STOCHASTIC MODELS
2016, VOL. 32, NO. 1, 20–51
http://dx.doi.org/10.1080/15326349.2015.1069202

Taylor & Francis
Taylor & Francis Group

Throughput and delay analysis of an opportunistic tree algorithm

R. Block and B. Van Houdt

Department of Mathematics and Computer Science, University of Antwerp–iMinds, Antwerp, Belgium

ABSTRACT

Tree algorithms are a well-known class of random access algorithms with a provable maximum stable throughput under the infinite population model (as opposed to ALOHA or the binary exponential backoff algorithm). In this article, we propose a tree algorithm for opportunistic spectrum usage in cognitive radio networks. A channel in such a network is shared among so-called primary and secondary users, where the secondary users are allowed to use the channel only if there is no primary user activity. The tree algorithm designed in this article can be used by the secondary users to share the channel capacity left by the primary users.

We analyze the maximum stable throughput and mean packet delay of the secondary users by developing a tree structured Quasi-Birth Death Markov chain under the assumption that the primary user activity can be modeled by means of a finite state Markov chain and that packets lengths follow a discrete phase-type distribution.

Numerical experiments provide insight on the effect of various system parameters and indicate that the proposed algorithm is able to make good use of the bandwidth left by the primary users.

ARTICLE HISTORY
Received April 2014
Accepted June 2015

KEYWORDS
Cognitive radio; delay; maximum stable throughput; random access; tree algorithms

MATHEMATICS SUBJECT CLASSIFICATION
91B70; 90B18; 68M12

1. Introduction

As the number of wireless devices increases day by day, the need for more bandwidth increases with it. In order to cope with the resulting spectrum scarcity problem, the concept of opportunistic spectrum access has been proposed.[1] The main idea of this concept is to allow unlicensed or secondary users (SUs) on channels that are idle. Whenever licensed or primary user (PU) activity is detected, the SUs should vacate the channel immediately, ensuring the caused interference is kept at a minimum. Since its introduction, many medium access control (MAC) protocols have been introduced and analyzed.[8, 9] While many of these MAC algorithms are either based on a CSMA/CA-like random access scheme (e.g., Refs.[12, 14]) or use some form of centralized control with channel reservation (e.g., Refs.[5, 7]), our aim is to investigate

CONTACT B. Van Houdt benny.vanhoudt@uantwerpen.be Department of Mathematics and Computer Science, University of Antwerp–iMinds, Middelheimlaan 1, B-2020 Antwerp, Belgium.

whether we can design an effective random access scheme that belongs to the class of tree algorithms for opportunistic channel access.

Tree algorithms [6, 10, 25, 2, 18, 16] form a well-studied class of random access algorithms that were the first to have a provable maximum stable throughput (MST) above 0 under the infinite population model (see Section 2.1). Under this model users arrive to the system according to a Poisson process and leave the system as soon as they transmitted their packet successfully. The MST is defined as the highest possible (Poisson) input rate λ (times the mean packet length if different from 1) for which a user spends a finite amount of time in the system with probability one. Tree algorithms are said to support free access if new users can join at any time, which is a desirable property for SUs in a cognitive radio setting. The most important existing results for tree algorithms with free access include Refs.[17, 25], which cover the basic and modified q-ary tree algorithm, Refs.[27, 13] which analyzes the impact of channel errors, Refs.[24, 11] which considers variable length packets and many more (see Refs.[20, 19]). For blocked access, several variations have been considered as well; they include channels with capture,[21] control subchannels,[23] multi-reception capabilities,[15] etc.

In all of the above studies, the channel shared by the users is available at all times, in contrast to a cognitive radio setting. Our objective therefore exists in designing and analyzing a tree algorithm for opportunistic spectrum usage that is able to cope with possibly long periods of channel unavailability. In order to demonstrate whether the designed algorithm uses the remaining spectrum in an effective manner, we show that the achieved MST during the periods of PU inactivity is close to the MST of the original tree algorithm. Numerical experiments also provide insights on the effect of various system parameters on the MST and mean packet delay of the SUs.

The system model we consider is a centralized network in which an unlimited set of clients try to communicate with a single access point (AP). All these nodes have two channels to their disposal: a data channel for which they are unlicensed and a (very narrow) control channel they can use freely and are licensed for. The nodes are informed about possible PU activity by the AP in an error-free manner. However, whereas the AP is equipped with the necessary hardware to transmit data and scan for PU activity simultaneously, the clients may have hardware to assess only one of the two channels at a time. This keeps the clients low-cost and their energy consumption low. Besides being perfectly detectable by the AP, the only other assumption we make about PUs is that their activity periods can be modeled using a finite state discrete time Markov chain (MC), where some states correspond to PU activity and the others to PU inactivity.

Among the SUs, the data channel is divided into time slots. The length of each slot should exceed the time needed by the AP to scan for PU activity. Packet transmissions can start only at the beginning of each slot, and packets can span multiple slots. At the end of each slot, the AP broadcasts error-free feedback that will inform the SUs about possible PU activity and whether or not a successful transmission occurred (see Section 2.2 for more details). Whenever the PUs become active in a

slot, its SU content is considered unusable and will have to be resent at a later time. The algorithm also relies on channel reservation: if the first segment of a packet is transmitted successfully, the channel remains reserved for the remaining segments of the packet, even if PU activity occurs in between.

In order to analyze the MST and mean packet delay of the SUs, we define a tree-structured Quasi-Birth-Death (QBD) Markov chain, by extending the ideas introduced in Ref.[26] It should be noted that constructing such an MC for the opportunistic tree algorithm is far more challenging compared to the basic algorithm with packets of length one (as in Ref.[26]) as the obvious generalization leads to matrices of impractical size for numerical evaluation. As such, the system is observed only at specific points in time, which significantly complicates the transition probabilities as well as the computation of the mean delay from the steady-state probabilities of the chain.

The remainder of the article is structured as follows. In Section 2 the opportunistic tree algorithm is introduced along with the channel model used to assess its performance. Section 3 contains a brief overview of the class of tree-structured QBD MCs and their main characteristics. Next, Section 4 indicates how these MCs can be used to model the opportunistic tree algorithm, where we use a stepwise approach to capture all of the features of the new algorithm and channel model. In Section 5 we indicate how the MST and mean delay can be determined using this Markov chain. Numerical results and a discussion of the impact of various parameters on the performance can be found in Section 6.

2. Algorithm and channel model

2.1. *Basic binary tree algorithm*

The basic binary TA, also known as the Capetanakis-Tsybakov-Mikhailov or CTM algorithm, with free access was introduced in Refs.[6, 25] and was initially analyzed using the following standard model:

1. Time is slotted into fixed-length time slots, and packets have a fixed length equal to one time slot. Packet transmissions are allowed to start only at the beginning of a time slot.
2. The channel is error-free, meaning transmissions are unsuccessful if and only if more than one user transmits simultaneously, where simultaneous transmissions are called collisions.
3. Error-free binary feedback is provided at the end of each time slot that indicates whether the slot contained a collision or not.
4. Infinite population: new users arrive on the channel according to a Poisson process and leave the channel after successfully transmitting their packet.

This standard model has been relaxed in a number of ways, allowing channel errors,[27, 13] variable packet lengths,[24, 11] Markovian arrivals,[26] etc.

The operation of the basic binary TA can be described as follows. Users with a packet ready for transmission are called active, and the other users are called

inactive. Each user maintains a variable called *counter*, and when a user becomes active, this variable is initialized to zero. An active user whose counter equals zero is allowed to transmit in the next time slot (hence the term *free access* as new users are always free to join). Users with a counter value larger than zero have to defer from transmitting until their counter value becomes zero. These users are called the *backlogged users*. All active users listen to the channel feedback and update their counter values as follows:

1. If the slot holds a collision, all backlogged users increase their counter by 1, while users involved in the collision each flip a coin.* If the result is heads, the counter value remains zero; otherwise, it increases to 1.
2. If the slot was idle or contained a successful transmission, all backlogged users decrease their counter by 1. If the slot held a successful transmission, the successful user becomes inactive.

A more intuitive way of describing the algorithm is to say that the users involved in a collision split into two groups. The users in the first group, with counter value zero, may retransmit in the next slot, while the users in the second group wait until all the packets of the first group are successfully transmitted, including any new packets that arrive during the resolution of the first group. If new collisions occur during the resolution of a group, the above-mentioned method is applied recursively. The counter value of a user thus represents the number of groups that are still ahead before the user may retransmit.

One can also describe the operation of the basic tree algorithm using an (initially empty) stack of groups, where each group contains zero or more users. At the start of a slot, the top of stack (ToS) is popped and the new arrivals are added to that group. All the users in the group then transmit. If this results in a collision, meaning the group contained more than one user, the group is randomly split into two groups, and both groups are pushed on the stack. If the transmission was successful or the slot was idle, meaning the group contained at most one user, the group is discarded at the end of the slot. In other words, collisions increase the stack size by one, while idle and success slots decrease its size by one. This procedure is then repeated in each slot. Using this representation, the counter variable of a user represents the user's position in the stack. For this reason, tree algorithms are also called stack algorithms.

2.2. *Opportunistic binary tree algorithm*

In this section we introduce the opportunistic binary tree algorithm (OBTA) and the cognitive radio channel model.

2.2.1. *OBTA channel model*

We start by listing some key differences between the standard channel model discussed in the previous section and the cognitive radio channel model:

* The coin may be biased, and its probability p of being heads is a protocol parameter. In our numerical experiments, we have set $p = 0.5$, while minor improvements in the performance can be expected when optimizing p.

1. The PU activity is introduced, and the stations using the OBTA are considered SUs, who are allowed to transmit packets only when the PUs are inactive.
2. A second channel is introduced on which the stations using OBTA can rely to exchange small control packets during periods of PU activity.
3. Both channels are slotted, but the length of the slots is no longer related to the packet length, but should exceed the time the receiver needs to determine whether the PU is active or not.
4. The packet length is no longer fixed, but packets spanning multiple slots are split into segments that fit within a single slot.
5. The receiver provides 8 types of feedback (using 3 bits) instead of binary feedback at the end of each time slot. New users are able to receive the feedback provided at the end of the time slot in which they became active.

2.2.2. *OBTA feedback*

The eight possible feedback values are *idle*, *collision*, *success_cont*, *success_done*, *pu_start*, *pu_active*, *pu_end*, and *pu_end_cont*. The first 4 feedback values are used to support the variable packet length and indicate that there is no PU activity and have the following meaning

- *idle*: the slot contains no SU activity,
- *collision*: the slot holds a collision among the SUs,
- *success_cont*: the successful reception of a segment of a packet takes place, which is not the last segment,
- *success_done*: the final segment of a packet is successfully transmitted in the slot.

The remaining four values are used to support the PU activity as follows:

- *pu_start*: a period of PU activity started in this slot and the channel was not reserved,
- *pu_active*: a period of PU activity started and the channel was reserved or a period of PU activity continues,
- *pu_end*: the PU activity period has ended (all the PUs became inactive) and the next slot is not reserved,
- *pu_end_cont*: the PU activity period has ended (all the PUs became inactive) and the next slot is reserved by a SU for the next segment of a packet.

To provide the latter type of feedback, the receiver scans each slot for PU activity.

