QBDs with Marked Time Epochs: a Framework for Transient Performance Measures

B. Van Houdt and C. Blondia
University of Antwerp
PATS Research Group
Middelheimlaan 1, B-2020 Antwerpen, Belgium
{benny.vanhoudt,chris.blondia}@ua.ac.be

Abstract

A framework to assess transient performance measures is introduced by generalizing the theory of the quasi Birth-and-Death (QBD) paradigm to QBDs with marked time epochs (QBD^m). The distinction with the classical QBD process is that certain time epochs get marked according to a specific set of Markovian rules. Our interest lies in obtaining the system state at the n-th marked time epoch. The steady state vector of a so-called reset Markov chain is used to obtain the above-mentioned system state (either by approximation or in an exact manner). A fast algorithm, with limited memory usage, based on solving a single quadratic matrix equation, a set of Sylvester matrix equations and fast Fourier transforms is proposed. The generality and flexibility of our framework is demonstrated on a set of queueing systems and applied to dimensioning a video playout buffer and studying the transient throughput of a wireless random access algorithm.

1 Introduction

A procedure to approximate the queue length distribution at time t^* and the waiting time distribution of the n-th customer in a queue with Markovian arrivals and phase-type services (i.e., the D-MAP/PH/1 queue) was introduced in [23]. Using the same underlying concepts and ideas, we develop a new framework, termed quasi Birth-and-Death processes with marked time epochs (QBD^m), to assess transient performance measures in a plug-and-play mode. Apart from introducing this framework, the main contribution of the paper lies within the significant improvement achieved—both in terms of the time and memory complexity—when computing the steady state vector of the associated reset Markov chain. Amongst others, this computational gain is realized by reducing the problem of solving a quadratic matrix equation involving large matrices to a single quadratic matrix equation and a set of Sylvester matrix equations of a much smaller dimension. Also, a low memory solution to the boundary condition, that avoids the storage of large matrices, is provided.

Transient performance measures have long been recognized as being complementary to steady state measures [24, 11, 4, 16, 18]. Either because there exist a need to understand the initial behavior of a system, or simply because the system has no steady state. Transient studies are also motivated as a means to investigate the impact of the initial system state in a simulation environment.

The paper is structured as follows. In Section 2 we introduce the notion of a QBD with marked time epochs. We demonstrate the flexibility of this framework in Section 3 on a class of queueing systems. A new and improved computational algorithm to assess transient performance measures is discussed in Section 5, whereas some numerical data is presented in Section 6

2 QBD Markov chains with marked time epochs

In order to develop a general framework to establish transient performance measures, we introduce the notion of a quasi Birth-and-Death (QBD) Markov chain (MC) with *marked* time epochs. We start by reiterating the definition of the well known QBD MC. A QBD MC [10] is characterized by a transition matrix $\bar{P}$ of the form

$$\bar{P} = \begin{bmatrix} \bar{B}_1 & \bar{B}_0 & 0 & 0 & \dots \\ \bar{B}_2 & \bar{A}_1 & \bar{A}_0 & 0 & \ddots \\ 0 & \bar{A}_2 & \bar{A}_1 & \bar{A}_0 & \ddots \\ 0 & 0 & \bar{A}_2 & \bar{A}_1 & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix}, \qquad (1)$$

where $\bar{B}_1$ and $\bar{A}_1$ are square matrices of dimension l and d, respectively. Such a MC has an infinite state space Ω, whose states are labeled $1, 2, \ldots$. The set of states $\mathcal{U}_0 = \{1, \ldots, l\}$ is referred to as level zero of the MC, whereas the set of states $\mathcal{U}_i = \{l+(i-1)d+1, \ldots, l+id\}$ is referred to as level $i > 0$ of the MC. For further use, relabel the states of level $i > 0$ as $\{1, \ldots, d\}$. Thus, transitions from and to level 0 are governed by the $\bar{B}_s$ matrices. The $\bar{A}_s$ matrices describe the possible transitions from level $i > 0$ to $i' = i-1, i$ and $i+1$ (with $i' > 0$). Throughout the paper we assume that $\bar{A} = \bar{A}_0 + \bar{A}_1 + \bar{A}_2$ is irreducible (which is the case for nearly all applications). A QBD with marked time epochs (QBD^m) has the additional property that

$$\bar{A}_s = \bar{A}_s^u + \bar{A}_s^m, \quad \bar{B}_s = \bar{B}_s^u + \bar{B}_s^m,$$

for some $\bar{A}_s^m, \bar{A}_s^u, \bar{B}_s^m$ and $\bar{B}_s^u$ nonnegative ($s = 0, 1$ and 2), and $\bar{B}_2^m e = \bar{A}_2^m e$, with e a column vector with all its entries equal to one (this last constraint can be dropped if necessary). The (j, j')-th entry of the matrix $\bar{A}_s^m$ gives the probability that, at time t, a transition occurs from state j of level $i > 0$ to state j' of level $i - s + 1$ (> 0) and time epoch t is marked, while $\bar{A}_s^u$ gives the probability for the same event, but without marking time t. Whether a time epoch t is marked, therefore depends on the transition from time epoch t to $t+1$. The matrices $\bar{B}_s^m$ and $\bar{B}_s^u$ have the same obvious interpretations. For later use, define $\bar{A}^m$ as $\bar{A}_0^m + \bar{A}_1^m + \bar{A}_2^m$.

We refer to the initial time epoch as time $t = 0$ and limit ourselves to the case where the MC $\bar{P}$ is in state $q_{ini} \in \mathcal{U}_0$ at time $t = 0$. The initial state q_{ini} is determined by a stochastic $1 \times l$ vector α_{ini}, i.e., q_{ini} equals j with probability $(\alpha_{ini})_j$. In this paper we develop a procedure to compute (either exact or by approximation) the state of the Markov chain characterized by $\bar{P}$ at the n-th marked time epoch. Before doing so, we present some examples to demonstrate the general nature and flexibility of this framework.

