or the S&Q model. During periods when there is no class-1 traffic present, we utilize the queue for other means (time 0, 4–5, 10–14 and 16–18). Notice, the number of priority 3 jobs on the stack might become larger than K, the capacity of its waiting room in the $3Q$ model, as we simply push all class-3 arrivals on the stack when a class-1 job is in service (time 5–13). Further on, it will become apparent that the number of class-3 jobs waiting for service in the 3Q model is identical to the minimum of K and the number of waiting class-3 jobs in the S&Q model. There is no need to keep track of the correct number of class-3 jobs at all times, it suffices that the number is identical to the one of the 3Q system when the server becomes available for the class-3 jobs. Thus, the number of class-3 job losses is identical in both models, however, the times at which the these losses occur need not to coincide.

Assume that a class-1 job is completed and there are no other class-1 jobs ready to start service, that is, the server is now available for class-2 or class-3 traffic (time 4, 10 and 16). Moreover, the queue used to store the class-1 jobs is now empty and can be utilized for other purposes as long as there is no class-1 traffic in the system. In such a case we start looking for a class-2 job in the stack, by popping jobs from the stack until we encounter a class-2 job. This class-2 job is placed in the service facility. As the service times within a class are iid, the identity of and order in which the jobs of a specific class are served is irrelevant, as long as an interrupted job is continued when the server becomes available again to this job's priority class. This is realized by assuming that the class-2 jobs mutually switch positions in the stack such that the oldest ones are on top.

The class-3 jobs that are removed from the stack while searching the stack for a class-2 job, are stored in the queue (time 4, 10, 14 and 16). Thus, as long as there are no class-1 jobs, we use the queue (temporarily) to store class-3 jobs. If the queue is full (with class-3 jobs) when popping a class-3 job from the stack, the job leaves the system without being performed (time 10 and 14). As with the class-2 jobs, we assume that the class-3 jobs also mutually exchange positions such that the youngest jobs get dropped first, while the elder receive service before the younger ones (that is why the lost jobs in the $3Q$ model immediately move to the lowest class-3 positions on the stack at time 5 and 8 in the S&Q model). If a class-2 job is completed (and there is no class-1 arrival) we start to pop jobs from the stack in search of another class-2 job (time 14). If no such job is encountered, meaning that the stack is empty and all class-3 jobs are now present in the queue, we start executing the priority 3 jobs.

Finally, if a class-2 job arrives while a class-3 job is being performed, it will interrupt its service and the class-3 job returns to the queue. Hence, we do not push the entire contents of the queue on the stack in such case. Class-3 arrivals that occur while a class-2 or class-3 job is being executed are always directly stored in the queue (time 5), or lost if the queue is full, while class-2

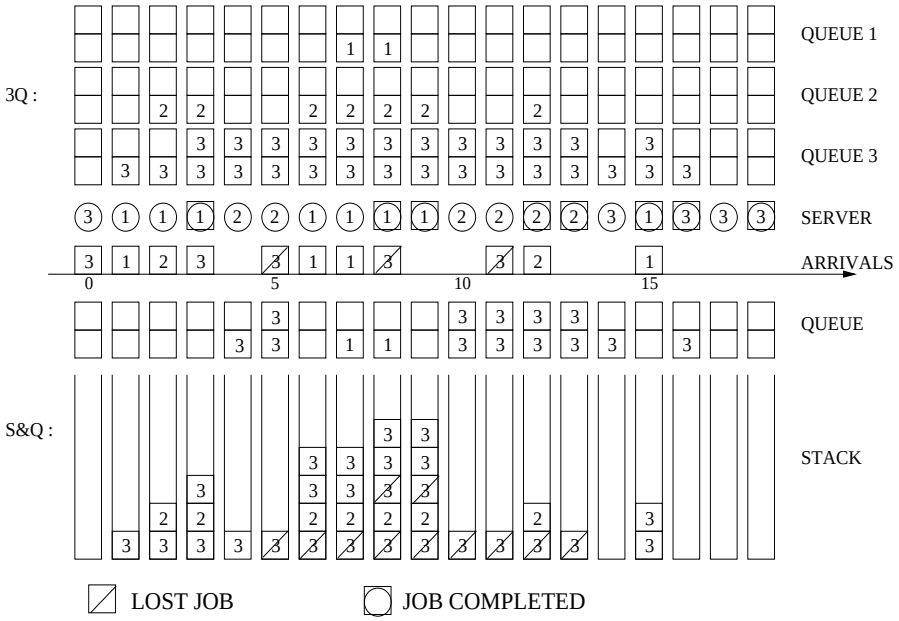


Figure 1: System dynamics during a busy period for the $3Q$ and $S\&Q$ system

arrivals are pushed onto the stack in such cases. The execution of class-3 jobs can only start when the stack is empty (time 0, 14 and 16-18), therefore, the number of lost jobs during a busy period is identical in both the $3Q$ and $S\&Q$ system.

3 Priority Queueing Model

In this section we present a detailed description of the priority queueing system used to demonstrate our novel approach in detail. As indicated in the introduction, the model can be extended in various directions, e.g., phase-type services, nonpreemptive systems, continuous time models, etc. Our model is the discrete time version of the $3Q$ model discussed in the previous section, with the following arrival process and service time requirements. The job arrival process is a MMAP[3] process with m phases. It is characterized by a set of four matrices D_0, D_1, D_2 and D_3. The (i, j)-th entry of D_0 represents the probability that the phase changes from i (at time n) to j (at time $n+1$), while no arrivals occur (at time n). Similarly, D_i gives the probability for the same event, but with a class-i job arrival (at time n), for $i = 1, 2$ and 3. Define θ as the stochastic vector that satisfies $\theta(D_0 + D_1 + D_2 + D_3) = \theta$. The arrival rate of the class-i jobs λ_i can be computed as $\lambda_i = \theta D_i e_m$ (where e_m is a column vector of size m with all its entries equal to one). The time needed to process a class-i job is assumed to be geometric with a mean of $1/p_i$ time

slots.

