QBD Markov Chains on Binomial-Like Trees and its Application to Multilevel Feedback Queues

B. Van Houdt*, J. Van Velthoven and C. Blondia

University of Antwerp,
Middelheimlaan 1,
B-2020 Antwerpen, Belgium

*Corresponding author. Current address: University of Antwerp, Department of Mathematics and Computer Science, Middelheimlaan 1, B-2020 Antwerpen, Belgium, benny.vanhoudt@ua.ac.be, tel: +32 3 265 38 91.

Abstract

A matrix analytic paradigm, termed Quasi-Birth-Death Markov chains on binomial-like trees, is introduced and a quadratically converging algorithm to assess its steady state is presented. In a bivariate Markov chain $\{(X_t, N_t), t \geq 0\}$, the values of the variable X_t are nodes of a binomial-like tree of order d, where the i-th child has i children of its own. We demonstrate that it suffices to solve d quadratic matrix equations to yield the steady state vector, the form of which is matrix geometric. We apply this framework to analyze the multilevel feedback scheduling discipline, which forms an essential part in contemporary operating systems.

Keywords: QBD Markov chains, tree-like processes, matrix-analytic methods, multilevel feedback queues

Over the last few decades, broad classes of frequently encountered queueing models have been analyzed by *matrix-analytic methods* [3, 11, 12, 13]. The embedded Markov chains in these models are two-dimensional generalizations of the classic M/G/1 and GI/M/1 queues, and quasi-birth-death (QBD) processes. Matrix-analytic models include notions such as the Markovian arrival process (MAP) and the phase-type (PH) distribution, both in discrete and continuous time. Considerable efforts have been put into the development of efficient and numerically stable methods for their analysis [3]. There is also an active search for accurate matching algorithms for both PH distributions and MAP arrivals when modeling communication systems, e.g., [1, 5, 6, 9, 15].

One of the more recent paradigms within the field of matrix-analytic methods has been its generalization to discrete-time bivariate Markov chains $\{(X_t, N_t), t \geq 0\}$, in which the values of X_t are the nodes of a d-ary tree (and N_t takes integer values between 1 and h). Depending on the transitions allowed in such a tree-structured Markov chain, several classes have been identified: (a) M/G/1-type [18], (b) GI/M/1-type [21], (c) QBD-type [20] and (d) tree-like processes [4], where the latter two classes were shown to be equivalent [19]. In each of these models the transitions fulfill a spacial homogeneity property, that is, the transition probability between two nodes depends only on the spacial relationship between the two nodes and not on their specific values. Various iterative algorithms to determine, among others, the steady state probability vector of these types of Markov chains have been developed. As opposed to the standard M/G/1-, GI/M/1- and QBD-type chains (which correspond to the case where $d = 1$) such algorithms exhibit either only linear convergence or are quadratic, but require an iterative procedure at each step or the solution of a large system of linear equations (e.g., [4]).

Instead of considering a state space that is organized as an infinite strip (the standard case) or as an infinite d-ary tree, we consider a different type of state space, termed a binomial-like tree (due to its similarity with binomial trees). A binomial-like tree consists of an infinite number of nodes, where the root node has d children. Also, every node that is the i-th child of its parent node, has i children of its own. Thus, when considering discrete-time Markov chains $\{(X_t, N_t), t \geq 0\}$ on such a state space, the value of X_t can be represented as an *decreasingly ordered* string of integers (between 1 and d). Indeed, let $\emptyset$ represent the root node and let X_t be the i-th child of a node represented as J, then X_t can be written as $J + i$ (where a + denotes the concatenation to the right).

A QBD-type Markov chain on such a state space is obtained by restricting the transitions in a manner similar to a tree-like process. However, to improve the flexibility of the framework, the range of the variable N_t may depend on the rightmost integer of X_t. Apart from discussing the ergodicity condition of such a Markov chain, we also develop an algorithm with quadratic convergence to determine its steady state vector, by showing that the matrices needed to characterize this vector can be obtained by solving a series of d quadratic matrix equations, for which various algorithms and software tools exist (e.g., [2]).

We further demonstrate that this new type of Markov chain is very effective in analyzing multilevel feedback queues, originally introduced in the late 1960s [16, 7, 10], which appear in many

modern textbooks on operating systems, e.g., [17]. Also, many contemporary operating systems support some form of multilevel feedback queues (e.g., Solaris). Multilevel feedback queues process jobs in a round-robin order, but whenever a job completes its quantum, its level is increased by one (all jobs start as level 0 jobs). Ongoing jobs are distributed over various FIFO queues, depending on their current level and a strict priority mechanism is used to visit these queues (where queue 0 has the highest priority), see Section 3 for a more detailed description.

Feedback queues are inspired by the optimality of scheduling orders such as shortest-process-next (SPN) or shortest-remaining-time-next (SRTN), which minimize the mean delay by favoring short jobs, but do not rely on a priori knowledge of the job durations. Instead the service time attained so far is used to determine the priority of a job. To avoid starvation effects for large jobs, the quantum received by a level i job typically increases as a function of i. Other queueing disciplines that are very closely related to multilevel feedback queues include least-attained-service, foreground-background and shortest elapsed time. Many papers have been devoted to the analysis of multilevel feedback queues, a detailed survey on this topic was written by Nuyens and Wierman [14]. As can be seen from this survey, most of the work focuses on the limiting case where the number of levels is infinite and the quantum q approaches zero.

We demonstrate that the behavior of a multilevel feedback queue can be captured by a QBD-type Markov chain on a binomial-like tree, when jobs arrive in batches, have generally distributed service times with finite support and the number of levels is large enough such that any job starting service in the last level completes service before its quantum expires. Notice, due to the finite support assumption, this corresponds to having an infinite number of levels. To the best of our knowledge, it is the first time that the joint queue length distribution of a multilevel feedback queue is derived in an analytical manner. Moreover, our model can be easily extended to the case where jobs receive different service quanta depending on their current level.

In Section 1 we formally introduce binomial-like trees and the class of QBD Markov chains having such a tree as their state space. A necessary and sufficient stability condition is established in Section 2 and the stationary vector is shown to have a matrix geometric form, provided that it exists. We also indicate how to develop a quadratically converging algorithm to compute the steady state vector. In Section 3 we indicate how to model a multilevel feedback queue as a QBD Markov chain on a binomial-like tree. We end with some illustrative numerical examples.