2.2.3. *OBTA algorithm*

To ensure minimal interference for the PU, the SUs should stop transmitting as soon as PU activity is detected. However, as they are unaware of PU activity until the end of the slot, some overlap between the SUs and the PUs is possible. A segment transmitted during this overlap is considered lost and should be retransmitted. Likewise, the SUs are unaware of the deactivation of the PU after a period of activity until they receive confirmation of this at the end of the slot. Consequently, the first slot after a period of PU activity will be idle. Clearly, the SUs need to be able to receive feedback during the activity periods of the PU. Therefore, all the stations switch to the

second channel upon receiving feedback equal to *pu_start* or *pu_active* and switch back to the first, broader, channel when they receive feedback equal to *pu_end* or *pu_end_cont*.

The OBTA algorithm is constructed out of the basic binary tree algorithm (BTA) by introducing a few modifications, but it keeps the core idea of recursively splitting collisions intact. The modifications make sure that if the PU becomes active, the SU transmissions are suspended until the PU is known to be inactive again. They also introduce the notion of channel reservation (as in Refs.[24, 11]): once the first segment of a packet is successfully transmitted, the channel is reserved among the SUs for the remainder of the packet, meaning the other SUs will not interfere with the transmission until the packet is fully transmitted, even if there is PU activity in between. The modifications are as follows:

1. New users still initialize their counter values to zero. However, users with a counter value equal to zero may only transmit in the next slot, if the feedback in the previous slot differed from *pu_start*, *pu_active*, *pu_end_cont*, and *success_cont* (except for the user who reserved the channel and who is clearly allowed to transmit after *pu_end_cont* or *success_cont* feedback).
2. Backlogged users decrease their counter only after receiving *idle*, *success_done*, or *pu_end* feedback.
3. Every active user increases his counter value by one if the feedback is either *pu_start* or *collision*.

Remark that any users that became active during a slot in which the channel was not reserved and the PU became active will immediately increase their counter value to one due to the *pu_start* feedback.

The above algorithm causes collisions among a large group of SUs after long periods of PU activity, leading to a lower SU channel utilization. To avoid these, an extra bit, called *flag*, is added to the feedback, a threshold T is added as a protocol parameter, and a new variable named *tracker* is introduced at the receivers side. This variable tracks the number of consecutive slots with *pu_start*, *pu_active*, *pu_end_cont*, or *success_cont* feedback. If the variable reaches the threshold T, the receiver resets its value to 0 and, in addition, sets the flag in the feedback (irrespective of the feedback value). Upon receiving feedback with the flag set, all active users increase their counter value by one with the exception of the SU who has reserved the channel (if there is such a SU). In this way, large groups of new arrivals are avoided by periodically grouping new arrivals and pushing them on the stack. An overview of the operation of the algorithm can be found in Figure 1.

3. Tree-structured QBD MC: A review

The class of tree-structured Quasi-Birth-Death Markov chains was introduced in Refs.[22, 28, 3] In this section we briefly review its main characteristics. Consider a discrete time bivariate Markov chain $\{(X_t, N_t), t \geq 0\}$, in which the values of X_t form a $(d+1)$-ary tree, and where N_t takes integer values between 1 and m. X_t is referred to as the node and N_t as the auxiliary variable of the Markov chain at time t. A

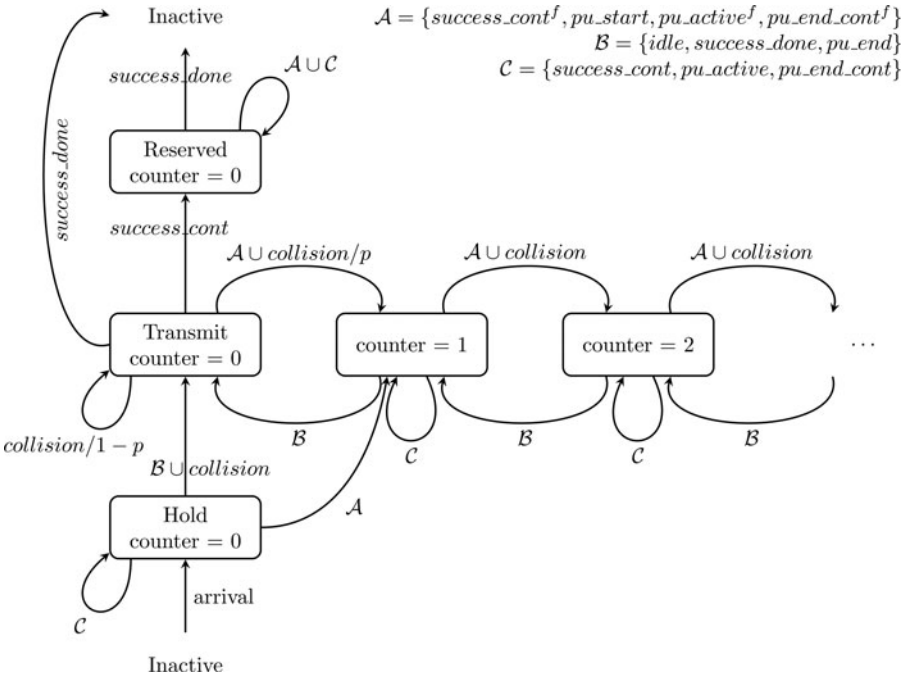


Figure 1. Flow chart showing the different states and transitions of the sender and its counter variable. The arc labels depict which type of feedback corresponds to which transition: when feedback is received that belongs to the depicted set, the corresponding transition is made (note for notation, no difference is made between a single element and its corresponding singleton). The transitions are fully defined given the current state and the received feedback, with 1 exception. If the sender is in the "Transmit" state and the type of feedback received is *collision*, the counter is increased by 1 w.p. p and remains 0 w.p. $1-p$. Feedback in which the flag bit is set, is marked by ^{f}. Senders are allowed to transmit only when in the "Transmit" state or in the "Reserved" state and if the feedback is different from *pu_start* and *pu_active*.

$(d+1)$-ary tree is a tree for which each node has exactly $d+1$ children. The root node is denoted as ϕ, the remaining nodes are denoted as strings of integers between 0 and d. Throughout the article we use lower-case letters to represent integers and upper-case letters to represent strings of integers when referring to the nodes of a tree. We use + to denote concatenation on the right. For example, if $J = j_1j_2j_3$, then $J + k = j_1j_2j_3k$.

The Markov chain $\{(X_t, N_t), t \geq 0\}$ is called a tree-structured QBD Markov chain if the following limitations apply to its transitions. First, if (X_t, N_t) is of the form $(J + k, i)$, then X_{t+1} must be either J, $J + s$ or $J + k + s$ for some $s \in \{0, \ldots, d\}$. In other words, from any node all transitions go either to the parent, a sibling, or a child node. Second, the transition probabilities should be independent of J (and of k in case of a transition to a child node), meaning they can be written in the form:

- $d_k^{i,j} = Pr[(X_{t+1}, N_{t+1}) = (J, j) | (X_t, N_t) = (J + k, i)]$,
- $a_{k,s}^{i,j} = Pr[(X_{t+1}, N_{t+1}) = (J + s, j) | (X_t, N_t) = (J + k, i)]$,
- $u_s^{i,j} = Pr[(X_{t+1}, N_{t+1}) = (J + k + s, j) | (X_t, N_t) = (J + k, i)]$.

Define the $m \times m$ matrices D_k, $A_{k,s,}$ and U_s such that entry (i, j) is equal to $d_k^{i,j}$, $a_{k,s}^{i,j}$ and $u_s^{i,j}$, respectively. Finally, when $(X_t, N_t) = (\phi, i)$ for some $i \in \{0, \ldots, m\}$, then $(X_{t+1}, N_{t+1}) = (s, j)$ with probability $u_s^{i,j}$ and $(X_{t+1}, N_{t+1}) = (\phi, j)$ with probability $f^{i,j}$, where the latter probabilities are grouped in an $m \times m$ matrix F.

The Markov chains constructed in this article do not require transitions among sibling nodes, as such $A_{k,s} = 0$ for all k and s. This implies that the positive recurrence of the Markov chain can be determined as follows. First, compute the matrix V as the limit of the following iterative method

$$V_{n+1} = \sum_{k=0}^{d} U_k (I - V_n)^{-1} D_k,$$

starting with $V_0 = 0$. Having obtained V, the matrices R_k, $0 \leq k \leq d$, are determined via

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

where entry (i, j) of R_k contains the expected number of visits to state $(J + k, j)$ given that we start in (J, i) before visiting node J again. The Markov chain (MC) is positive recurrent if and only if the spectral radius of $R = R_0 + \ldots + R_d$ is less than one, i.e., $sp(R) < 1$. Provided that the MC is positive recurrent, denote the steady state probability of state (J, i) as $\pi(J, i)$ and let $\pi(J) = (\pi(J, 0), \ldots, \pi(J, d))$, then

$$\pi(J) = \pi(\phi) R_{j_1} R_{j_2} \ldots R_{j_n},$$

for $J = j_1 j_2 \ldots j_n$. The boundary vector $\pi(\phi)$ can be obtained as

$$\pi(\phi) = \pi(\phi) \left(F + \sum_{k=0}^{d} R_k D_k \right),$$

such that $\pi(\phi)(I - R)^{-1} e = 1$, in which e is a column vector of m ones. Finally, for later use, we also introduce the matrices G_k, $0 \leq k \leq d$, where entry (i, j) of G_k corresponds to the probability that, given the MC starts in state $(J + k, i)$, the first visit to node J occurs in state (J, j). The matrices G_k can be expressed as $G_k = (I - V)^{-1} D_k$. Note, the matrices G_k are stochastic whenever the chain is positive recurrent.

4. Analytical model

In our model, new users arrive according to a Poisson process with rate λ. A user leaves the system as soon as he has successfully transmitted an entire packet (that may consist of multiple segments). In other words, there is a one-to-one correspondence between users and packets, and the packet delay is identical to the time that its associated user spends in the system. The probability of having x new arrivals over a time span of y slots is denoted by $b_x^{(y)} = e^{-\lambda y} (\lambda y)^x / x!$, and $b_{x+}^{(y)}$ is used to denote

the probability of having x arrivals or more. To ease the notation, we define $b_x^{(y)} = 0$ if $x < 0$.

Packet lengths are assumed to be distributed according to a discrete time phase type (DPH) distribution of order n with representation (γ, G). In other words, the probability that a packet is composed of k segments is given by $\gamma G^{k-1} g$, where $g = e - Ge$ holds the probabilities of absorption. We also assume that a packet is composed of at least one segment, i.e., $\gamma e = 1$. The state space of the corresponding discrete-time MC is denoted as $S_G \cup s_g$, with s_g the absorbing state.

The PU activity is modeled using a discrete-time MC with state space S_{pu} and transition matrix P_{pu}. The PU's state space is partitioned into two subsets S_i and S_a, the former containing the states in which the PU is considered idle, the latter in which the PU is considered active. Its transition matrix is likewise partitioned into submatrices containing the probabilities the PU stays or becomes (in)active:

$$P_{pu} = \begin{bmatrix} P_{ii} & P_{ia} \\ P_{ai} & P_{aa} \end{bmatrix}.$$

In order to introduce the tree-structured QBD MC that captures the behavior of the OBTA algorithm, we will use a stepwise approach: we start by briefly describing the MC used in Ref.[26] for the case where there is no PU activity and all packets have a length of one time slot; next, we consider variable packet lengths as well as the use of the *tracker* variable, and, finally, we add the PU activity.

