On the Maximum Stable Throughput of Tree Algorithms with Free Access

G.T. Peeters and B. Van Houdt
University of Antwerp - IBBT, Middelheimlaan 1, B-2020 Antwerp, Belgium

***Abstract*—A simple numerical procedure is presented to determine the maximum stable throughput (MST) of various tree algorithms with free access by defining an associated multitype branching process such that the criticality of the branching process corresponds to the stability of the tree algorithm. More precisely, a bisection algorithm is proposed that only requires the computation of the dominant eigenvalue of the expectation matrix of the branching process, the size of which is typically below 20, at each step. Using this novel approach, many existing results on free access tree algorithms are reproduced without much effort (e.g., channels with/without noise, variable length packets, interference cancellation, etc.). Furthermore, in an almost plug-and-play manner, the MST of the free access equivalent of many existing results on tree algorithms with blocked access is established (e.g., channels with capture, control signals, multi reception capabilities, etc.). The method can be applied to the class of independent and identically distributed arrival processes, which includes the Poisson process as a special case. Apart from determining the MST, the probability that a size i collision is resolved in a finite amount of time when the arrival rate exceeds the MST, can also be computed. Some limitations of the proposed methodology are discussed as well.**

***Index Terms*—Communication system performance, Information rates, Multiaccess communication, Stochastic processes**

I. Introduction

Tree algorithms [1]–[6] form a well studied class of multiple access algorithms that were the first to have a provable maximum stable throughput (MST) above 0. The is defined as the highest possible (Poisson) input rate λ for which a packet has a finite delay with probability one. Two types of tree algorithms can be distinguished when looking at the manner in which new packets are transmitted: free access and blocked access algorithms. Under free access, new packets are transmitted immediately[1] (at the start of the next slot boundary), whereas under blocked access slightly more involved rules apply (e.g., gated access, windowed/grouped access). The latter class of algorithms typically achieves higher MSTs, is often easier to analyze and is therefore studied more often.

The most important existing results for free access include [2], [7] which studies the basic and modified q-ary algorithm, [8], [9] which deals with channel errors, [10] which considers variable length packets and [11] where an interference cancellation mechanism is deployed. For blocked access many more variations have been considered, they include channels with capture [12], control subchannels [13], multi reception capabilities [14], etc.

[1]Unless a carrier-sense mechanism is present, in which case they are transmitted immediately after the end of the current ongoing transmission.

The standard approach to analyze the MST of a free access algorithm [2], [7], [10] relies on functional equations and requires some (involved) mathematics. In this paper we propose a novel method to determine the MST of a free access algorithm and demonstrate the ease of applying this methodology on various existing and novel tree algorithms with free access. More precisely, we demonstrate how to reproduce the results in [2], [7]–[11] and determine among others the MST of the free access equivalent of [12]–[14], in an almost plug-and-play manner. In each of these settings we analyze multiple algorithms. It should also be clear from this demonstration that the method can also be applied without much effort when some of these channel conditions are combined.

The idea behind the proposed methodology exists in associating a multitype branching process [15] with each of these tree algorithms (where the arrival process is part of the process), such that the criticality of the branching process corresponds to the stability of the tree algorithm. As the criticality of a branching process is easily checked by comparing the dominant eigenvalue of its expectation matrix with one, we can set up a simple bisection algorithm that requires hardly any work during each step, allowing us to determine a highly accurate value for the MST. The approach is also applicable if we relax the Poisson arrival process assumption to any independent and identically distributed arrival process.

The method contains one drawback, being that we need to upper bound the number of users that can transmit simultaneously in a single slot by some $d > 0$. Even though this could potentially increase the MST, we demonstrate that even moderate values of d (≤ 20) typically suffice to get highly accurate results (15 digits or more) for the maximum stable throughput, while setting d as large as a few hundred does not jeopardize the efficiency of the proposed method.

In [11], [16], [17] the MST of a number of free access algorithms was determined by constructing a tree structured Quasi-Birth-Death Markov chain (that made use of the same parameter d) such that the recurrence of the chain corresponded to the stability of the algorithm. In principle, we can use this methodology for each of the algorithms analyzed here, even if we wish to include Markovian components (such as Markovian errors or capture). However, setting up these Markov chains is much more involved (see [11], [17]). Moreover, determining the stability for a specific arrival rate requires the solution of a nonlinear matrix equation that is solved using a fixed point iteration with only linear convergence. Even for the basic cases in [11], [17] computing the MST to only as many as 5 digits becomes very time consuming.

Finally, apart from computing the MST of a free access algorithm, we will demonstrate that its corresponding branching process can also be used to determine the probability that a size i conflict gets resolved in a finite amount of time when the arrival rate exceeds the MST. We wish to remark that the association between tree algorithms with free access and branching processes is not new [18], however, the approach taken and problems studied in [18] are of a very different nature.

The paper is structured as follows. Section II discusses the standard information theoretical model for multiple access communications and the operation of the basic tree algorithm. The required background information on multitype branching processes and a general description of the branching process approach to determine the maximum stable throughput of a free access tree algorithm is given in Section III. Section IV demonstrates this approach on ten different classes of tree algorithms. Some limitations and other performance measures are briefly discussed in Section V.

II. Tree Algorithms

In this section we briefly discuss the standard information theoretical model for multiple access and the basic operation of a tree algorithm when operating under this model. As demonstrated in Section IV, the proposed method in this paper also applies when some of these assumptions are further relaxed.

A. Channel model

The standard information theoretical model used by a multitude of authors (for a detailed treatment of these assumptions we refer to [3]–[5]) assumes the following type of channel and user behavior:

- $S1$. *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.
- $S2$. *Infinite population*: there is an infinite set of users whose packet generation process is a Poisson process with rate $\lambda > 0$.
- $S3$. *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.
- $S4$. *Error-free feedback*: at the end of each slot, the receiver is assumed to provide binary (collision or not) or ternary (idle/success/collision) feedback to the transmitters.

With regard to assumption $S2$, the method developed in this paper applies to any independent and identically distributed (i.i.d.) arrival process, such that the probability of having $k \geq 0$ new arrivals in an arbitrary time slot is b_k. The Poisson process corresponds to the special case where $b_k = \frac{\lambda^k}{k!}\exp(-\lambda)$. We will also consider channels with errors, capture, multiple reception capabilities, collision detection, control subchannels, variable length packets, etc.