3 Demonstrating the applicability of QBD^{m}s

We will demonstrate the flexibility of our framework on the D-MAP/PH/1 queue. Obviously, any other (queueing) system that fits within the QBD paradigm would be suitable (several examples of QBDs can be found in [10, Chapter1], [5], [22]). Moreover, M/G/1- and GI/M/1-type Markov chains may be viewed as a special cases of QBDs [17]. Our choice for the D-MAP/PH/1 queue was driven by [8], where the transient behavior of a D-MAP/PH/1 queue was used to study the playout buffer of a video source.

D-MAP arrival processes [2] form a versatile class of tractable Markovian arrival processes, which, in general, are non-renewal. A D-MAP is defined by a set of two positive matrices D_0 and D_1: the entries $(D_1)_{j_1,j_2}$ represent the probability of having a transition from state j_1 to j_2 and a customer arrival, while a transition from state j_1 to j_2 without an arrival will occur with probability $(D_0)_{j_1,j_2}$.

A discrete time phase-type (PH) distribution can be represented in matrix form as (c, β, T), where c is a scalar, β a $1 \times c$ stochastic vector and T a $c \times c$ substochastic matrix [15]. The s-th component of the vector β represents the probability that a customer starts his service in phase s. Let $T^* = e - Te$, then the s-th entry of T^* denotes the probability that a customer completes his service provided that he is in phase s at the current time epoch. Finally, the (s_1, s_2)-th entry of T equals the probability that a customer continues his service in phase s_2 at the next time epoch provided that he is in phase s_1 at the current time epoch. The set of discrete time PH distributions is known to be very useful in approximating service time distributions encountered in communications networks [9]. A single work conserving server is considered.

It is well known [2] that the D-MAP/PH/1 queue forms a QBD Markov chain with a transition matrix $\bar{P}$ as in Eqn. (1), where $\bar{B}_0 = D_1 \otimes \beta$, $\bar{B}_1 = D_0$, $\bar{B}_2 = D_0 \otimes T^*$, $\bar{A}_0 = D_1 \otimes T$, $\bar{A}_1 = (D_0 \otimes T) + (D_1 \otimes T^*\beta)$ and $\bar{A}_2 = D_0 \otimes T^*\beta$, with $\otimes$ the matrix Kronecker product. Notice, depending on whether the load of the queue $\rho < 1$, the Markov chain $\bar{P}$ is stationary or not. Next, we discuss some transient performance measures that can be obtained via our framework.

System state at time $t = t^*$ In order to obtain the system state at time t^*, we can set $n = t^* + 1$ and

$$\bar{A}_s^m = \bar{A}_s, \quad \bar{B}_s^m = \bar{B}_s,$$

meaning we mark each time epoch and as such the n-th marked epoch is simply the system state at time t^* (as the first time epoch is time 0).

Waiting time of the n-th customer To compute the waiting time of the n-th customer in a D-MAP/PH/1 queue, it suffices to know the system state at the n-th arrival epoch. This time epoch corresponds to the n-th marked time epoch if we set

$$\begin{array}{lllll} \bar{A}_0^m & = & \bar{A}_0, & \bar{A}_1^m = D_1 \otimes T^*\beta, & \bar{A}_2^m = 0, \\ \bar{B}_0^m & = & \bar{B}_0, & \bar{B}_1^m = 0, & \bar{B}_2^m = 0. \end{array}$$

Notice, we mark each time epoch in which an arrival occurs.

Work left behind by the n-th customer To compute the work left behind by the n-th customer in a D-MAP/PH/1 queue, it suffices to know the system state at the n-th departure epoch. We can compute the system state at this time

epoch by considering the n-th marked time epoch if we set

$$\begin{array}{rcll} \bar{A}_0^m & = & 0, & \bar{A}_1^m = D_1 \otimes T^* \beta, \quad \bar{A}_2^m = \bar{A}_2, \\ \bar{B}_0^m & = & 0, & \bar{B}_1^m = 0, \qquad\qquad\quad \bar{B}_2^m = \bar{B}_2. \end{array}$$

Observe that each time epoch in which a departure occurs, is marked.

System state at the n-th visit to state j of the arrival process Let D_0^j and D_1^j equal D_0 and D_1, except that all its entries on the j-th row are set to zero, respectively. Denote the matrices A_s and B_s, for $s = 0, 1$ and 2, with D_i replaced by D_i^j, for $i = 0$ and 1, as A_s^j and B_s^j. By setting

$$\bar{A}_s^u = \bar{A}_s^j, \quad \bar{B}_s^u = \bar{B}_s^j,$$

we find that the n-th marked time epoch corresponds to the n-th visit to the D-MAP state j. This concept can be extended in a straightforward manner to the n-th visit to a set of states $\{j_1, \ldots, j_r\}$.

Furthermore, combinations of the examples above, e.g., the system state at the n-th arrival generated in state j of the arrival process, also fall within the reach of our framework.

4 Reset Markov chains

We denote $\pi^m(n)$ as the probability vector associated with the n-th marked time epoch, i.e., $\pi^m(n) = (\pi_0^m(n), \pi_1^m(n), \ldots)$ with $\pi_0^m(n)$ a $1 \times l$ and $\pi_s^m(n)$, for $s > 0$, a $1 \times d$ vector. In order to compute $\pi^m(n)$ efficiently (by approximation or exact), we make use of a so-called reset Markov chain. The concept of reset Markov chains was first introduced in [23], where the system state at time t^* and the waiting time of the n-th customer in a D-MAP/PH/1 queue was studied. Although the reset Markov chain introduced here is a simple abstraction of the MCs in [23], the algorithm used to compute the steady state vector π of the reset MC is by far more efficient in terms of both its time and memory complexity. This computational gain was achieved in three successive steps: (i) as opposed to using the cyclic reduction (CR) algorithm [13] to compute R in an iterative manner, we provide a direct recursive scheme, (ii) the structure of the boundary condition is exploited to compute π_0 and (iii) we make use of a FFT algorithm to compute the π_s ($s > 0$) components of π from π_0 and R.