4.1. *Fixed length packets and no PU activity*

If we assume that all the packets have a length of one time slot and there is no PU activity, the OBTA algorithm reduces to the basic binary TA described in Section 2.1, the evolution of which can be described by means of a stack. As such, we start by introducing the tree-structured QBD MC $\{(X_t, N_t), t \geq 0\}$ of Ref.[26] simplified to the case of Poisson arrivals. Under Poisson arrivals the system state of the basic binary TA is completely determined by the content of the stack, denoted as $s_x s_{x-1} \ldots s_2 s_1 s_0$ where the stack contains $x + 1$ groups and the ToS contains s_0 users. In order to obtain a tree-structured QBD MC, let N_t represent the number of users that will transmit in time slot t, i.e., it is s_0 plus the number of new arrivals in slot $t - 1$. The string $X_t = s_x s_{x-1} \ldots s_2 s_1$ represents the remainder of the stack, i.e., in time slot t there are s_i backlogged users with their counter value equal to i, for $i = 1, \ldots, x$. Note, s_x may be equal to zero if the bottom group of the stack contains no users.

Whenever a collision occurs, meaning $N_t > 1$, the users who transmitted are split into two groups and the counter value of the second group becomes one, while all the backlogged users increase their counter by one. In other words, the stack size increases by one and thus a transition occurs to a child node of X_t. The child node is determined by the number of users that selected the second group, while N_{t+1} is determined by the number of users that select the first group plus the number of

new arrivals that occurred in time slot t. Hence,

$$u_k^{i,i'} = Pr\left[(X_{t+1}, N_{t+1}) = (J+k, i') | (X_t, N_t) = (J, i)\right] = B(i,k) b^{(1)}_{i'-(i-k)},$$

for $i > 1$, where $B(i,k) = \binom{i}{k} p^{i-k}(1-p)^k$ is the probability that k of the i users select the second group.

If time slot t is idle or holds a success (and $X_t \neq \phi$), meaning $N_t = 0$ or 1, the backlogged users decrease their value by one, and N_{t+1} is determined by the number of users who had their counter value equal to one plus the number of new arrivals in slot t. Therefore, idle and successful slots cause a transition to the parent node of X_t and

$$d_k^{i,i'} = Pr[(X_{t+1}, N_{t+1}) = (J, i') | (X_t, N_t) = (J+k, i)] = b^{(1)}_{i'-k},$$

for $i = 0, 1$. When $X_t = \phi$ and slot t is idle or holds a success, N_{t+1} simply reflects the number of arrivals in slot t and $X_{t+1} = \phi$:

$$f^{i,i'} = Pr[(X_{t+1}, N_{t+1}) = (\phi, i') | (X_t, N_t) = (\phi, i)] = b^{(1)}_{i'},$$

for $i = 0, 1$.

The bivariate process $\{(X_t, N_t), t \geq 0\}$ is close to being a tree-structured QBD MC but requires nodes to have an infinite number of children and N_t to have an infinite range, as there is no upper bound on the number of users that can have the same counter value. To circumvent this issue, Ref.[26] introduced a model parameter d, such that at most d packets are transmitted simultaneously in a time slot, while the remaining packets are dropped. Note this immediately implies that there can be at most d users as well with the same counter value. Thus, by introducing d, the range Ω of N_t becomes $\{0, \ldots, d\}$, and each node has exactly $d+1$ children. Introducing this parameter, d may seem a little harsh, but its impact is nearly nonexistent if we pick d such that only a very small fraction ϵ of the packets is lost, e.g., $\epsilon = 10^{-15}$. In fact, after selecting a specific value of d we can solve the MC and determine the loss rate induced by d. Numerical experiments showed that small values of d, e.g., $d = 15$, suffice to get highly accurate results (unless we have long periods of channel reservation and/or PU activity combined with a large threshold T[†]). The parameter d causes the following modifications when $i' = d$:

$$\begin{aligned} u_k^{i,d} &= B(i,k) b^{(1)}_{(d-(i-k))+}, \\ d_k^{i,d} &= b^{(1)}_{(d-k)+}, \\ f^{i,d} &= b^{(1)}_{d+}. \end{aligned}$$

This truncation idea will also be used for the more advanced models, but we will skip the details for the sake of brevity.

[†] This can be understood by noting that during periods in which new arrivals are not allowed to transmit (either due to channel reservation and/or PU activity), the number of users in the hold state increases (see Figure 1). Thus, the longer these periods last, the higher the probability that we need to drop some packets. The threshold parameter introduces an upper bound on the length of these periods; therefore, smaller T values allow us to use smaller d values without causing much packet loss.

4.2. *Variable length packets and no PU activity*

We will now adapt the tree-structured QBD MC used to model the basic binary TA to incorporate variable packet lengths. In this case we will also model the impact of the tracker variable. In other words, we model the OBTA algorithm on a channel without PU activity. It may seem odd at first to use the tracker variable and threshold T in such a setting as its use was motivated by long periods of PU activity. However, its use can also improve the performance even without PU activity, provided that the packet length distribution is highly variable. In this case, very long packets will still result in large collisions, and these can be avoided using the tracker variable. If we set the threshold value $T = \infty$, the OBTA algorithm nearly coincides with the TA of Ref.[24] in the absence of PU activity, except that in Ref.[24] users are not able to distinguish between the successful transmission of an arbitrary segment and the last segment of a packet. As such, all the successful transmissions are followed by an idle slot that marks the end of a packet. To analyze the TA of Ref.[24], we can make use of the MC developed in this section if we simply add one more segment to each packet that corresponds to this idle slot (see Section 6 for more details).

In the absence of PU activity, the OBTA algorithm uses four feedback values (and the flag): *idle*, *collision*, *success_cont*, and *success_done*. One way to construct a tree-structured QBD MC is to adapt N_t such that it captures the number of users that transmit in slot t as well as the phase of the packet in case a single user is transmitting and the value of the tracker variable. As this leads to a significant increase in the size of the matrices characterizing the MC, this approach is not very appealing. Instead, we will no longer observe the system at each time slot.

If the threshold value $T = \infty$, we observe the system only at the time slots for which the feedback in the previous slot differed from *success_cont*. In other words, we do not observe the system during the transmission of the second to last segment of a successfully transmitted packet. In this case, the range Ω of N_t and X_t, where t is now the t-th observed slot, remains the same as do all the transitions, except for those from a state of the form $(J, 1)$, where the first segment of a packet is successfully transmitted. These transitions become

$$d_k^{1,i'} = Pr[(X_{t+1}, N_{t+1}) = (J, i')|(X_t, N_t) = (J + k, 1)] = \sum_{s=1}^{\infty} (\gamma G^{s-1} g) b_{i'-k}^{(s)},$$

as $\gamma G^{s-1} g$ is the probability that the packet consists of s segments. Note in this case we can also easily relax the assumption on the phase-type distributed packet length.

If we use a finite threshold value T, the above approach causes a problem. If we skip all the slots corresponding to the second to last segment of a packet, we may need to add several groups to X_t in a single transition from a state of the form (J, i), because the threshold value may be reached several times during the transmission of a single packet, and each time this occurs, we need to push a group to the stack. Transitions that add several items to X_t are, however, not allowed in a tree-structured QBD. To solve this problem, we skip only the slots corresponding to the

second to last segment of a packet for which the *flag* is not set (i.e., for which the threshold value is not reached). This implies that we now need to extend the range Ω of N_t to keep track of the phase of the packet whenever we observe the system in a slot where a segment is successfully transmitted. Hence, the range Ω of N_t becomes $\{0, 2, 3, \ldots, d\} \cup \{(1, j), j \in S_G\}$.

The transitions that only involve states of the form (J, i) with $i \neq 1$ remain identical to the case of the basic binary TA, but some changes are clearly needed when a state of the form $(J, (1, j))$ is involved. Let us first consider the case where we make a transition to a slot in which the first segment is successfully transmitted. These are identical to the binary basic TA except that we also initialize the phase of the packet length distribution, yielding for $i = 0, 2, \ldots, d$

$$\begin{aligned} u_k^{i,(1,j')} &= B(i,k) b_{1-(i-k)}^{(1)} \gamma_{j'}, \\ d_k^{i,(1,j')} &= b_{1-k}^{(1)} \gamma_{j'}, \\ f^{i,(1,j')} &= b_1^{(1)} \gamma_{j'}, \end{aligned}$$

as $\gamma_{j'}$ is the probability to start in phase j'. If we are in a state of the form $(J, (1, j))$ during the t-th observed slot and the $t+1$-st observed slot corresponds to a slot in which the flag is set, we get a transition to a child node as a group of new arrivals is pushed on the stack. Hence,

$$u_k^{(1,j),(1,j')} = (G^T)_{j,j'} b_k^{(T)},$$

as the pushed group consists of all the new packets that arrived during the T consecutive slots with *success_cont* feedback.

With probability $(G^z g)_j$ the remaining length of a packet, currently in phase j, is $z+1$ segments. Consequently, $\sum_{z=0}^{T-1} (G^z g)_j$ expresses the probability that the packet transmission ends before the threshold is reached (again), in which case a group is popped from the stack and the new arrivals that occurred during the $z+1$ remaining slots of the packet transmission are added to this group:

$$\begin{aligned} d_k^{(1,j),i'} &= \sum_{z=0}^{T-1} (G^z g)_j b_{i'-k}^{(z+1)}, \\ d_k^{(1,j),(1,j')} &= \sum_{z=0}^{T-1} (G^z g)_j b_{1-k}^{(z+1)} \gamma_{j'}, \end{aligned}$$

for $i' \neq 1$ with $f^{(1,j),(1,j')} = d_0^{(1,j),(1,j')}$ and $f^{(1,j),i'} = d_0^{(1,j),i'}$.

4.3. *Variable length packets with PU activity*

In the previous section we observed the system after the slots with *idle*, *collision*, or *success_done* feedback or when the flag bit was set in the feedback. When PU activity is introduced, we additionally observe after a slot with either *pu_start* or *pu_end* feedback (unless the channel is reserved). The system state N_t clearly needs

to be extended to incorporate the state of the PU. The range Ω of N_t now equals

$$\Omega = \underbrace{\{(0,s),(1,s),\ldots,(d,s)|s \in S_i\}}_{\Omega_{in}} \cup \underbrace{\{s|s \in S_a\}}_{\Omega_{ac}} \cup \underbrace{\{(1,j,s)|j \in S_G, s \in S_{pu}\}}_{\Omega_{res}},$$

where the first two sets Ω_{in} and Ω_{ac} correspond to slots without a channel reservation and without or with PU activity, respectively, while the last set Ω_{res} corresponds to a reserved channel *with* or *without* PU activity. Thus, when an SU reserves the channel and the packet transmission is interrupted by PU activity, the system makes a transition to a state of Ω_{res} with $s \in S_a$.