1 QBDs on Binomial-Like Trees

A *binomial-like* tree of order d consisting of an infinite set of nodes Ω is constructed as follows. Let $\emptyset \in \Omega$ be the root node of the tree, having d children (i.e., neighbors) labeled 1 to d. Each i-th child node with a label of the form $J + i$, has exactly i children of its own, labeled $J + i + 1$ to $J + i + i$. Thus, the set of all nodes is given by

$$\Omega = \{\emptyset\} \cup \{J | J = j_1\ j_2\ \ldots\ j_k, k \geq 1, 1 \leq j_l \leq d, j_l \geq j_{l'}, 1 \leq l \leq l' \leq k\},$$

that is, Ω includes all strings J of integers between 1 and d that are ordered descendingly. A binomial-like tree differs from the well-known binomial tree of order d [8] in the sense that the i-th child of a node has i children, instead of $i-1$. Therefore, a binomial-like tree has an infinite number of nodes, whereas a binomial tree of order d has exactly 2^d nodes.

Define a discrete time bivariate Markov chain $\{(X_t, N_t), t \geq 0\}$ in which the values of X_t are represented by nodes of a binomial-like tree of order d, for $d \geq 1$, and where the range of N_t is defined as $\{1, \ldots, h_{f(X_t,1)}\}$, where $f(X_t, 1)$ represents the rightmost integer of the string X_t (and as $\{1, \ldots, h_\emptyset\}$ if $X_t = \emptyset$). We will refer to X_t as the *node* and to N_t as the *auxiliary* variable of the Markov chain (MC) at time t. With some abuse of notation, we shall refer to this MC as (X_t, N_t). Notice, the number of states belonging to a node may depend on the rightmost integer of the node label, meaning sibling nodes may contain a different number of states, but two nodes that are the i-th child of some nodes J_1 and J_2 both hold exactly h_i states. Throughout this paper, we use the '+' to denote the concatenation on the right and '-' to represent the deletion from the right. Let $f(J, k)$, for $J \neq \emptyset$, denote the k rightmost elements of the string J, then $J - f(J, 1)$ represents the parent node of J.

The following restrictions need to apply for a MC (X_t, N_t) to be a QBD on a binomial-like tree. At each step the chain can only make a transition to its parent (i.e., $X_{t+1} = X_t - f(X_t, 1)$, for $X_t \neq \emptyset$), to itself ($X_{t+1} = X_t$), or to one of its own children ($X_{t+1} = X_t + s$ for some $1 \leq s \leq f(X_t, 1)$ or $1 \leq s \leq d$ if $X_t = \emptyset$). Moreover, the chain's state at time $t+1$ is determined as follows:

$$P[(X_{t+1}, N_{t+1}) = (J', j) | (X_t, N_t) = (J, i)] =$$
$$\begin{cases} f^{i,j} & J' = J = \emptyset, \\ b_k^{i,j} & J' = J \neq \emptyset, f(J,1) = k, \\ d_{k,l}^{i,j} & J \neq \emptyset, f(J,2) = lk, J' = J - f(J,1) = J - k, \\ d_k^{i,j} & J = k, J' = \emptyset, \\ u_{k,s}^{i,j} & J \neq \emptyset, f(J,1) = k, J' = J + s, \\ u_s^{i,j} & J = \emptyset, J' = s, \\ 0 & otherwise \end{cases}$$

Notice, the transition probability between two nodes depends only on the rightmost element of their labels and not on their specific values. We can now define the $h_\emptyset \times h_\emptyset$ matrix F, the $h_k \times h_k$ matrix B_k, the $h_k \times h_l$ matrix $D_{k,l}$, the $h_k \times h_\emptyset$ matrix D_k, the $h_k \times h_s$ matrix $U_{k,s}$ and the $h_\emptyset \times h_s$ matrix U_s with their $(i,j)^{th}$ elements given by $f^{i,j}$, $b_k^{i,j}$, $d_{k,l}^{i,j}$, $d_k^{i,j}$, $u_{k,s}^{i,j}$ and $u_s^{i,j}$, respectively. This completes the description of the tree-like process. Notice, a QBD on a binomial-like tree is fully characterized by the matrices F, B_k, $D_{k,l}$, D_k, $U_{k,s}$ and U_s, for $1 \leq s \leq k \leq l \leq d$, meaning in general by $1 + d(d+4)$ matrices (see Figure 1).

Clearly, denoting e as a column vector of the appropriate dimension and with all its entries equal

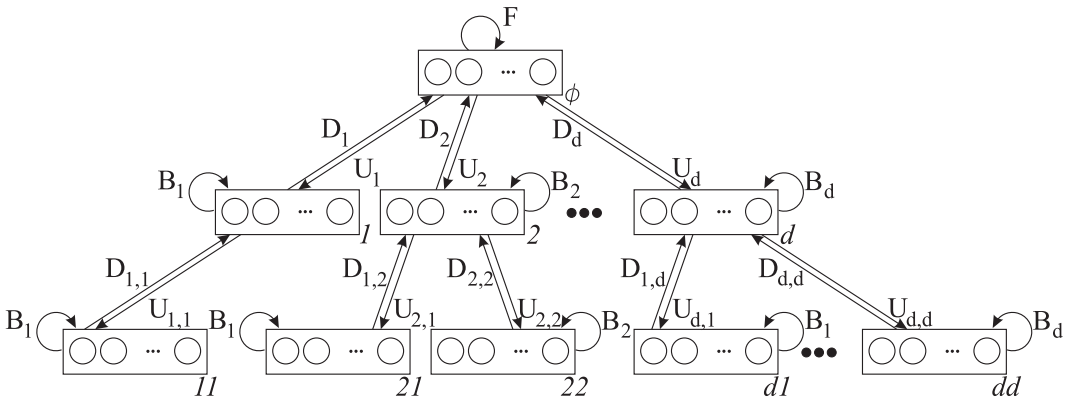


Figure 1: Structure of the first three levels of a binomial-like tree Markov chain and the matrices characterizing its transitions

to one, we have

$$B_k e + D_{k,l} e + \sum_{s=1}^{k} U_{k,s} e = e, \tag{1}$$

$$B_k e + D_k e + \sum_{s=1}^{k} U_{k,s} e = e, \tag{2}$$

$$Fe + \sum_{s=1}^{k} U_s e = e, \tag{3}$$

for $1 \leq k \leq l$. This implies that $D_k e = D_{k,l} e$ for all $k \leq l$. Further notice, setting $d = 1$ reduces the binomial-like tree to an infinite strip and the associated QBD to the classic QBD Markov chains where the number of states at level 0 may differ from the number of states on level $i > 0$.