The key property of the proposed method is that we can approximate the system state at the n-th marked time epoch $t^m(n)$, by considering the system state at the $Z_{k,n}$-th marked time epoch $t^m(Z_{k,n})$, where $Z_{k,n}$ is a negative binomially distributed random variable—which is the discrete time counterpart of the Erlang distribution—with k phases and a mean n, for k ($\leq n$) sufficiently large. In some cases we can actually set $k = n$ and obtain exact results, however k cannot always be set to n as the corresponding reset MC might become periodic. The idea to approximate time t by an Erlang distribution is not new and has been explored some time ago to obtain transient probabilities of finite state continuous time Markov chains [19, 3]. The choice for the negative binomial distribution (NBD) to approximate time $t^m(n)$ is in some sense optimal. Telek [20] has proven that the discrete time PH distribution with k phases, a mean $m_u \geq k$ and a minimal coefficient of variation is the NBD with parameters $(k/m_u, k)$. Thus, the closest we can get to a deterministic distribution with a mean m_u, if we make use of a *discrete* PH distribution with at most k phases, is the NBD with parameters $(p = k/m_u, k)$. Moreover, choosing the NBD as a reset time also provides the reset MC with useful structural properties that can be exploited when computing its steady state vector.

Let us now explain how to compute the system state at time $t^m(Z_{k,n})$ via a steady state analysis. Consider the stochastic process that evolves according to the transition matrix $\bar{P}$, but that is repeatedly reset when leaving the $Z_{k,n}$-th marked time epoch. Meaning, if we perform a Bernoulli trial each time a transition out of a marked time epoch takes place, with parameter $p = k/n$, the system is reset whenever k successes have occurred. The reset counter is then defined as the number of pending successes before the next reset event. Clearly the reset counter takes values in the range $\{1, 2, \ldots, k\}$. Although adding the reset counter variable as an additional auxiliary variable to the Markov chain $\bar{P}$ increases the size of its transition blocks by a factor k, we will demonstrate that there is no need to store any matrix of dimension kl or kd. After adding the reset counter as an additional auxiliary variable to the Markov chain $\bar{P}$, the reset process becomes an MC characterized by the transition matrix $P_{k,n}$:

$$P_{k,n} = \begin{bmatrix} B_1^{k,n} + C_0^{k,n} & B_0^{k,n} & 0 & 0 & \ldots \\ B_2^{k,n} + C_1^{k,n} & A_1^{k,n} & A_0^{k,n} & 0 & \ddots \\ C_1^{k,n} & A_2^{k,n} & A_1^{k,n} & A_0^{k,n} & \ddots \\ C_1^{k,n} & 0 & A_2^{k,n} & A_1^{k,n} & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix},$$

where

$$\begin{array}{rcl} A_s^{k,n} & = & (I \otimes (\bar{A}_s^u + (1-p)\bar{A}_s^m)) + (M_0^k \otimes p\bar{A}_s^m), \\ B_s^{k,n} & = & (I \otimes (\bar{B}_s^u + (1-p)\bar{B}_s^m)) + (M_0^k \otimes p\bar{B}_s^m), \\ C_0^{k,n} & = & M_1^k \otimes (p(\bar{B}_0^m e + \bar{B}_1^m e)\alpha_{ini}), \\ C_1^{k,n} & = & M_1^k \otimes (p\bar{A}^m e \alpha_{ini}), \end{array}$$

for $s = 0, 1$ or 2, and $p = k/n$. The $k \times k$ matrix M_0^k has ones on the first diagonal below its main diagonal and

all other entries equal to zero, while M_1^k has only one entry differing from zero being its last entry on the first row, which equals one. Notice, the block $C_1^{k,n}$ appearing on the second block row and first block column is a consequence of having $A_2^m e = B_2^m e$.

Let $\pi^{k,n} = (\pi_0^{k,n}, \pi_1^{k,n}, \pi_2^{k,n}, ...)$, with $\pi_0^{k,n}$ a $1 \times kl$ and $\pi_s^{k,n}$ ($s > 0$) a $1 \times kd$ vector, be the steady state vector of $P_{k,n}$. Moreover, let $\pi_s^{k,n} = (\pi_{s,1}^{k,n}, \ldots, \pi_{s,k}^{k,n})$, with $\pi_{s,j}^{k,n}$, for $j = 1, \ldots, k$, a $1 \times d$ ($1 \times l$) vector for $s > 0$ ($s = 0$). Conditions for the existence of the vector $\pi^{k,n}$ are discussed in Section 5. Provided that $\pi^{k,n}$ exists, the system state $\pi^m(Z_{k,n})$ at time $t^m(Z_{k,n})$, used as an approximation to $\pi^m(n)$, is the stochastic vector proportional to:

$$(\pi_{0,1}^{k,n}.\phi_0, \pi_{1,1}^{k,n}.\phi_1, \pi_{2,1}^{k,n}.\phi_1, \ldots)p,$$

where ϕ_0 and ϕ_1 are the transposed vectors of $\bar{B}_0^m e + \bar{B}_1^m e$ and $\bar{A}^m e$, respectively and '.' denotes the point-wise vector product. An efficient algorithm to compute $\pi^{k,n}$ and conditions that guarantee its existence are presented in Section 5.

5 Computing the stationary vector $\pi^{k,n}$ of $P_{k,n}$

The quasi Birth-and-Death (QBD) reset Markov chain developed in Section 4 is characterized by a transition matrix P of the form:

$$P = \begin{bmatrix} B_1 + C_0 & B_0 & 0 & 0 & \ldots \\ B_2 + C_1 & A_1 & A_0 & 0 & \ddots \\ C_1 & A_2 & A_1 & A_0 & \ddots \\ C_1 & 0 & A_2 & A_1 & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix}.$$

where A_i, for $i = 0, 1$ and 2, can be written as a $k \times k$ block matrix

$$A_i = \begin{bmatrix} E_i & 0 & \ldots & \ldots & 0 \\ F_i & E_i & \ddots & \ddots & \vdots \\ 0 & F_i & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & F_i & E_i \end{bmatrix},$$