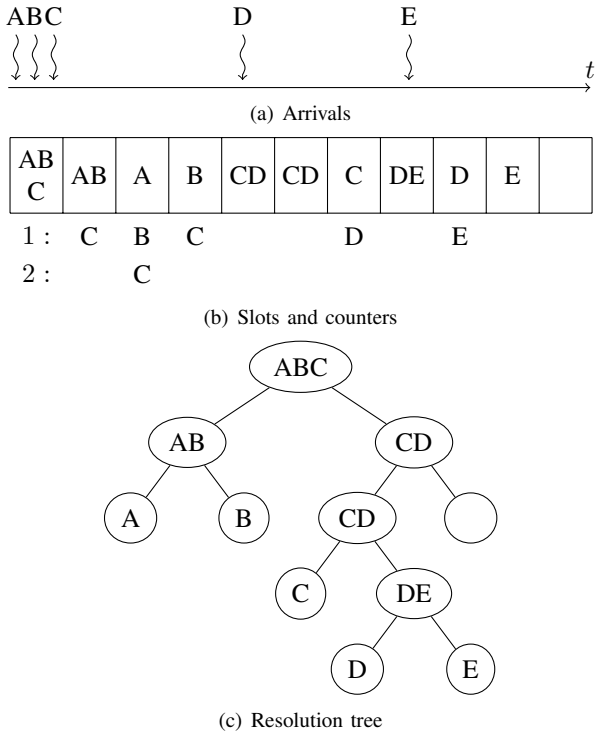

Fig. 1. Illustration of a free access tree algorithm, applied on an initial collision of three users. Free access allows new arrivals to occur during the conflict resolution interval. Each active user maintains a counter, that reflects the number of (possibly empty) groups that need to be resolved until retransmission.

B. Basic tree algorithm

The basic principle of a *binary* tree algorithm exists in resolving collisions by splitting the group of colliding users recursively, until each user receives his own slot. More specifically, suppose we have an initial collision slot with $n > 1$ users, which we will also refer to as a slot holding n users. Each of the n users, typically by means of a coin flip, chooses either group A or group B (sometimes referred to as the left or right group respectively). In the next slot, only group A is allowed to retransmit, while group B has to wait until A is completely resolved. Indeed, if A contains more than one user, we still have a collision; therefore group A splits again into two subgroups. The algorithmic solution to have group B wait, relies on the use of counters. Once a user becomes active, it maintains a counter which reflects the number of (possibly empty) groups that need to be resolved until (re)transmission. After becoming active, this counter is initialized at zero, meaning the user may transmit immediately (as we are considering free access algorithms). A user that transmits successfully becomes inactive; otherwise his counter is either set to zero or one, depending on the outcome of the coin flip. When an active user who did not transmit (i.e., with a counter > 0) receives feedback that indicates that a conflict occurred on the channel, his counter is increased by one (as one of the groups that needs to be resolved first is split into two groups); while a success or idle slot decreases his counter by one (as one group got resolved). Each conflict and the associated group splits can be visualized using a tree. Note due

to the free access, a tree will typically resolve more packets than those part of the initial collision, as shown in Figure 1.

The q-ary algorithm is a generalization of the binary one, where the group of colliding users split in $q \geq 2$ groups. Active users that did not transmit increment their counter by $q-1$ when a collision occurs, as $q-1$ extra groups are created.

III. Maximum stable throughput via branching process theory

In this section we provide a general description of our proposed methodology after reviewing the necessary multitype branching process results.

A. Branching process background

A multitype branching process is a process which describes the evolution of the individuals part of some population. Each individual is characterized by a type i, with $i \in \{1, \ldots, m\}$ for some $m > 0$ finite. The population of users evolves as follows. Given a single individual of a given type i, this individual will die (after an exponential amount of time with some parameter μ_i, which is irrelevant for our purpose) either with or without generating some offspring. Whether offspring is generated is independent of the number of individuals and their types present in the population. In other words, with some probability a_i the individual dies without offspring, while with some probability $A_{i,j_1 j_2 \ldots j_r}$ the individual dies and generates an offspring of r individuals, the type of the k-th being j_k. Clearly we have,

$$a_i + \sum_{r \geq 1} \sum_{j_1}^{m} \cdots \sum_{j_r}^{m} A_{i,j_1 j_2 \ldots j_r} = 1, \tag{1}$$

for $i = 1, \ldots, m$. Notice that this type of branching process is equivalent to a process that allows individuals to change types or to generate offspring without dying, by regarding the rightmost child as the continued life of the parent. When the population initially starts with a single individual of type i, its evolution can be depicted using a tree rooted by a type i branch, where each type k individual is represented as a type k branch.

An important question related to such a branching process is whether a tree rooted by a type i branch, for $i = 1, \ldots, m$, becomes extinct in a finite amount of time with probability one. To answer this question, it suffices to determine the $m \times m$ expectation matrix M, whose (i,j)-th entry $M_{i,j}$ contains the expected number of type j children of a single type i individual. It is well known [15] that a multitype branching process becomes extinct with probability one if and only if $sp(M) \leq 1$, with $sp(M)$ the largest eigenvalue of the (nonnegative) matrix M, provided that the matrix M is irreducible (meaning, for each $i, j \in \{1, \ldots, m\}$ there exists an $n \geq 0$ such that the (i,j)-th entry of M^n is strictly positive[2]). For our purpose it will suffice to determine the matrix M, meaning there is no need to compute the individual probabilities $A_{i,j_1 j_2 \ldots j_r}$ and a_i.

[2]This can be checked in $O(m+p)$ time using a Breadth-First-Search algorithm, where p is the number of nonzero entries in M.

B. Free access tree algorithms as branching processes

In this section we provide a general description of our methodology to determine the maximum stable throughput of a free access tree algorithm by relying on the above-mentioned branching theory result. In the next section we provide a detailed discussion for a multitude of free access algorithms.

Roughly speaking, we will associate a branch with some of the time slots part of the multiple access channel. Typically, these will be the slots holding $c \geq 2$ packets (including both new transmissions and retransmissions), that is, the collision slots. In some special cases, we will also associate a branch with idle or successful slots, or even distinguish between two types of slots that hold c packets, see Section IV. A slot holding c packets is therefore regarded as a type $c \geq 2$ branch. Depending on the splitting decisions and possibly new arrivals, a collision slot may induce zero or more new collision slots, which corresponds to the creation of offspring. More precisely, when a collision of c users occurs, the c users involved are split into a number of groups, say G_1 to G_q. Let n_i be the number of new arrivals that transmit in the slot that allows group G_i to retransmit and c_i the number of users that selected group G_i. Then, the number of type $k \geq 2$ children of a type c branch, equals the number of groups for which $c_i + n_i = k$.