Next we introduce a number of matrices that play a crucial role when studying the stability and stationary behavior of a QBD on a binomial-like tree. Let V_k denote the $h_k \times h_k$ matrix whose $(i,j)^{th}$ element is the taboo probability that starting from a state of the form $(J+k, i)$, the process eventually returns to node $J+k$ by visiting $(J+k, j)$, under the taboo of node J. These taboo probabilities only depend on k as $D_k e = D_{k,l} e$, for all $k \leq l$. One readily establishes that the following relationships hold:

$$V_k = B_k + \sum_{s=1}^{k} U_{k,s} (I - V_s)^{-1} D_{s,k}, \tag{4}$$

Let the $(i,j)^{th}$ element of the $h_k \times h_s$ matrix $R_{k,s}$ denote the expected number of visits to state $(J + ks, j)$ before visiting node $J + k$ again, given that $(X_0, N_0) = (J + k, i)$ and let R_s hold the expected number of visits to state (s, j), under taboo of node $\emptyset$, given $(X_0, N_0) = (\emptyset, i)$. These matrices obey the following equations:

$$R_{k,s} = U_{k,s} (I - V_s)^{-1}, \tag{5}$$

$$R_s = U_s(I - V_s)^{-1}. \tag{6}$$

Finally, define the $(i,j)^{th}$ entry of the matrix $G_{k,l}$ as the probability that, starting from a state of the form $(J + lk, i)$, we eventually visit node $J + l$ by visiting state $(J + l, j)$. Similarly, let G_k hold the probabilities that the root node is visited through state $(\emptyset, j)$ starting from state (k, i). The G-matrices can be expressed in terms of the V_k matrices by observing that:

$$G_{k,l} = (I - V_k)^{-1} D_{k,l}, \tag{7}$$

$$G_k = (I - V_k)^{-1} D_k. \tag{8}$$

2 Steady State Analysis

Throughout this section, we assume that the QBD process $\{(X_t, N_t), t \geq 0\}$ defined in the previous section is irreducible. In practice some transient states are allowed as their corresponding entries in the steady state vector will automatically become zero when applying the proposed algorithm. Clearly, the MC (X_t, N_t) is ergodic if and only if

$$D_k e + V_k e = e,$$

for all $1 \leq k \leq d$, which can be reformulated as $G_{k,l}e = e$, for all $k \leq l$, by rewriting it as $(I - V_k)^{-1} D_k e = e$ and using $D_k e = D_{k,l} e$, as well as equation (7). An alternate stability condition will be given further on.

Define

$$\begin{aligned} \pi_i(J) &= \lim_{t \to \infty} P[X_t = J, N_t = i], \\ \pi(\emptyset) &= (\pi_1(\emptyset), \pi_2(\emptyset), \ldots, \pi_{h_\emptyset}(\emptyset)), \\ \pi(J') &= (\pi_1(J'), \pi_2(J'), \ldots, \pi_{h_{f(J',1)}}(J')), \end{aligned}$$

for $J' \neq \emptyset$. As a direct consequence of [11, Theorem 5.2.1] the following matrix-geometric form can be revealed for the components $\pi(J)$ of the steady state vector:

$$\pi(s) = \pi(\emptyset) R_s, \tag{9}$$

$$\pi(J + ks) = \pi(J + k) R_{k,s}. \tag{10}$$

The first set of balance equations can be rewritten as

$$\pi(\emptyset) = \pi(\emptyset)(F + \sum_{s=1}^{d} U_s (I - V_s)^{-1} D_s),$$

to provide us with the vector $\pi(\emptyset)$. Using (9) and (10), the normalization condition $\sum_J \pi(J) e = 1$

can be rewritten as

$$\pi(\emptyset)\left(e+\sum_{s=1}^{d} R_s\Phi_s\right)=1,$$

with

$$\Phi_1 = (I-R_{1,1})^{-1}e, \tag{11}$$

$$\Phi_k = (I-R_{k,k})^{-1}\left(e+\sum_{s=1}^{k-1} R_{k,s}\Phi_s\right), \tag{12}$$

for $1 \leq k \leq d$. If the MC (X_t, N_t) is ergodic, then the series $\sum_J \pi(J)e$ must converge, meaning $\Phi_1, \Phi_2, \ldots, \Phi_d$ must converge, and this implies that the spectral radius of the matrices $R_{k,k}$, for $1 \leq k \leq d$, is less than one and $(I-R_{k,k})^{-1}$ exists.

The steady state vectors $\pi(J)$ can therefore be determined from the matrices V_k, for $1 \leq k \leq d$. Equation (4), for $k=1$, equals

$$V_1 = B_1 + U_{1,1}(I-V_1)^{-1}D_{1,1},$$

which has an identical form as the matrix equation for the U-matrix of a classic QBD Markov chain [11]. Moreover, $D_{1,1}e + B_1e + U_{1,1}e = e$, therefore V_1 is the U-matrix of a QBD Markov chain characterized by $(D_{1,1}, B_1, U_{1,1})$. Thus, we can compute V_1 by solving the quadratic matrix equation

$$Y_1 = D_{1,1} + B_1Y_1 + U_{1,1}Y_1^2,$$

and setting $V_1 = B_1 + U_{1,1}Y_1$. Any algorithm, including the logarithmic or cyclic reduction algorithm, to solve this type of equation can be used to obtain Y_1 [2]. Rewriting equation (4) for V_k, for $k = 2, \ldots, d$ gives

$$V_k = \left(B_k + \sum_{s=1}^{k-1} U_{k,s}(I-V_s)^{-1}D_{s,k}\right) + U_{k,k}(I-V_k)^{-1}D_{k,k}.$$

Having obtained V_1 to V_{k-1}, V_k is also a U-matrix of a classic QBD characterized by $(D_{k,k}, \bar{B}_k, U_{k,k})$, where $\bar{B}_k$ is defined as the expression between brackets. It is readily checked that $D_{k,k}e + \bar{B}_ke + U_{k,k}e = e$ whenever $V_ke + D_ke = e$. Hence, $V_k = \bar{B}_k + U_{k,k}Y_k$, where Y_k solves the quadratic equation

$$Y_k = D_{k,k} + \bar{B}_kY_k + U_{k,k}Y_k^2.$$

In conclusion, to obtain the matrices V_k, it suffices to solve d quadratic matrix equations, each of these can be solved using an algorithm with quadratic convergence.