by setting $E_i = \bar{A}_i^u + (1-p)\bar{A}_i^m$ and $F_i = p\bar{A}_i^m$. The key in finding the steady state probability vector $\pi = (\pi_0, \pi_1, \ldots)$ of P, where π_0 and π_i, for $i > 0$, have the same dimension as B_1 and A_1, respectively, is to solve the following nonlinear equation (if π exists):

$$R = A_0 + RA_1 + R^2 A_2, \tag{2}$$

as P is a special case of a GI/M/1 type MC [14]. Due to the structure of the matrices A_0, A_1 and A_2 the smallest nonnegative solution R of the above equation is a block triangular block Toeplitz (btbT) matrix. A btbT matrix X is characterized by its first block column as follows:

$$X = \begin{bmatrix} X_1 & 0 & \ldots & \ldots & 0 \\ X_2 & X_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ X_{k-1} & \ddots & \ddots & X_1 & 0 \\ X_k & X_{k-1} & \ldots & X_2 & X_1 \end{bmatrix}.$$

Observe that the matrices B_0, B_1 and B_2 of the MC characterized by $P_{k,n}$ are also btbT. An efficient algorithm to compute R is presented in Section 5.1.

The product between two btbT matrices as well as the inverse of a btbT matrix is again btbT. Hence, having found R, the matrix $R_b = B_0(I - A_1 - RA_2)^{-1}$ is btbT, since B_0 is (notice, the blocks of the btbT matrix R_b are not square). The steady state probability vector $\pi = (\pi_0, \pi_1, \ldots)$ of the GI/M/1-type matrix P is then found as [14]:

$$\begin{aligned} \pi_0 &= \pi_0 \left\{(B_1 + R_b B_2) + \right. \\ &\quad \left. (C_0 + R_b(I - R)^{-1} C_1)\right\}, \qquad (3) \\ \pi_1 &= \pi_0 R_b, \qquad (4) \\ \pi_s &= \pi_{s-1} R, \qquad (5) \end{aligned}$$

for $s > 1$, while π_0 and π_1 are normalized as $\pi_0 e + \pi_1 (I - R)^{-1} e = 1$. Thus, π exists if all of the following three criteria are met:

1. R has a spectral radius $sp(R) < 1$, i.e., $(I - R)^{-1} = \sum_{i=0}^{\infty} R^i$ exists. A btbT matrix is invertible if its block on the main diagonal is, therefore, it suffices to show that $sp(R_1) < 1$ (see Eqn. (6) for its definition). The matrix $E = E_0 + E_1 + E_2 = \bar{A} - p\bar{A}^m$ is irreducible, for $p < 1$, as $\bar{A}$ is. By applying Corollary 1.3.1 in [14], we find that the requirement $sp(R_1) < 1$ is always met as E is irreducible, but not stochastic (for $p < 1$).

2. The spectral radius of $A_1 + RA_2$ has to be less than one, for R_b to exist. This condition can be simplified to $sp(E_1 + R_1 E_2) < 1$ as $A_1 + RA_2$ is a btbT matrix.

3. Eqn. (3) has a strict positive solution, except for those entries of π_0 corresponding to possible transient states. Denote $BC[R] = (B_1 + R_b B_2) + (C_0 + R_b(I - R)^{-1} C_1)$. By direct calculation, using Eqn. (2), the matrix $BC[R]$ can be shown to be stochastic, i.e., $BC[R]e = e$. Hence, the only true requirement is that $BC[R]$ is not reducible.

Conditions 2 and 3 are not very demanding and should be valid for all practical purposes (as was the case for the nu-

merical examples in Section 6, as well as those reported in [23]).

5.1 Computing the rate matrix R

Let $R_1, \ldots, R_k$ denote the k blocks characterizing the btbT matrix R. Exploiting the structural properties of Eqn. (2) one finds that the R_s matrices obey the following set of relations:

$$R_1 = E_0 + R_1 E_1 + R_1^2 E_2, \quad (6)$$
$$K_{s,4} = K_{s,1} R_s K_{s,2} + R_s K_{s,3}. \quad (7)$$

where, for $s = 2, \ldots, k$,

$$K_{s,1} = R_1, \; K_{s,2} = E_2, \; K_{s,3} = E_1 + R_1 E_2 - I,$$
$$-K_{s,4} = R_{s-1} F_1 + \left(\sum_{j=2}^{s-1} R_j R_{s-j+1} \right) E_2 + \left(\sum_{j=1}^{s-1} R_j R_{s-j} \right) F_2 + 1_{\{k=2\}} F_0,$$

where 1_A equals 1 if A is true and 0 otherwise. Notice, each of the four matrices $K_{s,j}$, $j = 1, \ldots, 4$, is only a function of the E_i, F_i and $R_1, \ldots, R_{s-1}$ matrices. As such, we can compute the first block column of R, which fully determines R, by solving one quadratic equation (Eqn. (6)) and $k-1$ Sylvester matrix equations (Eqn. (7) starting with $s = 2$ to k). We propose to use the Cyclic Reduction (CR) algorithm [13] to solve the former and refer to Appendix A where a Hessenberg algorithm (HA) is given for the latter. Given the computational complexity of the CR and HA algorithm, a time and memory complexity of $O(d^3 k^2)$ and $O(d^2 k)$, respectively, is achieved to compute R (where d is the dimension of the R_s blocks).

Remark 1 In the special case where $E_0 = 0$, e.g., setting $k = n$ when computing the waiting time of the n-th customer in a D-MAP/PH/1 queue in an exact manner, the expressions for R_s simplify to:

$$R_1 = 0,$$
$$R_s = K_{s,4} (I - E_1)^{-1},$$

for $s > 1$. $(I - E_1)$ is invertible as E_1 is a substochastic matrix (due to Hadamard's theorem). Meaning R can be computed using simple matrix products and sums, without the need to solve any quadratic or Sylvester matrix equation. In this particular case $R^n = 0$, yielding $\pi_s = 0$ for $s > n$. For the D-MAP/PH/1 example mentioned above, the diagonal block of the btbT matrix R_b is also equal to zero, hence, $\pi_n = 0$ as well. Which is obvious as the n-th customer cannot find more than $n-1$ customers present in the queue at his arrival epoch.