By defining a branching process in the above manner, we find that a tree rooted by a type $c \geq 2$ branch will go extinct in a finite amount of time with probability one if and only if the conflict of size c is resolved in a finite amount of time with probability one. If all conflicts get resolved in a finite amount of time with probability one, the multiple access algorithm is stable. Hence, to test the stability of an algorithm, we must check whether the dominant eigenvalue of the corresponding matrix M is less than or equal to one. Notice, as the amount of offspring is influenced by the number of new arrivals, the probabilities b_i of having i new arrivals in a slot affect the matrix M, as expected.

The current model still has one unresolved issue: we associated a type c branch with each slot holding c packets. Clearly, this number c is in principle unbounded, meaning we require an infinite number of types in the branching process. However, the branching processes considered above only allow a finite number of types. To resolve this issue, we introduce an approximation, that turns out to have little to no influence on the maximum stable throughput as demonstrated in detail in the next section[3]. More precisely, we assume that at most d users can coexist in a slot. Whenever a slot appears that holds more than d packets, due to the arrival of new packets, we assume that some of these new arrivals are dropped (i.e., lost). As demonstrated in the next section (where we investigate the influence of d on the accuracy of the resulting maximum stable throughput), it turns out that even moderate values of d (e.g., $d \leq 20$), result in highly accurate values for the maximum stable throughput.

Hence, to determine the maximum stable throughput of a free access algorithm, we propose the following approach. Define a branching process (with d sufficiently large) by asso-

[3]A similar approximation was used in [11], [16], [17] where tree algorithms are modeled using tree structured Markov chains.

ciating a branch with (some of) the slots part of the multiple access channel and determine the expectation matrix M, which is a function of λ, the arrival rate of the (Poisson) arrival process. As the dominant eigenvalue $sp(M)$ is a function of the arrival rate λ, we determine the MST numerically by finding the point where this function reaches a value of 1. This can be done easily by employing a bisection algorithm on $(0,1)$. The number of iterations required can be further reduced by relying on the Brent/Dekker root finding algorithm on $(0,1)$. For the model with multiple reception capabilities of Section IV-E, we clearly need to apply the bisection algorithm on $(0,m)$ as the MST can be larger than one. For some of the models with a noisy channel (see Section IV-C), only the arrival rates $\lambda \in (\lambda_{min}, \lambda_{max})$ result in stability, meaning $sp(M)-1$ has two zeros in $(0,1)$. To determine both, we first apply a golden section search on $(0,1)$, which returns the arrival rate $\bar{\lambda}$ with $sp(M)$ minimal (assuming $sp(M)$ is a unimodal function of λ). Next, we apply the bisection algorithms on both $(0,\bar{\lambda})$ and $(\bar{\lambda},1)$ to find λ_{min} and λ_{max}.

In the next section, we treat several existing and new tree algorithms with free access using this branching process approach and determine their maximum stable throughput. From this multitude of examples, it should be clear that many other variations of these free access algorithms can be analyzed in an almost plug-and-play manner using this methodology.

IV. Examples

A. Basic tree algorithm

1) Binary tree algorithm: We start by studying the binary tree algorithm and associate a type c branch to each slot holding c packets for $c \geq 0$, meaning initially we also include idle and success slots (as opposed to the discussion in the previous section). Due to the assumptions made by the standard model in Section II-A, slots holding only one or zero packets are successfully received, hence, we state that idle or success slots do not induce any other slots. Collision slots however result in a conflict and the users involved in the collision are always split into two new groups. The two slots in which these two groups retransmit may be regarded as the offspring of the collision slot, meaning a collision slot induces two other slots (that may also hold some new arrivals). Colliding users select the first group with probability p, and the second group with probability $1-p$, where setting $p=0.5$ is known to maximize the MST, which can be intuitively understood by the symmetric operation of the algorithm.

The operation of the standard binary tree algorithm can be captured by a branching process with the following finite expectation matrix M obtained by truncating the number of packets that is allowed to coexist in a slot to d. To construct M, we first define the size $d+1$ square matrix $B^{(2)}$, where entry $B^{(2)}_{i,j}$ contains the expected number of groups with j packets after the size i collision is split (i.e., new arrivals are not taken into account by B)

$$B^{(2)}_{i,j} = \begin{cases} C^i_j(p^j(1-p)^{i-j} + p^{i-j}(1-p)^j) & i \geq 2, i \geq j \\ 0 & otherwise, \end{cases} \tag{2}$$

where C^i_j is defined as the number of possible ways to choose j out of i items. Indeed, the expected number of groups that hold j packets equals the expected number of left branches with j packets, which equals $C^i_j p^j(1-p)^{i-j}$, plus the expected number of right branches holding j packets, which equals $C^i_j p^{i-j}(1-p)^j$.

To determine the types of the 2 offspring slots, we must include the possible new arrivals. We notice that a type i slot actually corresponds to a slot holding $i-k \geq 0$ packets that require retransmission, together with k new arrivals, for some $k \geq 0$. Therefore, the expectation matrix is given by $B^{(2)}E$ with E a square matrix of size $d+1$ with

$$E = \begin{pmatrix} b_0 & b_1 & b_2 & b_3 & \cdots & \sum_{k=d}^{\infty} b_k \\ & b_0 & b_1 & b_2 & \cdots & \sum_{k=d-1}^{\infty} b_k \\ & & b_0 & b_1 & \cdots & \sum_{k=d-2}^{\infty} b_k \\ & & & b_0 & \cdots & \sum_{k=d-3}^{\infty} b_k \\ & & & & \ddots & \vdots \\ & & & & & \sum_{k=0}^{\infty} b_k = 1 \end{pmatrix},$$

where b_k characterizes the independent and identically distributed (i.i.d.) arrival process, with $b_k = \frac{e^{-\lambda}\lambda^k}{k!}$ representing the Poisson arrival process.

The branching process defined thus far holds type 0 and 1 branches, however, as these branches never generate any offspring, the finiteness of the tree is not influenced if all these branches where pruned. This means that we can simply ignore type 0 and 1 branches and study the dominant eigenvalue $sp(M)$, with

$$M = \chi_2(B^{(2)}E), \tag{3}$$

where $\chi_k(X)$ removes the first k rows and columns of a matrix X. Under Poisson arrivals all entries of M are nonzero and as such the matrix is irreducible (if $0<p<1$).