From the probabilistic interpretation of the Y_k matrices, one finds that $Y_k = G_{k,k}$. Also, if the $G_{k,k}$ matrices are stochastic, then so are all $G_{k,l}$ matrices, for $k < l$, because of (7) and $D_{k,k}e = D_{k,l}e$. Therefore, using the classic QBD stability condition [11], we find that the MC (X_t, N_t) is stable if and only if

$$\theta_k D_{k,k} e < \theta_k U_{k,k} e,$$

where θ_k is the stochastic invariant vector of the stochastic matrix $D_{k,k} + \bar{B}_k + U_{k,k}$. Notice, this condition is not readily checked as the matrices V_k need to be determined first for $k = 1, \ldots, d-1$ in order to compute the matrices $\bar{B}_2$ to $\bar{B}_d$.

We end this section by introducing a simple procedure for some quantities that are useful when deriving performance measures. Let Π_k be the sum of all $\pi(J)$ vectors for which $f(J,1) = k$, then

$$\Pi_d = \pi(\emptyset) R_d (I - R_{d,d})^{-1}, \tag{13}$$

$$\Pi_k = \left(\pi(\emptyset) R_k + \sum_{s=k+1}^{d} \Pi_s R_{s,k} \right) (I - R_{k,k})^{-1}, \tag{14}$$

for $k = 1, \ldots, d-1$. From these and the vectors Φ_k, the probability $P[N(J,k) = t]$ of having t integers k on the string can be expressed as, for $t > 0$:

$$P[N(J,k) = t] = \Pi_k (I - R_{k,k}) R_{k,k}^{t-1} (I - R_{k,k}) \Phi_k. \tag{15}$$

This equation can be justified by noticing that $\Pi_k(I - R_{k,k}) = \left(\pi(\emptyset) R_k + \sum_{s=k+1}^{d} \Pi_s R_{s,k} \right)$ corresponds to the sum of all $\pi(J)$ vectors for which $f(J,1) = k$, but $f(J,2) \neq kk$. As a consequence, $\Pi_k (I - R_{k,k}) R_{k,k}^{t-1}$ equals the sum of all $\pi(J)$ vectors for which $f(J,t) = k \ldots k$, but $f(J,t+1) \neq kk \ldots k$. Multiplying this result with $(I - R_{k,k})\Phi_k = \left(e + \sum_{s=1}^{k-1} R_{k,s} \Phi_s \right)$ is therefore exactly the probability of having t integers k on the string as stated in (15).

3 The Multilevel Feedback Queue

The multilevel feedback queue is a scheduling discipline introduced in the late 1960s [16, 7, 10], that has become an important CPU scheduling principle found in many operating systems textbooks. The goal of a multilevel feedback queue scheduler is to fairly and efficiently schedule a mix of processes with a variety of execution characteristics (e.g., I/O-bound or CPU-bound). By introducing different priority levels and by controlling how a process moves between priority levels, processes with different characteristics can be scheduled as appropriate. The scheduler attempts to provide a compromise between several desirable metrics, such as response time for interactive jobs, throughput for compute-intensive jobs, and fair allocations for all jobs. A multilevel feedback queue gives preference to short and I/O-bound jobs and quickly establishes the nature of a process and schedules it accordingly.

In its pure form, the multilevel feedback queue uses a set of priority queues and operates as follows. All new job arrivals are termed class-0 jobs and are placed in the priority 0 FIFO queue. The scheduler processes these class-0 jobs in a time sharing manner, that is, each class-0 job is processed for a maximum of q_0 time units. If a job is not completed before the quantum q_0 expires, it becomes a class-1 job and is placed in the priority 1 FIFO queue, where it is set to wait until the priority 0 queue is empty. When the scheduler finds the priority 0 queue empty, it continues serving class-1 jobs. A class-1 job receives at most q_1 time units of uninterrupted service before becoming a class-2 job, etc. In general,

- A strict priority discipline is used between all FIFO queues.
- A class-k job can only receive service when there are no class-l jobs with $l < k$ present.
- When a class-k job enters the CPU, it is allowed to occupy the processor for a maximum of q_k time units.
- If a class-k job does not end within its quantum of length q_k, it becomes a class-$(k+1)$ job and is placed at the back of the priority $k+1$ FIFO queue.
- New jobs arriving while a class-k job, with $k > 0$, is in service will not preempt the class-k job before it has received q_k time units of service, unless the class-k job completes.

Thus, a multilevel feedback queue will favor short jobs, without having a priori knowledge about the job lengths. Starvation of large jobs can be counteracted by assigning longer quanta q_k to the lower priority classes.

Most work in this area has focused on systems with an infinite number of classes and on the limiting case where $q_k = q \to 0$ for all k. Using the framework introduced in the previous sections, we demonstrate how one can analyze the multilevel feedback queue under the following assumptions:

1. Time is slotted and all quanta q_k are expressed as multiples of the time unit.
2. The quanta $q_k = 1$ for all k. This assumption can be relaxed without great difficulty, we restrict ourselves to $q_k = 1$ to simplify the Markov chain description.
3. A (possibly empty) batch of jobs arrives at each slot boundary. The batch size in consecutive slots is assumed independent and identically distributed. We denote b_i as the probability of having a size i batch, for $i \geq 0$. Without much effort, one can also incorporate a batch Markovian arrival process (D-BMAP), if needed.
4. The processing time S of all jobs is independent and identically distributed. Denote $s_i = P[S = i]$ as the probability that a job requires i time units of processing. This general distribution is assumed to have a finite support, i.e., there exists some d such that $P[S > d+1] = 0$.
5. The number of classes is (greater than or) equal to $d+1$ and are labeled class-0 to class-d. Notice, a class-d job always ends after one time unit of processing.
6. All class-k queues have a finite buffer size equal to N.

3.1 Markov Model

Denote c_t^i as the number of class-i jobs present in the system at time t. When observing the system just prior to the slot boundaries, a Markov chain $(C_t)_{t\geq 0}$ is obtained by keeping track of the queue contents c_t^i of all $d+1$ queues at the end of time slot t. Let $m(t)$ be the smallest index i such that $c_t^i > 0$ (or d if no such index exists). We represent the tuple $(c_t^{m(t)+1}, \ldots, c_t^d)$ as an ordered string of integers $J_t = \psi(c_t^{m(t)+1}, \ldots, c_t^d)$ between 1 and d as follows:

$$\psi(c_t^{m(t)+1}, \ldots, c_t^d) \stackrel{def.}{=} \overbrace{d \ldots d}^{c_t^d} \overbrace{(d-1) \ldots (d-1)}^{c_t^{d-1}} \ \ldots \ \overbrace{(m(t)+1) \ldots (m(t)+1)}^{c_t^{m(t)+1}}. \tag{16}$$

Thus, the tuple $(J_t, (m(t), c_t^{m(t)}))$ fully characterizes the system at time t and provides us with a Markov chain having a binomial-like tree of order d as its state space. Let us consider the different transitions that might occur in such a Markov chain, for now assume all queues are of infinite size ($N = \infty$). In all cases, the job in progress at time t will finish with probability $p_{m(t)+1} = P[S = m(t)+1 | S > m(t)]$.