As we are dealing with a discrete time model, we need to specify the order in which simultaneously occurring events take place. The following order is respected in our model: (i) A possible arrival at time n occurs first. The arriving job is placed in its corresponding waiting room. If a class-1 or class-3 job finds its waiting room full upon arrival, it immediately abandons the system (without being performed). (ii) Next, a possible service completion occurs, making the server ready to process another (waiting) job. Even if there is no service completion, the job being processed returns to its corresponding waiting room in case a job with a priority exceeding that of the job in progress arrived during step (i). (iii) If the server is empty after step (ii), the next job to be performed is selected from the waiting rooms according to the rules of the priority system. Notice, if a class-1 (or 3) job arrival coincides with a job completion and there were already K class-1 (or 3) jobs waiting to be processed, the newly arriving job will abandon the system due to the above-mentioned event order. Other orders of events can be considered as well, but may increase the complexity of the tree-like process reduction technique introduced further on. When setting up a Markov chain, we observe the system at time n just prior to any events, e.g., arrivals, service completions, etc.

4 The Markov Chain

In this section we make use of the relationship between the $3Q$ and $S\&Q$ model to analyze the priority queueing system described in Section 3. A Markov chain with a tree structured state space [16, 20] that captures the system dynamics of the $S\&Q$ model is set up. Let M_n denote the phase of the MMAP[3] job arrival process at time n. Due to the geometric nature of the job durations it suffices to keep track of the class of the job in progress. Let $S_n = i$, for $i = 1, 2$ and 3, if a class-i job is executed at time n and let $S_n = 0$ if there are no jobs in the system. Define Q_n as the queue contents at time n (where Q_n takes values in the range 0 to K) and X_n as the stack contents at the same time instant.

The stack contents will be denoted as a string of integers $J = j_1 j_2 \dots j_{|J|}$, where $j_i = 2$ or 3 for $i = 1, \dots, |J|$ and $|J|$ represents the length of the string. The empty string is denoted as $J = \emptyset$. The rightmost element of the string (i.e., $j_{|J|}$) corresponds with the class of the job at the top of the stack, while the leftmost element reflects the class of the bottom entry. Elements pushed onto or popped from the stack are therefore added to or removed from the right-hand side of the string J. Let $f(J, i)$ denote the string holding the i rightmost elements of the sting J (for $i \leq |J|$). We use the '+' to denote the concatenation on the right and '-' to represent the deletion from the right. For example, if $J_1 = 3223$ and $J_2 = 322$, then $J_1 + J_2 = 3223322$ and $J_1 - f(J_1, 2) = 32$. We also use the shorthand $J = 3^r$ if J is a string of

length r with all its elements equal to 3. For convenience we also denote the string $J = \emptyset$ as 3^0. We refer to the set of states with $X_n = J$ as node J of the tree. The node $J = \emptyset$ is further split into two subnodes $\emptyset_0$ and $\emptyset_1$. With some abuse of notation we sometimes write $J + \emptyset_1 = \emptyset_1 + J = \emptyset_0 + J = J + \emptyset_0 = J$.

It is not hard to see that (X_n, S_n, Q_n, M_n) forms a Markov chain for the $S\&Q$ model of the priority queueing system discussed in Section 3. We partition its state space into three sets of states, when joined the first two sets form the node $X_n = \emptyset$. For each of the three sets below the job arrival process has a range equal to $1, \ldots, m$.

- $X_n = \emptyset_0$: there are no class-1 or class-2 jobs in the system. This implies that $(S_n, Q_n) = (0, 0)$ or $(3, q)$ for $q = 0, \ldots, K$. Notice, whenever the Markov chain is in one of these states we use the queue to store class-3 jobs.

- $X_n = \emptyset_1$: the stack is empty and a class-1 or class-2 job is processed. If $S_n = 1$, meaning a class-1 job is being performed, we use the queue for class-1 jobs (and the range of Q_n equals $0, \ldots, K$). The empty stack indicates that there are no class-2 or class-3 jobs in the system. If $S_n = 2$, we utilize the queue for class-3 jobs. The range of Q_n is also equal to $0, \ldots, K$. Apart from the class-2 job in service, there are no other class-1 or class-2 jobs present.

- $X_n = J$, with $|J| > 0$: the stack is not empty. From the description of the $S\&Q$ model we know that $S_n = 1$ or 2. As in the $X_n = \emptyset_1$ case, the range of Q_n also equals $0, \ldots, K$ and the class of the jobs in the queue depends in the same manner on the value of S_n.

To facilitate the description of the transition probabilities, we introduce the following notation. Let $P_{J,J'}((s, q) \to (s', q'))$ be an $m \times m$ matrix whose (i, i')-th element equals the probability that $(X_{n+1}, S_{n+1}, Q_{n+1}, M_{n+1}) = (J', s', q', i')$ given that $(X_n, S_n, Q_n, M_n) = (J, s, q, i)$. When presenting the expression for these $m \times m$ matrices, we will only briefly touch upon some of the equalities, the others are found in an analogue manner.

(A) Let us start with the case where $X_n = \emptyset_0$. This means that there are no class-1 or class-2 jobs in the system and the queue is available/utilized for class-3 jobs. The Markov chain remains in the node $\emptyset_0$ unless there is a class-1 or class-2 job arrival at time n (see Eq. (1)). From hereon we denote $(1 - p_i)$

as $\bar{p}_i$.

$$P_{\emptyset_0,\emptyset_0}((s,q) \to (s',q')) =$$
$$\begin{cases} D_0 & s = s' = q = q' = 0, \\ D_3 & s = 0, s' = 3, q = q' = 0, \\ p_3 D_0 & s = 3, s' = 0, q = q' = 0, \\ & \text{or } s = s' = 3, K > q = q' + 1 > 0, \\ \bar{p}_3 D_0 + p_3 D_3 & s = s' = 3, q = q' < K, \\ \bar{p}_3 D_3 & s = s' = 3, q = q' - 1 < K \\ \bar{p}_3 D_0 & s = s' = 3, q = q' = K, \\ p_3(D_0 + D_3) & s = s' = 3, q = q' + 1 = K, \\ 0 & \textit{otherwise}. \end{cases} \tag{1}$$