Remark 2 In the special case where $E_2 = 0$, which occurs when we set $k = n$ when computing the system state at the n-th departure epoch of a D-MAP/PH/1 queue in an exact manner, the expressions for R_s simplify to:

$$R_1 = E_0 (I - E_1)^{-1},$$
$$R_s = K_{s,4} (I - E_1)^{-1},$$

for $s > 1$. Hence, as in Remark 1, the matrix R can be computed using simple matrix products and sums. Moreover, having either $R_1 = 0$ or $E_2 = 0$ suffices to meet the 2-nd condition to guarantee the existence of π as $sp(E_1) < 1$.

5.2 Solving the boundary condition

In order to efficiently compute π_0 we can take advantage of the specific structure of $BC[R]$. As B_1 and B_2 are btbT, so is $(B_1 + R_b B_2)$, while C_0 and C_1 have all their entries equal to zero, except for their last block column, which implies that only the last block column of $(C_0 + R_b (I - R)^{-1} C_1)$ is nonzero. When the nonzero entries of a matrix Z are all positioned in the last block column, we refer to Z as an lbC matrix. Thus, $BC[R]$ is the sum of a btbT and an lbC matrix and this sum is the transition matrix P_0 of the MC, characterized by P, when censored on the states corresponding to π_0. Let $\pi_0 = (\pi_{0,1}, \ldots, \pi_{0,k})$, where $\pi_{0,i}$, for $i = 1, \ldots, k$, is a $1 \times l$ vector, and denote the j-th state corresponding to $\pi_{0,i}$ as $\langle i, j \rangle$. Next, define G_i, for $i = 2, \ldots, k$, as an $l \times l$ matrix whose (j, j')-th entry equals the expected number of visits to state $\langle k - i + 1, j' \rangle$ starting in state $\langle k, j \rangle$ until the first return to some state $\langle k, s \rangle$, for $1 \leq s \leq l$. These matrices allow us to express the components of π_0 as a function of $\pi_{0,k}$ as follows:

$$\pi_{0,k-i+1} = \pi_{0,k} G_i,$$

for $i = 2, \ldots, k$. Let $Y_1, \ldots, Y_k$, resp. $Z_1, \ldots, Z_k$, be the blocks characterizing the btbT matrix $(B_1 + R_b B_2)$ and the lbC matrix $(C_0 + R_b (I - R)^{-1} C_1)$. Denote $\mathcal{S}_i = \{\langle i, j \rangle \, | j = 1, \ldots, l\}$ as the set of states corresponding to $\pi_{0,i}$ for $i = 1, \ldots, k$. Let $P_{0;i}$ be the transition matrix when censoring P_0 on $\left(\cup_{j=1}^{k-i+1} \mathcal{S}_j\right) \cup \mathcal{S}_k$, for $i = 2, \ldots, k+1$ (see Eqn. (8)). Given the stochastic interpretation of G_i, for $i = 2, \ldots, k$, we have:

$$G_i = V_{k-i+2,i} (I - Y_1)^{-1}$$

and $\pi_{0,k}$ is the solution of $\pi_{0,k} = \pi_{0,k} P_{0;k+1}$, where $P_{0;k+1} = W_1$. Expressions for the $V_{k-i+2,j}$, for $i = 2, \ldots, k$ and $j \geq i$, and W_{k-i+2}, for $i = 2, \ldots, k+1$,

$$P_{0;i} = \begin{bmatrix} Y_1 & 0 & 0 & \dots & 0 & Z_1 \\ Y_2 & Y_1 & 0 & \ddots & 0 & Z_2 \\ Y_3 & Y_2 & Y_1 & \ddots & 0 & Z_3 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ Y_{k-i+1} & Y_{k-i} & Y_{k-i-1} & \ddots & Y_1 & Z_{k-i+1} \\ V_{k-i+2,k} & V_{k-i+2,k-1} & V_{k-i+2,k-2} & \dots & V_{k-i+2,i} & W_{k-i+2} \end{bmatrix} \tag{8}$$

matrices can be found via the following recursive relations:

$$\begin{aligned} V_{k,j} &= Y_j, \\ V_{k-i+2,j} &= V_{k-(i-1)+2,j} + V_{k-(i-1)+2,i-1} \\ &\quad (I - Y_1)^{-1} Y_{j-(i-1)+1}, \\ W_k &= Y_1 + Z_k, \\ W_{k-i+2} &= W_{k-(i-1)+2} + V_{k-(i-1)+2,i-1} \\ &\quad (I - Y_1)^{-1} Z_{k-(i-1)+1}, \end{aligned}$$

for $i > 2$ and $j \geq i$. Therefore, the following algorithm, the time and memory complexity of which equal $O(l^3k^2)$ and $O(l^2k)$ respectively, can now be devised to compute $\pi_{0,k}$ and the matrices G_i:

1. Let $G_1 = Y_1 + Z_k$ and set $G_i = Y_i$ for $i = 2, \dots, k$,

2. for $i = 3$ to $k+1$ do
   $G_{i-1} = G_{i-1}(I - Y_1)^{-1}$
   for $j = i$ to k do
   $G_j = G_j + G_{i-1} Y_{j-(i-1)+1}$
   end
   $G_1 = G_1 + G_{i-1} Z_{k-(i-1)+1}$
   end

3. $\pi_{0,k}$ is the solution of $\pi_{0,k} = \pi_{0,k} G_1$ and $\pi_{0,k} e = 1$.

Observe that, after the $(i-2)$-th iteration in step 2, we have $G_1 = W_{k-i+2}$ and $G_j = V_{k-i+2,j}$, for $j \geq i$, while the computation of G_j is completed for $j = 2, \dots, i-1$.