Assume there are no new job arrivals at time t. (A1) If $c_t^{m(t)} > 1$ and the job in progress finishes, $m(t+1) = m(t)$, $c_{t+1}^{m(t+1)} = c_t^{m(t)} - 1$ and $J_{t+1} = J_t$. If the job in progress did not complete, $m(t)+1$ needs to be added to the string J_t to get J_{t+1}. (A2) On the other hand, if $c_t^{m(t)} = 1$ and the job completes at time t, we will delete[1] all the integers equal to $m(t+1) = f(J_t, 1)$ from the string J_t to obtain J_{t+1} and set $c_{t+1}^{m(t+1)}$ equal to number of integers removed. If the job does not complete, $m(t+1) = m(t)+1$ and we remove all (if any) integers equal to $m(t)+1$ from the string J_t, while setting $c_{t+1}^{m(t+1)}$ equal to one plus the number of removed $m(t)+1$ values. Clearly, some of these transitions are not allowed in a QBD MC, as multiple integers are potentially deleted from the string J_t at once.

If one or multiple new jobs do arrive, we distinguish between having (B1) $m(t) = 0$ and (B2) $m(t) > 0$, in both cases $m(t+1) = 0$. (B1) In this particular case we simply increase the value of $c_t^{m(t)}$ taking into account the new arrivals and add a 1 to the string J_t if the job in progress did not complete. (B2) If there were no class-0 jobs present, we first add a single $m(t)+1$ (if there is no job completion) followed by a series of $c_t^{m(t)} - 1$ integers $m(t)$. $c_{t+1}^{m(t+1)}$ will hold the number of new arrivals. Again, the QBD does not allow us to add a series of integers at once to the string J_t.

To reduce the Markov chain $(J_t, (m(t), c_t^{m(t)}))$ to a QBD having a binomial-like tree as its state space, we will slightly expand the state space of the chain. Currently, we have $J_t \in \Omega$ and $(m(t), c_t^{m(t)}) \in \{(k,n) | k = 0, \ldots, f(J_t, 1) - 1, n = 1, \ldots, N\}$ if $J_t \neq \emptyset$. Otherwise $(m(t), c_t^{m(t)}) \in \{(k,n) | k = 0, \ldots, d, n = 0, \ldots, N\}$. Remark that the number of states of a node depends on the rightmost integer of the string identifying the node. By limiting the range of the variable $c_t^{m(t)}$ to N, we automatically enforce that all queues have a finite capacity equal to N. Although the string J_t might contain more than N identical integers i, the ones that should have been deleted earlier (due to a buffer overflow), are deleted as soon as service is provided to the class-i queue (because

[1] Provided that the string $J \neq \emptyset$, otherwise the string remains identical to $\emptyset$.

all these integers need to be stored into the $c_t^{m(t)}$ variable first).

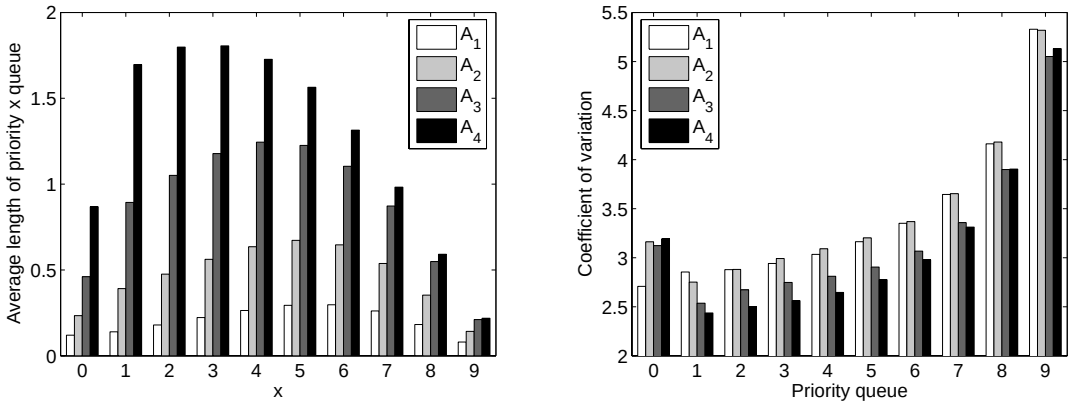
The transitions described in step (A2) can be avoided by adding a set of states $\{(-,n)|0 \leq n \leq N\}$ to each node variable J_t. Whenever we need to remove a series of $k > 1$ identical integers $m(t+1)$ from the string J_t, we can remove them one at a time. First, we enter the state $(-,0)$ after which the first $m(t+1)$ is removed from J_t and a transition to state $(-,1)$ is made, subsequently we visit the states $(-,2),(-,3)$ to $(-,k-1)$, each time removing one $m(t+1)$ from the string.[2] While removing the last $m(t+1)$ we enter state $(m(t+1),k)$. Notice, such transitions can be captured by a QBD as a transition to a parent node may depend on the rightmost integer of both the node and its parent, thus we can stay in states of the form $(-,x)$ as long as both these integers remain identical. As a consequence, there is no need to store the class $m(t+1)$ in the auxiliary variable when removing a series of identical integers, which is beneficial for the block size of the matrices involved.

The transitions of step (B2) are slightly more complicated in the sense that we add a list of k identical integers $m(t)$ possibly preceded by a single $m(t)+1$. To distinguish between these two scenario's we add two sets of additional states: $\{(+,n)|1 \leq n \leq N\} \cup \{(+1,n)|1 \leq n \leq N\}$. For such transitions, the variable $c_{t+1}^{m(t+1)}$ will hold the nonzero size of the batch arrival. If needed, we start by adding $m(t)+1$ to the string J_t and setting the auxiliary variable equal to $(+1,k)$, to indicate that k identical integers need to be added to the string. The value of these integers can be deduced by subtracting one from the rightmost element of the string (hence, the 1 in the notation). Subsequently, we visit states $(+,k-1)$ to $(+,1)$ and each time add a single integer to the string (the value of which is identical to the current rightmost element of the string, except for the first step where we might need to subtract one). Finally, we enter state $(0,x)$, where x is drawn from the batch size distribution, knowing it concerns a nonempty batch. Thus, we postpone determining the batch size until we add the last element to the string. We can support all these transitions by the QBD framework as a transition to a child node is allowed to depend on the rightmost element of the current node. For reasons of completeness, the matrices B_k, $D_{k,l}$, $U_{k,s}$, F, D_k and U_s, for $1 \leq s \leq k \leq l \leq d$, describing the QBD on the binomial-like tree are presented in the appendix. Having obtained the steady state vectors $\pi(J)$ of this expanded (QBD) Markov chain, we can deduce the steady state of the chain $(J_t,(m(t),c_t^{m(t)}))$ by applying a censoring argument.