The expression for $q = K$ can be understood by recalling that, due to the event order, an arriving job that finds the queue fully occupied is lost irrespective of whether there is a service completion at this particular time. This has to be kept in mind when considering other, future expressions with $q = K$. In the event of a class-2 arrival, we find $X_{n+1} = \emptyset_1$ and the queue remains available to the class-3 jobs (see Eq. (2), line 1 with $s' = 2$ and line 3–5). This is no longer the case if a class-1 job arrival occurs, because we need to push all q class-3 jobs present in the queue on the stack (together with the class-3 job residing in the server unless this job ended at time n); hence, $X_{n+1} = 3^{\max(K,q(+1))}$ (see Eq. (2), line 1–2, and (6), line 4–6):

$$P_{\emptyset_0,\emptyset_1}((s,q) \to (s',q')) =$$
$$\begin{cases} D_{s'} & s = 0, s' = 1 \text{ or } 2, q = q' = 0, \\ p_3 D_1 & s = 3, s' = 1, q = q' = 0, \\ p_3 D_2 & s = 3, s' = 2, q = q' < K, \\ \bar{p}_3 D_2 & s = 3, s' = 2, q = q' - 1 < K \\ D_2 & s = 3, s' = 2, q = q' = K, \\ 0 & \textit{otherwise}. \end{cases} \tag{2}$$

(B) Next, assume that $X_n = \emptyset_1$. If the (class-1 or class-2) job in progress finishes and leaves no other class-1 or class-2 jobs behind in the system, we have $X_{n+1} = \emptyset_0$ (see Eq. (3), notice that the q jobs in the queue are of class-3 if $s = 2$). Otherwise, we remain in the node $\emptyset_1$ (see Eq. (4) and (5) for $J = \emptyset_1$) unless (a) there is still a class-1 job in the system at time $n+1$ and a class-2 or class-3 job arrived at time n which we need to push on the stack (see Eq. (6) and Eq. (7), lines 1–2 with $s = 1$), (b) the class-2 job in progress does not end at time n and a class-1 or class-2 job arrives (see Eq. (7), line 2 for $s = 2$ and line 3), or (c) a class-2 job finishes at time n with at least one queued class-3

job, while a new class-1 job arrives (see Eq. (6), line 3).

$$P_{\emptyset_1,\emptyset_0}((s,q)\to(s',q')) = \begin{cases} p_s D_0 & s=1 \text{ or } 2, s'=0, q=q'=0, \\ p_1 D_3 & s=1, s'=3, q=q'=0, \\ p_2 D_0 & s=2, s'=3, K>q=q'+1>0, \\ p_2 D_3 & s=2, s'=3, q=q'<K \\ p_2(D_0+D_3) & s=2, s'=3, q=q'+1=K \\ 0 & otherwise. \end{cases} \tag{3}$$

For $J \neq \emptyset_0$, we have

$$P_{J,J}((1,q)\to(s',q')) = \begin{cases} p_1 D_0 & s'=1, K>q=q'+1>0, \\ \bar{p}_1 D_0 + p_1 D_1 & s'=1, q=q'<K, \\ \bar{p}_1 D_1 & s'=1, q=q'-1<K, \\ \bar{p}_1 D_0 & s'=1, q=q'=K \\ p_1(D_0+D_1) & s'=1, q=q'+1=K \\ p_1 D_2 & s'=2, q=q'=0, \\ 0 & otherwise, \end{cases} \tag{4}$$

and

$$P_{J,J}((2,q)\to(s',q')) = \begin{cases} p_2 D_1 & s'=1, q=q'=0, \\ \bar{p}_2 D_0 + p_2 D_2 & s'=2, q=q'<K, \\ \bar{p}_2 D_3 & s'=2, q=q'-1<K, \\ \bar{p}_2(D_0+D_3)+p_2 D_2 & s'=2, q=q'=K \\ 0 & otherwise. \end{cases} \tag{5}$$

For $r>0$, one finds

$$P_{J,J+3^r}((s,q)\to(s',q')) = \begin{cases} p_1 D_3 & J\neq\emptyset_0, s=s'=1, q=q'+1>0, r=1, \\ \bar{p}_1 D_3 & J\neq\emptyset_0, s=s'=1, q=q', r=1, \\ p_2 D_1 & J\neq\emptyset_0, s=2, s'=1, q=r, q'=0, \\ p_3 D_1 & J=\emptyset_0, s=3, s'=1, q=r<K, q'=0, \\ \bar{p}_3 D_1 & J=\emptyset_0, s=3, s'=1, q=r-1<K, q'=0, \\ D_1 & J=\emptyset_0, s=3, s'=1, q=r=K, q'=0, \\ 0 & otherwise. \end{cases} \tag{6}$$

While $J\neq\emptyset_0$ and $r\geq 0$ yields

$$P_{J,J+23^r}((s,q)\to(s',q')) = \begin{cases} p_1 D_2 & s=s'=1, q=q'+1>0, r=0, \\ \bar{p}_s D_2 & s=s'=1 \text{ or } 2, q=q', r=0, \\ \bar{p}_2 D_1 & s=2, s'=1, q=r, q'=0, \\ 0 & otherwise. \end{cases} \tag{7}$$

(C) Let us now consider the case $X_n \neq \emptyset$. The transitions to nodes of the form $J + 3^r$ and $J + 23^r$, for $r \geq 0$, are analogue to the $J = \emptyset_1$ scenario and are given by Eq. (4–7). We further distinguish between the case where X_n is of the form 3^r, for some $r > 0$ and $J + 23^r$ for some $r \geq 0$. The only nonzero transitions that remain untouched are those that correspond to a (class-1 or class-2) service completion that causes a decrease in the current stack size (meaning no class-1 or class-2 arrival can take place at time n). If $J = 3^r$, for some $r > 0$, we end up in node $\emptyset_0$ as all the class-3 jobs are popped from the stack (in search for a class-2 job); hence, for $r > 0$

$$P_{3^r,\emptyset_0}((s,q) \to (s',q')) = \begin{cases} p_1 D_0 & s=1, s'=3, q=0, q'=r-1<K-1, \\ p_1 D_3 & s=1, s'=3, q=0, q'=r<K, \\ p_1(D_0+D_3) & s=1, s'=3, q=0, q'=K-1<r, \\ p_2 D_0 & s=2, s'=3, q'=q+r-1<K-1, \\ p_2 D_3 & s=2, s'=3, q'=q+r<K, \\ p_2(D_0+D_3) & s=2, s'=3, q'=K-1<q+r, \\ 0 & otherwise. \end{cases} \tag{8}$$