Finally, the MST can be calculated using a bisection algorithm on $(0,1)$ that determines the dominant eigenvalue of the expectation matrix M at each step. As $B^{(2)}$ is independent of the arrival process, we only need to recalculate the matrices E and M during each iteration.

2) q-ary tree algorithm: The standard binary tree algorithm has been generalized to a q-ary tree algorithm, where instead of two slots, a colliding user can now choose out of q slots, each with probability p_i (such that $\sum_{i=1}^{q} p_i = 1$). In a manner completely analogous to that for binary tree algorithms we can construct a branching process by first constructing the matrix $B^{(q)}$:

$$B^{(q)}_{i,j} = \begin{cases} C^i_j \sum_{k=1}^{q} p_k^j (1-p_k)^{i-j} & i \geq 2, i \geq j \\ 0 & otherwise. \end{cases} \tag{4}$$

The stability can then be determined with the resulting expectation matrix $M^{(q)}$:

$$M^{(q)} = \chi_2(B^{(q)}E), \tag{5}$$

where the matrix E remains identical to the one of the binary algorithm.

q	MST
2	0.360177027958044626828452 8
3	0.401599370184180932389 2300
4	0.399222826314194568457 7376
5	0.387241407537505377882 4552
6	0.373354598594310805938 1336
7	0.359731123648665978160 2567

TABLE I
MST OF THE BASIC q-ARY TREE ALGORITHM.

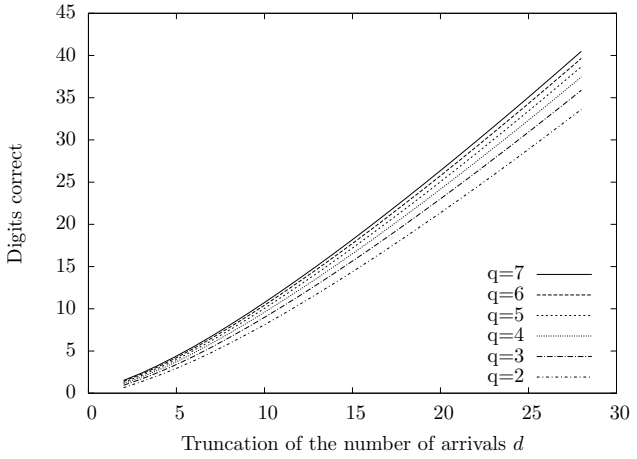

Fig. 2. Influence of the truncation d on the approximated MST for the basic q-ary tree algorithm.

3) Results: The MST of the basic q-ary tree algorithm is given in Table I for $q = 2$ to 7. As these are known results, we are mainly interested in the influence of d on the obtained MST. The most accurate result for the binary scheme we found in the literature was 0.360177147 [19], which seems to contradict Table I. However, this result was obtained from the fact that the MST corresponds to the smallest positive root of the equation:

$$\frac{2e^{-2\lambda}}{2\lambda-1}\sum_{i\geq 0}2^i e^{\frac{2\lambda}{2^i}}\left(e^{\frac{-\lambda}{2^i}}(1-\frac{\lambda}{2^i})-1-2(\frac{\lambda}{2^i})^2+2\frac{\lambda}{2^i}\right)=1. \tag{6}$$

This equation is however numerically troublesome (for instance, 2^i grows rapidly, while the terms between brackets tend to zero). Alternatively, this condition can be rewritten as in [7] in a numerically more suitable form, with $q = 2$:

$$\frac{q(1-\lambda)-1}{\lambda q^2 e^{-\lambda q/(q-1)}} = \sum_{i\geq 1}\frac{i}{i+1}\left(\frac{i-iq^{-1}}{1-q^{-i}}-\frac{1}{q}\right)\left(\frac{q}{q-1}\right)^i\frac{\lambda^i}{i!}. \tag{7}$$

We used Maple to solve this equation for i up to 1000, with 100 digits precision, which gave a perfect match with our findings in Table I.

Figure 2 shows the accuracy of the MST obtained by the branching process for d ranging from 2 to 30. These results were obtained by running our algorithm in 50 digit precision, using Maple, when compared with the 100 digit evaluation of (7). As expected, increasing d tends to increase accuracy of the MST and more surprisingly an accuracy of more than 15 digits is acquired using d values as small as 20. Hence, the amount of computation time is minimal as the Brent/Dekker root finding algorithm requires about 10 iterations and each iteration only involves computing the dominant eigenvalue of a positive size $d-1$ matrix. These results also indicate that the instability is not so much caused by the occurrence of occasional collisions between many packets, but mainly by the frequency of low order collisions. Further, for larger splitting factors q truncation is less damaging. This can be understood intuitively using the following example: suppose two successive slots hold a_1 and a_2 new arrivals with a_1 and a_2 close to d. Then, on average $a_1/q + a_2$ new arrivals will transmit simultaneously (if $p_1 = 1/q$) in the second slot, meaning large q values cause fewer lost packets due to truncation.

B. Modified q-ary tree algorithm

In the standard q-ary tree algorithm, $c \geq 2$ users involved in a collision might all select the last group to retransmit. If the feedback allows us to distinguish between an idle slot and a success, we could easily detect this situation, whenever a collision is followed by a series of $q-1$ idle slots. As the next slot is guaranteed to hold a collision, this group can be split immediately (i.e., the collision slot can be skipped). This modification was proposed by Massey [6] and others.

To model this modified algorithm, in which certain slots can be skipped, we need to adapt our branching process. There are a number of ways to incorporate these skipped slots. Previously, we associated with each slot holding $c \geq 2$ users a branch of type c. One way to deal with the skipped collisions is to treat a skipped slot with $c \geq 2$ users also as a branch of type c. However, in this *virtual* slot, we do not allow any new arrivals. Note that, although we loose the close connection between real slots and branches, this does not influence the MST. Whenever a conflict is solved in a finite number of slots with probability one, it is also solved in a finite number of slots with probability one if we do count the skipped slots. It is also possible to achieve a one-to-one correspondence between real slots and branches as in the previous section, however this slightly complicates the description of the expectation matrix $\bar{M}^{(q)}$.