3.2 Performance Measures

To conclude this section, we present some illustrative numerical examples. In these examples, we take a closer look at the influence the batch size and the processing time distribution have on the queue length of each queue individually and the average buffer occupancy of the entire system. The average response times (per job length) is also discussed. Consider a multilevel feedback queue with 10 priority classes where each buffer can store at most $N = 30$ jobs.
The processing time of a job follows a discrete uniform distribution with $P[S = i] = 0.1$, for

[2] Remark that there is no real need to include state $(-,0)$, however it allows us to give a somewhat easier description of the transition matrices.



(a) Average queue length

(b) Dispersion of the queue length distribution

Figure 2: Queue lengths in a multilevel feedback queue with 10 priorities and different arrival characteristics

$i = 1, \ldots, 10$. The mean arrival rate equals 0.12 jobs per slot, resulting in a total load of $\rho = 0.66$. We consider 4 different arrival processes, given by

$A_1 :\ b_0 = 0.88$, $b_1 = 0.12$, and, $b_i = 0$, for $i > 1$,

$A_2 :\ b_0 = 0.94$, $b_1 = 0.03$, $b_i = 0.01$, for $2 \leq i \leq 4$, and, $b_i = 0$, for $i > 4$,

$A_3 :\ b_0 = 0.98$, $b_6 = 0.02$, and, $b_i = 0$, for $i \notin \{0, 6\}$,

$A_4 :\ b_0 = 0.99$, $b_{12} = 0.01$, and, $b_i = 0$, for $i \notin \{0, 12\}$.

Notice, $A_1 \leq_{cx} A_2 \leq_{cx} A_3 \leq_{cx} A_4$ in the convex ordering sense. Figure 2(a) shows the average queue length for each of the 10 priority classes. As the batch size distribution becomes larger, the average queue length increases for each queue. Furthermore, the priority of the queue attaining the highest average occupancy increases with the burstiness of the arrival process. From Figure 2(b) it can be observed that the coefficient of variation, i.e., the ratio of the standard deviation to the mean, is larger for queues with a lower priority. Furthermore, for a burstier arrival process, the coefficient of variation of the queue length is slightly smaller.

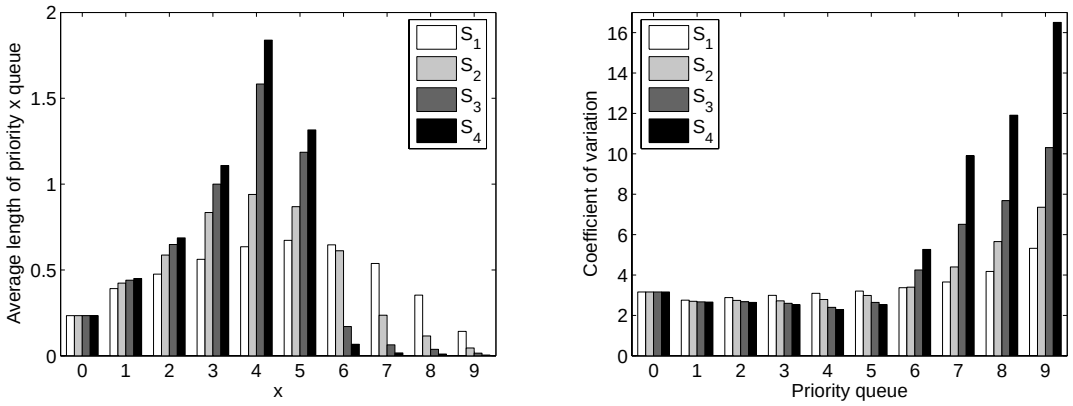
In the following example, we keep the arrival process fixed, while varying the job processing times. New jobs enter the multilevel feedback queue according to the second arrival process discussed in the previous example, i.e., $b_0 = 0.94$, $b_1 = 0.03$, $b_i = 0.01$, for $2 \leq i \leq 4$, and $b_i = 0$ for $i > 4$. Furthermore, we consider four different job processing time distributions given by:

$S_1 :\ P[S = i] = 0.1$ for $1 \leq i \leq 10$,

$S_2 :\ P[S = i] = 0.05$ for $1 \leq i \leq 3$, $8 \leq i \leq 10$, $P[S = i] = 0.175$ for $4 \leq i \leq 7$,

$S_3 :\ P[S = i] = 0.025$ for $1 \leq i \leq 4$, $7 \leq i \leq 10$, $P[S = i] = 0.4$ for $i \in \{5, 6\}$,

$S_4 :\ P[S = i] = 0.01$ for $1 \leq i \leq 4$, $7 \leq i \leq 10$, $P[S = i] = 0.46$ for $i \in \{5, 6\}$.



(a) Average queue length

(b) Dispersion of the queue length distribution

Figure 3: Queue lengths in a multilevel feedback queue with 10 priorities under varying processing time distributions

Each of these distributions has an average, resp. maximum, processing time of 5.5, resp. 10, slots. Also, $S_1 \geq_{cx} S_2 \geq_{cx} S_3 \geq_{cx} S_4$ in the convex ordering sense. The queue length results obtained for these scenarios are presented in Figure 3. It can be observed that for the uniform processing time distribution, the average queue length is more spread out over the different queues compared to a processing time distribution where the probability mass is more centered around the mean. Considering the coefficient of variation, we observe larger values for the queues with a lower priority similar to the previous example.

