A multiaccess tree algorithm with free access, interference cancellation and single signal memory requirements

G. T. Peeters, B. Van Houdt, C. Blondia

University of Antwerp, Department of Mathematics and Computer Science, Performance Analysis of Telecommunication Systems Research Group - IBBT, Middelheimlaan 1, B-2020 Antwerp, Belgium

Abstract

Tree algorithms are a well studied class of collision resolution algorithms for solving multiple access control problems. Successive interference cancellation, which allows one to recover additional information from otherwise lost collision signals, has recently been combined with tree algorithms with blocked access (Yu and Giannakis, Infocom 2005, pp. 1908-1916), providing a substantially higher maximum stable throughput (MST): 0.693 for Poisson arrivals, given an infinite number of memory locations for storing signals. We propose a novel tree algorithm for a similar problem, but with two relaxed model assumptions: free access is supported and a single signal memory location suffices. A study of the maximal stable throughput of this algorithm is provided using matrix analytic methods; as a result, an MST of 0.5698 for Poisson arrivals is achieved. Our methodology also allows us to investigate the MST when the multiple access channel is subject to Markovian arrival processes.

Key words: multiple access, tree algorithm, successive interference cancellation

1 Introduction

Multiple access channels have been used as key components in the design of various access network technologies. For instance, random access schemes are used to share the available bandwidth in 802.11 networks as well as in 10 and 100Mbit Ethernet systems (in combination with carrier-sense and/or collision-detection mechanisms). In point-to-multipoint access networks, such as hybrid-fiber-coaxial (HFC) networks (i.e., DOCSIS networks) and DVB-RCS satellite networks, random access channels are supported such that end-users can specify their uplink bandwidth requirements to the network via fixed length control messages in a multiple access manner. Although all these random access channels rely on the well known binary exponential backoff (BEB) algorithm (or a simple ALOHA scheme), tree algorithms have been recognized as important (if not, superior) contenders during the development of the 802.14 standard [6,5] for HFC networks (however, the 802.14 standardization process was prematurely terminated by the introduction of the DOCSIS standard).

Tree algorithms also strongly outperform the class of backoff algorithms (including the BEB) in terms of their maximum stable throughput (MST) [1]. In the standard information theoretical setting, the MST is defined as the highest possible (Poisson) input rate for which a packet has a finite delay with probability one. The first tree algorithms were independently developed in the late 1970s by Capetanakis [3] and Tsybakov, Mikhailov and Vvedenskaya [12]. As such one often refers to this first class of tree algorithms as the CTMV algorithms. These algorithms were the first to have a provable MST above zero. Afterwards new tree algorithms were developed with MSTs as high as 0.4878 using the standard information theoretical multiple access model [1,4,11].

A random access protocol consists of two components: the channel access protocol (CAP) and the collision resolution algorithm (CRA). The CAP specifies the rules that users need to follow when transmitting a new packet for the first time. The CRA informs the users about the algorithm used to resolve collisions (i.e., simultaneous transmissions). The easiest CAP is *free access*, meaning new packets may be transmitted without further delay. Other important CAPs include *blocked* (or gated) and *windowed* (or grouped) access. In both these cases (some of) the ongoing conflicts on the channel need to be resolved before new packets are allowed to access the channel, implying that some channel monitoring is required even when users are inactive. All known algorithms with an MST close to 0.4878 require some form of channel monitoring, while the highest known MST for a free access algorithm is only 0.4076.

Meanwhile, the 0.4878 MST realized under the standard information theoretical model, has been exceeded in various manners by introducing additional mechanisms not available under the standard model, such as energy measurement techniques to determine the collision multiplicity [8] and an additional control field/bit with separate feedback [7]. Recently, the SICTA algorithm which uses successive interference cancellation (SIC) mechanisms, was designed and was shown to achieve an MST as high as 0.693 [16]. SICTA uses a blocked access CAP and requires a (theoretically) unbounded amount of memory for storing signals.

In this paper we introduce a novel tree algorithm using SIC for the more difficult free access CAP. Moreover, our algorithm stores at most one signal at a time, achieving minimal memory requirements. To determine the MST of our algorithm, we rely on matrix analytic methods. More specifically, as in [13,14] for the standard CTMV algorithms, we develop a tree-like process [2] such that this process is ergodic if and only if the algorithm is stable under the specified arrival process. Using a bisection algorithm, we subsequently determine the MST for various Markovian arrival processes (and not just the Poisson process for which an MST of 0.5698 is realized). Using the approach taken in [13], the novel algorithm does not naturally result in a tree-like process and some additional steps are required (as in [14] for some of the more advanced CTMV members). The originality of the modeling approach lies exactly in these steps as a more efficient approach is taken that does not require us to expand the state space (as was done in [14]).