Note in the case of a single inactive PU state s and no active PU states, the range Ω of N_t contains one more additional state $(1, s)$ compared to the previous section. This state corresponds to a slot in which a single user is transmitting its first segment of a packet. In the case of no PU activity, we can remove this state by immediately initializing the phase of the packet and thus by letting N_t equal $(1, j, s)$ instead of $(1, s)$. However, in the presence of PU activity, this transmission may or may not fail depending on whether the PU becomes active. Hence, if we immediately initialize the phase of the packet length whenever a single user transmits, we would need two sets of states of the form $(1, j, s)$ in order to be able to distinguish between a reserved or unreserved channel, depending on whether it is the first segment of the packet that is being transmitted or not. Instead, we will postpone the initialization of the packet length until the first segment is successfully transmitted, allowing us to reduce the first set to a single state $(1, s)$.

4.3.1. *Transitions with $N_t \in \Omega_{in}$ and $N_{t+1} \notin \Omega_{res}$*

Given that the PU is inactive and remains inactive during a slot, the transitions corresponding to idle slots and collision slots remain nearly identical; we merely take the postponed initialization of the packet length distribution and the transition of the PU into account. This leads to

$$\begin{aligned} u_k^{(i,s),(i',s')} &= B(i,k) b_{i'-(i-k)}^{(1)} (P_{ii})_{s,s'}, \\ d_k^{(0,s),(i',s')} &= b_{i'-k}^{(1)} (P_{ii})_{s,s'}, \\ f^{(0,s),(i',s')} &= b_{i'}^{(1)} (P_{ii})_{s,s'}, \end{aligned}$$

for $i \geq 2$ and $s, s' \in S_i$. Furthermore, if the PU becomes active while the channel is not reserved, all active users increase their counter value by one:

$$u_k^{(i,s),s'} = b_{k-i}^{(1)} (P_{ia})_{s,s'},$$

for $i \geq 0$, $s \in S_i$ and $s' \in S_a$. The case where $N_t = (1, s)$ and $N_{t+1} \in \Omega_{in}$ will be covered at the end of this section.

4.3.2. *Transitions with $N_t \in \Omega_{ac}$ and $N_{t+1} \notin \Omega_{res}$*

When the PU is active and the channel is unreserved, we observe the system again if either the threshold is reached or the PU becomes idle again before reaching the threshold. In the former case, a new group solely existing of new arrivals is pushed

onto the stack, resulting in

$$u_k^{s,s'} = b_k^{(T)} \left(P_{aa}^T\right)_{s,s'},$$

with $s, s' \in S_a$. In the latter case, in which the PU becomes inactive before the threshold is reached (again), the accumulated arrivals are added to the ToS and all of them will transmit in the next slot. As the probability of reaching an idle state s' after $z+1$ steps from state s is given by $(P_{aa}^z P_{ai})_{s,s'}$, we have

$$d_k^{s,(i',s')} = \sum_{z=0}^{T-1} (P_{aa}^z P_{ai})_{s,s'} b_{i'-k}^{(z+1)},$$

for $i' \geq 0$, $s \in S_a$ and $s' \in S_i$.

4.3.3. *Remaining transitions*

We end this section by discussing the transitions where $N_t = (1, j, s)$, with $s \in S_{pu}$, and $N_t = (1, s)$, with $s \in S_i$, when the PU does *not* become active during the transmission of the first segment. In both cases the next observation point is either when the last segment of the packet is transmitted or when the flag bit is set. To compute the probabilities of the corresponding transitions, we define a substochastic matrix P_H that keeps track of both the PU and the packet length distribution during the channel reservation periods. The rows and columns of P_H correspond to the states $\{(j, s) | j \in S_G, s \in S_{pu}\}$. As P_H captures the evolution during a single time slot, each transition incorporates a transition of the PU and, if the PU was idle, a transition of the packet length distribution. Hence,

$$P_H = \begin{bmatrix} G \otimes P_{ii} & I_{|S_G|} \otimes P_{ia} \\ I_{|S_G|} \otimes P_{ai} & I_{|S_G|} \otimes P_{aa} \end{bmatrix}.$$

Further, define the matrix P_h as

$$P_h = \begin{bmatrix} g \otimes P_{ii} \\ 0 \end{bmatrix}.$$

Using these matrices, the probability of reaching the threshold when the channel is reserved, i.e., when N_t is of the form $(1, j, s)$, can be expressed as $(P_H^T)_{(j,s),(j',s')}$, $j, j' \in S_G$, $s, s' \in S_{pu}$, and the probability of finishing the packet transmission before the threshold is reached as $\sum_{z=0}^{T-1} (P_H^z P_h)_{(j,s),(g,s')}$, $s' \in S_i$. Hence, if the channel is reserved, we have

$$u_k^{(1,j,s),(1,j',s')} = b_k^{(T)} \left(P_H^T\right)_{(j,s),(j',s')},$$

$$d_k^{(1,j,s),(i',s')} = \sum_{z=0}^{T-1} \left(P_H^z P_h\right)_{(j,s),s'} b_{i'-k}^{(z+1)}.$$

Finally, if the channel was not reserved, i.e., when N_t is of the form $(1, s)$, and the PU does not become active during the transmission of the first segment, we also need

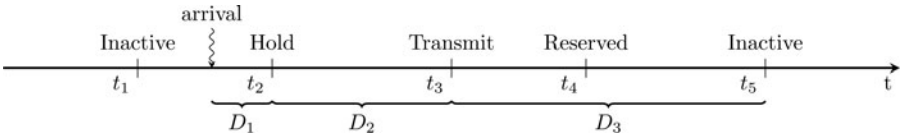


Figure 2. The three parts of the time that a tagged user spends in the system. t_1 is the time at which we observe the system prior to the arrival of the tagged user, t_2 is the time the tagged user exits the Hold state, t_3 is the first time that the tagged user enters the Transmit state, t_4 is the time at which the tagged user manages to reserve the channel by successfully transmitting his first segment, and at time t_5 the tagged user exits the system (i.e., successfully transmits his last segment).

to take care of the postponed packet length initialization. This yields

$$u_k^{(1,s),(1,j',s')} = b_k^{(T)} \left(\begin{bmatrix} \gamma G \otimes P_{ii} & 0 \end{bmatrix} P_H^{T-1} \right)_{s,(j',s')},$$

$$d_k^{(1,s),(i',s')} = ((\gamma \otimes I) P_h)_{s,s'} b_{i'-k}^{(1)} + \left(\begin{bmatrix} (\gamma G) \otimes P_{ii} & 0 \end{bmatrix} \sum_{z=0}^{T-2} P_H^z P_h \right)_{s,s'} b_{i'-k}^{(z+2)},$$

as γP_h is the probability that a packet consisting of a single segment is successfully transmitted on an unreserved channel and γG captures both the initialization and the first transition of the packet length distribution. We would like to remark that $f^{(1,j,s),(i',s')} = d_0^{(1,j,s),(i',s')}$ and $f^{(1,s),(i',s')} = d_0^{(1,s),(i',s')}$ as having no backlogged users can be regarded as having empty groups on the stack.

5. Performance measures

Once the matrices F, U_k, and D_k, $0 \leq k \leq d$ are constructed, determining the maximum stable throughput (MST) is not hard. We merely use the method described in Section 3 to check for system stability in combination with a simple root finding algorithm such as bisection to find the largest arrival rate λ for which $sp(R) < 1$. Calculating the mean delay, however, is far more complicated. Although the approach is somewhat similar to Ref.[26], the computation is considerably more involved as we do not observe the system at the end of each time slot, and some new users are also immediately pushed onto the stack due to the presence of the tracker variable.

To calculate the expected time a tagged user is active, we split this time into three parts (see Figure 2). The first part is the time between the actual arrival, and next observation point in the model. When referencing to Figure 1, the user spends this time in the *Hold* state. This corresponds to the time between the arrival and t_2 in Figure 2. The second part is the time that elapses between the sender leaving the *Hold* state and entering the *Transmit* state for the first time, that is, the time between t_2 and t_3 in Figure 2. The final part is the time between the sender's first transmission attempt and final transmission, i.e., the successful transmission of the last segment of the packet (i.e., the time between t_3 and t_5). The random variables denoting the length of these periods are denoted as D_1, D_2, and D_3.

Lemma 5.1. Let D_1 be the time between a tagged arrival and the next observation point, then

$$E[D_1] = \frac{1}{2}\frac{\sum_{x\in\Omega}\delta_x\sum_{z=1}^{T}P[L_x=z]z^2}{\sum_{x\in\Omega}\delta_x\sum_{z=1}^{T}P[L_x=z]z},$$

where the random variables L_x, for $x \in \Omega$, are defined as

$$P[L_x=z] = \begin{cases} 1 & x=(i,s), 0\le i\le d, i\neq 1, s\in S_i, \\ (P_{aa}^{z-1}P_{ai}e)_s & x=s, z=1,\dots,T-1, s\in S_a, \\ (P_{aa}^{T}e)_s + (P_{aa}^{T-1}P_{ai}e)_s & x=s, z=T, s\in S_a, \\ (\gamma g P_{ii}e)_s + (P_{ia}e)_s & x=(1,s), z=1, s\in S_i, \\ \left([\gamma G\otimes P_{ii}\ \ 0]\,P_H^{z-2}P_h e\right)_s & x=(1,s), z=2,\dots,T-1, s\in S_i, \\ \left([\gamma G\otimes P_{ii}\ \ 0]\left(P_H^{T-1}+P_H^{T-2}P_h\right)e\right)_s & x=(1,s), z=T, s\in S_i, \\ (P_H^{z-1}P_h e)_{(j,s)} & x=(1,j,s), z=1,\dots,T-1, j\in G, \\ & s\in S_{pu}, \\ (P_H^{T}e)_{(j,s)} & x=(1,j,s), z=T, j\in G, s\in S_{pu}, \end{cases} \quad (1)$$

and δ_x with $x \in \Omega$ is entry x of the row vector

$$\delta = \pi(\phi)(I-R)^{-1}.$$

Proof. Let L_x be the random variable representing the number of slots that occur between two consecutive points of observation, given that $N_t = x$ at the first point of observation. Its probability distribution can be readily deduced from the model and is given by (1).

To calculate the probability that the tagged user arrives between two consecutive observation points that are z slots apart, we note that due to the Poisson arrivals, the tagged user is z_1/z_2 times more likely to arrive in an interval of length z_1 than in an interval of length z_2. Therefore, we can express the required probability as $P[L_x = z]z/\sum_{z=1}^{T}(P[L_x = z]z)$ given that $N_t = x$ at the first of the two consecutive points of observation. Denote δ_x as the probability that $N_t = x$ at an arbitrary point of observation, and let δ be the row vector with entry x equal to δ_x, then due to the form of the steady state probabilities

$$\delta = \pi(\phi)(I-R)^{-1}.$$

Therefore, the probability that the tagged user arrives in a period of length z between to arbitrary consecutive points of observation can be expressed as

$$\frac{\sum_{x\in\Omega}\delta_x P[L_x=z]z}{\sum_{x\in\Omega}\delta_x\sum_{z=1}^{T}P[L_x=z]z},$$

and the result is established by noting that the position of the tagged used is uniformly distributed within this period due to the Poisson arrivals. □

For further use, define ξ_x and $\xi_x^{(2)}$, respectively, as the means and second moments of L_x and let ξ and $\xi^{(2)}$ denote the corresponding column vectors. In other words, $\xi_x = \sum_{z=1}^{T} zP[L_x = z]$ and $\xi_x^{(2)} = \sum_{z=1}^{T} z^2 P[L_x = z]$. Hence, due to the previous lemma, we have $E[D_1] = \frac{\delta\xi^{(2)}}{2\delta\xi}$.