In Table 3.2 we present the average queue lengths over all 10 priority queues for each of the discussed scenarios (as well as the total loss rate ν due to the finite nature of the queues). How this loss rate can be computed, will be discussed later on. As mentioned earlier, the average queue length increases together with the burst size of newly arriving jobs. Considering the processing time distribution, we observe that the average queue length is minimal for the uniform distribution and increases for distributions where the probability mass is less spread out.

scenario	avg queue length	loss rate	scenario	avg queue length	loss rate
A_1	2.041 jobs	1.93e-7	S_1	4.650 jobs	1.38e-4
A_2	4.650 jobs	1.38e-4	S_2	4.896 jobs	2.55e-4
A_3	8.789 jobs	2.08e-3	S_3	5.378 jobs	9.72e-4
A_4	12.566 jobs	8.90e-3	S_4	5.732 jobs	1.48e-3

Table 1: Average queue length in the 10-class multilevel feedback queue

From the average queue length we can also deduce the average response time for the different jobs in the system. Denote jobs with a required processing time of $i+1$ slots as type-i jobs, i.e., a type i job has to pass through priority queues $0, \ldots, i$ before leaving the system. Using Little's law we can compute the average response time of a type-i job who was allowed to enter the first queue. We start by determining the input rate of the different buffers, taking into account jobs that leave

the system after having received their required processing time as well as jobs that are dropped due to buffer overflow. For priority queue 0, the input rate equals the average rate at which a job arrives in the system, that is, $\lambda_0 = \sum_{i=1}^{\infty} i b_i$. The input rate λ_i of queue i ($i > 0$) can be obtained from the output rate μ_{i-1} of queue $i-1$, which can be computed by

$$\mu_{i-1} = \pi_{i-1}(\emptyset)e + \sum_{j=i}^{d} \Pi_{j,i-1} e, \tag{17}$$

with

$$\begin{aligned} \pi_{k,l}(J) &= \lim_{t\to\infty} P[X_t = J, (m(t), c_t^{m(t)}) = (k,l)], \\ \pi_k(J) &= (\pi_{k,1}(J), \ldots, \pi_{k,N}(J)), \\ \pi(J) &= (\pi_0(J), \ldots, \pi_d(J)), \end{aligned}$$

and $\Pi_{k,l}$ is defined as the sum off all vectors $\pi_l(J)$ for which $f(J,1) = k$, analogous to equations (13) and (14). That is, in (17) we sum all probabilities corresponding to states where queue $i-1$ is active. The arrival rate λ_i is then given by $\lambda_i = (1-p_i)\mu_{i-1}$.

Applying Little's law to each of these $d+1$ queues, we see that the mean time spent in queue i of all jobs passing through queue i equals N_i/μ_i (notice, as the queue is finite, we have to use μ_i and not λ_i). As the type of a type-j job with $j \geq i$ is irrelevant for the time it spends in queue i, this mean value is correct for all type-j jobs passing through queue i. We denote the average response time of a type-i job that entered our system as RT_i. Thus, if a type-i job is only partially processed, it also contributes to RT_i. As all the type-0 jobs entering the system get processed by queue 0, we have $RT_0 = N_0/\mu_0$. For the type-1 jobs, there are two components: N_0/μ_0 for the mean time a class-1 customer spends in queue 0, plus the mean time it spends in queue 1, provided that it enters queue 1. The probability that a type-1 job enters this queue is μ_1/λ_1. In conclusion, $RT_1 = N_0/\mu_0 + N_1/\lambda_1$. Similarly, one establishes that

$$RT_i = \frac{N_0}{\mu_0} + \sum_{k=1}^{i} \frac{N_k}{\lambda_k}.$$

Figure 4 presents the average response time of the different types of jobs in the 8 scenarios. As could be expected, the burstiness of the arriving jobs has a large influence on this average response time. For the processing time distributions we observe that the distribution that leads to the smallest average response time depends on the type of customer. For example, for a customer with a required processing time of 6 slots, a larger distribution in the convex ordering sense leads to a smaller average response time, whereas for a customer of type 10, the opposite observation can be made.

From the input and output rate of each queue we can also obtain the different loss rates, i.e., the loss rate ν_i ($i > 0$) of queue i is given by $\nu_i = \mu_i - \lambda_i$. The total loss rate of the system then

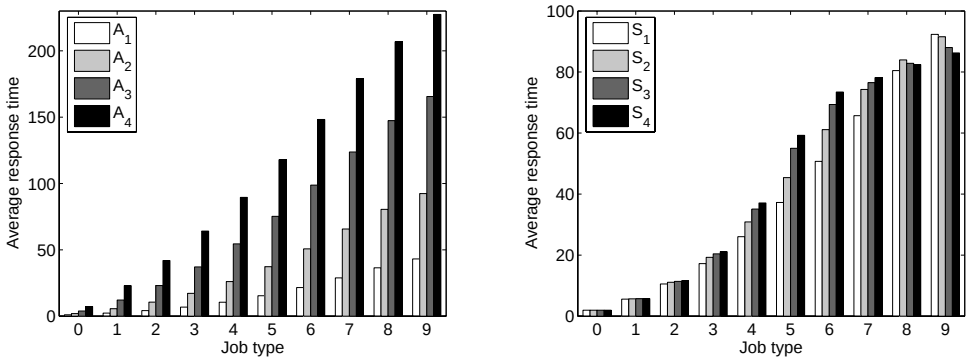


Figure 4: Average response time of type t customers in a multilevel feedback queue with 10 priorities under varying arrival characteristics and with different processing time distributions

equals $\nu = \sum_{k=0}^{d} \nu_k$. In Table 3.2 this total loss rate is presented for the different scenarios under consideration.

A Appendix A: Transition Matrices

In this appendix we will briefly describe the transition matrices of the expanded QBD Markov chain $(J_t, (m(t), c_t^{m(t)}))$. Since three types of additional states were added, we have $h_k = (3+k)N + 1$, for $k = 1, \ldots, d$. The first block row corresponds to a transition from an additional state of type $(-, n)$, with $0 \leq n \leq N$, the second and the third block row to a transition from a state of type $(+1, n)$, respectively $(+, n)$, with $1 \leq n \leq N$. The first block row will therefore consist of $N + 1$ rows, whereas the other block rows have only N rows. To reduce the notational complexity, we introduce the following matrices of which the appropriate dimension depends on the context:

- $M_{i,j}$ is a zero matrix with a one in row i, column j.
- e_i is a zero column vector with a one on entry i.
- $b^1 = (b_1, b_2, \ldots, b_N)$, where $b_j = 0$ if j exceeds the maximum batch size.
- B_T^1 denotes an upper triangular Toeplitz matrix with its first row equal to b^1.
- I^+ denotes a zero matrix with ones on the upper diagonal.
- $I^=$ denotes a zero matrix with ones on the main diagonal.
- I^- is a zero matrix with ones on the lower diagonal.
- I^{--} is a zero matrix with ones on the diagonal below the lower diagonal.