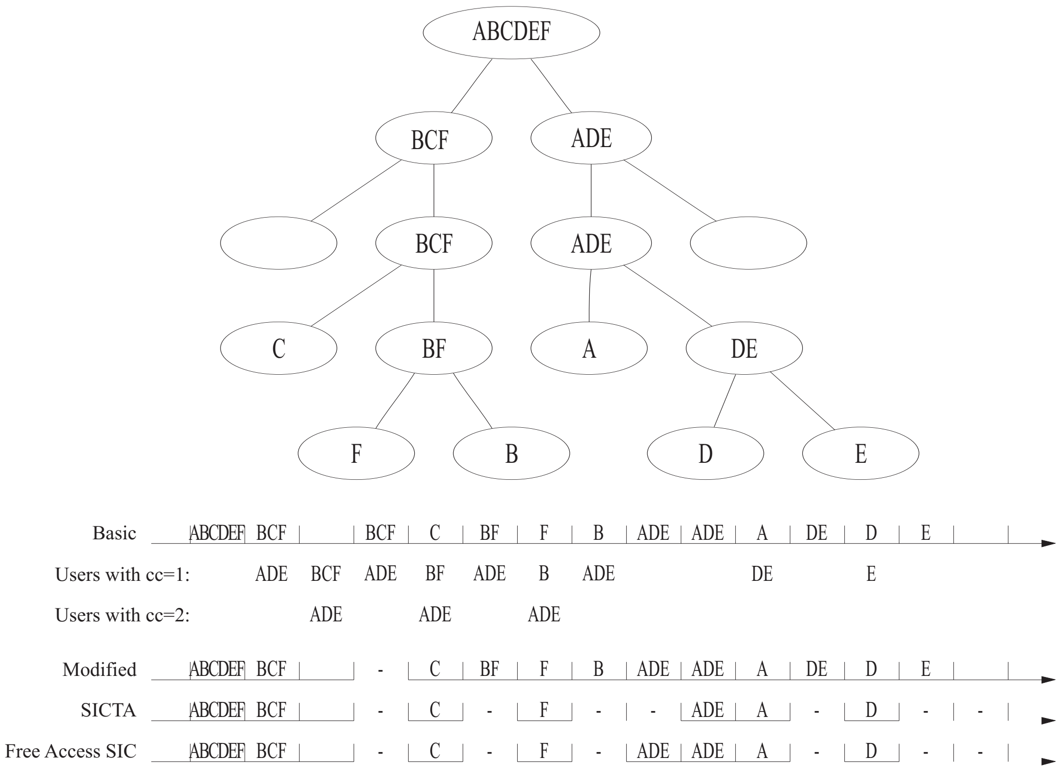


Fig. 1. *Illustration of the basic tree algorithm, where the collision of six users A, B, C, D, E and F transmitting simultaneously in the first slot, is resolved. This process is often visualized using a tree, providing an overview of the splitting decisions; a depth-first, left-to-right traversal of this tree corresponds with the transmission sequence. The modified, SICTA and our proposed free access algorithm allow us to skip some slots, as indicated with a -. For our proposed free access solution, the tree can be considered as a free access scenario where no other arrivals occur during the first 9 or more slots.*

Tree algorithms form a class of CRAs. The idea behind the basic binary tree algorithm is that each time a collision occurs, the colliding terminals split themselves into two groups using random numbers: a terminal chooses the first group with probability p and the second group with probability $1 - p$, which may be regarded as flipping a (biased) coin. Typically, fair coins are considered, i.e., $p = 0.5$; however, in some cases biased coins can provide a slightly higher MST. In the next slot, the terminals of the first set retransmit, while the second set has to defer transmission until the first set is completely resolved. In case more than one terminal selected the first subset, the next slot holds another collision and the first subset splits again into two groups. This recursive procedure is repeated until all conflicts are solved, after which new packets may be transmitted in case of blocked access. Note, for free access, new arrivals may occur during this process and thus extend the resolution period. The operation of the basic binary CTMV algorithm is illustrated by Figure 1. Here, we see that it suffices for each terminal to keep track of a single counter that indicates the number of sets that remain to be resolved before it may retransmit. Users whose counter equals zero transmit, other users in- or decrease their counter by one, depending on whether there is a collision or not.

In order to improve the MST, the basic tree algorithm can be slightly modified. This modification, also proposed by Massey [9], consists of skipping *doomed* slots. These doomed slots occur when a conflict is followed by an empty slot, in which case the following slot will certainly contain another collision. Successive interference cancellation allows us to skip even more slots, which further improves the throughput, as demonstrated by SICTA [16]. When two signals A and B occur in the same slot, interference cancellation only requires the retransmission of signal A in order to recover signal B from the joint signal (where signal A and B each consist of one or multiple messages). Due to the blocked access CAP, SICTA allows us to skip every right child in the contention tree, since the "subtraction" of the signal of its sibling from the previous conflict (its parent) provides its contents anyway. Free access prevents us from assuming that every second subset can be skipped, thus requiring a very different approach.

We start this paper by discussing all the multiple access model assumptions, which characterize a free access medium, capable of successive interference cancellation with limited signal memory. Based on this model, we continue the description of the novel random access algorithm, which is analyzed using matrix analytical methods in the following section. Finally, we provide numerical results, indicating the MST for a variety of Markovian arrival processes.

2 Slotted Multiaccess Model Assumptions

The proposed algorithm and its analysis are based on the following assumptions, describing a multiaccess model with interference cancellation. The first series of assumptions ($S1$ to $S4$) are standard assumptions that have been used by a multitude of authors (for a detailed treatment of these assumptions we refer to [4,11,1]).

$S1$. *Slotted system*: the channel is divided in fixed length time slots; each user is allowed only to start transmitting at the beginning of a time slot; all packets have the same length equal to one time slot.
$S2$. *Error-free reception* by the receiver: a slot is either received as an idle, success or a collision slot, depending on whether zero, one or more packets are transmitted.
$S3$. *Infinite population*: there is an infinite set of users, generating packets that are assumed to be unique. Notice, infinite populations provide a pessimistic estimation for finite systems by considering each packet as a virtual station.
$S4$. *Immediate, error-free feedback*: at the end of each slot, the receiver is assumed to provide immediately feedback to the transmitters.

Note that we do not restrict ourselves to default Poisson (or Bernoulli) arrivals, as our analysis technique will allow us to study any discrete-time batch Markovian arrival process (D-BMAP)[13].

The assumptions ($I1$ to $I3$) specific to the interference cancellation mechanism follow now. Consider two signals a and b, where b contains the combination of signals $B_1, \ldots, B_n$. We denote $a - b$ as the interference cancellation operation, which only results in a valid signal if a consists of $B_1, \ldots, B_n, A_1, \ldots, A_m$, and has $A_1, \ldots, A_m$ as a result.

