Transient analysis of tree-like processes and its application to random access systems

J. Van Velthoven, B. Van Houdt* and C. Blondia
University of Antwerp
Middelheimlaan 1
B-2020 Antwerp, Belgium
email: {jeroen.vanvelthoven,benny.vanhoudt,chris.blondia}@ua.ac.be

ABSTRACT

A new methodology to assess transient performance measures of tree-like processes is proposed by introducing the concept of tree-like processes with marked time epochs. As opposed to the standard tree-like process, such a process marks part of the time epochs by following a set of Markovian rules. Our interest lies in obtaining the system state at the n-th marked time epoch as well as the mean time at which this n-th marking occurs. The methodology transforms the transient problem into a stationary one by applying a discrete Erlangization and constructing a reset Markov chain. A fast algorithm, with limited memory usage, that exploits the block structure of the reset Markov chain is developed and is based, among others, on Sylvester matrix equations and fast Fourier transforms. The theory of tree-like processes generalizes the well-known paradigm of Quasi-Birth-Death Markov chains and has various applications. We demonstrate our approach on the celebrated Capetanakis-Tsybakov-Mikhailov (CTM) random access protocol yielding new insights on its initial behavior both in normal and overload conditions.

1. INTRODUCTION

Over the last three decades, broad classes of frequently encountered queueing models have been analyzed by *matrix-analytic methods* [22, 23, 18]. 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 [5]. There is also an active search for accurate matching algorithms for both PH distributions and MAP arrivals when modeling communication systems, e.g., [2, 7, 15].

One of the more recent paradigms within the field of matrix-analytic methods has been its generalization to discrete-time bivariate Markov chains $\{(X_k, N_k), k \geq 0\}$, in which the values of X_k are the nodes of a $(d+1)$-ary tree (and N_k takes values between 1 and h). By limiting the type of transitions that can occur in such a Markov chain, tree structured Markov chains of the M/G/1- [27], GI/M/1- [36] and QBD-type [35] have been developed. 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. Setting $d = 0$ reduces these Markov chains to the standard M/G/1-, GI/M/1- and QBD-type chains.

The set of tree-like processes [4] was originally introduced as a subclass of the tree structured QBD Markov chains. Meanwhile, it has been shown that any tree structured QBD can be reduced to a tree-like process [30]. Typical applications of tree-like processes include single server queues with a LIFO service discipline [36, 27, 14, 13]. In [29, 34, 31] tree-like processes were used to analyze a set of well-known random access algorithms, called tree or splitting algorithms. Some recent work also indicates that tree-like processes can be used to study priority queues [32]. Various numerical methods exist to examine the stability of a tree-like process and to determine its stationary behavior.

In this paper we focus on the transient behavior of tree-like processes. Our contribution is both of theoretical and practical nature. First, we introduce a new framework called *tree-like processes with marked time epochs* to assess transient performance measures in a unified manner. Such a process distinguishes itself from the common tree-like process in the sense that as it evolves it also marks part of the time epochs. Any transient problem that can be linked to the n-th marking, for some n, of some tree-like process with marked time epochs, can be dealt with using our methodology. Second, to analyze this new process (that is, to generate numerical results) we developed an efficient algorithm that uses a *discrete Erlangization* to approximate the system state at the n-th marking. The main building block of this algorithm is a so-called *reset* Markov chain. Such a Markov chain allows us to reformulate the transient problem at hand into a stationary problem of an expanded Markov chain. When performing this stationary analysis we further exploit the

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

block triangular block Toeplitz (btbT) structure of the matrices involved.

We also demonstrate our approach by analyzing the transient behavior of the celebrated binary tree algorithm by Capetanakis, Tsybakov and Mikhailov (CTM) [9, 28], for random access systems. The key characteristic of the CTM algorithm is its underlying tree (or stack) structure. Tree algorithms have received considerable attention over the last three decades and are still analyzed within the context of various networking areas. The use of tree algorithms in cable-TV networks (i.e., Hybrid-Fiber Coaxial networks) has been proposed by various authors including [17, 16, 8], where the focus was either on the frame structure, large round-trip times or the finite population. A lot of this work was motivated by the activities of the IEEE 802.14 working group on tree algorithms [12]. Tree algorithms as a means to solve collision resolution problems have been addressed in the context of Ad-Hoc networks [26], where the residual battery energy affects the splitting process, and W-CDMA systems [1] with a priority random-access protocol. A unified probabilistic treatment to assess the asymptotic behavior of tree algorithms has also appeared more recently [24, 20], avoiding the need to resort to complex analysis techniques.

By considering the transient regime of tree algorithms we can get an understanding of the initial protocol dynamics even when the system is overloaded (that is, the overall input rate λ is above the maximum stable throughput). To the best of our knowledge, it is the first time that analytical results of this type are provided for the CTM protocol. The link between the CTM algorithm and tree-like processes necessary to apply our new methodology was established in [29]. The transient behavior of other protocol variations can be investigated in a similar manner by relying on the results presented in [34].

Transient measures for discrete time Markov chains are often obtained through an iterative or recursive approach, e.g., one starts with the initial probability vector and subsequently multiplies this vector with the transition matrix P (while exploiting its structure). Even if the event of interest occurs rather early in time, such an approach causes extra difficulties for tree-like processes as the number of states that can be visited by the chain at time t grows exponentially fast. Problems like this are circumvented by our approach as we perform a single steady state analysis of an expanded Markov chain.

The paper is structured as follows. We start by giving some background information on tree-like processes, including their definition and some of the key equations related to their stationary behavior (Section 2). This is followed by the introduction of our new framework, called tree-like processes with marked time epochs (Section 3.1). Afterward, a discrete Erlangization technique is applied to construct the reset Markov chain that transforms the transient problem in a stationary one (Section 3.2). A key role in obtaining the reset MC's stationary behavior is played by the V matrix, for which several algorithms are presented (Section 3.3). Next, we demonstrate how to derive the transient measure of interest from the matrix V (Section 3.4). We apply our framework to assess a number of transient performance measures associated with the well-known CTM protocol (Section 4).

The work presented in this paper is in the same spirit as [33], which was limited to the special case where $d = 0$, i.e., the standard set of QBD processes.

2. SOME BACKGROUND ON TREE-LIKE PROCESSES

The set of tree-like processes [4] was first introduced as a subclass of the set of tree structured Quasi-Birth-Death Markov chains and afterward shown to be equivalent to them [30]. This section provides some background information on this type of discrete time Markov chains (MCs). Consider a discrete time bivariate MC $\{(X_t, N_t), t \geq 0\}$ in which the values of X_t are represented by nodes of a $(d+1)$-ary tree, for $d \geq 0$, and where N_t takes integer values between 1 and h. We will refer to X_t as the *node* and to N_t as the *auxiliary* variable of the MC at time t. With some abuse of notation, we shall refer to this MC as (X_t, N_t). The root node of the $(d+1)$-ary tree is denoted as $\emptyset$ and the remaining nodes are denoted as strings of integers, where each integer takes a value between 0 and d. For instance, the k-th child of the root node is represented by k, the l-th child of the node k by kl, and so on. Throughout this paper, we use the '+' to denote the concatenation on the right and '-' to represent the deletion from the right. For example, if $J = k_1 k_2 \dots k_n$, then $J + k = k_1 k_2 \dots k_n k$. Let $f(J,k)$, for $J \neq \emptyset$, denote the k rightmost elements of the sting J, then $J - f(J,1)$ represents the parent node of J.