Lemma 5.2. Let D_2 be the time that the tagged user spends in the system between exiting the Hold state and entering the Transmit state for the first time, then

$$E[D_2] = \Upsilon W \Phi_1,$$

where

$$\Upsilon = \frac{\pi(\phi)\bar{F} + \sum_{k=0}^{d} \delta R_k \bar{D}_k + \delta \sum_{k=0}^{d} \bar{U}_k}{(\pi(\phi)\bar{F} + \sum_{k=0}^{d} \delta R_k \bar{D}_k + \delta \sum_{k=0}^{d} \bar{U}_k)e}, \tag{2}$$

and the matrices $\bar{F}$, $\bar{U}_k$, and $\bar{D}_k$ are identical to F, U_k, and D_k defined in Section 4, except that $b_z^{(y)}$ and $b_{z+}^{(y)}$ are replaced by $ze^{-\lambda y}(\lambda y)^z/z!$ and $z(1 - \sum_{l=1}^{z-1} e^{-\lambda y}(\lambda y)^l/l!)$, respectively. Further, W is a diagonal matrix with

$$W_{x,x} = \begin{cases} 1, & x \in \Omega_{ac} \cup \Omega_{res}, \\ 0, & otherwise, \end{cases}$$

and Φ_1 is the unique solution of the following system of linear equations:

$$\Phi_1 = \xi + \sum_{s=0}^{d} U_s(\Phi_1 + G_s\Phi_1).$$

Proof. The time D_2 is the time the tagged user spends in the system between exiting the *Hold* state and his first transmission attempt. If the tagged user is in the *Hold* state and the feedback is *idle*, *collision*, *success_done*, or *pu_end*, he immediately transfers to the *Transmit* state, meaning $D_2 = 0$. However, if the received feedback is $success\_cont^f$, pu_start, pu_active^f, or $pu\_end\_cont^f$, the tagged user increases his counter to 1 and has to wait until his counter is 0 to start his first transmission attempt. It is possible to identify these two cases by looking at the system state when the tagged user exits the *Hold* state, i.e., at t_2 in Figure 2. Indeed, if at time t_2 the channel is reserved or the PU is active (or both), the tagged user does not send in the next slot but increases his counter to 1 and thus contributes to D_2. In all other cases (meaning the channel was idle, there was a collision, or the last segment of a packet was successfully transmitted in the slot the tagged user arrived), D_2 equals zero.

Next, we derive an expression for the probability that the system is in state $x \in \Omega$ when exiting the Hold state (i.e., at the first observation point after the tagged arrival, that is, at time t_2 in Figure 2), and we group these probabilities in a row vector Υ. To calculate Υ, we use a slightly modified version of the transition matrices of the model. The gist of the idea is that we look at the system state at an arbitrary point of observation, make a single transition, and note that the tagged user is z_1/z_2 times

more likely to be part of a transition during which z_1 new arrivals occur instead of z_2. Therefore, we define $\bar{F}$, $\bar{U}_k$, and $\bar{D}_k$ in the same manner as F, U_k, and D_k in Section 4, but with $b_z^{(y)}$ and $b_{z+}^{(y)}$ replaced by $ze^{-\lambda y}(\lambda y)^z/z!$ and $z(1-\sum_{l=1}^{z-1} e^{-\lambda y}(\lambda y)^l/l!)$, respectively. Thus, to determine Υ, we observe the system state at an arbitrary point of observation, make a single transition according to these *scaled* transition matrices, and renormalize the result. Clearly, the scaled transition matrix $\bar{F}$ is used for the transition from the root node ($X_t = \phi$), $\bar{D}_k$ applies only in a state for which the node variable is of the form $X_t = J + k$, while the $\bar{U}_k$ matrices apply to all states. The probability of being in the root node at an arbitrary point in time is given by $\pi(\phi)$ and the probability of being in a state for which X_t is of the form $J + k$ can be expressed as δR_k. Consequently, the row vector Υ can be computed as (2).

When $x \in \Omega_{ac} \cup \Omega_{res}$ at time t_2, the time D_2 corresponds to a fundamental period of the MC with initial state x. In the OBTA algorithm, the fundamental period is the time it takes to resolve the current event on the channel (PU activity, packet transmission, or collision) and allow for the next group to transmit (those with counter value 1). If the current event results in adding new groups to the stack in front of the designated group, the time needed to resolve the events corresponding to those groups is included as well. In the MC, this corresponds to the time needed to reach a state belonging to node J, given that we started in a state of the form $(J + k, x)$. We define the column vector Φ_1 such that entry x holds the expected length of the fundamental period with initial state $N_t = x$. To determine $(\Phi_1)_x$, we first count the expected number of slots needed until the next point of observation. Remark that this quantity was already discussed in Lemma 5.1 and is given by ξ_x. At this point with probability $(D_k e)_x$, a transition occurs to the parent node J, and we are done (note, $D_k e = D_0 e$ for all k). However, with probability $(U_s)_{x,y}$ a transition to state $(J + k + s, y)$ occurs, meaning we now first need to count the time needed to return to a state of the form $(J + k, z)$, and then we need to count the time needed to reach a state belonging to J. The former is given by $(\Phi_1)_y$, and the latter is obtained using $(G_s)_{y,z}$, which holds the probability that the chain is in state $(J + k, z)$ at the end of a fundamental period given that we started in state $(J + k + s, y)$, and $(\Phi_1)_z$. In matrix notation, this leads to the following formula:

$$\Phi_1 = \xi + \sum_{s=0}^{d} U_s(\Phi_1 + G_s\Phi_1).$$

Finally, to calculate $E[D_2]$, we define the diagonal matrix W

$$W_{x,x} = \begin{cases} 1, & x \in \Omega_{ac} \cup \Omega_{res}, \\ 0, & \text{otherwise}, \end{cases}$$

such that $E[D_2]$ can be expressed as $\Upsilon W \Phi_1$ as $D_2 = 0$ whenever $x \notin \Omega_{ac} \cup \Omega_{res}$ at time t_2 and is equal to $(\Phi_1)_x$ for $x \in \Omega_{ac} \cup \Omega_{res}$. □

Before we continue with the calculation of $E[D_3]$, we define

$$(N_k)_{x,y} = \begin{cases} 1, & x = (i,s), y = s', 0 \le i \le k, s \in S_i, s' \in S_a, \\ \frac{k}{i}, & x = (i,s), y = (i',s'), \max(2,k) \le i \le d, 0 \le i' \le d, s, s' \in S_i, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

and $(M_k)_{x,y} = 1 - (N_k)_{x,y}$, which we then use in the following lemma.

Lemma 5.3. Let D_3 be the time between the first transmission attempt of the tagged user and the successful transmission of his last segment, then

$$E[D_3] = \Psi\Phi_2,$$

with

$$\Psi = \Upsilon(I - W) + \frac{\delta \sum_{k=0}^{d} \bar{U}_k W G_k}{(\pi(\phi)\bar{F} + \sum_{k=0}^{d} \delta R_k \bar{D}_k + \sum_k \delta \bar{U}_k)e},$$

and

$$\Phi_2 = \xi + \sum_{k=0}^{d} [(M_k \circ U_k)\Phi_2 + (N_k \circ U_k)(\Phi_1 + G_k\Phi_2)],$$

where Υ, W, $\bar{F}$, $\bar{D}_k$, $\bar{U}_k$, ξ, N_k, and M_k are defined in or after Lemma 5.1 and $\circ$ denotes the Hadamard or entrywise matrix multiplication.

Proof. Define the column vector Φ_2 with entries $x \in \Omega_{res} \cup \Omega_{in} \backslash (0, s)$, $s \in S_{pu}$, as follows:

- If $x \in \Omega_{in} \backslash (0, s)$, then $(\Phi_2)_x$ represents the expected number of slots until the last segment of the tagged user is successfully transmitted, given that the tagged user attempts to transmit his first segment in the current slot, the phase of which is x.
- If $x \in \Omega_{res}$, $(\Phi_2)_x$ represents the expected number of slots until the last segment of the tagged user is successfully transmitted, given that the tagged user has reserved the channel and the phase of the current slot is x.

For notational purposes, the entries $(\Phi_2)_x$ are also introduced further on for $x \in \Omega_{ac} \cup (0, s)$, but these values are of no use.

After obtaining Φ_2 (see below), we determine the distribution of the phases $x \in \Omega$ in which the tagged user attempts his first transmission (at time t_3 in Figure 2) and let the row vector Ψ contain these values. Assuming we are able to determine both Φ_2 and Ψ, the expectation of D_3 can clearly be computed as

$$E[D_3] = \Psi\Phi_2.$$

In Lemma 5.2 we computed Υ, entry x of which holds the probability of being in phase x at time t_2. Further, as $D_2 = 0$ whenever $W_{x,x} = 0$, where W is defined in Lemma 5.2, the phase at time t_3 is also x when $W_{x,x} = 0$. Hence, $(\Upsilon(I - W))_x$ contains the probability that the system is in phase x at time t_3, and the tagged user is

immediately allowed to transmit when exiting the Hold state. However, in case the tagged user is not allowed to transmit after exiting the Hold state, ΥW clearly does not match the phase distribution at time t_3. To calculate the latter distribution, we note that as the tagged user is immediately pushed onto the stack when exiting the Hold state, a transition to a child node must take place. Hence,

$$\Upsilon W = \frac{\delta \sum_{k=0}^{d} \bar{U}_k W}{(\pi(\phi)\bar{F} + \sum_{k=0}^{d} \delta R_k \bar{D}_k + \delta \sum_{k=0}^{d} \bar{U}_k)e},$$

as $\bar{F}$ and $\bar{D}_k$ do not contain any transitions in which arrivals are pushed onto the stack. We also know that, as the tagged user is pushed onto the stack, the current state must be of the form $(J + k, x)$. Furthermore, the state in which the tagged user attempts his first transmission must be of the form (J, s), and we are interested in the probability distribution of s. Remark that entry (i, j) of the matrix G_k, which we introduced at the end of Section 3, contains the probability that the first visit to node J occurs in state (J, j), given that the MC starts in state $(J + k, i)$. Thus, the phase distribution at time t_3 can be calculated as

$$\Psi = \Upsilon(I - W) + \frac{\delta \sum_{k=0}^{d} \bar{U}_k W G_k}{(\pi(\phi)\bar{F} + \sum_{k=0}^{d} \delta R_k \bar{D}_k + \sum_{k=0}^{d} \bar{U}_k)e}.$$

The final component we still need to obtain is Φ_2. This computation proceeds in a similar but more involved manner as Φ_1. The mean time until the next point of observation is again given by ξ_x. With probability $(D_0 e)_x$ (again, $D_0 e = D_k e$ for all k), the last segment of the tagged user is successfully transmitted at the next point of observation, and we are done. Otherwise, with probability $(U_k)_{x,y}$ a transition to phase y of the k-th child of the current node occurs and the following cases have to be considered:

1. If $x = (1, j, s)$ or $(1, s)$ and $y = (1, j', s')$, we simply need to add $(\Phi_2)_y$ as the channel is still or became reserved by the tagged user.
2. If $x = (i, s)$ with $1 \leq i \leq k$ and $s \in S_i$ and $y = s'$ with $s' \in S_a$, the PU became active and we have to wait a fundamental period until the tagged user can transmit again, followed by $(\Phi_2)_z$ if the fundamental period ends in phase z.
3. If $x = (i, s)$ with $\max(2, k) \leq i \leq d$ and $s \in S_i$ and $y = (i', s')$ with $s' \in S_i$, then a collision occurred in the current slot. As there is a transition to the k-th child node, k of the i users chose the second group, meaning with probability k/i the tagged user selected the second group. If the tagged user selected the first group, the additional delay can be expressed as $(\Phi_2)_y$. Otherwise, the tagged user first has to wait a fundamental period, followed by $(\Phi_2)_z$ if the fundamental period ends in phase z.