$I1$. Analogue to Yu and Giannakis [16], interference cancellation, operating on two signals a and b allows the receiver to distinguish between the following four scenarios:
(1) $a - b$ contains one signal. In this case, $a - b$ contains the transmitted signal of one terminal.
(2) The reverse case of scenario 1: $b - a$ contains one signal, which is also successfully received.
(3) a is identical to b, in which case no additional transmissions can be decoded.
(4) Neither of the other scenarios applies.
$I2$. The receiver is able to store a single signal ss, which is either a received signal, or the result of an interference cancellation operation $a - b$ (provided that this contains a meaningful signal). Note that this condition is far less restrictive than the assumption made in [16], where an (in principle) infinite amount of memory is required to store intermediate signals.
$I3$. Each message is accompanied with a single bit, indicating whether the message is transmitted for the first time. In case of a successful slot, this allows the receiver to distinguish between a new transmission and a retransmission. The field carrying this bit is not used by the interference cancellation operation; otherwise, a retransmitted signal would differ from the first trans-

mission attempt.

A similar control bit/field (or mini-slot) as in assumption $I3$ was proposed in [7]— resulting in six possible slot outcomes—to realize a throughput above the celebrated 0.4878 FCFS algorithm [1].

Finally, we will consider a random access algorithm with *free* access, meaning terminals that generate a new packet are allowed to access the channel immediately. This is important to emphasize, because although a windowed access 0.4878 algorithm has been devised under the standard assumptions $S1$ to $S4$ (and Poisson arrivals), the highest achieved throughput under free access is, to the best of our knowledge, only 0.4076 [11].

3 Algorithm

The basic approach of our proposed algorithm consists of performing a standard tree algorithm. As in SICTA, some slots are skipped if enough information can be derived from the previous slots and the signal memory. Determining slots that can be skipped is straightforward in the case of SICTA, since every right child is retrieved by subtracting the signal of its (left) sibling from their common parent. With free access however, this does not hold in general, since new arrivals may render the subtracted signal meaningless. As such, we will skip a right slot only if the interference cancellation guarantees sufficient knowledge about the contents of this slot.

A second difference with SICTA occurs when the resolution of one conflict requires us to solve another conflict first. Here, the limited memory prevents us from storing both conflict signals. Therefore, we only skip a right child of a collision slot, if we are certain of its content and its left sibling is successful or idle. In Figure 1, this is why the first ADE slot cannot be skipped, as opposed to SICTA.

In the original CTMV algorithm [3,12], it suffices for each user to store a single counter $cc \geq 0$. This integer counter represents the number of sets that still require resolution until he is allowed to retransmit. However, the use of the control bit/field in our algorithm also requires the use of a single flag $first$ per user, indicating whether the current message has been transmitted once, or more than once.

The individual counters cc are updated on the receipt of channel feedback, provided by a receiver. In our algorithm, this feedback can attain five possible values: $\cdot/\cdot/\cdot$, $\cdot/Co/\cdot$, $\cdot/\cdot/Sc$, $Sr/Co/\cdot$ and $Sr/\cdot/\cdot$. The mnemonics here will further on be associated with "skip right" (Sr), "collision" (Co) and "skip collision" (Sc).

In a sense, this feedback can be related with the binary and ternary feedback of the basic and modified binary CTMV algorithm. In case of the basic CTMV, the feedback generated by the receiver indicates whether there was a collision. Thus, the feedback would either be $\cdot/Co/\cdot$ or $\cdot/\cdot/\cdot$: collision or no collision (note, an idle slot is not considered a collision).

The modified CTMV algorithm improves the basic protocol, but requires an additional feedback value to indicate the occurrence of an idle slot. An idle slot appearing immediately after a collision indicates that the next slot is guaranteed to hold a collision, meaning, it can be skipped. In the modified CTMV algorithm the terminal decides to skip slots based on the feedback value *idle*. Alternatively, our proposed feedback $\cdot/\cdot/Sc$ would move this decision to the receiver side.

The receiver side

The receiver side behavior can be summarized in ten rules ($R1$ to $R10$), which will provide the appropriate feedback to the users, based on the content of the current slot cs and a single memory location ss, potentially holding a saved signal. In this process, the receiver will store at most one signal at a time, being in ss, to serve as a reference for interference cancellation. Notice, the control bit/field is only used to distinguish between rules $R7$ and $R8$. If the current node is a left branch of the resolution tree, the memory location ss will contain the joint signal of all the users who selected this node or its corresponding right branch. Otherwise (i.e., for right branches and the root node), the location ss should be empty, that is, $ss = \emptyset$.

In Table 1, all receiver rules are summarized. Indeed, if the receiver recovers a single message from either $ss - cs$ or $cs - ss$, this message is correctly received and the feedback flag Sr is used. In the first case ($R2$), the right node only contained the already correctly recovered message, while in the second ($R3$), all conflicting users selected the left node (and a single new transmission took place).

When $ss \neq \emptyset$, an idle slot ($R9$) or a success without successful interference cancellation ($R7$ and $R8$), both indicate that the right set (i.e., the next slot) is certainly in conflict; therefore, the feedback Sc is provided. Recall, when $ss \neq \emptyset$, the current slot is necessarily a left node in the conflict resolution tree. If a success also causes the recovery of a second message via $ss - cs$ ($R5$), the right node must hold a single message (which we just recovered) and can be skipped.

Rule	Conditions		Actions	
	Current slot	IC	Feedback	Saved signal
R1	Collision	$ss = cs$	$Sr/Co/\cdot$	$ss' = cs$
R2	Collision	$ss - cs$ valid	$Sr/Co/\cdot$	$ss' = cs$
R3	Collision	$cs - ss$ valid	$Sr/Co/\cdot$	$ss' = ss$
R4	Collision	otherwise	$\cdot/Co/\cdot$	$ss' = cs$
R5	Success	$ss - cs$ valid	$Sr/\cdot/\cdot$	$ss' = \emptyset$
R6	Success	$ss = \emptyset$	$\cdot/\cdot/\cdot$	$ss' = \emptyset$
R7	Success and new	otherwise	$\cdot/\cdot/Sc$	$ss' = ss$
R8	Success and old	otherwise	$\cdot/\cdot/Sc$	$ss' = ss - cs$
R9	Idle	$ss \neq \emptyset$	$\cdot/\cdot/Sc$	$ss' = ss$
R10	Idle	$ss = \emptyset$	$\cdot/\cdot/\cdot$	$ss' = \emptyset$

Table 1
All possible scenarios a receiver may encounter; depending on each state, different feedback is provided to the users, while updating the saved signal.