To obtain $\bar{M}^{(q)}$, we construct the square matrix $P^{(q)}$ with $d+1$ rows, entry $P^{(q)}_{i,j}$ of which contains the expected number of *virtual* slots with j users induced by a slot holding i packets:

$$P^{(q)}_{i,j} = \begin{cases} p_q^i (b_0)^{q-1} & i = j \geq 2, \\ 0 & otherwise. \end{cases} \tag{8}$$

Notice, a virtual slot occurs whenever all i users pick the last group and no new arrivals occur in the first $q-1$ slots (otherwise they are not idle). As we do not allow any new arrivals to occur in a virtual slot, we can construct the matrix $\bar{M}^{(q)}$, representing the branching process, as follows:

$$\bar{M}^{(q)} = \chi_2((B^{(q)} - P^{(q)})E + P^{(q)}). \tag{9}$$

The maximum stable throughput for $q = 2$ to 7 is given in Table II together with the optimal p_q value (for the q-ary scheme setting $p_1 = \ldots = p_{q-1}$ is known to be optimal).

q	$p_1 = \ldots = p_{q-1}$	MST
2	0.40680	0.393225073128056
3	0.31454	0.407614789045566
4	0.24445	0.400851418664151
5	0.19791	0.387803252352080
6	0.16575	0.373582232584529
7	0.14241	0.359834533583306

TABLE II
MST OF THE MODIFIED q-ARY TREE ALGORITHM.

These results are in perfect agreement with all prior published results.

C. Tree algorithms on a noisy channel

In this section we study both the basic and modified q-ary tree algorithms when errors occur on the channel [8], [6, Section 5.2], [20], [9]. The following two types of errors are considered: (i) an idle slot is incorrectly perceived as a collision and (ii) a successful transmission is regarded as a collision. We assume that these errors are memoryless and that the first event occurs with probability δ and the second with probability ϵ, where δ is typically smaller than ϵ. This model corresponds with [8], while the limited case were $\delta = \epsilon$ was analyzed in [9] for the basic q-ary algorithm with free access.

1) Basic q-ary tree algorithm: The adaptations required to incorporate errors are limited. Apart from having branches that correspond to slots holding $c \geq 2$ packets, we also introduce type 0 and 1 branches that correspond to slots holding 0 or 1 packets that are incorrectly perceived as collisions. Hence, as opposed to Section IV-A2 where we removed all the type 0 and 1 branches as they never generate any offspring, slots holding 0 or 1 packet now do create offspring with probability δ and ϵ, respectively. As such, we define $B_e^{(q)}$

$$(B_e^{(q)})_{i,j} = \begin{cases} C_j^i \sum_{k=1}^{q} p_k^j (1-p_k)^{i-j} & i \geq 0, i \geq j \\ 0 & otherwise, \end{cases} \quad (10)$$

where only the first two rows differ from $B^{(q)}$. The stability can then be determined with the resulting expectation matrix $M_e^{(q)}$:

$$M_e^{(q)} = B_e^{(q)} E D_e, \quad (11)$$

where D_e is a diagonal matrix with entries $(\delta, \epsilon, 1, \ldots, 1)$ that multiplies the first two columns of $B_e^{(q)} E$ by δ and ϵ, respectively.

2) Modified q-ary tree algorithm: The modified scheme requires some more care. As for the modified scheme without channel errors, we have type $c \geq 2$ branches for both real and virtual slots holding c packets. We now also include real and virtual slots with 0 and 1 packet that were regarded as collisions. As such, we first consider $B_e^{(q)}$, the (i,j)-th entry of which gives the expected number of size j groups induced by a type i branch. With probability $p_q^i((1-\delta)b_0)^{q-1}$ the last group corresponds to a virtual slot and therefore no new arrivals need to be added to this slot. The real slots holding 0 or 1 packet will only correspond to a type 0 or 1 branch with probability δ and ϵ, respectively, while the skipped slots holding 0 or 1 packet always correspond to such a branch.

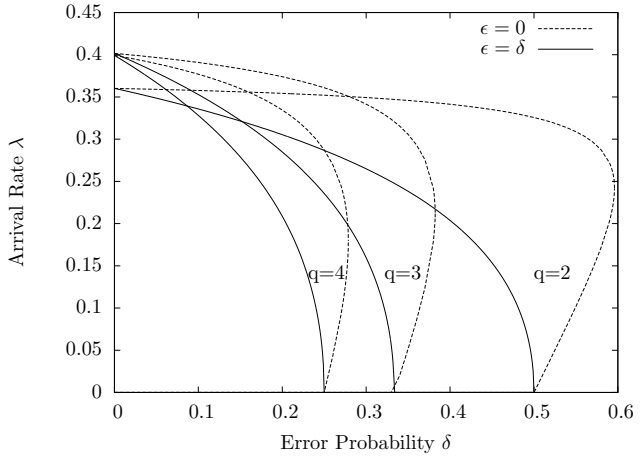

Fig. 3. Stability region for the basic q-ary tree algorithm as a function of the error probability δ, with $\epsilon = 0$ or δ.

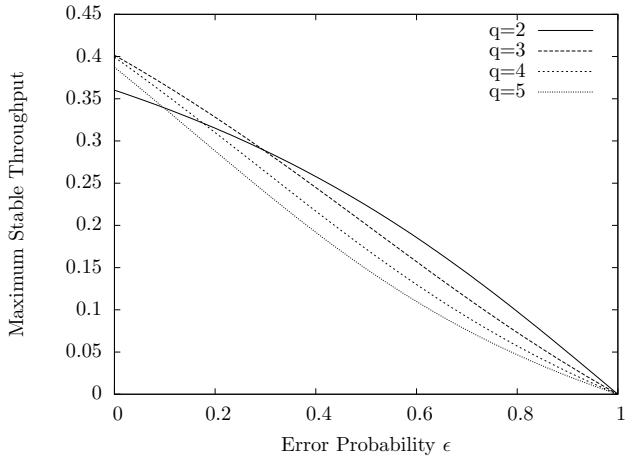

Fig. 4. MST for the basic q-ary tree algorithm as a function of the error probability ϵ when $\delta = 0$.

Therefore, the stability can be determined from the expectation matrix $\bar{M}_e^{(q)}$:

$$\bar{M}_e^{(q)} = (B_e^{(q)} - P_e^{(q)}) E D_e + P_e^{(q)}, \quad (12)$$

where

$$(P_e^{(q)})_{i,j} = \begin{cases} p_q^i((1-\delta)b_0)^{q-1} & i = j \geq 0, \\ 0 & otherwise. \end{cases} \quad (13)$$

3) Results: The results for the basic/modified q-ary scheme on a channel with errors, are somewhat different in nature as the algorithm is no longer necessarily stable for all $\lambda \in [0, \lambda_{max}]$ with λ_{max} the maximum stable throughput if $\delta > 0$. Instead there also exists a λ_{min} such that stability is only achieved for $\lambda \in [\lambda_{min}, \lambda_{max}]$.