The following restrictions need to apply for an MC (X_t, N_t) to be a tree-like process. 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 $0 \leq s \leq d$). Moreover, the chain's state at time $t+1$ is determined as follows:

$$P[(X_{t+1}, N_{t+1}) = (J', j) | (X_t, N_t) = (J, i)] = \begin{cases} \bar{f}^{i,j} & J' = J = \emptyset, \\ \bar{b}^{i,j} & J' = J \neq \emptyset, \\ \bar{d}_k^{i,j} & J \neq \emptyset, f(J,1) = k, J' = J - f(J,1), \\ \bar{u}_s^{i,j} & J' = J + s, s = 0, \dots, d, \\ 0 & otherwise \end{cases}$$

Notice, the transition probability between two nodes depends only on the spacial relationship between the two nodes and not on their specific values. We can now define the $h \times h$ matrices $\bar{D}_k$, $\bar{B}$, $\bar{F}$ and $\bar{U}_s$ with respective $(i,j)^{th}$ elements given by $\bar{d}_k^{i,j}$, $\bar{b}^{i,j}$, $\bar{f}^{i,j}$ and $\bar{u}_s^{i,j}$. This completes the description of the tree-like process. Notice, a tree-like process is fully characterized by the matrices $\bar{D}_k$, $\bar{B}$, $\bar{U}_s$ and $\bar{F}$.

Next we introduce a number of matrices that play a crucial role when studying the stability and stationary behavior of a tree-like process. The fundamental period of a tree-like process starting in state $(J+k, i)$ is defined as the first passage time from the state $(J+k, i)$ to one of the states (J, j), for $j = 1, \dots, h$. Let G_k, for $0 \leq k \leq d$, denote the matrix whose $(i,v)^{th}$ element is the probability that the MC is in state (J, v) at the end of a fundamental period which started in state $(J+k, i)$. Let the $(i,v)^{th}$ element of the matrix R_k denote the expected number of visits to state

$(J+k, v)$ before visiting node J again, given that $(X_0, N_0) = (J, i)$. Finally, let V denote the matrix whose $(i, v)^{th}$ element is the taboo probability that starting from state $(J + k, i)$, the process eventually returns to node $J + k$ by visiting $(J + k, v)$, under the taboo of the node J. Notice, due to the restrictions on the transition probabilities, the matrix V does not depend on k. Yeung and Alfa [35] were able to show that the following expressions hold for these matrices:

$$G_k = (I - V)^{-1} D_k, \tag{1}$$

$$R_k = U_k (I - V)^{-1}, \tag{2}$$

$$V = B + \sum_{s=0}^{d} U_s G_s. \tag{3}$$

Combining these equations, we have the following relation:

$$V = B + \sum_{s=0}^{d} U_s (I - V)^{-1} D_s. \tag{4}$$

Provided that the tree-like process $\{(X_t, N_t), t \geq 0\}$ is ergodic (which is the case if and only if the all the G_k matrices are stochastic or likewise if and only if the spectral radius of $R = R_0 + R_1 + \ldots + R_d$ is less than one), define

$$\begin{aligned} \bar{\pi}_i(J) &= \lim_{t \to \infty} P[X_t = J, N_t = i], \\ \bar{\pi}(J) &= (\bar{\pi}_1(J), \bar{\pi}_2(J), \ldots, \bar{\pi}_h(J)). \end{aligned}$$

The vectors $\bar{\pi}(J)$ can be computed from $\bar{\pi}(\emptyset)$ using the relation $\bar{\pi}(J + k) = \bar{\pi}(J) R_k$ and $\bar{\pi}(\emptyset)$ is found by solving the boundary condition $\bar{\pi}(\emptyset) = \bar{\pi}(\emptyset)(\sum_k R_k D_k + F)$ with the normalizing restriction that $\bar{\pi}(\emptyset)(I - R)^{-1} e = 1$ (where e is a column vector with all its entries equal to one).

3. TREE-LIKE PROCESS WITH MARKED TIME EPOCHS

3.1 Process Definition

In order to develop a novel framework to derive transient performance measures in a unified manner, we introduce the tree-like process *with marked time epochs*. As such a tree-like process evolves, each time epoch is either marked or not, where the marking rules typically depend on the performance measure of interest. In Section 3.3 and 3.4 we will present a number of algorithms that allow us to compute (i) the system state at the n-th marked time epoch and (ii) the average time epoch at which the n-th marking occurs.

Consider a MC $\{(X_t, N_t, M_t), t \geq 0\}$, where the values of X_t are nodes of a $(d + 1)$-ary tree, N_t takes values that range from 1 to h and $M_t = m$ or u. (X_t, N_t, M_t) is a tree-like process with marked time epochs if the transition probabilities are of the following form, for $y = m$ and u:

$$P[(X_{t+1}, N_{t+1}, M_{t+1}) = (J', j, y) | (X_t, N_t) = (J, i)] = \begin{cases} \bar{f}^{i,j,y} & J' = J = \emptyset, \\ \bar{b}^{i,j,y} & J' = J \neq \emptyset, \\ \bar{d}_k^{i,j,y} & J \neq \emptyset, f(J,1) = k, J' = J - f(J,1), \\ \bar{u}_s^{i,j,y} & J' = J + s, s = 0, \ldots, d, \\ 0 & otherwise \end{cases}$$

Notice, the value of M_{t+1} is affected by (X_t, N_t) and (X_{t+1}, N_{t+1}), but not by M_t (given (X_t, N_t)); hence, the value of M_0 is irrelevant. Next, define the $h \times h$ matrices $\bar{D}_k^y$, $\bar{B}^y$, $\bar{F}^y$ and $\bar{U}_s^y$, for $y = m$ and u, in the obvious manner. Furthermore, let

$$\begin{aligned} \bar{D}_k &= \bar{D}_k^m + \bar{D}_k^u, \\ \bar{B} &= \bar{B}^m + \bar{B}^u, \\ \bar{U}_s &= \bar{U}_s^m + \bar{U}_s^u, \\ \bar{F} &= \bar{F}^m + \bar{F}^u, \end{aligned}$$

for $0 \leq k, s \leq d$. Remark that $\{(X_t, N_t), t \geq 0\}$ is a tree-like process characterized by the matrices $\bar{D}_k$, $\bar{B}$, $\bar{F}$ and $\bar{U}_s$. We state that the t^{th} time epoch of this tree-like process is marked if and only if $M_{t+1} = m$. Therefore, the nonnegative matrix $\bar{D}_k^m$ contains the probabilities that $(X_t, N_t) = (J + k, i)$, while $(X_{t+1}, N_{t+1}) = (J, j)$ and time epoch t gets marked. $\bar{D}_k^u$ contains the probabilities for the same event, but without marking time t. A similar interpretation can be given for the other matrices. Whether a time epoch t is marked therefore depends on the transition from time t to time $t + 1$. We refer to the initial time epoch as time $t = 0$ and when performing the transient analysis we limit ourselves to the case where the tree-like process is in the root node at time $t = 0$, that is, the probability that we start in state $(\emptyset, i)$ is given by the i^{th} entry of the stochastic vector α_{ini}.

Apart from the introduction of this new framework, one of the main contributions in this paper lies in the development of efficient algorithms to compute the system state, either in an exact or approximated manner, of the MC (X_t, N_t) at the n-th marked time epoch. As a byproduct we will also derive the average time at which this n-th marking occurs, which is often an important performance measure on its own. For this purpose, we will combine the concepts of a discrete Erlangization and a reset Markov chain. The idea of applying an Erlangization to assess transient results dates back to Ross [25], who applied this technique to approximate the system state at time t of a finite continuous time Markov chain. Reset Markov chains were also used in the context of [33].