	$\cdot/\cdot/\cdot$	$\cdot/Co/\cdot$	$\cdot/\cdot/Sc$	$Sr/\cdot/\cdot$	$Sr/Co/\cdot$
$cc = 0$	-1	$0/1$	-1	-1	$-1*$ or $0/1$
$cc = 1$	0	2	$0/1$	-1	-1
$cc \geq 2$	$cc - 1$	$cc + 1$	cc	$cc - 2$	cc

Table 2
Actions for each user, depending on the feedback. $*$ applies only if the current packet was sent for the first time. A coin flip distinguishes between 0 and 1, in case of $0/1$, respectively for choosing the left or right branch.

When experiencing a conflict with $ss = cs$, an empty right node is detected and the feedback flag Sr is used ($R1$). The remaining three rules ($R4$, $R6$ and $R10$) correspond to the standard CTMV behavior.

The transmitter side

The transmitter part of the algorithm consists of updating the counter value cc. Based on this value, a user may (re)transmit a message ($cc = 0$), postpone its transmission ($cc > 0$), or consider the transmission successful ($cc < 0$). Updates on this value are based on both the current value of the counter, and the feedback provided by the receiver. A summary is given in Table 2. We will discuss each feedback signal separately.

Feedback $\cdot/\cdot/\cdot$ In this case, we proceed similar to the idle or success case in the basic CTMV. As such, all counter values cc can be decremented by one. If a user transmitted, the counter value of which must have equaled 0, it will decrease cc to -1, indicating a successful transmission.

Feedback $\cdot/Co/\cdot$ This scenario is also identical to the operation in the basic CTMV, when a slot holds a collision. As such, all counter levels have to be incremented by one, except those that transmitted in the previous slot, having 0 as their counter value. For these users, a random decision has to be made to determine whether they join either the first branch (cc becomes 0), or the second (cc becomes 1).

Feedback $\cdot/\cdot/Sc$ With this feedback, the receiver signals that the slot was successful or idle, and that he has enough information to skip the next slot, being a collision. As such, all users which were not involved in either of these slots ($cc \geq 2$) keep their counter value fixed (as the number of unresolved sets remains identical). The possible user that was successful ($cc = 0$) becomes inactive, while the users of the right branch ($cc = 1$) have to make a random decision.

Feedback $Sr/\cdot/\cdot$ Here, the receiver signals that the slot holds a successful transmission and enough information is available to skip the next slot, which holds a single message. As such, all users which are not involved in either of these slots ($cc \geq 2$) can decrease their counter by 2, while the two users that are in the current and next slots ($cc = 0$ or 1) may regard their messages as successfully transmitted.

Feedback $Sr/Co/\cdot$ In this final case, the receiver reports that the slot was unsuccessful, but he has enough information to detect either a single or no message in the right branch. The possible user of the right branch ($cc = 1$) can consider himself as successful; while the users who transmitted in the previous slot have to make a splitting decision, except for a user who transmitted for the first time; he can also consider his message as successfully received. As such, all users who are not involved in either of these branches ($cc \geq 2$) do not alter their counter value.

4 Analysis through Matrix Analytic Methods

In [13,14] it was shown that the dynamics of the basic and modified CTMV algorithms under D-BMAP arrivals, can be captured using tree-like processes [2], which are equivalent to the class of tree-structured Quasi-Birth-Death Markov chains (QBD MC) [15]. For each algorithm, a tree-like process was constructed such that the Markov chain was positive recurrent if and only if the CTMV algorithm is stable for the D-BMAP process under consideration, which allowed us to determine the maximum stable throughput (MST) for a variety of arrival processes (including Poisson).

In this section we construct a similar tree-like process to determine the MST for the algorithm introduced in the previous section. The construction is however far less obvious as for the basic CTMV algorithm due to the complications caused by the interference cancellation features. We start by briefly describing the main properties of tree-like QBD MCs.

A tree-like process is a discrete-time bivariate Markov chain $\{(X_t, N_t), t \geq 0\}$, where X_t takes values on a $d+1$-ary tree (and is termed the node variable) and N_t takes integer values between 1 and h (and is termed the auxiliary variable). A $d+1$-ary tree is a tree where each node (including the root node) has exactly $d+1$ children labeled 0 to d. As such, each node in a $d+1$-ary tree is uniquely represented by a string of integers $J = j_1 j_2 \dots j_n$, with $0 \leq j_k \leq d$ and $k \leq n$, via the path one needs to follow to descend from the root node, denoted as $\emptyset$, to the node J. Strings of integers are written in upper case, while an individual integer is represented in lower case. As a concatenation operator, we will use the + symbol, so that if J denotes $j_1 \dots j_n$, $J + k$ represents the string $j_1 \dots j_n k$.