A plot that depicts the stability region for $q = 2$ to 4, while $\epsilon = 0$ or δ is given in Figure 3, these are in agreement with [8], [9]. For $\epsilon = 0$, λ_{min} becomes larger than zero when δ exceeds $1/q$. Indeed, having $\delta > 1/q$ suffices to cause instability for any arrival rate, unless errors cannot occur during a success and the arrival rate is large enough to guarantee enough successes. Figure 4 depicts the MST when

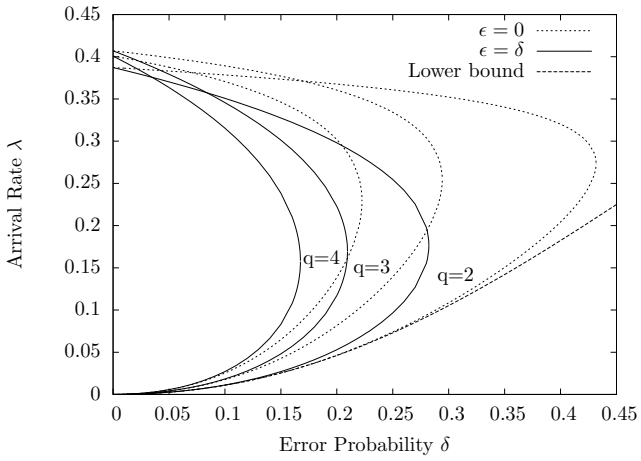

Fig. 5. Stability region for the modified q-ary tree algorithm as a function of the error probability δ, with $\epsilon = 0$ or δ.

$\delta = 0$ and indicates that under high error rates the binary scheme becomes superior.

For the binary modified scheme, the existence of a $\lambda_{min} > 0$ can be understood intuitively as follows. Suppose an idle slot is incorrectly perceived as a collision. As long as no new arrivals or errors occur, the channel will remain idle and the number of unresolved groups remains identical as all the right slots are skipped (which causes the well known deadlock in case of blocked access). However, each time an error occurs during this interval without new arrivals, the number of unresolved groups increases by one. Thus, if the arrival rate is sufficiently small compared to the error rate, these false collisions cause instability as the number of unresolved groups will grow to infinity.

More formally, for the binary scheme, we find that entry $(0,0)$ of the matrix $\bar{M}_e^{(2)}$ can be written as

$$(\bar{M}_e^{(2)})_{0,0} = \exp(-\lambda)(1+\delta) - \exp(-2\lambda)(1-\delta)\delta. \quad (14)$$

When $(\bar{M}_e^{(2)})_{0,0} > 1$ the dominant eigenvalue $sp(\bar{M}_e^{(2)})$ is guaranteed to exceed one, implying instability. From the above equation one finds

$$\lambda_{min} \geq -\log\left(\frac{(1+\delta) - \sqrt{(1-\delta)^2 + 4\delta^2}}{2(1-\delta)\delta}\right). \quad (15)$$

Note, this inequality holds for all ϵ including $\epsilon = 0$. The stability region for $q = 2$ to 4 is shown in Figure 5 for $\epsilon = 0$ or δ, together with the lower bound for λ_{min}. When $\delta = 0$, λ_{min} becomes zero again and we get similar curves as in Figure 4 for the basic q-ary scheme.

D. Tree algorithms on a channel with collision detection

Consider a channel that supports a collision detection mechanism [6, Section 5.3], meaning collisions on the channel can be detected before the end of the time slot. Define $\ell_c < 1$ as the fraction of a time slot needed to detect a collision. We further assume that the collision feedback is also issued as soon as the collision is detected, allowing us to shorten the length of collision slots in our analysis. Hence, slots following a collision slot will typically carry fewer new arrivals as less time was available for them to arrive. As before, let b_i and $b_i^{(cd)}$ be the probability of having i new arrivals in an ordinary and collision slot, respectively. Thus, under Poisson arrivals, we have $b_i = \exp(-\lambda)\lambda^i/i!$ and $b_i^{(cd)} = \exp(-\lambda\ell_c)(\lambda\ell_c)^i/i!$.

1) Basic q-ary tree algorithm: Once more we associate a type c branch with every slot holding a collision of c packets. Each collision is split into q groups. The first group will retransmit in the next slot, which immediately follows the collision slot. Hence, the number of new arrivals added to the first group is determined by $(b_i^{(cd)})_{i\geq 0}$. For the remaining $q-1$ groups, it is important to remark that they will receive a slot after the last slot of the conflict resolution interval (CRI) that solved all prior groups. By definition, the last slot of a CRI never holds a collision, thus all the remaining groups will receive new arrivals according to $(b_i)_{i\geq 0}$.

This results in the following expectation matrix. Define E_{cd} as E, but with b_i replaced by $b_i^{(cd)}$ and for $k = 1$ to q define $B_k^{(q)}$ as

$$(B_k^{(q)})_{i,j} = \begin{cases} C_j^i p_k^j (1-p_k)^{i-j} & i \geq 2, i \geq j \\ 0 & otherwise. \end{cases} \quad (16)$$

Notice, $\sum_k B_k^{(q)} = B^{(q)}$ and $(B_k^{(q)})_{i,j}$ represents the expected number of type j groups after splitting a type i branch that form the k-th group. Set

$$M_{cd}^{(q)} = \chi_2((B^{(q)} - B_1^{(q)})E + B_1^{(q)}E_{cd}). \quad (17)$$

2) Modified q-ary tree algorithm: When analyzing the MST of the modified scheme using collision detection, we first associate a different type of branch to a real collision of $c \geq 2$ packets (type c) and a skipped collision of the same multiplicity (type $c^{(cd)}$). This is useful as the left child of a skipped collision also receives new arrivals according to b_i, as opposed to $b_i^{(cd)}$ for a real collision, because the left child of a skipped collision follows an idle slot, while the left child of a real collision follows the collision itself.