3.2 Discrete Erlangization and reset Markov chains

Denote the i^{th} entry of the $1 \times h$ vector $\pi_n^m(J)$ as the probability that the Markov chain $\{(X_t, N_t), t \geq 0\}$ is in state (J, i) at the n-th marked time epoch. The key property of a discrete Erlangization is that we can approximate the system state at the n-th marking $t^m(n)$ by considering the system state at the $Z_{l,n}$-th marked time epoch $t^m(Z_{l,n})$, where $Z_{l,n}$ is a negative binomially distributed (NBD) random variable with l phases and a mean n, for $l(\leq n)$ sufficiently large. Setting $l = n$ would result in exact results, however l cannot always be set to n as the reset MC might become periodic (see further on). Using the NBD as a reset time implies, among others, that the transition blocks of the reset MC presented below have a special structure that can be exploited when computing the MC's steady state probabilities.

To compute the system state at time $t^m(Z_{l,n})$, we make use of an expanded Markov chain, called a reset Markov chain. The key feature of such a Markov chain is that it reformulates the transient problem of computing $t^m(Z_{l,n})$, into a steady state analysis. Consider the stochastic process

that evolves according to $\{(X_t, N_t, M_t), t \geq 0\}$, but that is repeatedly reset when leaving the $Z_{l,n}$-th marked time epoch (meaning a transition occurs to state $(X_0 = \emptyset, i, m)$ with probability $(\alpha_{ini})_i$). Hence, if we perform a Bernoulli trial with parameter $p = l/n$ each time we have a transition out of a marked time epoch, the system is reset whenever l successes have occurred. We define the reset counter as being the number of pending successes before the next reset event. It is clear that this reset counter takes values in the range $\{1, 2, \ldots, l\}$. When we add this reset counter as an additional auxiliary variable to the tree-like process $\{(X_t, N_t), t \geq 0\}$, we obtain a Markov chain $\{(\mathcal{X}_t, \mathcal{N}_t), t \geq 0\}$, where $\mathcal{N}_t$ has the set $\{(i,j)|0 \leq i \leq l, 0 \leq j \leq h\}$ as its range. Remark, if we succeed in computing the stationary behavior of the expanded MC $(\mathcal{X}_t, \mathcal{N}_t)$, we can easily find the system state just prior to a reset event, which is exactly the system state at time $t^m(Z_{l,n})$. The transition probabilities of the reset MC can be written as

$$P[(\mathcal{X}_{t+1}, \mathcal{N}_{t+1}) = (J', (i', j'))|(\mathcal{X}_t, \mathcal{N}_t) = (J, (i,j))] = \begin{cases} f^{i,j,i',j'} & J' = J = \emptyset, \\ b^{i,j,i',j'} & J' = J \neq \emptyset, \\ d_k^{i,j,i',j'} & |J| > 1, f(J,1) = k, \\ & J' = J - f(J,1), \\ c_k^{i,j,i',j'} & |J| > 1, f(J,1) = k, J' = \emptyset, \\ d_k^{i,j,i',j'} + c_{1,k}^{i,j,i',j'} & J = k, J' = \emptyset, \\ u_s^{i,j,i',j'} & J' = J + s, s = 0, \ldots, d, \\ 0 & otherwise \end{cases}$$

where $|J|$ represents the length of the string J. Let $d_k^{i,j,i',j'}$, $c_k^{i,j,i',j'}$, $b^{i,j,i',j'}$, $f^{i,j,i',j'}$ and $u_s^{i,j,i',j'}$ be the $((i-1)h+j, (i'-1)h+j')^{th}$ entry of the $lh \times lh$ matrix D_k, C_k, B, F and U_s, respectively. One readily establishes that these matrices are given by:

$$\begin{aligned} D_k &= (I \otimes (\bar{D}_k^u + (1-p)\bar{D}_k^m)) + (S_0 \otimes p\bar{D}_k^m), \\ B &= (I \otimes (\bar{B}^u + (1-p)\bar{B}^m)) + (S_0 \otimes p\bar{B}^m), \\ U_s &= (I \otimes (\bar{U}_s^u + (1-p)\bar{U}_s^m)) + (S_0 \otimes p\bar{U}_s^m), \\ C_k &= S_1 \otimes p\bar{T}_k^m e\alpha_{ini}. \end{aligned} \quad (5)$$

where $\bar{T}_k^m = \bar{D}_k^m + \bar{B}^m + \sum_{s=0}^d \bar{U}_s^m$. In these equations I denotes the $l \times l$ unity matrix, e is a column vector of the appropriate dimension with all entries equal to one and S_0 is an $l \times l$ matrix with ones on the first diagonal below the main diagonal and all other entries equal to zero. The matrix S_1 has only one entry different from zero, being the last entry on the first row, which equals one. The matrix F that governs the transitions from the root node to itself can be written as

$$\begin{aligned} F &= (I \otimes (\bar{F}^u + (1-p)\bar{F}^m)) + (S_0 \otimes p\bar{F}^m) \\ &\quad + \left(S_1 \otimes p\left(\bar{F}^m e + \sum_{s=0}^d \bar{U}_s^m e\right)\alpha_{ini}\right). \end{aligned}$$

Consider the reset MC $\{(\mathcal{X}_t, \mathcal{N}_t), t \geq 0\}$ and define its transition matrix $P^{l,n}$ of infinite (countable) size such that the strings are ordered lexicographically—that is, $J < J'$ if either $|J| < |J'|$ or there exists an $k > 0$ such that the $J - f(J,k) = J' - f(J',k)$ and the $(|J| - k + 1)$-th entry of J is smaller than that of J'. The block matrix $P^{l,n}$ has the same structure as that of any tree-like process (see [4]), therefore, the components $\pi^{l,n}(J)$, defined as

$$\begin{aligned} \pi_{(i,j)}^{l,n}(J) &= \lim_{t\to\infty} P[\mathcal{X}_t = J, \mathcal{N}_t = (i,j)], \\ \pi_i^{l,n}(J) &= (\pi_{(i,1)}^{l,n}(J), \pi_{(i,2)}^{l,n}(J), \ldots, \pi_{(i,h)}^{l,n}(J)), \\ \pi^{l,n}(J) &= (\pi_1^{l,n}(J), \pi_2^{l,n}(J), \ldots, \pi_l^{l,n}(J)), \end{aligned}$$

of its stationary probability vector have the same form as those of a tree-like process, i.e., $\pi^{l,n}(J+k) = \pi^{l,n}(J)R_k$, except for $\pi^{l,n}(\emptyset)$ (see Section 2). Hence, we can easily compute the necessary components of $\pi^{l,n} = (\pi^{l,n}(\emptyset), \pi^{l,n}(1), \pi^{l,n}(2), \ldots)$ once the $lh \times lh$ matrix V, defined by Eq. (3), and the vector $\pi^{l,n}(\emptyset)$ are obtained. As lh can be considerably large, we will speed up their computation by exploiting the structure of the D_k, B, U_s and C_k matrices. Three such algorithms to determine V are presented in the next section. A sufficient condition for the existence of $\pi^{l,n}$ is that the average reset time of $(\mathcal{X}_t, \mathcal{N}_t)$ is finite, irrespective of whether the MC (X_t, N_t) is ergodic. The finiteness of the average reset time is often obvious from an operational point of view. If not, it can be verified rigorously by checking whether the special radius of $R = R_0 + \ldots + R_d$ is less than one (the G_k matrices of $(\mathcal{X}_t, \mathcal{N}_t)$ need not to be stochastic due to the reset events).