For $\{(X_t, N_t), t \geq 0\}$ to be a tree-like QBD MC the following restrictions need to apply. First of all, only transitions from a node to its parent, to the node itself and to one of its children are allowed. Moreover, the probability of a transition from a state of the form $(X_t, N_t) = (J + k, i)$ may only depend on i, except for transitions going to one of the parent states (J, i'), in which case they may also depend on k. Or, more formally:

$$P[(X_{t+1}, N_{t+1}) = (J', i')|(X_t, N_t) = (J, i)] = \tag{1}$$

$$\begin{cases} f^{i,i'} & J' = J = \emptyset, \\ b^{i,i'} & J' = J \neq \emptyset, \\ d_k^{i,i'} & J \neq \emptyset, f(J,1) = k, J' = J - f(J,1), \\ u_s^{i,i'} & J' = J + s, s = 0, \ldots, d, \\ 0 & otherwise, \end{cases} \tag{2}$$

where $f(J, 1)$ denotes the last element of J, and $J - f(J, 1)$ the deletion of the last element of J. The probabilities $f^{i,i'}$, $b^{i,i'}$, $d_k^{i,i'}$ and $u_s^{i,i'}$ are the $(i, i')^{th}$ elements of the $h \times h$ matrices F, B, D_k and U_s respectively. These $2d + 4$ matrices fully characterize the tree-like process and the matrices $B + D_k + \sum_{s=0}^{d} U_s$ are stochastic for all $k = 0, \ldots, d$.

The basic binary CTMV algorithm can be modeled using a tree-like QBD MC as follows: the node variable $X_t = j_1, \ldots, j_n$ represents the string of backlogged terminals with j_i being the number of users whose counter cc equals i and the auxiliary variable $N_t = (Y_t, Z_t)$, with Y_t the number of transmissions at slot t and Z_t the D-BMAP state at time epoch t (see [13] for details). Consider the same stochastic process (X_t, N_t), but for the tree algorithm proposed in the previous section. This stochastic process (X_t, N_t) is not Markovian, as illustrated below.

Suppose that in slot t, $X_t = J + 4$ and $Y_t = 1$. Consider the following two scenarios for slot $t - 1$:

- $X_{t-1} = J$ and $Y_{t-1} = 5$. Here, the receiver can deduce that slot $t + 1$ holds a collision, causing this slot to be skipped. Thus, in case of a split of 4 into $2 - 2$, this give us $X_{t+1} = J + 2$, and $Y_{t+1} = 2$.
- $X_{t-1} = J + 4 + 1$ and $Y_{t-1} = 1$. In this case, the following slot is not known to be a conflict (since there is no relevant saved signal), and thus will not be skipped, resulting in $X_t = J$ and $Y_t = 4$ for $t + 1$.

Hence, simply keeping track of all the cc values is insufficient as we also need some info about the feedback provided due to the stored signal ss. Also, even if the process was Markovian, one finds that the process contains transitions to sibling and grand-parent nodes, which are not allowed in the tree-like framework.

To circumvent these difficulties we could extend the state space of the stochastic process $\{(X_t, N_t), t \geq 0\}$ and insert some artificial time epochs to create a tree-like process as was done in [14] for other CTMV variants. However, we will use a more efficient technique, starting from the non-Markovian stochastic process $\{(X_t, N_t), t \geq 0\}$ that requires no additional states. The idea behind this technique is to replace “difficult” states with a different, existing state which has exactly the same outgoing transition probabilities.

Intuitively, the resulting MC can be seen as an MC where each left branch slot t, which causes us to skip the corresponding right branch node, is replaced by a “fictive” slot t'. This technique will not only provide us a process that is Markovian, but also eliminates transitions from a state to its grand-parent (which would happen when a collision of two users splits into one and one) or a sibling node, resulting in a tree-like QBD MC.

Assume a given realization $\{(X_t(w), (Y_t(w), Z_t(w))),\ t \geq 0\}$ of the stochastic process $\{(X_t, (Y_t, Z_t)),\ t \geq 0\}$. The chain $\{(\mathcal{X}_t, \mathcal{N}_t = (\mathcal{Y}_t, Z_t)),\ t \geq 0\}$ is constructed recursively, starting with the initial state $t = 0$, according to the following rules (Ri refers to corresponding receiver action in Table 1). Also notice that we continue making use of Z_t, thus it suffices to discuss $\mathcal{X}_t$ and $\mathcal{Y}_t$. Figure 2 illustrates this procedure.

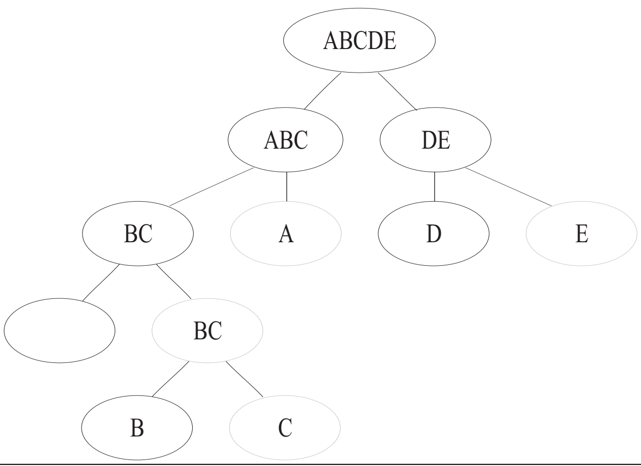
Initial state
Set $(\mathcal{X}_0(w), \mathcal{Y}_0(w)) := (X_0(w), Y_0(w))$

Following states
(A) If $((\mathcal{X}_{t-1}(w), \mathcal{Y}_{t-1}(w)) = (J, k)$ where $k \geq 2$:

- Rules $R1$ to $R3$ and $R5$ cause us to skip the right branch. Therefore, there is no need to add a 0 or 1 to $X_t(w)$ as this 0 or 1 would be removed during the next transition:
  a. $(X_t(w), Y_t(w)) = (J + 0, k)$, then $((\mathcal{X}_t(w), \mathcal{Y}_t(w)) = (J, k)$ ($R1$)
  b. $(X_t(w), Y_t(w)) = (J + 1, (k - 1))$, then set $((\mathcal{X}_t(w), \mathcal{Y}_t(w)) = (J, (k - 1))$ ($R2, 5$)
  c. $(X_t(w), Y_t(w)) = (J + 0, (k + 1))$, then $((\mathcal{X}_t(w), \mathcal{Y}_t(w)) = (J, k)$ ($R3$)
- Rule $R9$ implies that we will skip the right collision as a collision is split into an idle and a collision slot, therefore we simply store k in the auxiliary variable $\mathcal{Y}_t(w)$:
  d. $(X_t(w), Y_t(w)) = (J + k, 0)$, then $((\mathcal{X}_t(w), \mathcal{Y}_t(w)) = (J, k)$
- If a single new arrival occurs, rule $R7$ may apply, in which case the 1 will be removed during the next transition and the k would be split, therefore we simply store k in $\mathcal{Y}_t(w)$:
  e. $(X_t(w), Y_t(w)) = (J + k, 1)$, then $((\mathcal{X}_t(w), \mathcal{Y}_t(w)) = (J, k)$ ($R7$)