The order of events, in particular the fact that an arrival occurs before a job completion, explains why $q' = K - 1$ in line 3 and 6 of Eq. (8) when a class-3 arrival occurs with $q + r \geq K$. If $X_n = J + 23^r$, for $r \geq 0$, the events discussed by Eq. (8) cause the removal of r class-3 jobs and one class-2 job from the stack:

$$P_{J+23^r,J}((s,q) \to (s',q')) = \begin{cases} p_1 D_0 & s=1, s'=2, q=0, q'=r<K, \\ p_1 D_3 & s=1, s'=2, q=0, q'=r+1<K+1, \\ p_1(D_0+D_3) & s=1, s'=2, q=0, q'=K\leq r, \\ p_2 D_0 & s=s'=2, q'=q+r<K, \\ p_2 D_3 & s=s'=2, q'=q+r+1<K+1, \\ p_2(D_0+D_3) & s=s'=2, q'=K\leq q+r, \\ 0 & otherwise. \end{cases} \tag{9}$$

The steady state probabilities (and related performance measures) of a Markov chain (MC) with a tree structured state space can be computed via matrix analytic methods, provided that it falls within the M/G/1 or GI/M/1 paradigm [16, 20]. A MC is of the M/G/1 type if the node variable X_n can only *decrease* in length by one at a time. Similarly, for a GI/M/1 type MC the node variable X_n may not *grow* in length by more than one at a time. As neither of these restrictions applies to our MC, a reduction is needed if we want to rely on these matrix analytic results. In the next section we introduce an efficient reduction technique that reduces our MC to a tree-like process [3]. The set of tree-like processes coincides with the intersection of the set of M/G/1 and GI/M/1 type MCs as any tree-structured QBD MC can be reformulated as a tree-like process [17].

5 The tree-like process

In order to reduce the Markov chain (X_n, S_n, Q_n, M_n) to a tree-like process we need to eliminate all transitions that add/remove multiple jobs to/from the stack in a single transition. Moreover, while doing so we have to make sure that transitions from a node $J \neq \emptyset_0$ to J, $J+2$ or $J+3$ do not depend on J, while transitions from $J \neq \emptyset$ to $J' = J - f(J,1)$ may be influenced by $f(J,1)$, the top element on the stack, but not by $J' = J - f(J,1)$. We shall reduce the MC $(X_n, N_n = (S_n, Q_n, M_n))$ to a tree-like process by constructing an expanded MC $(\mathcal{X}_k, \mathcal{N}_k = (\mathcal{S}_k, \mathcal{Q}_k, \mathcal{M}_k))$. The technique used to construct this expanded MC is similar to Ramaswami's [12] to reduce a classic M/G/1-type MC to a QBD MC or to the approach taken in Van Houdt *et al* [18] to construct a tree structured QBD. The key idea behind this expanded MC is that whenever a transition occurs that adds a string of the form $J' = 3^{r+1}$ or $J' = 23^r$, for $r > 0$, to the node variable $X_n = J$, we split this transition into $r+1$ transitions[3] that each add one integer to the node variable $\mathcal{X}_k$. After adding the first element (a 2 or 3), we set $\mathcal{S}_k = 3^+$ to indicate that we still need to push a series of 3s to the stack (the exact number of 3s is stored in the $\mathcal{Q}_k$ variable). Likewise we split transitions that pop a string $J' = 23^r$ or 3^r, with $r > 0$, from the stack into $r+1$ transitions and let $\mathcal{S}_k = 2^-$ to indicate that we are popping elements from the stack until we encounter a class-2 job[4] ($\mathcal{Q}_k$ now indicates the number of class-3 jobs that was popped from the stack, without getting lost, since the last class-1 job left the system).

More formally, assume a given realization $(X_n(w), N_n(w) = (S_n(w), Q_n(w), M_n(w)))$ of the MC $(X_n, N_n = (S_n, Q_n, M_n))$. The expanded chain $(\mathcal{X}_k, \mathcal{N}_k = (\mathcal{S}_k, \mathcal{Q}_k, \mathcal{M}_k))$ is constructed as follows. The range of $\mathcal{S}_k$ equals $1, 2, 2^-$ and 3^+ if $J \neq \emptyset_0$, while $\mathcal{Q}_k$ takes values in $0, \ldots, K$ for $\mathcal{S}_k = 1, 2$ and $1, \ldots, K$ for $\mathcal{S}_k = 2^-, 3^+$. Thus, we add $2K$ states to each node $J \neq \emptyset_0$ of the original MC (X_n, N_n).

Initial state: If $(X_0(w), N_0(w)) = (J, (s, q, i))$, then set $(\mathcal{X}_0(w), \mathcal{N}_0(w)) = (J, (s, q, i))$. Also, set $n = 0$ and $k = 0$, n represents the steps of the original chain and k represents the steps of the expanded chain. We will establish a one-to-one correspondence between the state $(J, (s, q, i))$ of the original chain and the state $(J, (s, q, i))$ of the expanded chain.

[3]When a transition from $J = \emptyset_0$ to 3^{r+1} occurs we split the transition into $r+2$ steps, where the first step goes from $\emptyset_0$ to $\emptyset_1$. The advantage of splitting this type of transitions into $r+2$ steps as opposed to $r+1$ is that we can use the same transition matrix to go from node $\emptyset_1$ to $J = 3$ as from $J = J'$ to $J' + 3$ for any $J' \neq \emptyset$.

[4]Transition from $J = 3^r$ to $\emptyset_0$ are split into $r+1$ steps, where the last step goes from $\emptyset_1$ to $\emptyset_0$. Therefore, we can use the same transition matrix to go from node $J = 3$ to $\emptyset_1$ as from $J = J' + 3$ to J' for any $J' \neq \emptyset$.

Transition Rules: We distinguish between three possible cases: $\mathcal{S}_k(w) \neq 2^-$ or 3^+, $\mathcal{S}_k(w) = 2^-$ and $\mathcal{S}_k(w) = 3^+$.
(A) $\mathcal{S}_k(w) \neq 2^-, 3^+$: Consider $(X_n(w), (S_n(w), Q_n(w), M_n(w)))$, and do one of the following:

1a. Assume $X_n(w) = J$ and $(X_{n+1}(w), S_{n+1}(w), Q_{n+1}(w)) = (J + 23^r, 1, 0)$ for some $r > 0$, then set $\mathcal{X}_{k+1}(w) = J + 2$ and $\mathcal{N}_{k+1}(w) = (3^+, r, M_{n+1}(w))$.