To express the size $2(d-1)$ expectation matrix $\bar{M}_{cd}^{(q)}$, we first introduce

$$(P_{cd}^{(q)})_{i,j} = \begin{cases} p_q^i b_0^{(cd)} (b_0)^{q-2} & i = j \geq 0, \\ 0 & otherwise. \end{cases} \quad (18)$$

Notice, the (i,j)-th entry holds the expected number of skipped collisions consisting of j packets induced by a real collision of i packets. The offspring of a skipped collision is identical to the offspring of any collision in a system without collision detection, thus the expectation matrix becomes:

$$\bar{M}_{cd}^{(q)} = \begin{bmatrix} \chi_2(C_1 E + B_1^{(q)} E_{cd}) & \chi_2(P_{cd}^{(q)}) \\ \chi_2((B^{(q)} - P^{(q)})E) & \chi_2(P^{(q)}) \end{bmatrix}, \quad (19)$$

with $C_1 = (B^{(q)} - B_1^{(q)} - P_{cd}^{(q)})$. The dominant eigenvalue of this matrix is identical to the one of the following size $d-1$ matrix:

$$\chi_2(C_1 E + B_1^{(q)} E_{cd}) + \chi_2(P_{cd}^{(q)}) C_2 \chi_2((B^{(q)} - P^{(q)})E), \quad (20)$$

where C_2 is a size $d-1$ diagonal matrix with entries $1/(1 - P_{i,i}^{(q)})$ for $i = 2, \ldots, d$. The above matrix corresponds to the expectation matrix of the branching process that only associates branches with real collisions.

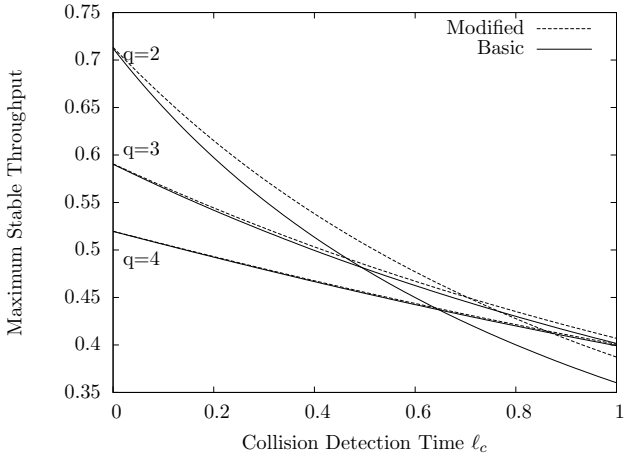

Fig. 6. Maximum stable throughput on a channel with collision detection using fair coins ($p_1 = \ldots = p_q = 1/q$).

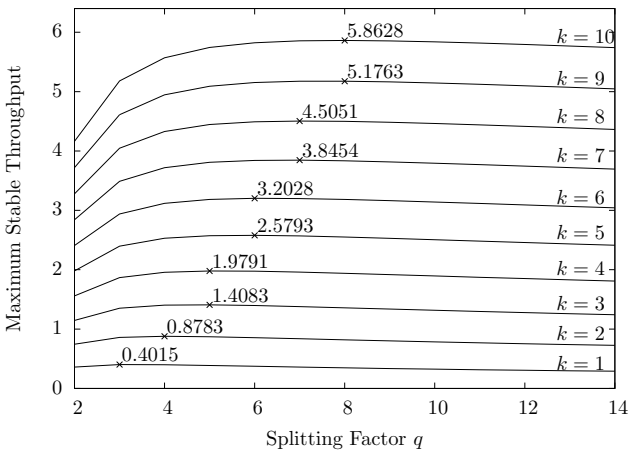

Fig. 7. MST for the basic q-ary tree algorithm on a channel with multiple reception capabilities as a function of the splitting factor q.

k	basic MST($1/2$)	mod MST(p_1)	p_1
2	0.7442511116	0.76241672333	0.46049
3	1.1454969435	1.15342900688	0.48581
4	1.5586068844	1.56181625774	0.4902
5	1.9802585084	1.98150077748	0.49827
6	2.4083773927	2.40884285823	0.49940
7	2.8415984822	2.84176851091	0.49980
8	3.2789739965	3.27903486348	0.49993
9	3.7198159722	3.71983741049	0.49998
10	4.1636069203	4.16361437217	0.49999

TABLE III
MST OF THE BINARY TREE ALGORITHM ON A CHANNEL WITH MULTIPLE RECEPTION CAPABILITIES.

3) Results: Figure 6 shows the MST as a function of the collision detection time ℓ_c when using fair coins ($p_1 = \ldots = p_q = 1/q$). It demonstrates that the binary scheme becomes superior when collisions can be detected early, while the gain achieved by the modified algorithm diminishes quickly. This is not entirely unexpected as the modified scheme skips collisions, that now require less capacity on the channel (when compared to a success or an idle slot). There is however still a gain of 0.1% as ℓ_c approaches zero, which can be understood by remarking that the new arrivals interact differently with the retransmissions for both algorithms even if the collision slots have a near zero length.

Fair coins are no longer optimal when a collision detection mechanism is present. One can achieve higher MSTs by increasing p_1 somewhat, which is no surprise as the first of the q groups formed by a collision receives fewer new arrivals (according to $b_i^{(cd)}$) when compared to the remaining $q-1$ groups.

E. Tree algorithms on a channel with multiple reception capabilities

In this section we consider a multiple access channel with multiple reception capabilities [14], [21]. On such a channel all simultaneous transmissions involving k or less packets can be retrieved successfully by relying on coding techniques. Hence, any collision consisting of at most k packets is resolved immediately and the receiver acknowledges this using the *no collision* or *success* feedback signal (depending on whether the feedback is binary or ternary).

To determine the maximum stable throughput for the basic and modified q-ary algorithm, it suffices to notice that any type c branch with $c \in \{2, \ldots, k\}$ no longer generates any offspring. Hence, for the binary scheme the expectation matrix becomes

$$M_m^{(q)} = \chi_{k+1}(B^{(q)}E), \tag{21}$$

while for the modified one we have

$$\bar{M}_m^{(q)} = \chi_{k+1}((B^{(q)} - P^{(q)})E + P^{(q)}). \tag{22}$$

Notice, there is no need to set the first $k+1$ rows of $B^{(q)}E$ or $(B^{(q)} - P^{(q)})E + P^{(q)}$ to zero as they are removed by the χ_{k+1} operation.