(B) If $((\mathcal{X}_{t-1}(w), \mathcal{Y}_{t-1}(w)) = (J, k)$ where $k \geq 3$ and $(X_t(w), Y_t(w)) = (J + (k - 1), 1)$, then $((\mathcal{X}_t(w), \mathcal{Y}_t(w)) = (J, (k - 1))$ as the 1 is removed in the next transition and $k - 1$ will be split (rule $R8$).

(C) Otherwise, rule $R4$, $R6$ or $R10$ applies and the standard CTMV behavior is followed; hence, set $(\mathcal{X}_t(w), \mathcal{Y}_t(w)) = (X_t(w), Y_t(w))$.



Basic:

$(\phi, 5) \triangleright (2,3) \triangleright (21,2) \triangleright (212,0) \triangleright (21,2) \triangleright (211,1) \triangleright (21,1) \triangleright (2,1) \triangleright (\phi,2) \triangleright (1,1) \triangleright (\phi,1)$

(X_t, Y_t):

$(\phi, 5) \triangleright (2,3) \triangleright (21,2) \triangleright (22,0) \triangleright \quad \triangleright (21,1) \triangleright \quad \triangleright \quad \triangleright (\phi,2) \triangleright (1,1) \triangleright$

$(\mathcal{X}_t, \mathcal{Y}_t)$:

$(\phi, 5) \triangleright (2,3) \triangleright (2,2) \triangleright (2,2) \triangleright \quad \triangleright (2,1) \triangleright \quad \triangleright \quad \triangleright (\phi,2) \triangleright (\phi,1) \triangleright$

Fig. 2. *Illustration of the Markov chain associated with a collision tree. The corresponding MC for the basic tree algorithm is provided as reference. The final MC* $(\mathcal{X}_t, \mathcal{Y}_t)$ *is constructed by translating the stochastic process* (X_t, Y_t) *using the provided rules.*

We note that this approach can also be applied for determining the MST of the modified binary CTMV algorithm; allowing us to half the number of states per node compared to the approach used in [14], causing some gain in the time and memory complexity of the approach.

The tree-like QBD MC $(\mathcal{X}_t, \mathcal{N}_t)$ still has one unresolved issue: each node has infinitely many children as an infinite number of users might hold the same *cc* value. Therefore, we introduce the MC $(\mathcal{X}_t^d, \mathcal{N}_t^d)$, which only allows d users with an identical *cc* value, causing each node to have exactly $d+1$ children. If more than d users attain the same *cc* value, we simply drop some messages such that only d users have this value. We can measure the impact of this operation by calculating the rate of lost messages (we refer to the end of this section for more details). We increase d (typically d between 10 to 20 suffices), until this rate is negligible ($< 10^{-6}$). For numerical evidence that this truncation operation has no significant effect (for d large enough) we refer to [13] and [14].

We can now proceed with the description of matrices D_k, U_s and B (the matrix F is not required to determine the stability of the chain) that characterize the tree-like process.

The matrices D_k hold the transition probabilities that the chain $(\mathcal{X}_t^d, \mathcal{N}_t^d)$ goes from state $(J+k,(i,j))$ to state $(J,(i',j'))$. From the previous discussion we see that this only occurs if $R6$ or $R10$ apply, therefore, as in [13] we have

$$(D_k)_{(i,j),(i',j')} = \begin{cases} (C_{i'-k})_{j,j'} & i \leq 1, i' \geq k, i' < d, \\ \sum_{l \geq d-k} (C_l)_{j,j'} & i \leq 1, i' \geq k, i' = d, \\ 0 & otherwise, \end{cases} \tag{3}$$

where the $\{C_n, n \geq 0\}$ matrices characterize the D-BMAP process and $(C_n)_{j,j'}$ holds the probability that n new arrivals occur, while the underlying state changes from state j to j'.

The matrices U_s hold the transition probabilities that the chain $(\mathcal{X}_t^d, \mathcal{N}_t^d)$ goes from state $(J+k,(i,j))$ to state $(J+ks,(i',j'))$. The first five lines are a consequence of the transitions a. to e. in (A), the fifth line also captures the transitions in (B). The matrices U_s' are identical to the ones specified in [13] for the basic binary CTMV algorithm (case (C)). The presence of the sixth line is necessitated by truncating the number of children in the tree to $d+1$, if d is large enough these transitions become irrelevant:

$$(U_s)_{(i,j),(i',j')} = \begin{cases} 0 & i' = i, s = 0, i \neq d \\ 0 & i' = i-1, s = 1 \\ 0 & i' = i+1, s = 0, i \neq d-1 \\ 0 & i' = 0, s > 1 \\ 0 & i' = 1, s > 1 \\ p_0^i \sum_{l \geq 2} (C_l)_{j,j'} & i \geq d-1, s = 0, i' = d, \\ (U_s')_{(i,j),(i',j')} & otherwise, \end{cases} \tag{4}$$

where $p_0 = p$, $p_1 = 1-p$ and $(U_s')_{(i,j),(i',j')}$ is defined by:

$$(U_s')_{(i,j),(i',j')} = \begin{cases} N_s^i p_0^{i-s} p_1^s (C_{i'-(i-s)})_{j,j'} & i > 1, i \geq s, i' \geq i-s, i' < d, \\ N_s^i p_0^{i-s} p_1^s \sum_{l \geq d-(i-s)} (C_l)_{j,j'} & i > 1, i \geq s, i' \geq i-s, i' = d, \\ 0 & otherwise, \end{cases} \tag{5}$$

where N_s^i denotes the number of different possible combinations of s from i different items.