5.3 Computing π from π_0 and R

By combining both algorithms, presented in Sections 5.1 and 5.2, with Eqn. (4) we can calculate π_0, R and π_1 in a $O(\max(d,l)^3k^2)$ time and a $O(\max(d,l)^2k)$ memory complexity. Using Eqn. (5), the next s components $\pi_2, \dots, \pi_{s+1}$ of π can be obtained in $O(d^2k^2s)$ time, using $O(d^2k + dks)$ memory. Moreover, as indicated below, the time complexity can be further reduced to $O((d^2N \log N) + (dN \log N + d^2N)s)$, where $N = 2^{\lceil \log_2(2k-1) \rceil}$, by making use of fast Fourier transforms (FFTs).

Fast vector and block triangular block Toeplitz multiplication In this section we present an FFT based algorithm to compute the product

$$(y_1, y_2, \dots, y_k) = (x_1, x_2, \dots, x_k) \begin{bmatrix} R_1 & 0 & \dots & 0 \\ R_2 & R_1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ R_k & R_{k-1} & \dots & R_1 \end{bmatrix},$$

where x_s and y_s are both $1 \times d$ vectors and and R_s a $d \times d$ matrix, for $1 \leq s \leq k$. The idea behind this algorithm is similar to [12, Chapter 2] where an FFT algorithm was introduced to multiply a block vector with a block Toeplitz matrix. Define $R(z) = \sum_{s=0}^{k-1} R_{s+1} z^s$, $x(z) = \sum_{s=0}^{k-1} x_{k-s} z^s$ and $\hat{y}(z) = \sum_{s=0}^{2k-2} \hat{y}_{s+1} z^s = x(z)R(z)$. That is, in matrix form

$$(\hat{y}_1, \dots, \hat{y}_{2k-1}) = (x_k, x_{k-1}, \dots, x_1) \begin{bmatrix} R_1 & R_2 & \dots & R_k & 0 & \dots & 0 \\ 0 & R_1 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & R_1 & R_2 & \dots & R_k \end{bmatrix}.$$

This shows that $y_s = \hat{y}_{k-s+1}$ for $s = 1, \dots, k$. In summary, an FFT based algorithm to compute $y = (y_1, \dots, y_k)$ proceeds as follows:

1. Evaluate $R(z)$ at the N-th roots of unity, ω_N^s, for $s = 0, \dots, N-1$ and $N \geq 2k-1$. In practice we choose $N = 2^{\lceil \log_2(2k-1) \rceil}$. This step requires d^2 discrete Fourier transforms (DFTs) of order N.

2. Evaluate $x(z)$ at the N-th roots of unity, ω_N^s, for $s = 0, \dots, N$ and $N \geq 2k-1$, using d DFTs of order N.

3. Compute the N products $\hat{y}(\omega_N^s) = x(\omega_N^s)R(\omega_N^s)$, for $s = 0, \dots, N-1$, resulting in a cost of $O(Nd^2)$.

4. Perform d inverse DFTs of order N to find the vectors $\hat{y}_s$, for $s = 0, \dots, 2k-2$. Set $y_s = \hat{y}_{k-s+1}$ for $s = 1, \dots, k$.

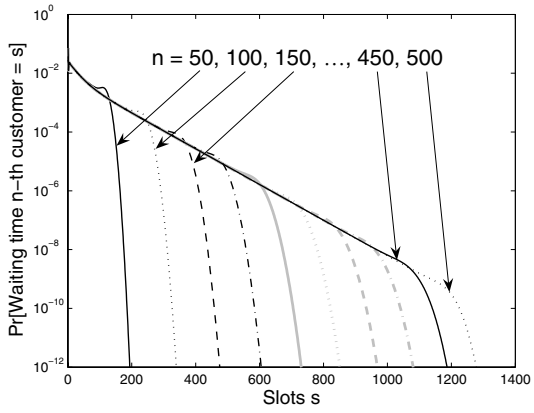

Figure 1. Dimensioning a video playout buffer

n	CPU time rate matrix R	Total CPU time
50	0.375	2.844
100	1.328	10.125
150	2.906	26.344
200	4.984	38.047
250	7.469	60.453
300	11.141	150.547
350	14.672	188.828
400	18.813	215.682
450	23.859	256.688
500	29.125	306.094

Table 1. A video playout buffer, Computation times in seconds

The total costs of this algorithm are dominated by step 1 and equal $O(d^2 N \log N)$, while the costs of the other steps are $O(dN \log N + d^2 N)$. It is worthwhile noting that when computing the components of π, we always multiply with the same matrix R. Hence, step 1 should only be performed once. This leads to a $O((d^2 N \log N)+(dN \log N+d^2 N)s)$ time complexity to compute the next s components of π.

6 Applications

Two sets of numerical examples are included: the first, amongst others, reproduces the results presented in [8] and is mainly used to demonstrate the computational strength of our framework, the second investigates the transient behavior of a wireless random access algorithm (FS-ALOHA [5]) under overload conditions.

6.1 Dimensioning a video playout buffer

In [8] it was argued that a video playout buffer can be dimensioned via the transient analysis of a D-MAP/PH/1 queue. More specifically, by considering the 10^{-9}-quantile of the sojourn time distribution of its n-th customer, the initial delay Δ that the player has to wait before starting the video playout, can be estimated. For the specifics of the 4-state D-MAP arrival process, as well as the 10-phase PH distribution, we refer to [8]. As explained in Section 3, this results in block matrices $\bar{A}_s$ of size 40×40. In order to produce exact results we have set $k = n$, for $n = 50, 100, \ldots, 500$ (see Figure 1). Notice, in [8] no results were reported for n above 142, as the inversion of the probability generating function already took many hours for this particular example.

Table 1 holds the CPU time (in seconds) required to compute the exact waiting time distribution of the n-th customer on a PC with a 2Ghz Intel Pentium processor and 512 MB RAM. The peak memory usage of the MATLAB session as reported by the windows task manager was 194 MB and the timing was performed by the MATLAB profiler. Notice, the R, A_s, etc. matrices involved are square with a dimension $d = 40n$ (however, during our computations we only need to store their first 40 columns). The CPU times can be further reduced by making use of an FFT algorithm to calculate the inverse of the btbT matrices $(I - R)$ and $(I - A_1 - RA_2)$ [13, Chapter2]. In the current MATLAB implementation, about 30% of the utilized CPU time was spent on inverting these matrices. The total CPU time contains a mild step behavior due to the FFT component. Finally, part of the increase in computational costs for n large is contributed by the need to compute more components π_s of π (as they decrease less rapidly).