3.3 Computing the V matrix

(A) The $lh \times lh$ matrix V is the smallest nonnegative solution to Eqn. (4) and can be computed by a standard fixed point iteration [4]. That is, the matrix V is obtained as $\lim_{N\to\infty} V[N]$ from the recursion

$$V[N+1] = B + \sum_{s=0}^{d} U_s(I - V[N])^{-1}D_s, \quad (6)$$

where $V[0] = B$. We can however benefit from the structural properties of the B, U_s and D_s matrices involved to make this recursion faster. The matrices B, U_s and D_s (for $(0 \leq s \leq d)$ are all lower block bidiagonal matrices (see Eqn. (5)). Let us denote the blocks on the main diagonal of a lower block bidiagonal matrix X as $X^{(0)}$ and the blocks below the main diagonal as $X^{(1)}$. Hence,

$$\begin{aligned} B^{(0)} &= \bar{B}^u + (1-p)\bar{B}^m, & B^{(1)} &= p\bar{B}^m, \\ U_s^{(0)} &= \bar{U}_s^u + (1-p)\bar{U}_s^m, & U_s^{(1)} &= p\bar{U}_s^m, \\ D_s^{(0)} &= \bar{D}_s^u + (1-p)\bar{D}_s^m, & D_s^{(1)} &= p\bar{D}_s^m. \end{aligned}$$

The lower block bidiagonal structure implies that V has a block triangular block Toeplitz (btbT) structure. As a consequence, it suffices to determine the first block column of V in order to know the entire matrix. That is, a btbT matrix Y is completely characterized by its first block column as follows:

$$Y = \begin{bmatrix} Y^{(0)} & 0 & \cdots & \cdots & 0 \\ Y^{(1)} & Y^{(0)} & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ Y^{(l-2)} & \ddots & \ddots & Y^{(0)} & 0 \\ Y^{(l-1)} & Y^{(l-2)} & \ldots & Y^{(1)} & Y^{(0)} \end{bmatrix}.$$

Notice also that the btbT structure is preserved by every operation used in Eqn. (6), meaning all $V[N], G_k$ and R_k matrices are also btbT. Making use of Eqn. (1–4), we can

calculate matrix $V[N+1]$ from matrix $V[N]$ using block column vectors only. First, set

$$R_k^{(i)}[N+1] = \begin{cases} U_k^{(0)} W^{(0)}[N] & i = 0, \\ U_k^{(0)} W^{(i)}[N] + U_k^{(1)} W^{(i-1)}[N] & i > 0, \end{cases}$$

where the btbT matrix $W[N] = (I - V[N])^{-1}$ can be calculated in a the time and memory complexity of $O(l^2h^3)$ and $O(lh^2)$, respectively. The time complexity can be further reduced to $O(lh^3 + l\log(l)h^2)$ by making use of fast Fourier transforms (see [19, Chapter 2]). Second, compute $V[N+1]$ as

$$V^{(i)}[N+1] = \begin{cases} B^{(0)} + \sum_k R_k^{(0)}[N+1] D_k^{(0)} & i = 0, \\ \sum_k (R_k^{(i)}[N+1] D_k^{(0)} + R_k^{(i-1)}[N+1] D_k^{(1)}) + \\ \quad 1[i=1] B^{(1)} & i > 0. \end{cases}$$

As a result, the amount of memory needed to compute V is only linear in l as opposed to square.

(B) A second approach is to compute the blocks $V^{(0)}, \ldots, V^{(l-1)}$ that characterize the matrix V recursively. If we consider only the first block row of Eqn. (4), we get

$$V^{(0)} = B^{(0)} + \sum_{s=0}^{d} U_s^{(0)} (I - V^{(0)})^{-1} D_s^{(0)}. \quad (7)$$

As $V^{(0)}$ has size $h \times h$ only, we can rely on the standard fixed point iteration presented in Eqn. (6) for its computation. Due to (4) we can establish the following equation for $V^{(i)}$, for $i > 0$:

$$V^{(i)} = K_i + \sum_{s=0}^{d} U_s^{(0)} W^{(0)} V^{(i)} W^{(0)} D_s^{(0)}, \quad (8)$$

with $W^{(0)} = (I - V^{(0)})^{-1}$ and

$$K_i = 1[i=1] B^{(1)} + \sum_{s=0}^{d} \left[U_s^{(1)} W^{(i-1)} D_s^{(0)} + U_s^{(0)} W^{(0)} \left(\sum_{k=1}^{i-1} V^{(k)} W^{(i-k)} \right) D_s^0 + \left(U_s^{(1)} W^{(i-2)} 1[i>1] + U_s^{(0)} W^{(i-1)} \right) D_s^{(1)} \right].$$

Notice, having computed $V^{(0)}, \ldots, V^{(i-1)}$ allows us to compute K_i (as these suffice to compute the $W^{(k)}$ matrices, for $k < i$, that characterize the btbT matrix $(I - V)^{-1}$).

Two iterative approaches [4] can be followed to retrieve $V^{(i)}$ from Eqn. (8). Both start by setting $V^{(i)}[0] = K_i$. A first iteration consists in generation the sequence $V^{(i)}[N]$ as

$$V^{(i)}[N+1] = K_i + \sum_{s=0}^{d} U_s^{(0)} W^{(0)} V^{(i)}[N] W^{(0)} D_s^{(0)}. \quad (9)$$

The second one generates a sequence by solving the Sylvester matrix equation

$$V^{(i)}[N+1] - U_0^{(0)} W^{(0)} V^{(i)}[N+1] W^{(0)} D_0^{(0)} = K_i + \sum_{s=1}^{d} U_s^{(0)} W^{(0)} V^{(i)}[N] W^{(0)} D_s^{(0)}. \quad (10)$$

For the sake of completeness we present an Hessenberg algorithm that solves a Sylvester matrix equation of the form $X + AXB = C$ in the Appendix. The Sylvester scheme will typically result in fewer iterations, however, more time is needed to perform a single iteration, making the first approaches often the fastest. Compared to the algorithm presented in **(A)** both schemes tend to be significantly faster. However, as reported in [4], the convergence of (9) and (10) is not guaranteed, as opposed to the convergence of algorithm **(A)**. For the application discussed in Section 4 we experienced no convergence problems for both (9) and (10).

3.4 Calculating the probability vectors $\pi_n^m(J)$

Having obtained the $lh \times lh$ matrix V, we are now in a position to compute the stationary measure $\pi^{l,n}$ of the reset MC $(\mathcal{X}_t, \mathcal{N}_t)$. Consider the Markov chain obtained by censoring $(\mathcal{X}_t, \mathcal{N}_t)$ on the states of the root node $(\emptyset, (i,j))$. Its transition matrix $P_\emptyset^{l,n}$ can be written as

$$P_\emptyset^{l,n} = (F + V - B) + (e - (F + V - B)e)((0, \ldots, 0, 1) \otimes \alpha_{ini}),$$

where $\otimes$ denotes the Kronecker matrix product. The matrix F covers the single step transitions, whereas $V - B$ holds all the paths from the root node to itself that start with a transition to one of its child nodes without the occurrence of a reset event (as Eqn. (3) implies that $\sum_{s=0}^{d} U_s (I-V)^{-1} D_s = V - B$). The return state (i,j) of all the remaining paths from $\emptyset$ to itself is of the form (l,j) and is determined by α_{ini} as the final transition of these paths is a reset event.