Thus, if we define the matrix N_k as in (3) and let $(M_k)_{x,y} = 1 - (N_k)_{x,y}$, the matrices N_k and M_k clearly contain on position (x, y) the probability that after a transition

$(U_k)_{x,y}$ we need to wait a fundamental period before the user can resume transmitting or not. Hence, we have the following equation for Φ_2:

$$\Phi_2 = \xi + \sum_{k=0}^{d}[(M_k \circ U_k)\Phi_2 + (N_k \circ U_k)(\Phi_1 + G_k\Phi_2)],$$

where $\circ$ denotes the Hadamard or entrywise matrix multiplication. □

Proposition 5.1. *The mean time that a tagged user remains in the system, that is, the mean delay, can be expressed as*

$$E[D_1] + E[D_2] + E[D_3] = \frac{\delta\xi^{(2)}}{2\delta\xi} + \Upsilon W\Phi_1 + \Psi\Phi_2,$$

where δ, $\xi^{(2)}$, ξ, Υ, W, Φ_1, Ψ, *and* Φ_2 *are defined in (or after) Lemma 5.1 to 5.3.*

6. Numerical results

In this section we present various results for the mean packet delay and the MST. As the throughput is typically defined as the fraction of the channel that is used to successfully transmit data, the MST is defined as the mean packet length times the maximum arrival rate λ for which the packet delay is finite with probability one.

6.1. *Variable packet lengths and no PU activity*

Analyzing the MST of the original CTM algorithm with variable packet length has proved to be challenging. In Ref.[24], Tsybakov and Fedortsev derived an upper and lower bound on the MST. Using functional equations, Jacquet and Merle[11] analyzed the MST of a modified version of the algorithm that gave priority to new users after a successful transmission and enabled them to determine the MST (a less computationally expensive method for getting the MST for this variation was introduced in Ref.[19]). The OBTA algorithm operates identical to the original CTM algorithm in the absence of PU activity and with the threshold $T = \infty$, except that it uses four types of feedback instead of the usual three (being, idle, success, and collision). More specifically, OBTA uses both *success_cont* and *success_done* as feedback for successful segments, such that users know when the last segment was successfully transmitted, whereas with the original CTM algorithm, users cannot make this distinction. As a result, all successful transmissions are followed by an idle slot. Thus, to analyze the original CTM algorithm, we can rely on the MC model of Section 4.2 if we simply add an additional segment to each packet that represents the idle slot. This can be easily done by constructing a new DPH (γ', G') for the packet length:

$$\gamma' = \begin{bmatrix}\gamma & 0\end{bmatrix},$$

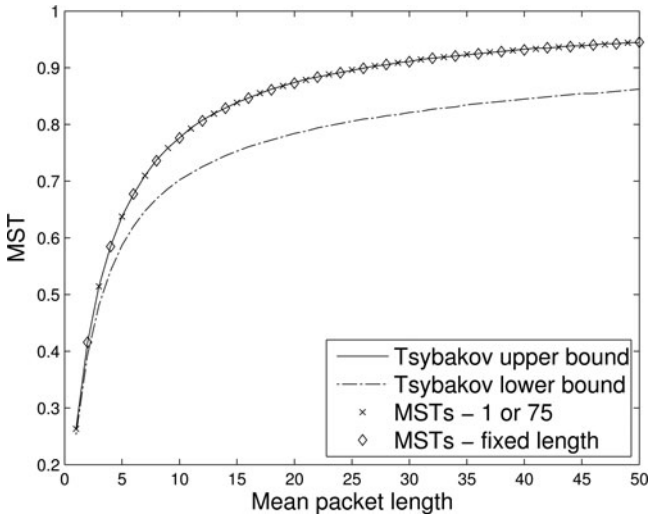


Figure 3. MST of the original CTM algorithm as a function of the mean packet length when compared to the lower and upper bound derived in Ref.[24].

and

$$G' = \begin{bmatrix} G & g \\ 0 & 0 \end{bmatrix}.$$

In this manner we calculated the MSTs for the original CTM algorithm with variable packet lengths and compared them to the bounds introduced in Ref.[24] The results for two different packet length distributions are shown in Figure 3: one with fixed packet lengths and one with packets of length 1 with probability c and length 75 with probability $1 - c$. The parameter c of the distribution is set such that the mean packet length matches a given mean, varying between 1 and 50. The upper and lower bounds depicted correspond to the case where all packets have either length 1 or 75. The upper bound for deterministic packet lengths is nearly identical to the one where all packets have length 1 or 75, while the lower bound is somewhat larger. The MST results were computed with $d = 40$ packets per slot, and the bisection algorithm was used to obtain a precision of 10^{-3}. Two important conclusions can be drawn from this figure: the MST seems quite insensitive to the shape of the distribution; the mean seems to matter the most. In fact, we also performed the same experiment with a uniform packet length distribution (with mean lengths between 1 and 50) and got a very similar MST (with an absolute difference in the order of 10^{-4}). Second, the upper bound in Ref.[24] and the MST results nearly coincide, where the maximum absolute difference found between the upper bound and the MSTs of the different distributions is in the order of 10^{-3}. From these results, it is even tempting to conjecture that the upper bound in Ref.[24] coincides with the MST.

The tree algorithm of Jacquet and Merle[11], which gives new users priority after successful transmissions, implicitly makes use of four types of feedback. As such, it is more fair to compare its performance with that of the OBTA algorithm, instead

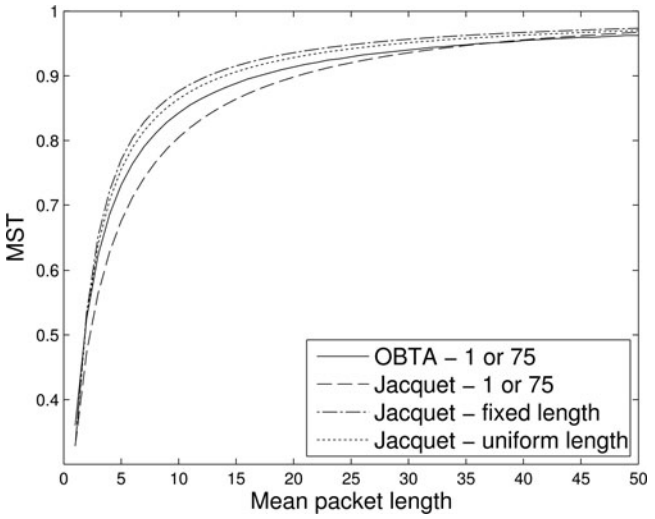


Figure 4. MST of OBTA with $T = \infty$ versus MST of the tree algorithm of Jacquet for different packet length distributions.

of with the original CTM algorithm. The MST results are depicted in Figure 4 for the same three packet length distributions considered before (i.e., fixed, uniform, or length 1 or 75). As the MST curves for the OBTA algorithm nearly coincide, only one is plotted. Figure 4 clearly indicates that the MST of Jacquet's algorithm depends not only on the mean of the distribution but also on the shape, while lower variance leads to a higher MST. The results show that Jacquet's algorithm achieves a higher MST than the OBTA algorithm provided that the variance of the packet length distribution is not too large. Note in the case that all packets have length 1 or 75, the variance is still fairly low if most of the packets have length 75, which is required to have a high mean. We should note that we are using a threshold value of $T = \infty$ for the OBTA algorithm, so some improvement in the MST of the OBTA algorithm can still be achieved by optimizing T as discussed in more detail in Section 6.2. When setting $T < \infty$, the MST does become more sensitive to the shape of the packet length distribution.

While the MST was nearly insensitive to the shape of the packet length distribution, the same does not hold for the mean packet delay, as can be seen in Figure 5. In this figure, the mean delay, normalized by the mean packet length, of OBTA (with $T = \infty$) and Jacquet's algorithms is plotted for $\lambda = 0.4$. The results show that the mean delay of both algorithms is very similar except for very small mean packet lengths (as the system is closer to instability). We also note that the normalized mean delay decreases as a function of the mean packet length, which is expected as the MST increases in such case. Finally, we observe higher delays for more variable packet length distributions, which is probably caused by the fact that short packets can get stuck behind longer packets.

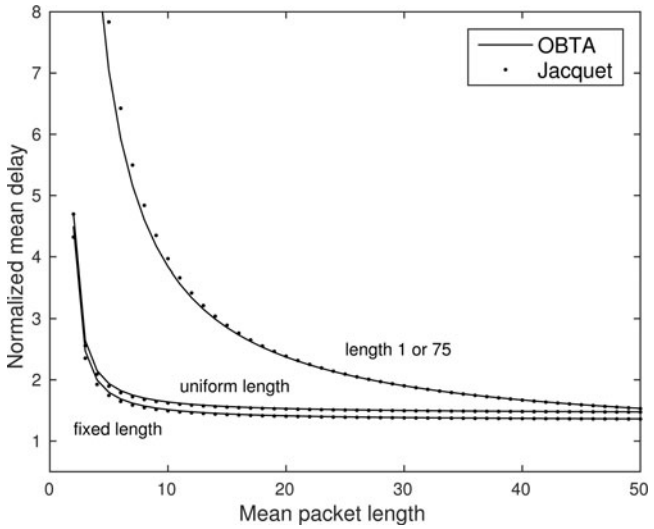


Figure 5. Delay comparison of the OBTA with the algorithm introduced by Jacquet for different packet length distributions. The delay was calculated for $\lambda = 0.4$.

6.2. *Optimal threshold value*

In this section we look at the impact of the threshold value T. As it seems natural that the optimal T is influenced by the mean packet length, we normalized the threshold value by the mean packet length. Figure 6 displays the MST as a function of the normalized threshold value for a fixed mean packet length of 20 and no PU activity. The packet lengths are either uniformly distributed or generated using a DPH that was constructed using the moment matching method given in Ref.[4], Appendix A], which matches the mean and SCV. The results indicate that the MST can be improved somewhat by optimizing T, while for non-optimal thresholds values the

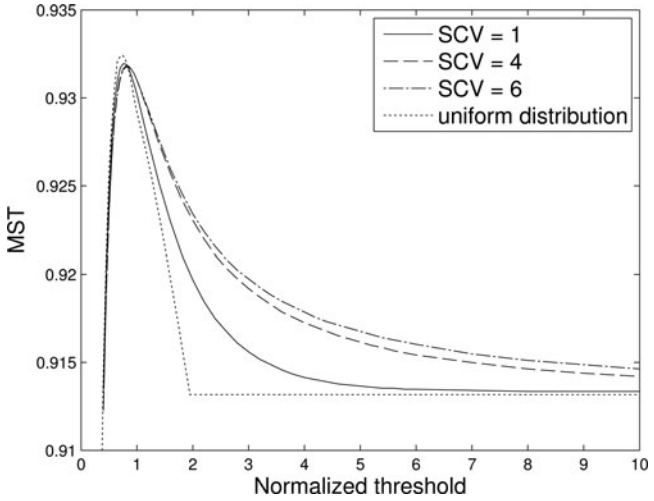


Figure 6. The MST versus the normalized threshold value in case of no PU activity for various packet length distributions with mean 20.