The matrix B holds the transition probabilities that the chain $(\mathcal{X}_t^d, \mathcal{N}_t^d)$ goes from state $(J,(i,j))$ to state $(J,(i',j'))$. The first line captures cases a, c, d and e of (A), the second and third case b. of (A) and (B).

$$B_{(i,j),(i',j')} = \begin{cases} (p_0^i + p_1^i)((C_0)_{j,j'} + (C_1)_{j,j'}) & i' = i, i > 1 \\ i(p_0^{i-1}p_1 + p_0 p_1^{i-1})(C_0)_{j,j'} & i' = i-1, i > 2 \\ 2p_0p_1(C_0)_{j,j'} & i' = 1, i = 2 \\ 0 & otherwise. \end{cases} \tag{6}$$

The key component to determine the stability of a tree-like process lies in the computation of the $h \times h$ matrix V [15], which is the smallest nonnegative solution to

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

One easily sees that the $(u,v)^{th}$ element of V represents the taboo probability that the first visit to node $J \neq \emptyset$ is to state (J,v) starting from state (J,u) under taboo of node $J - f(J,1)$, i.e., the parent node of node J.

A number of iterative schemes can be used to determine V [2]. For the numerical results we made use of the following basic iterative algorithm:

$$V[0] = B \tag{8}$$

$$V[N+1] = B + \sum_{s=0}^{d} U_s (I - V[N])^{-1} D_s. \tag{9}$$

The recursion is repeated until the infinity norm of $V[N+1] - V[N]$ is sufficiently small (i.e. $< 10^{-8}$). The stability of the tree-like process is then verified by checking whether the matrices

$$G_k = (I - V[N])^{-1} D_k$$

are (numerically) stochastic [15].

To compute the amount of dropped traffic due to truncating the number of children of a node to $d+1$, we can proceed as follows. First, define $R_k = U_k(I - V[N])^{-1}$. Next, we continue by computing the $1 \times h$ steady state vector π of the MC $\{(\mathcal{X}_t^d, \mathcal{N}_t^d = (\mathcal{Y}_t^d, Z_t)), t \geq 0\}$ when censored on the states of the root node $\emptyset$. By noticing that $F = D_0 + B$, one establishes that $\pi(D_0 + V) = \pi$, with π a stochastic vector (as $V = B + \sum_{s=0}^{d} U_s (I-V)^{-1} D_s$). Due to the matrix geometric form of the steady state vector of a tree-like process [15], the vector $\pi(I - (R_0 + R_1 + \ldots + R_d))^{-1}$ contains the probabilities that $(\mathcal{Y}_t^d, Z_t)$ equals (i,j) at an arbitrary point in time t. Denoting these probabilities as z_{ij}, we can count the number of successes λ_s per time epoch as follows:

$$\sum_{j,j'} \big(z_{1j} + z_{2j} 2 p_0 p_1 (C_0)_{j,j'} + \sum_{i>1} z_{ij} (p_0^i + p_1^i)(C_1)_{j,j'} + \sum_{i>2} z_{ij} i (p_0^{i-1} p_1 + p_0 p_1^{i-1})(C_0)_{j,j'} \big).$$

The loss rate due to d is then found as $\lambda - \lambda_s$, where λ is the mean arrival rate of the D-BMAP characterized by $\{C_n, n \geq 0\}$.

5 Numerical Examples

Using the method described thus far, we determined the maximal stable throughput for the following well known Markovian arrival processes: the Poisson process (abbreviated with $PP(\lambda)$), the Erlang process ($ER(\lambda, k)$) and the Markov modulated Poisson process ($M(\lambda_1, \lambda_2, e, f)$), where e and f reflect the mean sojourn time in state 1 and 2, respectively. The results are shown in Table 3. We observe that for Poisson arrivals, a stable throughput of 0.5693 is achieved using fair coins, while biased coins can slightly improve this result; choosing $p = 0.471$ improves the MST up to 0.5698. Intuitively, for blocked access instead of free access, our algorithm is expected to perform optimal when $p = 0.5$ due to its symmetric operation. Some more formal evidence of this property is given in the Appendix. The free access property however shifts the optimal p value as only right branch nodes are skipped and therefore, on average, the left branches receive slightly more new packets. As such, setting p somewhat smaller causes a minor improvement.

For the other arrival processes, we can see the influence of the correlation on both the throughput and optimal splitting probability p; higher burstiness typically results in both a lower MST and a lower optimal p. Intuitively, collisions will occur more often during the periods where the arrival process generates traffic at a rate above average. For positive correlated traffic, selecting the second group might therefore postpone some retransmissions for periods where the arrival process is less active. More deterministic processes on the other hand have an optimal p close to 0.5. The latter can be understood by considering that in case of a collision, typically two users transmit simultaneously; in which case each user should preferably choose a different subset, which is most likely to happen if $p = 0.5$.