The vector $\pi^{l,n}(\emptyset)$ corresponding to the root node $\emptyset$, is the invariant vector of $P_\emptyset^{l,n}$, normalized as

$$\pi^{l,n}(\emptyset)(I - R)^{-1} e = 1.$$

The matrix $P_\emptyset^{l,n}$ has a special structure, which we can exploit when computing $\pi^{l,n}(\emptyset)$. More precisely, it is the sum of a btbT matrix $(F + V - B)$ and an lbC matrix, where an lbC matrix is a matrix with its nonzero entries all positioned in the last block column. In order to compute $\pi^{l,n}(\emptyset)$ efficiently, we can therefore rely on the algorithm presented in [33, Section 5], the time and space complexity of which equal $O(m^3 l^2)$ and $O(m^2 l)$, respectively. The remaining components of the stationary measure $\pi^{l,n}$ for the nodes of the form $J + k$ are given by $\pi^{l,n}(J+k) = \pi^{l,n}(J) R_k$, where the matrices R_k (with $0 \le k \le d$) can be calculated via Eqn. (2). Fast Fourier transforms can be used to speed up the computation of the components $\pi^{l,n}(J+k)$, because of the btbT structure of R_k (see [33, Section 5] for details).

We now indicate how the probability vector $\pi_{Z_{l,n}}^m(J)$, used to approximate the system state $\pi_n^m(J)$ at the n-th marked time epoch, can be obtained from $\pi^{l,n}(J)$. The j-th entry of the vector $\pi_{Z_{l,n}}^m(J)$ holds the probability that the MC $(\mathcal{X}_t, \mathcal{N}_t)$ is in state $(J, (1,j))$ provided that a reset event occurs, hence,

$$\pi_{Z_{l,n}}^m(\emptyset) = \frac{(\pi_1^{l,n}(\emptyset).\phi_0)p}{c},$$

$$\pi_{Z_{l,n}}^m(J+k) = \frac{(\pi_1^{l,n}(J+k).\phi_{1,k})p}{c}.$$

Here, ϕ_0 and $\phi_{1,k}$ are the transposed vectors of $\bar{F}^m e +$

$\sum_{s=0}^{d} \bar{U}_s^m e$ and $\bar{T}_k^m e$ respectively and '.' denotes the pointwise matrix product. The normalization constant c, which guarantees that $\sum_J \pi_{Z_{l,n}}^m(J)e = 1$, is identical to the probability that a reset event occurs at an arbitrary time instant. It can be verified, using the structural properties of $\pi^{l,n}$, that $c = \gamma e$, where the $1 \times h$ vector γ is given by

$$\gamma/p = \pi_1^{l,n}(\emptyset).\phi_0 + \pi^{l,n}(\emptyset)(I-R)^{-1}\left(\sum_{k=0}^{d} R_k^{(*)}.(e\phi_{1,k})\right), \tag{11}$$

where $R_k^{(*)}$ represents the first block column of the btbT matrix R_k. Notice, in order to calculate the normalization constant c, it suffices to know the probability vector corresponding to the root node of the reset MC. The average time after which the $Z_{l,n}$-th reset event takes place $\bar{t}_n = 1/c$, which may be regarded as an approximation to the average time at which the n-th marking occurs, is thus found as a byproduct and is often of particular interest on its own. Exact results can be generated provided that setting $l = n$ does not cause periodicity in the MC $(\mathcal{X}_t, \mathcal{N}_t)$.

4. TRANSIENT ANALYSIS OF THE CTM PROTOCOL WITH FREE ACCESS

In this section we make use of our new framework to analyze the transient behavior of the celebrated Capetanakis-Tsybakov-Mikhailov (CTM) protocol [9, 28, 10, 11] for random multiple access communication. An abundance of papers has been devoted to evaluate its performance (see [21] for an extensive overview), to the best of our knowledge none have been devoted to its transient behavior. We will restrict ourselves to the basic binary CTM algorithm with free access. Other protocol variations are also within reach of our framework as their behavior can often be captured by a tree-like process (see [34]).

The following standard assumptions are made [3, 21]. A single channel is shared among an infinite population of users that transmit packetized messages. A user is said to becomes active as soon as he has a message ready for transmission. Users refrain from generating new messages while being active. The time is slotted and each slot has a fixed duration, equal to the length of a packet. Transmissions of a packet are assumed to occur at the beginning of a time slot and each transmission is within the reception range of every user. Binary feedback (collision/no collision) is immediately available at the end of each time slot. Simultaneous transmissions are destructive, in the sense that all packet involved become corrupt. However, a discrete-time batch Markovian arrival process (D-BMAP) is used to model the user activity, which greatly relaxes the standard Poisson assumption. Some of the other standard assumptions can be further relaxed if necessary.

The D-BMAP process [6] is the discrete-time counterpart of the BMAP and is characterized by a sequence of $h^* \times h^*$ matrices B_n, for $n \geq 0$, the entries of which are denoted by $b_n^{i,j}$. These matrices have the following probabilistic interpretation. If the D-BMAP is in state i $(1 \leq i \leq h^*)$ at the start of slot t, n users become active and the state of the D-BMAP equals j $(1 \leq j \leq h^*)$ at the beginning of slot $t+1$ with a probability $b_n^{i,j}$. The arrivals are assumed to occur on the boundary of time slot $t-1$ and t, meaning the n users that became active can transmit their packet as early as time slot t.

The CTM protocol is a collision resolution algorithm in which users continue to retransmit a packet until it is correctly received. Apart from the immediate feedback, users do not exchange any information about their activity on the channel, as a consequence contention has to be resolved without any additional information. In order to resolve contention the CTM protocol divides the users involved in a collision into two groups $GR(1)$ and $GR(2)$. The users of the first group $GR(1)$ retransmit their packet in the next time slot, while users of the second group $GR(2)$ wait until the first group is resolved (meaning all messages belonging to $GR(1)$ have to be transmitted successfully and the users of $GR(2)$ need to be aware of this). The algorithm has a recursive nature, in the sense that if $GR(1)$ causes another collision, $GR(1)$ is split into two groups $GR(1a)$ and $GR(1b)$, and so on. Notice, in this case the users of $GR(2)$ need postpone any retransmission attempts until both $GR(1a)$ and $GR(1b)$ have been resolved. The channel access protocol is assumed to be the free access protocol. This means that users that become active immediately—that is, at the start of the next slot boundary—access the channel.

By allowing the immediate participation of new arrivals in the contention tree, this problem can be regarded as a dynamic tree algorithm. The asymptotic behavior of tree algorithms that do not support this dynamic feature were studied in [24, 20]. Some transient statistics on these (non-dynamic) tree algorithms can also be derived from our setting. For instance, by assuming Poisson arrivals with $\lambda = 0$ and by considering a conflict of size n as the initial state, we could compute the mean time until the i-th success (for $i \leq n$) of a size n conflict. By repeatedly resetting the tree-like process at the i-th success, exact results can be realized for limited values of n. Numerical results not presented here, show that one typically observes a linear growth as a function of i.

In the CTM protocol each active user maintains an integer value, referred to as the current stack level. A user that became active during slot $t-1$ initializes its current stack level for slot t at zero. A user is allowed to send its packet during the current time slot if its current stack level equals zero. The current stack level of a user is updated as follows. If slot t does not hold a collision, all users with a current stack level for slot t equal to $i > 0$ set their current stack level for slot $t+1$ to $i-1$. If slot t however holds a collision, the current stack level for slot $t+1$ is set to $i+1$ and users with a current stack level for slot t equal to zero are divided into two groups. A user joins the first (resp. the second) group with a probability q (resp. $1-q$) and sets its current stack level for time $t+1$ to zero (resp. one). Remark, the current stack level maintained by a user may be regarded as the number of pending groups that require resolution before the user is allowed to retransmit.