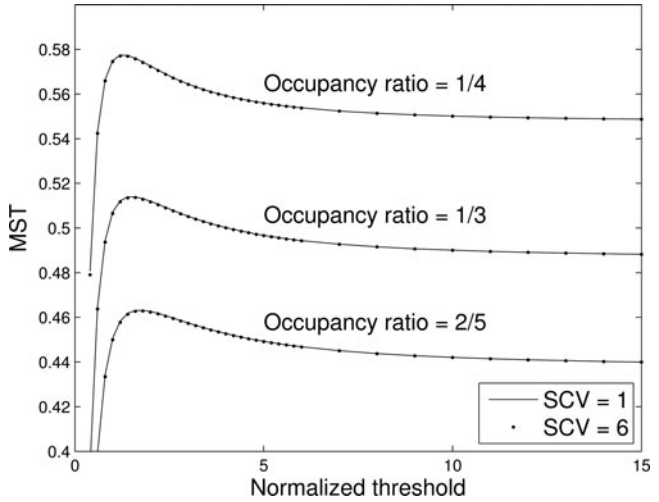


Figure 7. The MST versus the normalized threshold value, plotted for different occupancy ratios. The PU distributions consist of 2 moment matched DPH distributions with mean 400 for the active period and means 600, 800, and 1200 for the idle periods; the packet length distribution has mean 5 and SCV 2.

MST becomes quite dependent on the shape of the packet length distribution. For the uniform distribution and the moment matched distribution with SCV 1, the optimal relative threshold value is around 0.75; the value for the other distributions is closer to 0.8. Similar results were also observed when the mean packet length was set to 5. Furthermore, for the uniform distribution to have a mean of 20, the maximum packet length equals 39, hence any $T > 39$ results in the same MST. Note that by setting T equal to 0.8 times the mean packet length, the mean number of arrivals during an interval of length T is approximately 0.8 times the MST or somewhat below 1, meaning the mean size of the groups pushed on the stack due to the tracker reaching the value of T is somewhat below 1.

To examine the influence of the PU on the optimal threshold value, we repeated the above experiment for different PU models. Each PU model consisted of two DPHs, one for the idle period and one for the active period. The DPHs were constructed to match a given mean and SCV. The mean active time is set to 400, and the mean idle time is either 600, 800, or 1200 slots, respectively, which results in occupancy ratios of 2/5, 1/3, and 1/4 (the occupancy ratio is the fraction of time that the PUs are active). The SCV of the moment matched DPHs is either 1 or 6. The packet length distribution is a moment matched DPH with mean 5 and SCV 2. The results are shown in Figure 7 and indicate that the chosen SCVs for the idle and active period do not influence the MST much. On the other hand, the channel occupancy ratio does influence the optimal threshold value T. The observed optimal T values are approximately 1.2, 1.4, and 1.8 times the mean packet length and increase with the occupancy ratio. This is not unexpected as the mean number of arrivals during an interval of length T should be somewhat below 1, and this mean depends on the achieved MST divided by the mean packet length, where the former clearly decreases as the occupancy ratio increases.

6.3. *Benchmarking OBTA*

To get an idea of the performance of OBTA in the presence of PU activity, we compare the MST of OBTA in the presence of a PU, denoted as MST_{pu}, to the MST of the algorithm in the absence of a PU, denoted as MST_{no_pu}. To make the comparison fair, we look at the ratio between MST_{no_pu} and $MST_{pu}/(1 - occupancy\_ratio)$, as the channel can only be used by OBTA during periods of PU inactivity. We will refer to MST_{no_pu} as the benchmark value, as we cannot expect to see a substantially higher value for $MST_{pu}/(1 - occupancy\_ratio)$. As Section 6.2 suggests, the optimal threshold value T appears to be proportional to the occupancy ratio; we therefore divided the normalized threshold value by the occupancy ratio in case of PU activity. The normalized threshold value (before the division by the occupancy ratio) is 1 during the following experiments.

We may expect that OBTA performs poorly given that the inactivity periods are very short, as switching between PU active and inactive periods also makes some slots unusable for the SUs. Consider, for instance, the case where the PU activity is captured by

$$P_{pu} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix},$$

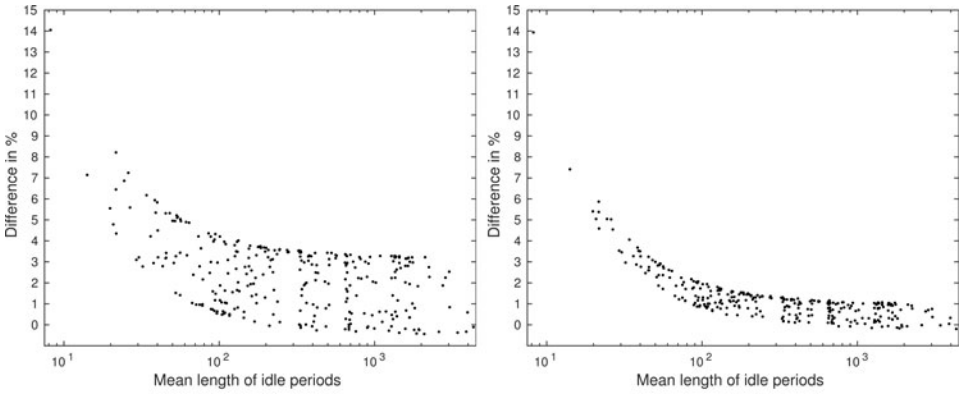
where $S_i = \{1\}$ and $S_a = \{2\}$. The corresponding system clearly has an occupancy ratio of 0.5, and each period of activity has a mean length of 2 slots. The MST of this system with a moment matched packet length distribution with mean 10 and SCV 1 and a normalized threshold value of 1 is 0.215. Hence, $MST_{pu}/(1 - occupancy\_ratio) = 0.430$, while the $MST_{no\_pu} = 0.869$.

Setting such artificial cases aside, we generated 300 random PUs matrices P_{pu} using the method discussed in Appendix A. The MST was calculated for each of the generated PU matrices in combination with a moment matched packet length distribution with means 10 and 40 and SCVs 1 and 6. The calculations were performed with a precision of 10^{-3}. For each MST, we calculated the ratio

$$\frac{MST_{no\_PU} - |MST_{PU}|}{|MST_{PU}|} \times 100,$$

where $|MST_{PU}| = MST_{pu}/(1 - occupancy\_ratio)$.

Figure 8a shows the ratio for a mean packet length of 10 and Figure 8b for a mean packet length of 40 slots. The SCVs for both packet length distributions is 1; the case with an SCV equal to 6 yields similar results. Both figures indicate that for short mean idle periods, the benchmark value is substantially higher than OBTA: there is a difference of 5–15% between the OBTA with the PU and the benchmark. However, the difference clearly decreases when the mean idle period length increases, especially when the mean packet length is 40. In this latter case we see that the benchmark performs approximately only 2% better (worst case). When the mean packet length is 10 slots, it performs approximately 5% better (worst case). We also note that in some cases OBTA even exceeds the benchmark by a fraction. For a mean



(a) Mean packet length 10. (b) Mean packet length 40.

Figure 8. Difference in percentage between the throughput of the OBTA with a PU and without a PU. The SCV of the packet length distribution is 1, and the normalized threshold value is 1/*occupancy_ratio*.

packet length of 10 slots, the biggest improvement when compared to the benchmark is approximately 0.43%; for a mean length of 40 slots, the maximum improvement found was 0.20%. This slight improvement may seem counterintuitive, but we believe this is merely due to the choice of the threshold T.

6.4. *Influence of activity-inactivity dependency*

In this section we examine the impact of correlation between the length of two consecutive PU activity/inactivity periods. The length of each period is captured by DPHs. The DPH for the idle period is denoted as (γ_i, P_{ii}) with $\gamma_i = [p_{i1}, p_{i2}]$ and $p_{ii} = e - P_{ii}e$; the DPH for the active periods is denoted as (γ_a, P_{aa}) with $\gamma_a = [p_{a1}, p_{a2}]$ and $p_{aa} = e - P_{aa}e$. The lengths have a hypergeometric distribution, and, in the default case, there is no correlation between consecutive periods. Each DPH consists of two phases, the first one representing a short period, and the second one representing a long period. Consequently, P_{ii} can be represented as

$$P_{ii} = \begin{bmatrix} q_{i1} & 0 \\ 0 & q_{i2} \end{bmatrix},$$

with $q_{i1} < q_{i2}$ and P_{aa} can be written down in a similar fashion using q_{a1} and q_{a2}. The absence of correlation between the idle and active periods means that when becoming active (idle), the active (idle) state is chosen according to γ_a (γ_i). Therefore, the resulting transition matrix can be denoted as

$$P_{pu,i} = \begin{bmatrix} P_{ii} & p_{ii}\gamma_a \\ p_{aa}\gamma_i & P_{aa} \end{bmatrix}.$$

To create dependency between consecutive periods, we define the transition matrix of the PU with dependency as

$$P_{pu,d} = \begin{bmatrix} q_{i1} & 0 & a(1-q_{i1}) & (1-a)(1-q_{i1}) \\ 0 & q_{i2} & (1-b)(1-q_{i2}) & b(1-q_{i2}) \\ c(1-q_{a1}) & (1-c)(1-q_{a1}) & q_{a1} & 0 \\ (1-d)(1-q_{a2}) & d(1-q_{a2}) & 0 & q_{a2} \end{bmatrix}.$$

Given that the first and third state represent the short periods, $a, c \in [0, 1]$ are the probabilities that if the PU switches from idle to active or vice versa, a short period follows a short period. Conversely, parameters $b, d \in [0, 1]$ represent the probabilities that a long period is followed by a long one.

Through varying the parameters a, b, c, and d, we can increase or decrease the correlation/anti-correlation between the lengths of consecutive periods. As we only wish to alter the correlation and not the distribution of the active/inactive PU periods, we need to keep the steady-state distribution of PU constant. This creates dependencies between the parameters a, b, c, and d:

$$(1-b) = \frac{\pi_3(1-q_{q1}) - a\pi_1(1-q_{i1})}{\pi_2(1-q_{i2})}, \tag{4}$$

and

$$(1-d) = \frac{\pi_1(1-q_{i1}) - c\pi_3(1-q_{a1})}{\pi_4(1-q_{a2})}. \tag{5}$$

As $b, d \in [0, 1]$, additional restrictions apply to a, c. These restrictions can be readily obtained from the equations above.

In the numerical experiments we considered two different PU settings:

1. The short idle periods consist of 10 slots on average and the long idle periods of 100; the mean active periods consist 8 and 80 slots, respectively. In the independent case, the probability of having a short active period is 0.65, and the probability of having a short idle period is 0.45. This setting corresponds to an occupancy ratio of 35.81%.
2. The mean short idle period is 15 slots and the long idle period 200. The mean length of the short active period is 20 slots, and the mean length of the long active period 120 slots. In case of no dependence, the probability of having a short active period is 0.75, and 0.35 for having a short idle period. This setting corresponds to an occupancy ratio of 24.97%.