The computation time required to investigate the stability of the tree-like process, rapidly increases as the arrival rate approaches the MST. For example, on a Pentium-M 1.86GHz using MATLAB R2006b for Linux, in case of Poisson arrivals with $p = 0.5$ and $d = 12$, the time to determine the stability of 0.56933 is 51s, while for 0.569 it only takes 3.6s. Also, as the correlation in the input traffic increases, more iterations are required which further increases the

	MST with $p = 0.5$	Optimal p	MST with optimal p
$M(0, \lambda, 300, 300)$	[.5352, .5357]	.434	[.5381, .5386]
$M(0, \lambda, 30, 30)$	[.5370, .5375]	.439	[.5394, .5399]
$PP(\lambda)$	[.56933, .56938]	.471	[.56983, .56988]
$ER(\lambda, 2)$	[.58670, .58675]	.481	[.58685, .58690]
$ER(\lambda, 3)$	[.59493, .59498]	.488	[.59500, .59505]
$ER(\lambda, 4)$	[.59915, .59920]	.493	[.59920, .59925]
$ER(\lambda, 5)$	[.60143, .60148]	.496	[.60143, .60148]
$ER(\lambda, 15)$	[.60303, .60308]	.500	[.60303, .60308]

Table 3
The MST for the proposed tree algorithm, with various Markovian arrival processes. For each case, an interval is given, where the first number indicates a rate for which the chain was stable, while the second number indicates a rate resulting in an unstable chain. The calculations were repeated with a (close-to) optimal splitting probability p, indicating the benefits of using biased coins.

computation times. This is why we present the stability results in the form of an interval, determining the stability up to 10 digits would be too time consuming.

Apart from knowing whether the chain is stable, we can also compute its upward drift, defined as the probability that a transition to the parent node is made minus the probability that we make a transition to a child node, that is

$$\pi(I - (R_0 + \ldots + R_d))^{-1} \left(D_k e - \sum_{s=0}^{d} U_s e \right),$$

where e is a column vector with all ones, and k any number between 0 and d, since all $D_k e$ are equal. This drift value is also used to determine the optimal p in case the stability interval of two p values overlaps.

A Blocked Access Analysis

Let us briefly repeat the operation of a blocked access random access algorithm. After an initial collision of n nodes, all new arrivals postpone their first transmission attempt until the n initial nodes have resolved their collision. The time elapsed from the initial collision until the point where the n nodes have transmitted successfully is called the collision resolution interval (CRI). Suppose that m new packets are generated during the CRI. Then, a new CRI starts (with m participants) when the previous CRI (with n nodes involved) ends.

Notice, the control bit/field introduced in our algorithm is not required when using blocked access. In this section we will establish a closed form expression for the mean time L_N required to resolve a conflict of size N. The methodology used is analogue to the analysis of the basic and modified CTMV algorithm by Mathys and Flajolet [10]. Clearly, $L_0 = L_1 = 1$ and L_N obeys the following recursion:

$$\begin{aligned}&L_N = 1 + \sum_{i=0}^{N} \binom{N}{i} p^i (1-p)^{N-i} \left(L_i + L_{N-i}\right) - p^N - (1-p)^N \\ &-Np(1-p)^{N-1} - Np^{N-1}(1-p) + \delta_{N,2} Np(1-p),\end{aligned}$$

where $\delta_{i,j}$ is one if $i = j$ and zero otherwise. Define $L(z) = \sum_{N=0}^{\infty} L_N z^N/N!$, then we find the following functional equation for $L(z)$:

$$\begin{aligned}&L(z) = e^z e^{-pz} L(pz) + e^z e^{-(1-p)z} L((1-p)z) + e^z - 2(1+z) \\ &-(e^{pz} - 1 - pz) - (e^{(1-p)z} - 1 - (1-p)z) - pz(e^{(1-p)z} - 1) \\ &-(1-p)z(e^{pz} - 1) + p(1-p)z^2.\end{aligned}$$

Setting $L^*(z) = \sum_{N=0}^{\infty} L_N^* z^N = e^{-z} L(z)$, the above equation yields

$$\begin{aligned}&L^*(z) - L^*(pz) - L^*((1-p)z) = 1 - e^{-pz} - e^{-(1-p)z} \\ &-(pze^{-pz} + (1-p)ze^{-(1-p)z}) + e^{-z} p(1-p)z^2.\end{aligned}$$

Equating the coefficients of z^k on both sides implies

$$L_k^*(1 - p^k - (1-p)^k) = (-1)^k/k!(k-1)\left(p^k + (1-p)^k + kp(1-p)\right).$$

This finally results in

$$L_N = 1 + \sum_{k=2}^{N} \binom{N}{k} (-1)^k (k-1) \frac{p^k + (1-p)^k + kp(1-p)}{(1 - p^k - (1-p)^k)}.$$

This expression is symmetric on $(0, 1)$ around $p = 1/2$, meaning switching p and $1 - p$ does not alter the value of L_N. This implies that an extremum is reached in $p = 1/2$.

References

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

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

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

[4] A. Ephremides, B. Hajek, Information theory and communication networks: an unconsummated union, IEEE Transactions on Information Theory 44 (6) (1998) 2416–2434.

[5] N. Golmie, F. Mouveaux, D. Su, A comparison of MAC protocols for hybrid fiber/coax networks: IEEE 802.14 vs. MCNS, in: Proc. of the 16th Int. Conf. on Comm., Vancouver, Canada, 1999.

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

[7] D. Kazakos, L. F. Merakos, H. Deliç, Random multiple access algorithms using a control mini-slot., IEEE Trans. Computers 46 (4) (1997) 473–476.

[8] S. Khanna, S. Sarkar, I. Shin, An energy measurement based collision resolution protocol, in: Proc. of the 18-th ITC conference, Berlin Germany, 2003.

[9] J. Massey, Collision resolution algorithms and random-access communication, in: G. Longo (ed.), Multi-Users Communication Networks, CISM Courses and Lectures No. 256, Springer Verlag, Wien-New York, 1981.

[10] P. Mathys, P. Flajolet, Q-ary collision resolution algorithms in random-access systems with free or blocked channel access, IEEE Transactions on Information Theory IT-31 (2) (1985) 217–243.

[11] G. Polyzos, Molle, Performance analysis of finite nonhomogeneous population tree conflict resolution algorithms using constant size window access, IEEE Transactions on Communications 35 (11) (1987) 1124–1138.

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

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

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

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

[16] Y. Yu, G. B. Giannakis, SICTA: a 0.693 contention tree algorithm using successive interference cancellation., in: INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Miami (USA), 2005.