6.2 FS-ALOHA: a wireless random access algorithm

For several decades, one of the key properties of a random access algorithm (RAA) has been its maximum stable throughput (MST). This is the maximum input rate for which the RAA can guarantee that each packet has a finite delay (with probability 1). However, even if the offered input traffic rate is above the MST, the channel might operate smoothly for some time (e.g., slotted ALOHA). Therefore, it is useful to gain insight on the initial (transient) behavior of the channel throughput.

In this section we demonstrate that the transient channel throughput of Fifo-by-Sets ALOHA (FS-ALOHA) can be determined using our framework. FS-ALOHA is an RAA that maintains the simplicity of ALOHA, while significantly improving its performance. It was specifically designed to operate in a wireless LAN environment where the uplink channel—that is, from the end-user to the network—is al-

located in a dynamic way (through a contention/reservation channel and a piggybacking mechanism). The uplink channel is partitioned in fixed length frames, where a fixed part (being T contention slots) of each frame is dedicated to the contention channel. The contention channel is used by a mobile station to report its bandwidth requirements via a request message, unless it has an ongoing data exchange with the network access point allowing the mobile node to piggyback the request to the end of a data transmission. For the results presented in Figure 2, we have set the protocol parameters S and N equal to 5. These two parameters play the following role. The T contention slots in each frame are divided into two disjoint sets of S and N slots such that $T = S+N$. The operation of FS-ALOHA is as follows: (A) Newly arrived requests are transmitted, for the first time, by randomly choosing one out of the S slots; this is the first set of S slots after the request was generated. If some of the transmissions taking place in the S slots of a frame are unsuccessful, because multiple MSs transmitted in the same slot, the unsuccessful requests are grouped into a Transmission Set (TS), which joins the back of the queue of TSs waiting to be served. (B) The other N slots are used to serve the queue of backlogged TSs on a FIFO basis. Backlogged TSs are served, one at a time, using slotted ALOHA, that is, all the requests part of the TS select one out of the N slots and are transmitted in this slot. The requests that were transmitted successfully leave the TS, the others retransmit in the N slots of the next frame using the same procedure. The service of a TS lasts until all the requests part of the TS have been successfully transmitted, in which case the service of the next TS, if there is another TS in the queue, starts service in the N slots of the next frame. If there are no queued TS, all $T = S + N$ slots are used for newly arriving customers.

A more detailed description of the operation and properties of FS-ALOHA would lead us to far astray and can be found in [5, 21]. In Figure 2 we consider an arrival rate of $\lambda = 3, 3.4, 3.7, 5$ and 8 request packets per frame, leading to an input rate of $\lambda/10$ requests/slot on the contention channel. The MST for FS-ALOHA (with $S = N = 5$) equals 0.348744 (see [5]). Thus, in three of the five cases ($\lambda = 3.7, 5$ and 8) the system is unstable and the number of backlogged stations grows to infinity as time evolves, while in the remaining two scenarios a steady state is reached.

In [5], it is shown that the behavior of FS-ALOHA can be captured by a QBD MC, the f-th time epoch of which corresponds to the f-th uplink frame. Hence, if we mark each time epoch of this QBD MC (by setting $A_s^m = A_s$ and $B_s^m = B_s$, for $s = 0, 1$ and 2), we can approximate the transient throughput during frame $f = 10, \ldots, 1000$, by assessing the system state at the $n = f + 1$-th marked time epoch. In this case setting $k = n$ would result in a periodic MC, as such the best approximation we can get is by letting

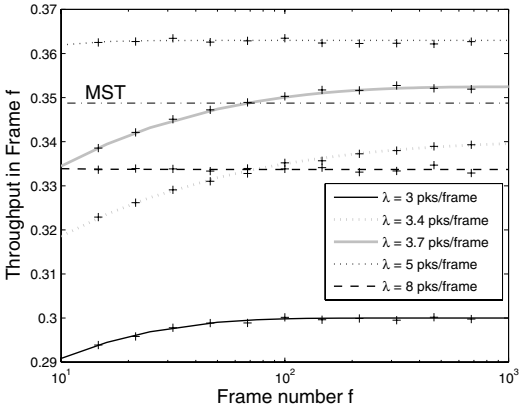

Figure 2. Transient throughput of FS-ALOHA

$k = n - 1 = f$. For $n > 200$, we have limited k to 200, as this suffices to get accurate results, i.e., $n = \min(f, 200)$. A simulation program confirms the high degree of accuracy that these approximated results have (that is, the '+' data in Figure 2). The simulated results were averaged over 10^5 runs. Another way to get an idea on the accuracy of the results, due to the choice of k, is to compare the results with a different, larger choice for k. For instance, having $k = \min(f, 300)$ only changes the transient throughput by $9.2\ 10^{-6}$ (for $n = 1000$ and $\lambda = 3.7$). Finally, exact transient throughput figures can also be gathered using a brute-force recursive technique.

We also computed the throughput in frame $f = 10^5$, with $k = 200$, and found a match up to the 14-th digit between the transient throughput in frame f and the limit of this throughput for t going to infinity. This limit equals λ if the system is stable and can be computed from the protocol parameters λ, S and N otherwise. It is worthwhile noting that the transient throughput of FS-ALOHA can be computed from π_0 and $\sum_{s=1}^{\infty} \pi_s = \pi_1 (I - R)^{-1}$, thus, there is no need to compute the components π_s separately. This is especially useful when studying the transient throughput in an overload scenario with f large (e.g., 10^5).