Let us now briefly indicate how to model the CTM protocol as a tree-like process (for more details see [29]). Consider the Markov chain $\{(\mathbb{X}_t, \mathbb{N}_t), t \geq 0\}$. Let $\mathbb{X}_t$ be the string holding the current stack level for slot t for all backlogged stations, i.e., stations with a current stack level (for slot t)

larger than zero. The sample space of the random variable $\mathbb{X}_t$ is $\Omega_1 = \{\emptyset\} \cup \{J : J = s_k \dots s_1, s_j \geq 0, 1 \leq j \leq k, k \geq 1\}$. For instance, if $\mathbb{X}_t = s_k \dots s_1$ there are $\sum_{i=1}^{k} s_i$ backlogged stations and the current stack level equals i for s_i of the backlogged stations. The auxiliary variable $\mathbb{N}_t$ holds both the number of active stations with a current stack level for slot t equal to zero and the state of the D-BMAP at the start of slot $t+1$. Its sample space is given by $\Omega_2 = \{(n, j) : n \geq 0, 1 \leq j \leq h^*\}$.

The Markov chain $(\mathbb{X}_t, \mathbb{N}_t)$ (with state space $\Omega_1 \times \Omega_2$) has a tree structure, but each node has an infinite number of children. To reduce this MC to a tree-like process one can approximate $(\mathbb{X}_t, \mathbb{N}_t)$ by a bivariate Markov chain $(\mathbb{X}_t^d, \mathbb{N}_t^d)$, obtained by setting a maximum value d on the number of stations that can have a same current stack level for slot t. This way, each node of $\mathbb{X}_t^d$ has only $(d+1)$ children and the variable $\mathbb{N}_t^d$ has a finite range of size $h = h^*(d+1)$. The introduction of the value d as a restriction on the number of stations having the same current stack level, implies that some packets get dropped whenever a situation occurs in which more than d packets attain the same current stack level value. The CTM protocol does not drop packets, thus, d has to be chosen large enough such that number of dropped packets can be neglected (e.g., $< 10^{-10}$). In practice, setting $d = 10$ or 15 often suffices.

For further use, we will now quickly reintroduce the matrices $\bar{D}_k$, $\bar{U}_s$ and $\bar{F}$ that fully characterize the tree-like process $(\mathbb{X}_t^d, \mathbb{N}_t^d)$. The matrices $\bar{D}_k$ cover the transition probabilities from state $(J+k, (i, j))$ to the state $(J, (i', j'))$. This happens when slot t does not hold a collision, which implies

$$\bar{D}_k((i,j),(i',j')) = \begin{cases} (B_{i'-k})_{j,j'} & i \leq 1, i' \geq k, i' < d, \\ \sum_{n \geq d-k} (B_n)_{j,j'} & i \leq 1, i' \geq k, i' = d, \\ 0 & otherwise. \end{cases} \tag{12}$$

The matrices $\bar{U}_s$ hold the transition probabilities of going from state $(J+k, (i, j))$ to the state $(J+ks, (i', j'))$, which happens when a collision occurs in slot t. Hence,

$$\bar{U}_s((i,j),(i',j')) = \begin{cases} C_s^i q^{i-s} \tilde{q}^s (B_{i'-(i-s)})_{j,j'} & i > 1, i \geq s, i' \geq i-s, i' < d, \\ C_s^i q^{i-s} \tilde{q}^s \sum_{n \geq d-(i-s)} (B_n)_{j,j'} & i > 1, i \geq s, i' \geq i-s, i' = d, \\ 0 & otherwise, \end{cases} \tag{13}$$

where $\tilde{q} = 1 - q$ and C_s^i denotes the number of different possible combinations of s from i different items. Finally, the transition probabilities that the process goes from state $(\emptyset, (i, j))$ to the state $(\emptyset, (i', j'))$ are given by the matrix $\bar{F}$. This matrix is given by

$$\bar{F}((i,j),(i',j')) = \begin{cases} (B_{i'})_{j,j'} & i \leq 1, i' < d, \\ \sum_{n \geq d} (B_n)_{j,j'} & i \leq 1, i' = d, \\ 0 & otherwise. \end{cases} \tag{14}$$

We are now in a position to apply our framework to assess some transient performance measures of the CTM protocol.

Let us first take a look at the transient throughput of the CTM algorithm. The throughput T_n in slot n equals the probability that a packet is successfully transmitted in slot n. This happens when there is exactly one station with its current stack level equal to zero for slot n. We can apply our framework by marking all time epochs, i.e., $D_k^u = B^u = U_s^u = F^u = 0$, as this probability is readily available from the system state at time n. More specifically, we have

$$T_n = \sum_{j=1}^{h^*} \pi_n^m(1, j), \text{ where } \pi_n^m(i,j) = \sum_J \pi_n^m(J, (i,j)),$$

and $\pi_n^m(J, (i, j))$ are the obvious $h = h^*(d+1)$ components of the vector $\pi_n^m(J)$. An simple expression for T_n can be found from the $1 \times h$ vector $\gamma = (\gamma_0, \gamma_1, \dots, \gamma_d)$, defined by Eqn. (11), with $\gamma_i = (\gamma_i(1), \dots, \gamma_i(h^*))$. That is, γ/c contains, for each combination of the current stack level i and the state of the D-BMAP process j, the probability that, at the n-th marking there are i stations with a current stack level equal to zero, while the state of the D-BMAP process is j. Hence,

$$T_n = \sum_{j=1}^{h^*} \frac{\gamma_1(j)}{c}.$$

We first present some results in case the user activity is modeled as a Poisson arrival process, that is, we set $B_n = e^{-\lambda}\lambda^n/n!$, for $n \geq 0$. Flajolet and Jacquet [11] have shown that the CTM algorithm with free access (with $q = 0.5$) is stable under a Poisson flow of arrivals if the arrival rate $\lambda < 0.360177$. We refer to this value as the maximum stable throughput (MST) of the random access algorithm.

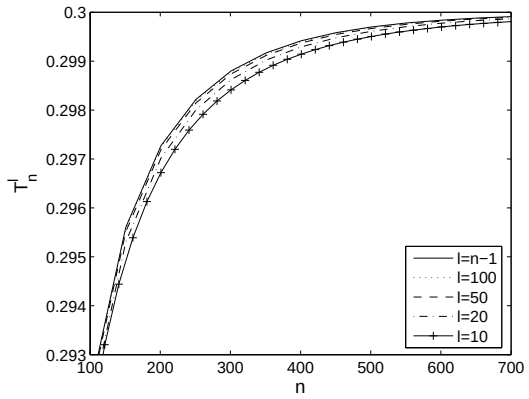

Figure 1: ***Throughput in slot n under Poisson arrivals: the influence of l on the accuracy***

Figure 1 depicts some approximations T_n^l for the transient throughput T_n for different values of l, the number of phases of the negative binomial distribution used to setup the reset process. T_n^l is computed in the same manner as T_n, except that the vectors $\pi_n^m(J)$ are approximated by $\pi_{Z_{l,n}}^m(J)$. The rate of the Poisson process $\lambda = 0.3$. As mentioned earlier, in some cases we cannot set l equal to n as this causes the MC $(\mathcal{X}_t, \mathcal{N}_t)$ to become periodic. Marking all time epochs will always cause periodicity, therefore the best approximation we can realize is by setting l equal to $n-1$. Figure 1 indicates that smaller values of l will result in a larger underestimation of the throughput T_n, however, for relatively small values