The packet length distribution was a moment matched DPH with with mean 5 or 15 and SCV 2 or 4. The normalized threshold values considered were 1, 3, 50, and ∞. We examined the impact of different types of correlation:

1. No correlation: $a = 1 - b = p_{a1}$ and $c = 1 - d = p_{i1}$.
2. Negative correlation: $b = d = 0$.
3. Mixed correlation (type 1): $b = 1$ and $d = 0$.
4. Mixed correlation (type 2): $b = 0$ and $d = 1$.
5. Positive correlation: $b = d = 1$.

Table 1. Average and maximum change (%) of the MST when compared to the independent case.

	Positive		Mixed 1		Negative		Mixed 2	
T	avg	max	avg	max	avg	max	avg	max
1	− 0.01	− 0.01	0.00	0.00	0.00	0.00	0.00	0.01
3	− 0.01	− 0.01	0.00	0.01	0.00	0.01	0.00	0.00
50	0.07	0.18	0.00	− 0.01	− 0.01	− 0.01	0.00	− 0.01
∞	0.50	0.93	− 0.03	− 0.08	− 0.04	− 0.10	− 0.03	− 0.08

Table 2. Average and maximum change (%) of the mean delay when compared to the independent case.

	Positive		Mixed 1		Negative		Mixed 2	
T	avg	max	avg	max	avg	max	avg	max
1	127.6373	196.0642	7.6693	−14.2966	10.2258	−16.4929	6.7175	−14.1622
3	117.9688	188.0498	7.1740	−13.6413	9.6436	−15.9569	6.2183	−13.4140
50	120.2805	188.0430	7.3057	−13.8729	9.8313	−16.0820	6.3443	−13.4630
∞	114.6996	184.5054	7.1154	−13.7859	9.6705	−15.7072	6.1523	−13.0812

The above parameter values are such that the values of a and c correspond to the most extreme cases that meet the above restrictions.

The MST was determined with a precision of 10^{-4}. After the MST is obtained for a particular setting, we determined its relative increase using $(MST_d/MST_i - 1)*100$, where MST_i represents the MST in the independent case and MST_d represents the MST in the dependent case. The mean and maximum differences over the different packet length distributions and PU settings are shown in Table 1. The results indicate that the correlation has hardly any effect on the MST unless $T = \infty$. In the latter case the impact was still below 1% in all cases, and the most pronounced in the positive correlation case.

The same experiment was repeated for the mean delay when the throughput was set equal to 0.15, 0.25, 0.35, and 0.45. The results are shown in Table 2. As opposed to the MST, the correlations have a much more profound impact on the mean delay, as expected. Furthermore, for each of the threshold settings, we see the largest difference in the case of positive correlation, confirming the MST results above. The other types of correlation turn out to be beneficial for the mean delay.

Appendix

A: *Generation of the random PUs*

To generate the 300 random P_{pu} matrices used in Section 6.3, we proceeded as follows. The main objective here is to make sure that the periods of inactivity are not too short (as is often the case for a random transition matrix), while having an occupancy ratio between 5 and 60 percent. To construct P_{pu} we started by selecting one

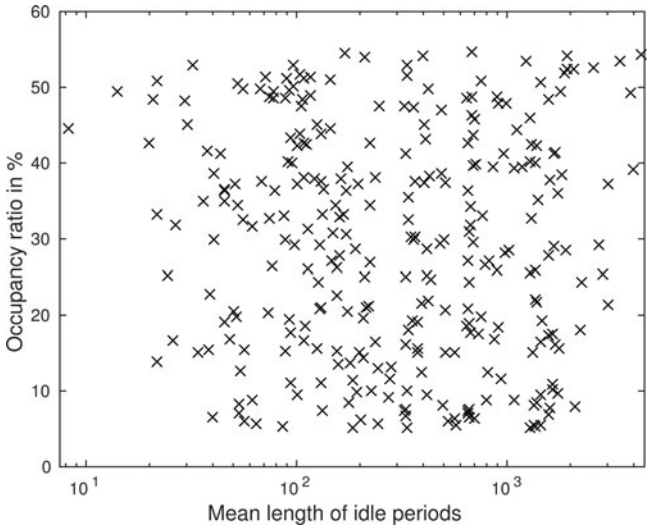


Figure A1. The mean length of the idle periods and the occupancy ratios of the generated P_{pu} matrices.

of the following four matrices

$$\begin{bmatrix} 100 & 100 & 1 & 1 \\ 100 & 100 & 1 & 1 \\ 1 & 1 & 100 & 100 \\ 1 & 1 & 100 & 100 \end{bmatrix}, \begin{bmatrix} 150 & 50 & 1 & 0 \\ 50 & 150 & 0 & 1 \\ 1 & 10 & 40 & 50 \\ 1 & 1 & 0 & 50 \end{bmatrix}, \begin{bmatrix} 50 & 20 & 1 \\ 100 & 1 & 1 \\ 1 & 20 & 100 \end{bmatrix} \text{ and } \begin{bmatrix} 20 & 1 \\ 1 & 15 \end{bmatrix}.$$

Next, we obtained P_{pu} by multiplying each entry in this matrix with a random number and renormalizing such that the row sums are equal to one. The corresponding idle state sets are, respectively, {1}, {1, 2}, and {1, 2} for the matrices of size 2, 3, and 4. Finally, the matrix P_{pu} was accepted provided that the occupancy ratio was less than 5% off a predefined target value, while the mean length of an inactive period had to exceed some threshold. The thresholds considered were 5, 20, 80, 320, 640, and 1280 slots, and the target occupancy ratios were 0.1, 0.2, 0.3, 0.4, and 0.5. For each threshold and target value, we generated 10 matrices. Figure A1 shows the resulting occupancy ratio and mean idle period of the generated matrices.

References

[1] Akyildiz, I.F.; Lee, W.-Y.; Vuran, M.C.; Mohanty, S. Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *Comput. Netw.* **2006**, *50*, 2127–2159.

[2] Bertsekas, D.; Gallager, R. *Data Networks*. Prentice-Hall Int., Inc.: Upper Saddle River, NJ, 1992.

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

[4] Boute, R.N.; Lambrecht, M.R.; Van Houdt, B. Performance evaluation of a production/inventory system with periodic review and endogenous lead times. *Naval Research Logistics* **2007**, *54*, 462–473.

[5] Buddhikot, M.M.; Kolodzy, P.; Miller, S.; Ryan, K.; Evans, J. DIMSUMNet: New directions in wireless networking using coordinated dynamic spectrum access. In: *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*. June 2005, pp. 78–85. DOI: 10.1109/WOWMOM.2005.36. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1443488.

[6] Capetanakis, J.I. Tree algorithms for packet broadcast channels. *IEEE Trans. Inform. Theory* **1979**, *25*, 319–329.

[7] Cordeiro, C.; Challapali, K.; Birru, D. IEEE 802.22: the first worldwide wireless standard based on cognitive radios. In: *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*, 2005. *DySPAN 2005*. pp. 328–337. DOI: 10.1109/DYSPAN.2005.1542649. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1542649.

[8] Cormio, C.; Chowdhury, K.R. A survey on MAC protocols for cognitive radio networks. *Ad Hoc Netw.* **2009**, *7*, 1315–1329.

[9] De Domenico, A.; Calvanese, E.S.; Di Benedetto, M.G. A survey on MAC strategies for cognitive radio networks. *IEEE Comm. Surveys Tutorials* **2010**, *14*, 21–44.

[10] Ephremides, A.; Hajek, B. Information theory and communication networks: An unconsummated union. *IEEE Trans. Inform. Theory* **1998**, *44*, 2416–2434.

[11] Jacquet, P.; Merle, E. Analysis of a stack algorithm for CSMA-CD random length packet communication. *IEEE Trans. Info. Theory* **1990**, *36*, 420–426.

[12] Jia, J.; Zhang, Q.; Shen, X. HC-MAC: A hardware-constrained cognitive MAC for efficient spectrum management. *IEEE J. Selected Areas Comm.* **2008**, *26*, 106–117. DOI: 10.1109/JSAC.2008.080110. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4413144.

[13] Liang, R.M.; Tan, H.H. On the error analysis of single-channel free-access collision resolution algorithms. *IEEE Aerospace Conference Proceedings, Vol. 1*. Big Sky, MT, **2000**, 129–140.

[14] Lien, S.-Y.; Tseng, C.-C.; Chen, K.-C. Carrier sensing based multiple access protocols for cognitive radio networks. In: *2008 IEEE International Conference on Communications*, 2008, pp. 3208–3214. DOI: 10.1109/ICC.2008.604. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4533640.

[15] Likhanov, N.B.; Plotnik, I.; Shavitt, Y.; Sidi, M.; Tsybakov, B.S. Random access algorithms with multiple reception capabilities and n-ary feedback channel. *Problemy Peredachi Informatsii* **1993**, *29*, 82–91.

[16] Massey, J.L. Collision resolution algorithms and random-access communication. *Multi-User Communication Networks*. Longo, G., Ed.; CISM Courses and Lectures No. 256. Wien-New York: Springer Verlag, 1981, 73–137.

[17] Mathys, P.; Flajolet, P. Q-ary collision resolution algorithms in random-access systems with free or blocked channel access. *IEEE Trans. Inform. Theory* **1985**, *31*, 217–243.

[18] Molle, M.L.; Polyzos, G.C. *Conflict Resolution Algorithms and Their Performance Analysis*. Tech. rep. University of Toronto, CS93-300, 1993.

[19] Peeters, G.T.; Van Houdt, B. On the maximum stable throughput of tree algorithms with free access. *IEEE Trans. Inf. Theory* **2009**, *55*, 5087–5099.

[20] Peeters, G.T.; Van Houdt, B.; Blondia, C. A multiaccess tree algorithm with free access, interference cancellation and single signal memory requirements. *Perf. Eval.* **2007**, *64*, 1041–1052.

[21] Sidi, M.; Cidon, I. Splitting protocols in presence of capture. *IEEE Trans. Inf. Theory* **1985**, *31*, 295–301.

[22] Takine, T.; Sengupta, B.; Yeung, R.W. A Generalization of the matrix M/G/1 paradigm for Markov chains with a tree structure. *Stoch. Models* **1995**, *11*, 411–421.

[23] Towsley, D.; Vales, P. Announced arrival random access protocols. *IEEE Trans. Comm.* **1987**, *35*, 513–521.
[24] Tsybakov, B.S.; Fedortsov, S.P. Local-area network with random-multiple-access communications. *Problemy Peredachi Informatsii* **1986**, *22*, 49–58.
[25] Tsybakov, B.S.; Mikhailov, V.A. Free synchronous packet access in a broadcast channel with feedback. *Problemy Peredachi Inform* **1978**, *14*, 32–59.
[26] Van Houdt, B.; Blondia, C. Stability and Performance of stack algorithms for random access communication modeled as a tree structured QBD Markov chain. *Stoch. Models* **2001**, *17*, 247–270.
[27] Vvedenskaya, N.D.; Tsybakov, B.S. Random multiple access of packets to a channel with errors. *Problemy Peredachi Informatsii* **1983**, *19*, 69–84.
[28] Yeung, R.W.; Alfa, A.S. The quasi-birth-death type Markov chain with a tree structure. *Stoch. Models* **1999**, *15*, 639–659.