1b. Otherwise, provided that $X_n(w) = J \neq \emptyset_0$ and $(X_{n+1}(w), S_{n+1}(w), Q_{n+1}(w)) = (J + 3^r, 1, 0)$ for some $r > 1$, let $\mathcal{X}_{k+1}(w) = J + 3$ and $\mathcal{N}_{k+1}(w) = (3^+, r - 1, M_{n+1}(w))$.

1c. For $X_n(w) = \emptyset_0$ and $(X_{n+1}(w), S_{n+1}(w), Q_{n+1}(w)) = (J + 3^r, 1, 0)$ for some $r > 0$, we define $\mathcal{X}_{k+1}(w) = \emptyset_1$ and $\mathcal{N}_{k+1}(w) = (3^+, r, M_{n+1}(w))$.

2a. If $X_n(w) = J + 23^r$ and $(X_{n+1}(w), S_{n+1}(w)) = (J, 2)$ for some $r > 0$, then define $\mathcal{X}_{k+1}(w) = J + 23^{r-1}$ and $\mathcal{N}_{k+1}(w) = (2^-, \max(Q_n(w) + 1 + a, K), M_{n+1}(w))$, where a is the number of class-3 job arrivals at time n.

2b. For $X_n(w) = 3^r$ and $(X_{n+1}(w), S_{n+1}(w)) = (\emptyset_0, 3)$ for some $r \geq 1$, set $\mathcal{X}_{k+1}(w) = 3^{r-1}$ (i.e., $\emptyset_1$ if $r = 1$) and $\mathcal{N}_{k+1}(w) = (2^-, \max(Q_n(w) + 1 + a, K), M_{n+1}(w))$.

3. In all other cases set $(\mathcal{X}_{k+1}(w), \mathcal{N}_{k+1}(w)) = (X_{n+1}(w), N_{n+1}(w))$

Next, increment both n and k by one.
(B) $\mathcal{S}_k(w) = 2^-$: $\mathcal{X}_k(w)$ can be written as $\emptyset_1, J + 2$ or $J + 3$ for some string J.

1. For $\mathcal{X}_k(w) = J+2$, set $\mathcal{X}_{k+1}(w) = J = X_n(w)$ and $\mathcal{N}_{k+1}(w) = (2, \mathcal{Q}_k(w), \mathcal{M}_k(w)) = N_n(w)$. Notice, n was incremented by one in (A).

2. While for $\mathcal{X}_k(w) = J + 3$, set $\mathcal{X}_{k+1}(w) = J$ and $\mathcal{N}_{k+1}(w) = (2^-, \max(\mathcal{Q}_k(w) + 1, K), \mathcal{M}_k(w))$.

3. Finally, when $\mathcal{X}_k(w) = \emptyset_1$, let $\mathcal{X}_{k+1}(w) = \emptyset_0 = X_n(w)$ and $\mathcal{N}_{k+1}(w) = (3, \mathcal{Q}_k(w) - 1, \mathcal{M}_k(w)) = N_n(w)$.

Next, increase k by one and do not alter the value of n.
(C) $\mathcal{S}_k(w) = 3^+$: $\mathcal{Q}_k(w)$ is either equal to or larger than 1 (due to (A1)).

1. For $\mathcal{Q}_k(w) = 1$, set $\mathcal{X}_{k+1}(w) = \mathcal{X}_k(w) + 3 = X_n(w)$ and $\mathcal{N}_{k+1}(w) = (1, 0, \mathcal{M}_k(w)) = N_n(w)$.

2. When $\mathcal{Q}_k(w) > 1$, set $\mathcal{X}_{k+1}(w) = \mathcal{X}_k + 3$ and $\mathcal{N}_{k+1}(w) = (3^+, \mathcal{Q}_k(w) - 1, \mathcal{M}_k(w))$.

Increase k by one and do not alter the value of n.

We are now in a position to characterize the nonzero transition probabilities of the expanded MC, i.e., the tree-process $(\mathcal{X}_k, \mathcal{N}_k)$. The transitions from node $\emptyset_0$ to $\emptyset_0$ are identical to those of the MC (X_n, N_n) and given by Eq. (1). Similarly, the transitions from a node $J \neq \emptyset_0$ to itself remain unaltered and are characterized by Eq. (4-5), meaning that for $s = 2^-$ or 3^+, the string $\mathcal{X}_k = J$ either increases or decreases in length. Looking at the transition rules, in particular at case (B3), we see that we need to *add* the following line to Eq. (3) in order to capture all transitions from $\emptyset_1$ to $\emptyset_0$:

$$P_{\emptyset_1,\emptyset_0}((s,q),(s',q')) = \\ \left\{ \begin{array}{ll} I_m & s = 2^-, s' = 3, q = q' + 1 > 0, \end{array} \right. \tag{10}$$

with I_m the $m \times m$ identity matrix. By rule (A1c), we know that apart from the transitions from $\emptyset_0$ to $\emptyset_1$ given by Eq. (2) we need to *add* the following lines

$$P_{\emptyset_0,\emptyset_1}((s,q),(s',q')) = \\ \left\{ \begin{array}{ll} p_3 D_1 & s = 3, s' = 3^+, 0 < q = q' < K, \\ \bar{p}_3 D_1 & s = 3, s' = 3^+, q = q' - 1 < K, \\ D_1 & s = 3, s' = 3^+, q = q' = K. \end{array} \right. \tag{11}$$

Notice, at time $k+1$ we are either in node $\emptyset_0$ or $\emptyset_1$ if $\mathcal{X}_k = \emptyset_0$. Let us now consider the case $\mathcal{X}_k = J$ and $\mathcal{X}_{k+1} = J + 3$. From Eq. (6) and rules (A1b), (A3) and (C) we find (for $J \neq \emptyset_0$)

$$P_{J,J+3}((s,q) \to (s',q')) = \\ \left\{ \begin{array}{ll} p_1 D_3 & s = s' = 1, q = q' + 1 > 0, \\ \bar{p}_1 D_3 & s = s' = 1, q = q', \\ p_2 D_1 & s = 2, s' = 1, q = 1, q' = 0, \\ & \text{or } s = 2, s' = 3^+, q = q' + 1 > 1, \\ I_m & s = 3^+, s' = 1, q = 1, q' = 0, \\ & \text{or } s = s' = 3^+, q = q' + 1 > 1, \\ 0 & \textit{otherwise}. \end{array} \right. \tag{12}$$