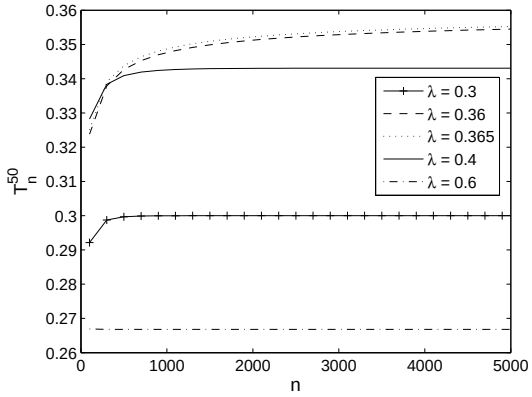

Figure 2: ***Throughput in slot n under Poisson arrivals for various arrival rates λ***

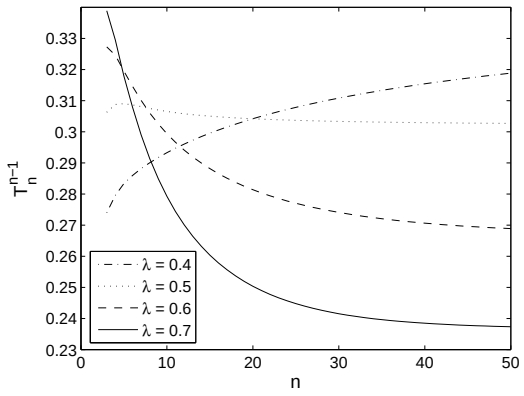

Figure 3: ***Throughput in the n-th slot***

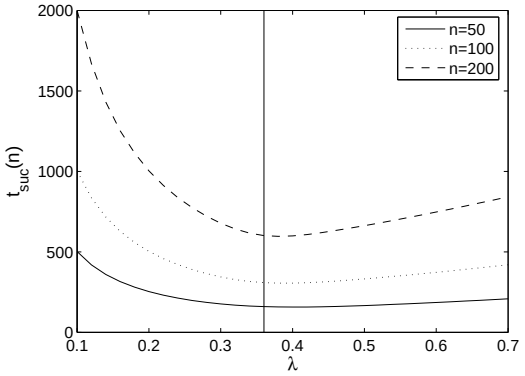

Figure 4: ***Average time $\bar{t}_{suc}(n)$ until the n-th success under Poisson input***

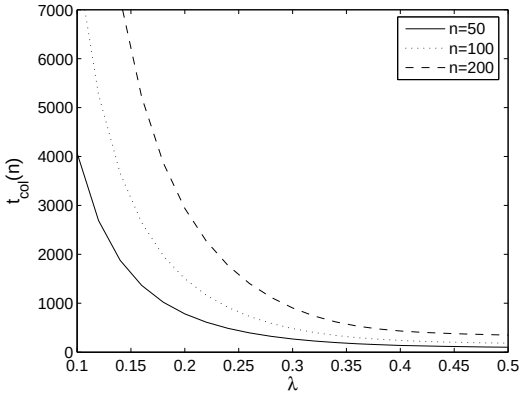

Figure 5: ***Average time $\bar{t}_{col}(n)$ until the n-th collision under Poisson input***

of l we still obtain a quite accurate approximation of the throughput compared to letting $l = n - 1$. Working with a smaller l value implies that less time and memory is needed for the computation of the matrix V and the probability vectors $\pi^m_{Z_{l,n}}(J)$. For $l = 100$, the computation time, as given by the Matlab Profiler, is close to 30 seconds using an Intel Pentium M 1.70GHz processor with 1GB of memory. The Windows Task Manager indicated a peak memory usage of 80Mb for the entire Matlab session during the calculation of this figure.

In Figure 2 we compare the transient throughput for different arrival rates λ. For $\lambda = 0.3$, we notice that the transient throughput rapidly converges to the input rate λ. When we take the arrival rate λ closer to the MST (e.g., $\lambda = 0.36$), the throughput grows more slowly over time, but still seems to reach the input rate for n large enough. One can also observe that taking values for λ above the MST, does not seem to result in an output rate that is above the MST. Actually, for slot $n = 10^7$ we have $T^{100}_n = 0.35996$ and $T^{100}_n = 0.35812$ for $\lambda = 0.36$ and $\lambda = 0.365$, respectively. For such arrival rates a somewhat higher throughput is often achieved in the first few slots, as shown in Figure 3. After this initial phase during which several stations get backlogged, the number of collisions increases as the new arrivals need to compete with some of these backlogged stations.

Next, we will calculate the average time $\bar{t}_{suc}(n)$ at which the n-th successful transmission takes place. To apply our framework, it suffices to mark all time epochs that correspond to a successful transmission. The matrices $\bar{D}^m_k$, $\bar{U}^m_s$ and $\bar{F}^m$ that mark a time epoch are calculated from the matrices $\bar{D}_k$, $\bar{U}_s$ and $\bar{F}$, given by Eqns. (12–14), as follows:

$$\bar{D}^m_k((i,j),(i',j')) = \begin{cases} \bar{D}_k((i,j),(i',j')) & \text{if } i = 1, \\ 0 & \textit{otherwise.} \end{cases},$$

for $k = 0, \ldots, d$ and

$$\bar{F}^m((i,j),(i',j')) = \begin{cases} \bar{F}((i,j),(i',j')) & \text{if } i = 1, \\ 0 & \textit{otherwise.} \end{cases}.$$

The entries of the matrices $\bar{U}_s$ are equal to zero for $i = 1$, hence, $\bar{U}^m_s = 0$. When computing $\bar{t}_{suc}(n)$, we can set l equal to n to generate exact results. Recall, $\bar{t}_{suc}(n) = \bar{t}_n = 1/c$, where c is the normalization constant defined in Section 3.4.

Figure 4 shows the mean time $\bar{t}_{suc}(n)$ that passes until the n-th successful transmission occurs, for $n = 50, 100$ and 200. If this rate is low, the number of slots needed to transmit n packets successfully is relatively high. When λ increases $\bar{t}_{suc}(n)$ decreases up to some point where a mini-

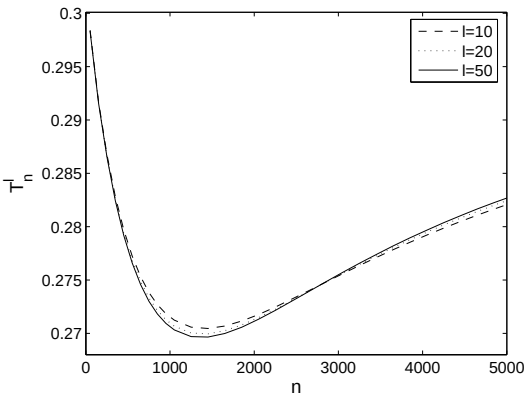

Figure 6: ***Throughput in the n-th slot***

mum is reached, afterward $\bar{t}_{suc}(n)$ increases again. This can be understood as follows. When λ is low, the total number of packets generated is small and therefore a considerable amount of time is required to generate n packets, most of these packets are successful during their first attempt. Higher values of λ reduce the time required to generate n packets, but at the same time also increases the mean number of transmission attempts needed. When the arrival rate becomes too large, the effect of the increased collision probability starts to dominate, which causes the minimum. Our results indicate that for large values of n, the rate λ for which $\bar{t}_{suc}(n)$ is minimal, can become larger than the MST.