Figure 2 shows that during the first couple of frames the mean number of backlogged stations grows as the throughput is below the input rate. However, for $\lambda = 3$ and 3.4 a steady state is reached as the transient throughput quickly converges to λ. The rate of convergence decreases as the input rate is closer to MST, which is obvious as the mean number of backlogged stations in steady state is larger in such cases. For the unstable scenarios, we observe that the number of backlogged stations starts to build from the first few frames and continues to do so indefinitely. Thus, as opposed the slotted ALOHA [6] there is no initial period during which the overloaded system operates smoothly (the MST of slotted ALOHA equals zero, as opposed to what

the flawed analysis of Abramson shows [1]).

For the sake of completeness, we briefly indicate how to solve a Sylvester matrix equation of the form $AXB + CX = D$ (Equation (7) can be written in this form by taking the transposed). The Hessenberg algorithm presented below can be found in [7], as well as a (computationally more expensive) Schur algorithm. We start with a Hessenberg-triangular decomposition of (A, C):

$$WAV = L_{bar}, \quad WCV = N_{bar},$$

where W and V are unitary (meaning $WW^* = VV^* = I$, with Z^* denoting the complex conjugate of a matrix Z), L_{bar} a Hessenberg and N_{bar} a triangular matrix. Let $U^*BU = T$ be a complex Schur decomposition of B, with U unitary and T a triangular matrix. Then, premultiplying and postmultiplying $AXB + CX = D$ by W and U, transforms the system to $L_{bar}YT + N_{bar}Y = F$, where $Y = V^*XU$ and $F = WDU$. Denote z_i and z_{ij} as the i-th column and (i, j)-th entry of a matrix Z, respectively. Equating the i-th column in $L_{bar}YT + N_{bar}Y = F$ leads to

$$[N_{bar} + t_{ii}L_{bar}]\, y_i = f_i - \sum_{j=1}^{i-1} t_{ji} L_{bar} y_j.$$

Thus, to compute X we need to solve d linear Hessenberg systems (where d denotes the dimension of the square matrices A, B, C and D). As such the Sylvester matrix equation $AXB + CX = D$ can be solved in time $O(d^3)$. Each of the decomposition algorithms required is a Matlab build-in function (i.e., hess and schur). Moreover, the '\' Matlab command to solve the d linear systems automatically recognizes the Hessenberg form, thereby solving each system in $O(d^2)$ time.

Acknowledgment

B. Van Houdt is a post-doctoral fellow of the FWO-Flanders.

References

[1] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall Int., Inc., 1992.

[2] C. Blondia. A discrete-time batch markovian arrival process as B-ISDN traffic model. *Belgian Journal of Operations Research, Statistics and Computer Science*, 32(3,4):3–23, 1993.

[3] R. Carmo, E. de Souza e Silva, and R. Marie. Efficient solutions for an approximation technique for the transient analysis of markovian models. Technical report, IRISA Publication Interne N 1067, 1996.

[4] G. Choudhury, D. Lucantoni, and W. Whitt. Multidimensional transform inversion with applications to the transient M/G/1 queue. *Annals of Applied Probability*, 4:719–740, 1994.

[5] D. V. Cortizo, J. García, C. Blondia, and B. Van Houdt. FIFO by sets ALOHA (FS-ALOHA): a collision resolution algorithm for the contention channel in wireless ATM systems. *Performance Evaluation*, 36-37:401–427, 1999.

[6] P. Flajolet. Evaluation de protocols de communication: aspects mathematics. Technical Report 797, INRIA, 1988.

[7] N. Higham and H. Kim. Solving a quadratic matrix equation by Newton's method with exact line search. *SIAM J. Matrix Anal. Appl.*, 23:303–316, 2001.

[8] T. Hofkens, K. Spaey, and C. Blondia. Transient analysis of the D-BMAP/G/1 queue with an application to the dimensioning of a playout buffer for VBR traffic. In *Proc. of Networking 2004*, Athens, Greece, 2004.

[9] A. Lang and J. L. Arthur. Parameter approximation for phase-type distributions. In *Matrix-Analytic Methods in Stochastic Models, (S. R. Chakravarthy and A. S. Alfa (Editors))*, pages 151–206, New York, 1996. Marcel-Dekker, Inc.

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

[11] D. Lucantoni, G. Choudhury, and W. Whitt. The transient BMAP/PH/1 queue. *Stochastic Models*, 10:461–478, 1994.

[12] B. Meini. *Fast algorithms for the numerical solution of structured Markov chains*. PhD thesis, University of Pisa, 1998.

[13] B. Meini. Solving QBD problems: the cyclic reduction algorithm versus the invariant subspace method. *Advances in Performance Analysis*, 1:215–225, 1998.

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

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

[16] L. L. Ny and B. Sericola. Transient analysis of the BMAP/PH/1 queue. *I.J. of Simulation*, 3(3-4):4–14, 2003.

[17] 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.

[18] A. Remke, B. Haverkort, and L. Cloth. Model checking infinite-state Markov chains. In *TACAS 2005, LNCS 3440, Springer*, pages 237–252, Feb 2005.

[19] S. Ross. Approximating transient probabilities and mean occupation times in continues-time markov chains. *Probability in the Engineering and Informational Sciences*, 1:251–264, 1987.

[20] M. Telek. Minimal coefficient of variation of discrete phase type distributions. In *3rd International Conference on Matrix Analytic Methods in Stochastic Models*, pages 391–400, Leuven, Belgium, 2000. Notable Publications Inc.

[21] B. Van Houdt and C. Blondia. Robustness properties of FS-ALOHA(++): a contention resolution algorithm for dynamic bandwidth allocation. *Mobile Networks and Applications (MONET), Special Issue on Performance Evaluation of QoS Architectures in Mobile Networks*, 8(3):237–253, 2003.

[22] B. Van Houdt and C. Blondia. The waiting time distribution of a type k customer in a MMAP[K]/PH[K]/c (c=1,2) queue using QBDs. *Stochastic Models*, 20(1):55–69, 2004.

[23] B. Van Houdt and C. Blondia. Approximated transient queue length and waiting time distributions via steady state analysis. *Stochastic Models*, 21(2-3):725–744, 2005.

[24] J. Zhang and E. Coyle. Transient analysis of quasi-birth-death processes. *Stochastic Models*, 5(3):459–496, 1989.