Similarly, by Eq. (7) and rules (A1a) and (A3), we have (for $J \neq \emptyset_0$)

$$P_{J,J+2}((s,q) \to (s',q')) = \\ \left\{ \begin{array}{ll} p_1 D_2 & s = s' = 1, q = q' + 1 > 0, \\ \bar{p}_s D_2 & s = s' = 1 \text{ or } 2, q = q', \\ \bar{p}_2 D_1 & s = 2, s' = 1, q = q' = 0, \\ & \text{or } s = 2, s' = 3^+, q = q' > 0, \\ 0 & \textit{otherwise}. \end{array} \right. \tag{13}$$

According to Eq. (9) and rules (A3) and (B1), transitions from node $J+2$ to J are governed by

$$P_{J+2,J}((s,q) \to (s',q')) = \begin{cases} p_1 D_0 & s=1, s'=2, q=q'=0, \\ p_1 D_3 & s=1, s'=2, q=0, q'=1, \\ p_2 D_0 & s=s'=2, q=q'<K, \\ p_2 D_3 & s=s'=2, q=q'-1<K, \\ p_2(D_0+D_3) & s=s'=2, q=q'=K, \\ I_m & s=2^-, s'=2, q=q', \\ 0 & otherwise. \end{cases} \tag{14}$$

Finally, Eq. (9) and rules (A2), (A3) and (B2) give rise to[5]

$$P_{J+3,J}((s,q) \to (s',q')) = \begin{cases} p_1 D_0 & s=1, s'=2^-, q=0, q'=1, \\ p_1 D_3 & s=1, s'=2^-, q=0, q'=2, \\ p_2 D_0 & s=2, s'=2^-, q=q'-1<K-1, \\ p_2 D_3 & s=2, s'=2^-, q=q'-2<K-1, \\ p_2(D_0+D_3) & s=2, s'=2^-, q=K-1 \text{ or } K, q'=K, \\ I_m & s=2^-, s'=2^-, q'=\min(q+1,K), \\ 0 & otherwise. \end{cases} \tag{15}$$

We define the following set of matrices. Let F be the $m(K+2) \times m(K+2)$ matrix holding the transition probabilities from node $\emptyset_0$ to itself, F_{01} the $m(K+2) \times m(4K+2)$ matrix characterizing the transitions from $\emptyset_0$ to node $\emptyset_1$, while F_{10} is the $m(4K+2) \times m(K+2)$ matrix containing the probabilities of going from node $\emptyset_1$ to $\emptyset_0$. Moreover, let A and U_k, for $k=2,3$, represent the square matrix of dimension $m(4K+2)$, holding the transition probabilities from a node $J \neq \emptyset_0$ to itself and node $J+k$, respectively. Finally, define D_k, for $k=2,3$, as the $m(4K+2) \times m(4K+2)$ matrix whose entries represent the transition probabilities to go from a node J of the form $J'+k$ to $J-f(J,1)=J'$ (see Figure 2).

6 Steady state probabilities and performance measures

In this section we indicate how to compute the steady state probabilities $\tilde{\pi}$ of the expanded MC $(\mathcal{X}_k, \mathcal{N}_k)$, from which we derive the invariant measure π of the MC (X_n, N_n) using a censoring argument. Several performance measures

[5]Some minor changes are needed if $K=1$.

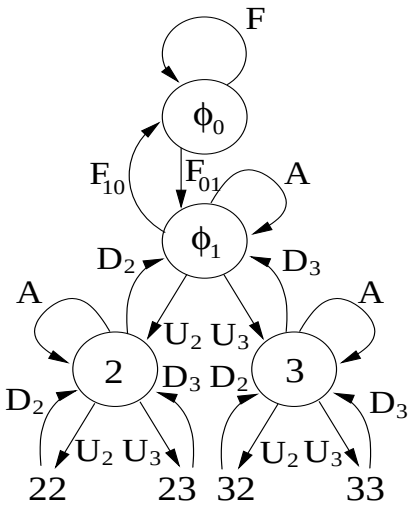


Figure 2: Transition diagram for the top levels of the tree

are obtained from $\tilde{\pi}$ and π. Define

$$
\begin{array}{lcl}
\tilde{\pi}_J(s,q,j) & = & \lim_{k\to\infty} P[\mathcal{X}_k = J, \mathcal{N}_k = (s,q,j)], \\
\tilde{\pi}_J(s,q) & = & (\tilde{\pi}_J(s,q,1),\ldots,\tilde{\pi}_J(s,q,m)), \\
\tilde{\pi}_J(s) & = & \left\{ \begin{array}{ll}
(\tilde{\pi}_J(s,0),\ldots,\tilde{\pi}_J(s,K)) & s=1,2, J\neq\emptyset_0 \\
(\tilde{\pi}_J(s,1),\ldots,\tilde{\pi}_J(s,K)) & s=2^-,3^+, J\neq\emptyset_0 \\
\tilde{\pi}_J(s,0) & s=0, J=\emptyset_0 \\
(\tilde{\pi}_J(s,0),\ldots,\tilde{\pi}_J(s,K)) & s=3, J=\emptyset_0
\end{array} \right. \\
\tilde{\pi}_J & = & \left\{ \begin{array}{ll}
((\tilde{\pi}_J(1),\tilde{\pi}_J(2),\tilde{\pi}_J(2^-),\tilde{\pi}_J(3^+)) & J\neq\emptyset_0, \\
((\tilde{\pi}_J(0),\tilde{\pi}_J(3)) & J=\emptyset_0
\end{array} \right.
\end{array}
\tag{16}
$$

These probabilities exist provided that the MC $(\mathcal{X}_k, \mathcal{N}_k)$ is ergodic. A simple algorithmic test that allows us to check its ergodicity is presented further on. From an operational point of view, it is clear that both the chains $(\mathcal{X}_k, \mathcal{N}_k)$ and (X_n, N_n) are stable if and only if $\lambda_1(1-p_{1,loss})/p_1 + \lambda_2/p_2 < 1$, that is, the offered load by the class-1 and class-2 jobs (minus the load of the lost class-1 jobs) should be less than one. This condition has been verified numerically for several examples. However, establishing formal proofs on the stability of tree-like processes is generally considered as hard [20], because no explicit stability condition in terms of the transition matrices has been found (as opposed to the classic QBD, M/G/1 and GI/M/1 type MCs [6, 10, 11]).