Similarly, define $\bar{t}_{col}(n)$ as the average time at which the n-th collision takes place. To determine $\bar{t}_{col}(n)$ we can apply our framework with

$$\bar{D}_k^m = \bar{F}^m = 0 \text{ and } \bar{U}_s^m = \bar{U}_s.$$

Remark, each time epoch where at least two stations have a current stack level equal to zero are marked, as these stations will transmit their packet in the next time slot causing a collision. The average time until the n-th collision is depicted in Figure 5, for $n = 50, 100, 200$. As one can expect, this average time decreases as the Poisson arrival rate increases.

Until now, we assumed that the users become active according to a Poisson process. As a final result, we will consider a two-state Markov modulated Poisson process as well. When in the first (the second) state, stations become active according to a Poisson process with rate λ_1 (λ_2). Transitions between the two states take place with a probability 0.001 at the end of each time slot, that is, the average sojourn time in both states is 1000 slots. We assume that the system is initialized in state 1. Figure 6 shows the transient throughput for different values of l for $\lambda_1 = 0.5$ and $\lambda_2 = 0.1$. Therefore the matrices B_n are given by

$$B_n = \begin{bmatrix} 0.999e^{\lambda_1}\lambda_1^n/n! & 0.001e^{\lambda_1}\lambda_1^n/n! \\ 0.001e^{\lambda_2}\lambda_2^n/n! & 0.999e^{\lambda_2}\lambda_2^n/n! \end{bmatrix}.$$

As can be seen in Figure 6, during the first 1000 to 1500 slots the throughput decreases, after which it slowly starts to grow toward the arrival rate $\lambda = 0.3$. This is in line with the previous observations: initially we have an overloaded system as $\lambda_1 = 0.5$ which causes the initial decrease as explained before. On average, after about 1000 slots, the state of the D-BMAP will change to $\lambda_2 = 0.1$, allowing the backlogged stations to access the channel more often. As the CTM protocol is still stable under this arrival process, the throughput will eventually converge to $\lambda = 0.3$ as n goes to infinity. How one checks the stability of the CTM protocol under D-BMAP arrivals is explained in [29].

APPENDIX

For reasons of completeness, we present a Hessenberg type algorithm to solve a Sylvester matrix equation of the type $X + AXB = C$.

Algorithm:

- Input: the matrices A, B and C.
- Apply a Hessenberg decomposition on the matrix A: $A = PHP^*$, where P^* denotes the complex conjugate of P and P is a unitary matrix (meaning $PP^* = I$).
- Let $U^*BU = T$ be a complex Schur decompostion of B, with U unitary and T a triangular matrix.
- By premultiplying $X + AXB = C$ by P^* and postmultiplying it by U, $X + AXB = C$ can be rewritten as $HYT + Y = F$, with H a Hessenberg and T a triangular matrix. Here, $Y = P^*XU$ and $F = P^*DU$.
- Denote z_i and z_{ij} as the i-th column and the (i, j)-th entry of a matrix Z, respectively. Then, considering the i-th column of the equation $HYT + Y = F$ gives us $[I + t_{ii}H]y_i = f_i - \sum_{j=1}^{i-1} t_{ji}Hy_j$.
- Hence, to compute X we need to solve d linear Hessenberg systems, where d denotes the dimension of the square matrices A, B and C.

This algorithm has a time complexity of $O(d^3)$. It is easy to implement as for instance each of the decompositions required is a Matlab built-in function. Moreover, the '\' Matlab command to solve the d linear systems recognizes the Hessenberg structure and solves each of the systems in $O(d^2)$ time. Notice, from Eqn. (10) that we only need to perform the Hessenberg and the Schur decomposition once because $A = U_0^{(0)}W^{(0)}$ and $B = W^{(0)}D_0^{(0)}$, which are independent of the values of k and N.

A. REFERENCES

[1] K. Al Agha, P. Jacquet, and N. Vvedenskaya. Analysis of the priority stack random-access protocol in W-CDMA systems. *IEEE Transactions on Vehicular Technology*, 51(3):588–596, 2002.

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

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

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

[6] 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.

[7] 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.

[8] O. Boxma, D. Denteneer, and J. Resing. Delay models for contention resolution in closed populations. *Performance Evaluation*, 53:169–185, 2003.

[9] J.I. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Trans. Inform. Theory*, 25(5):319–329, 1979.

[10] G. Fayolle, P. Flajolet, M. Hofri, and P. Jacquet. Analysis of a stack algorithm for random multiple-access communication. *IEEE Transactions on Information Theory*, IT-31(2):244–254, 1985.

[11] P. Flajolet and P. Jacquet. Analytic models for tree communication protocols. Technical Report 648, INRIA, 1987.

[12] N. Golmie, Y. Saintillan, and D.H. Su. A review of contention resolution algorithms for IEEE 802.14 networks. *IEEE Communication Surveys*, 2(1), 1999.

[13] Q. He and A.S. Alfa. The MMAP[K]/PH[K]/1 queues with a last-come-first-serve preemptive service discipline. *Queueing Systems*, 28:269–291, 1998.

[14] Q. He and A.S. Alfa. The discrete time MMAP[K]/PH[K]/1/LCFS-GPR queue and its variants. In *Proc. of the 3rd Int. Conf. on Matrix Analytic Methods*, pages 167–190, Leuven (Belgium), 2000.

[15] 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.

[16] P. Jacquet, P. Mühlethaler, and P. Robert. Performant implementations of tree collision resolution algorithms for CATV networks. Technical Report 4107, INRIA, 2001.

[17] A.J.E.M. Janssen and M.J.M. de Jong. Analysis of contention tree algorithms. *IEEE Transactions on Information Theory*, 46:2163–2172, 2000.

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

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

[20] A. Mohamed and P. Robert. A probabilistic analysis of some tree algorithms. *Annals of Applied Probability*, 15(4):2445–2471, 2005.

[21] M. L. Molle and G.C. Polyzos. Conflict resolution algorithms and their performance analysis. Technical report, University of Toronto, CS93-300, 1993.

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

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

[24] P. Robert. On the asymptotic behavior of some algorithms. *Random Structures and Algorithms*, 27(2):235–250, 2005.

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

[26] Y.E. Sagduyu and A. Ephremides. Energy-efficient collision resolution in wireless Ad-Hoc networks. In *IEEE Infocom*, San Francisco, 2003.

[27] 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.

[28] B. S. Tsybakov and V.A. Mikhailov. Free synchronous packet access in a broadcast channel with feedback. *Problemy Peredachi Inform*, 14(4):32–59, 1978.

[29] B. Van Houdt and C. Blondia. Stability and performance of stack algorithms for random access communication modeled as a tree structured QBD Markov chain. *Stochastic Models*, 17(3):247–270, 2001.

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

[31] B. Van Houdt and C. Blondia. Robustness of Q-ary collision resolution algorithms in random access systems. *Performance Evaluation*, 57:357–377, 2004.

[32] B. Van Houdt and C. Blondia. Analyzing priority queues with 3 classes using tree-like processes. *Submitted for publication*, 2005.

[33] B. Van Houdt and C. Blondia. QBDs with marked time epochs: a framework for transient performance measures. In *Proc. of QEST 2005*, pages 210–219. IEEE Computer Society, 2005.

[34] 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.

[35] 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.

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