Remark, for $* \in \{+, =, -, --\}$, I^* is not necessarily a square matrix. The $h_k \times h_k$ matrix B_k covers the transitions where $J_{t+1} = J_t$ and $f(J_t, 1) = k$. Hence,

$$B_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ p_1 b_0 M_{1,1} & 0 & 0 & p_1(B_T^1 + b_0 I^-) & 0 & 0 & \cdots & 0 \\ p_2 b_0 M_{1,1} & 0 & 0 & p_2 e_1 b^1 & p_2 b_0 I^- & 0 & \cdots & 0 \\ p_3 b_0 M_{1,1} & 0 & 0 & p_3 e_1 b^1 & 0 & p_3 b_0 I^- & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_k b_0 M_{1,1} & 0 & 0 & p_k e_1 b^1 & 0 & 0 & \cdots & p_k b_0 I^- \end{bmatrix}.$$

The $h_k \times h_l$ matrices $D_{k,l}$ contain the transition probabilities between state $(-, i)$ and state $(-, i+1)$, where $f(J_t, 1) = k$ and $f(J_{t+1}, 1) = l$. Therefore,

$$D_{k,k} = \begin{bmatrix} I^+ + M_{N+1,N+1} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix},$$

$$D_{k,l} = \begin{bmatrix} 0 & \cdots & 0 & I^= + M_{N+1,N} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \end{bmatrix},$$

for $l > k$. The non-zero entries of the matrix $D_{k,l}$ are positioned on the $(k+4)$-th block column of the first block row since $m(t+1) = k$. The $h_k \times h_s$ matrices $U_{k,s}$ represent the situations in which an integer is added to the string J_t and are given by

$$U_{k,1} = \Delta_{k,1} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & I^- & \frac{e_1 b^1}{1-b_0} \\ 0 & 0 & I^- & \frac{e_1 b^1}{1-b_0} \\ b_0 M_{11} & 0 & 0 & B_T^1 + b_0 I^- \\ 0 & 0 & (1-b_0) I^{--} & e_2 b^1 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$U_{k,s} = \Delta_{k,s} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & I^- & \frac{e_1 b^1}{1-b_0} & 0 & \cdots & 0 \\ 0 & 0 & I^- & \frac{e_1 b^1}{1-b_0} & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ b_0 M_{11} & (1-b_0) I^- & 0 & e_1 b^1 & 0 & \cdots & b_0 I^- \\ 0 & 0 & (1-b_0) I^{--} & e_2 b^1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix},$$

with $s \neq 1$ and $\Delta_{k,s}$ a diagonal matrix represented by

$$\Delta_{k,s} = diag(e^t, 1_{\{k=s+1\}} e^t, 1_{\{k=s\}} e^t, e^t, \ldots, (1-p_s)e^t, p_{s+1} e^t, e^t, \ldots, e^t),$$

where e^t denotes the transposed vector of e. In the matrix $U_{k,s}$ the non-zero (and in $\Delta_{k,s}$ the non-one) entries are located on the block rows 2, 3, $s+3$, and $s+4$. Finally, the following equations describe the matrices F, U_s and D_k:

$$F = \begin{bmatrix} b_0 & b^1 & 0 & 0 & \cdots & 0 & 0 \\ p_1 b_0 e_1 & p_1 (B_T^1 + b_0 I^-) & 0 & 0 & \cdots & 0 & 0 \\ p_2 b_0 e_1 & p_2 e_1 b^1 & p_2 b_0 I^- & 0 & \cdots & 0 & 0 \\ p_3 b_0 e_1 & p_3 e_1 b^1 & 0 & p_3 b_0 I^- & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_d b_0 e_1 & p_d e_1 b^1 & 0 & 0 & \cdots & p_d b_0 I^- & 0 \\ b_0 e_1 & e_1 b^1 & 0 & 0 & \cdots & 0 & b_0 I^- \end{bmatrix},$$

$$D_k = \begin{bmatrix} 0 & \cdots & 0 & I^+ + M_{NN} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \end{bmatrix},$$

$$U_1 = \Delta_1 \begin{bmatrix} 0 & 0 & 0 & 0 \\ b_0 M_{11} & 0 & 0 & B_T^1 + b_0 I^- \\ 0 & 0 & (1-b_0) I^{--} & e_2 b^1 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$U_s = \Delta_s \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ b_0 M_{11} & (1-b_0)I^- & 0 & e_1 b^1 & 0 & \cdots & b_0 I^- \\ 0 & 0 & (1-b_0)I^{--} & e_2 b^1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix},$$

for $s \neq 1$ and where $\Delta_s = diag(e^t, \ldots, e^t, (1-p_s)e^t, p_{s+1}e^t, e^t, \ldots, e^t)$. Remark, there are no additional states in the root node, hence $h_\phi = (d+1)N + 1$. The first row of the matrices F and U_s corresponds to a transition from an idle system.

References

[1] S. Asmussen, O. Nerman, and M. Olsson. Fitting phase-type distributions via the EM algorithm. *Scandinavian Journal of Statistics*, 23:419–441, 1996.

[2] D. Bini, B. Meini, S. Steffé, and B. Van Houdt. Structured Markov chain solver: software tools. In *Proc. of the SMCTools workshop*, Pisa, Italy, 2006.

[3] D. A. Bini, G. Latouche, and B. Meini. *Numerical methods for structured Markov chains*. Oxford University Press, 2005.

[4] 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.

[5] A. Bobbio, A. Horváth, M. Scarpa, and M. Telek. Acyclic discrete phase type distributions: Properties and a parameter estimation algorithm. *Performance Evaluation*, 54(1):1–32, 2003.

[6] L. Breuer. An EM algorithm for batch Markovian arrival processes and its comparison to a simpler estimation procedure. *Annals of Operations Research*, 112:123–138, 2002.

[7] E. Coffman and L. Kleinrock. Feedback queueing models for time-shared systems. *Journal of the Association for Computing Machinery*, 15:549–576, 1968.

[8] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press and McGraw-Hill, 1990.

[9] G. Horváth, P. Buchholz, and M. Telek. A MAP fitting approach with independent approximation of the inter-arrival time distribution and the lag correlation. In *Proc of QEST 2005*. IEEE Computer Society, 2005.

[10] L. Kleinrock. A continuum of time-sharing algorithms. In *Proc. of AFIPS SJCC*, 1970.

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

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

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

[14] M. Nuyens and A. Wierman. The foreground-background queue: a survey. *Submitted*, 2006.

[15] T. Ryden. An EM algorithm estimation in Markov-modulated Poisson processes. *Computational Statistics & Data Analysis*, 21(4):431–447, 1996.

[16] L. Schrage. The queue M/G/1 with feedback to lower priorities. *Management Science*, 18:466–474, 1967.

[17] W. Stallings. *Operating Systems: Internals and design principles*. Prentice Hall, 2005.

[18] 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.

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

[20] 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.

[21] 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.