As a special case of [20, Theorem 1] it follows that the vectors $\tilde{\pi}_J$ can be

written as

$$\begin{aligned}
(\tilde{\pi}_{\emptyset_0}, \tilde{\pi}_{\emptyset_1}) &= (\tilde{\pi}_{\emptyset_0}, \tilde{\pi}_{\emptyset_1}) \begin{bmatrix} F & F_{01} \\ F_{10} & A + R_2 D_2 + R_3 D_3 \end{bmatrix}, \\
(\tilde{\pi}_{\emptyset_0}, \tilde{\pi}_{\emptyset_1}) &= \frac{(\tilde{\pi}_{\emptyset_0}, \tilde{\pi}_{\emptyset_1})}{(\tilde{\pi}_{\emptyset_0} e_{m(K+2)} + \tilde{\pi}_{\emptyset_1} (I - (R_2 + R_3))^{-1} e_{m(4K+2)})}, \\
\tilde{\pi}_{J+k} &= \tilde{\pi}_J R_k,
\end{aligned} \tag{17}$$

for $J \neq \emptyset_0$. The matrices R_2 and R_3 are found as $U_2(I-V)^{-1}$ and $U_3(I-V)^{-1}$, respectively, while V is the smallest non-negative solution to[6]

$$V = A + U_2(I - V)^{-1} D_2 + U_3(I - V)^{-1} D_3.$$

Several algorithms to iteratively solve a non-linear matrix equation of this form can be used [3]. We will rely on the standard fixed-point iteration (FPI): let $V[0] = A$ and compute $V[N+1]$ as $V[N+1] = A + U_2(I - V[N])^{-1} D_2 + U_3(I - V[N])^{-1} D_3$. To speed up this iterative scheme we can make use of the structure of the matrices A, U_k, D_k and V. For instance, from the probabilistic interpretation of V one finds that V (and $V[N]$) has the following form:

$$V = \begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,2^-} & 0 \\ V_{2,1} & V_{2,2} & V_{2,2^-} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & V_{3^+,2^-} & 0 \end{bmatrix}, \tag{18}$$

where $V_{s,s'}$ are square $m(K+1)$ matrices for $s' \neq 2^-$, $V_{s,2^-}$ is an $m(K+1) \times mK$ matrix for $s = 1, 2$ and $V_{3^+,2^-}$ a square mK matrix. Having computed V, we can check the ergodicity of the chain $(\mathcal{X}_k, \mathcal{N}_k)$ by computing $G_2 = (I-V)^{-1} D_2$ and $G_3 = (I - V)^{-1} D_3$. The matrices G_2 and G_3 are both stochastic[7] if and only if the MC $(\mathcal{X}_k, \mathcal{N}_k)$ is ergodic (see [19]).

Due to the construction of the expanded MC $(\mathcal{X}_k, \mathcal{N}_k)$, the steady state probability vectors π_J (defined in the obvious way analogue to Eq. (16)) associated with the MC (X_n, N_n) can be expressed as

$$\begin{aligned}
\pi_{\emptyset_0} &= \frac{\tilde{\pi}_{\emptyset_0}}{c}, \\
\pi_J &= \frac{(\tilde{\pi}_J(1), \tilde{\pi}_J(2))}{c},
\end{aligned}$$

for $J \neq \emptyset_0$, with

$$c = 1 - \tilde{\pi}_{\emptyset_1} (I - (R_2 + R_3))^{-1} \begin{bmatrix} 0_{m(2K+2)} \\ e_{m2K} \end{bmatrix},$$

[6] I is the identity matrix of the appropriate dimension

[7] In practice, we consider the substochastic matrices G_k as stochastic if $G_k e_{m(4K+2)} > (1 - 10^{-12}) e_{m(4K+2)}$. We cannot check whether the rows of G_k exactly sum to one, as floating-point environments do not represent numbers in an exact manner.

where 0_x is a zero column vector of size x. To establish this result, we relied on the identity $\sum_{J\neq\emptyset_0}(\tilde{\pi}_J(2^-),\tilde{\pi}_J(3^+))e_{m2K} = 1-c$.

In the remainder of this section we focus on the performance measures of the class-3 jobs. Measures for the other two classes can be obtained as well. However, as the class-3 jobs do not affect class-1 or class-2 jobs, these measures can also be computed in a direct manner using more standard techniques for queues with 2 priority classes. The loss probability of the class-3 jobs can be computed as

$$p_{3,loss} = 1 - \frac{p_3\pi_{\emptyset_0}(3)e_{m(K+1)}}{\lambda_3}, \tag{19}$$

as the loss equals one minus the output rate of the completed class-3 jobs divided by the class-3 input rate. To compute the probability $P[W_3 = i]$ of having i the class-3 jobs waiting in the $3Q$ system, we start by defining the following row vectors, for $k = 0,\ldots,K-1$:

$$\begin{aligned}&(\nu_k(1),\nu_k(2),\nu_k(2^-,3^+)) = \\ &\quad \tilde{\pi}_{\emptyset_1}\left((I-R_2)^{-1}R_3\right)^k(I-R_2)^{-1}(I_{4K+2}\otimes e_m)/c,\end{aligned}$$

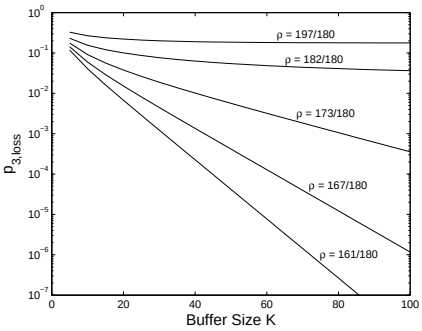
where $\nu_k(1)$ and $\nu_k(2)$ are both $1\times(K+1)$ vectors. The s-th entry of $\nu_k(1)$ gives us the probability of having (i) a class-1 job is in progress, (ii) $s-1$ class-1 jobs in the queue and (iii) k class-3 jobs on the stack in the $S\&Q$ model, while the s-th entry of $\nu_k(2)$, denoted as $(\nu_k(2))_s$ holds the same probability, but with (i) a class-2 job in service and (ii) $s-1$ class-3 jobs in the queue. Therefore, we have for $i = 0,1,\ldots,K-1$

$$\begin{aligned}P[W_3 = i] &= \pi_{\emptyset_0}(0,0)e_m 1[i=0] + \pi_{\emptyset_0}(3,i)e_m + \\ &\quad \nu_i(1)e_{K+1} + \sum_{s=0}^{i}(\nu_{i-s}(2))_{s+1},\end{aligned} \tag{20}$$

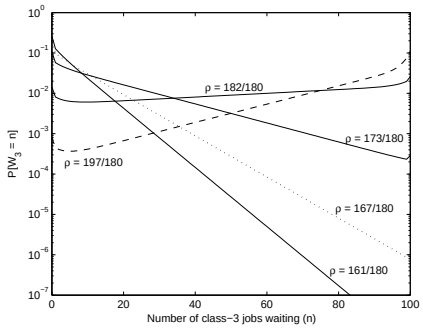
where $1[w] = 1$ if w is true and 0 otherwise. The class-3 queue in the $3Q$ model has size K, therefore, $P[W_3 = K] = 1 - \sum_{i=0}^{K-1}P[W_3 = i]$.

7 Numerical Examples

This section includes some numerical results to demonstrate the strength of our approach. We consider buffer sizes K from 5 to 100. The computation time for a $K = 100$ example is about 3 minutes on a 2GHz Pentium with 512Mb RAM, the peak memory usage by the Matlab 7.0 (for Windows) session was less than 145Mb as measured by the Windows task manager. The memory usage can easily be further reduced when necessary. Some of the results presented in this section were also validated my means of a simulation to check for implementation errors. We consider the following arrival process:



(a) Class-3 loss probability



(b) Class-3 buffer occupation for $K = 100$

$$
\begin{aligned}
D &= \begin{bmatrix} 1 - 1/500 & 1/500 \\ 1/250 & 1 - 1/250 \end{bmatrix}, \\
D_1 &= (1/8, 1/30)D, \quad D_2 = (1/20, 1/15)D, \\
D_3 &= (1/20, 1/x)D, \quad D_0 = D - D_1 - D_2 - D_3,
\end{aligned}
$$

with $x = 6, 8, 10, 12$ and 15. The mean service time of a priority 1, 2 and 3 job is 3, 5 and 6 time units, respectively. Thus, the sum of the class-1 and class-2 load $\rho_{12} = 101/180$. The total load ρ depends on the value of x as $\rho = 137/180 + 2/x$, which results in a load of $197/180, 182/180, 173/180, 167/180$ and $161/180$ for the x-values mentioned above. Notice, setting $x = 6$ or 8 creates an overloaded system.

Figure 3(a) depicts the class-3 loss probability $p_{3,loss}$ as a function of the buffer size K for various x values. As expected, increasing the buffer capacity K reduces the loss probability. However, for the two overload scenarios there is little use in supporting a larger capacity. Figure 3(b) shows the class-3 buffer contents distribution (in the $3Q$ model) for $K = 100$. Except for the two overload setups, the number of waiting type-3 jobs seems to decrease exponentially. When $\rho = 182/180$ we get a distribution that is fairly close to a uniform distribution which is intuitively what we expect as ρ is close to one, while for $\rho = 197/180$ we tend to have a full class-3 buffer due to the strong overload.

References

[1] A.S. Alfa. Matrix-geometric solution of discrete time MAP/PH/1 priority queue. *Naval Research Logistics*, 45:23–50, 1998.

[2] A.S. Alfa, B. Liu, and Q.M. HE. Discrete-time analysis of MAP/PH/1 multiclass general preemptive priority queue. *Naval Research Logistics*, 50:662–682, 2003.

[3] D.A. Bini, G. Latouche, and B. Meini. Solving nonlinear matrix equations arising in tree-like stochastic processes. *Linear Algebra Appl.*, 366:39–64, 2003.

[4] Q. He. Queues with marked customers. *Adv. in Appl. Probab.*, 28:567–587, 1996.

[5] Q. He and M.F. Neuts. Markov chains with marked transitions. *Stochastic Processes and their Applications*, 74:37–52, 1998.

[6] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods and stochastic modeling*. SIAM, Philadelphia, 1999.

[7] F. Machihara. On the queue with PH-Markov renewal preemption. *J. Oper. Res. Soc. of Japan*, 36:13–28, 1993.

[8] D.G. Miller. Computation of steady-state probabilities for M/M/1 priority queues. *Operations Research*, 29(5):945–958, 1981.

[9] R. Miller. Priority queues. *Annals of Mathematical statistics*, 31:86–103, 1960.

[10] M.F. Neuts. *Matrix-Geometric Solutions in Stochastic Models, An Algorithmic Approach*. John Hopkins University Press, 1981.

[11] M.F. Neuts. *Structured Stochastic Matrices of M/G/1 type and their applications*. Marcel Dekker, Inc., New York and Basel, 1989.

[12] V. Ramaswami. The generality of QBD processes. In *Advances in Matrix Analytic Methods for Stochastic Models*, pages 93–113. Notable Publications Inc., Neshanic Station, NJ, 1998.

[13] T. Takine. A nonpreemptive priority MAP/G/1 queue with two classes of customers. *J. Oper. Res. Soc. of Japan*, 39(2):266–290, 1996.

[14] T. Takine and T. Hasegawa. The workload in a MAP/G/1 queue with state-dependent services: its applications to a queue with preemptive resume priority. *Stochastic Models*, 10(1):183–204, 1994.

[15] T. Takine and T. Hasegawa. The nonpreemptive priority MAP/G/1 queue. *Operations Research*, 47(6):917–927, 1999.

[16] T. Takine, B. Sengupta, and R.W. Yeung. A generalization of the matrix M/G/1 paradigm for Markov chains with a tree structure. *Stochastic Models*, 11(3):411–421, 1995.

[17] B. Van Houdt and C. Blondia. Tree structured QBD Markov chains and tree-like QBD processes. *Stochastic Models*, 19(4):467–482, 2003.

[18] B. Van Houdt and C. Blondia. Throughput of Q-ary splitting algorithms for contention resolution in communication networks. *Communications in information and systems*, 4(2):135–164, 2005.

[19] R.W. Yeung and A.S. Alfa. The quasi-birth-death type Markov chain with a tree structure. *Stochastic Models*, 15(4):639–659, 1999.

[20] R.W. Yeung and B. Sengupta. Matrix product-form solutions for Markov chains with a tree structure. *Adv. in Appl. Probab.*, 26:965–987, 1994.