The maximum stable throughput for the binary scheme is given in Table III (where $d = 50$ was used to get accurate results for the larger k values), where $p = 1/2$ is still optimal for the basic scheme. For the modified scheme the optimal p value tends to $1/2$ as k increases. Further, the gain achieved by the modified scheme reduces as k grows, making it less worthwhile to exploit ternary feedback. Figure 7 indicates that the optimal splitting factor q grows as a function of the number of simultaneous transmissions k that can be decoded correctly (for the basic scheme with fair coins).

F. Interference cancellation tree algorithm

A free access tree algorithm that uses interference cancellation (IC) to improve the MST was introduced in [11]. Similar to the modified tree algorithm, some of the slots can be skipped, because the receiver can deduce its content from prior transmissions. This additional information is provided by IC, which allows one to "subtract" signals. More precisely, if a group of colliding packets splits into two subgroups, the second group can be reconstructed by subtracting the signal of the first from the signal of the original group. Under blocked access this allows one to skip the right child of any collision (as was done in [22], [23]). However,

free access prevents this approach, because new arrivals may occur in the first group, which obstructs the recovery of the right child via cancellation. The algorithm introduced in [11] solves this problem by skipping the second group only when the cancellation is guaranteed to succeed and makes use of a control field that indicates whether a particular packet is transmitted for the first time. The following cases allow us to skip the right child, where the notation x/y denotes that x and y packets selected the first and second group, respectively. The number of packets belonging to the parent slot is denoted as n; the sum of x and y exceeds n whenever new arrivals occur:

- $0/n$: this case is identical to the modified tree algorithm.
- $1/n-1$: a successful reception of one packet, together with the indication that it is retransmitted, reveals that the subtraction can produce the signal of the second group.
- $1/n$: this is similar to the previous case, but this time the packet is transmitted for the first time, meaning the second group is just a repetition of the parent group.
- $n/0$: if the signal of the first group is identical to that of the parent, we know that the second group is empty.
- $n-1/1$: if we perform the subtraction of the signal of the first group from that of the parent and decode a single retransmitted packet, we know that there is exactly one packet in the second group. Furthermore, this packet is also successfully received.
- $n+1/0$: if we perform the subtraction of the signal of the parent from that of the first group, and we decode a single new packet, we know that a new arrival occurred, while all the colliding packets selected the first group. Apart from skipping the second group, the first group continues with n instead of $n+1$ users.

To summarize, if all users choose either the first or the second group, with at most one arrival in the first, the second slot can be skipped (case 1,3,4 and 6). Similar, if all but one choose either the first or the second group with no arrivals in the first, the second can also be skipped (case 2 and 5). To model this algorithm as a branching process, we first define the probability $p_{rskip}(i,j)$ that the right slot induced by a size i collision can be skipped, provided that j out of i colliding packets choose the first group. Given the above observation, we find:

$$p_{rskip}(i,j) = \delta_{j=0 \vee j=i}(b_0+b_1) + \delta_{j=1 \vee j=i-1} b_0, \qquad (23)$$

with δ_X equal to one if X is true and zero otherwise. Using $p_{rskip}(i,j)$, we define $P^{(ic)}$:

$$P^{(ic)}_{i,j} = \begin{cases} C^i_j (p^j (1-p)^{i-j} p_{rskip}(i,j)) & i \geq 2, i \geq j \\ 0 & otherwise. \end{cases} \qquad (24)$$

This allows us to construct the expectation matrix $M^{(ic)}$:

$$M^{(ic)} = \chi_2((B^{(2)} - P^{(ic)})E + P^{(ic)} + R), \qquad (25)$$

with R defined as follows:

$$R_{i,j} = \begin{cases} p^i b_1 & 2 \leq j = i < d \\ -p^i b_1 & 3 \leq j = i+1 \leq d \\ 0 & otherwise. \end{cases} \qquad (26)$$

This R matrix corresponds to case 6 and captures the fact that only the initial n packets need to be resolved, thus a type n branch is created instead of a type $n+1$.

Using this branching process we find a maximum stable throughput of 0.56985336033524 when $p = 0.47103$, this result matches entirely with [11], where a time consuming iterative procedure for tree structured Quasi-Birth-Death Markov chains indicated that the MST was part of the interval $[0.56983, 0.56988]$ for p optimal.

G. Tree algorithms with control subchannels

In this section we determine the MST of some tree algorithms when a single slot consists of $g \geq 2$ control minislots and one data slot (with a length equal to one packet) [13]. Each of the g subchannels as well as the data channel provides separate feedback at the end of the slot (this is the DF case in [13]). These g control channels are used in the following manner: whenever a user transmits a packet on the data channel, the users also flips a fair g-sided coin and transmits a control signal in the corresponding control subchannel. We consider the same two types of feedback as in [13] for the control and data channels (resulting in 4 possibilities: BF/BF, BF/TF, TF/BF and TF/TF). The binary feedback (BF) on the control channel is somewhat different as it distinguishes between empty and nonempty slots (i.e., something/nothing), as opposed to the default collision/no collision.

The algorithms operate as follows. With BF/BF feedback, a collision creates a new slot for each nonempty subchannel, thus it operates as a g-ary splitting algorithm, except that it knows the identity of the empty groups and therefore refrains from assigning a slot to these groups. As such we can get the MST by looking at the expectation matrix

$$M^{(g)}_{BF/BF} = \chi_2(B^{(g)}_{BF/BF} E), \qquad (27)$$

where $B^{(g)}_{BF/BF}$ is identical to $B^{(q)}$, with $q=g$, except that the first column is equal to zero. As we cannot benefit in the usual way from ternary feedback on the data channel, we also use this algorithm for the BF/TF setting.

When TF/BF feedback is provided, a collision generates a slot for each subchannel holding a single signal, while for the subchannels holding a collision, two slots are provided by immediately splitting the collision in a binary manner. This implies that we are now faced with the following expectation matrix:

$$M^{(g)}_{TF/BF} = \chi_2(B^{(g)}_{TF/BF1} E + B^{(g)}_{TF/BF2} B^{(2)} E), \qquad (28)$$

where $B^{(g)}_{TF/BF1}$ is zero, except for the entries $(i,1)$ which are identical to those of $B^{(q)}$, with $q=g$, while $B^{(g)}_{TF/BF2}$ equals $B^{(q)}$, with $q=g$, except that the entries (i,j) for $j=0,1$ are zero.

Finally, under TF/TF feedback we can skip a guaranteed collision whenever a binary split of the TF/BF algorithm results in an empty left slot. The easiest way to express the expectation matrix is to assign a different type of branch to

g	BF/BF	TF/BF	TF/TF
2	0.376815	0.470771	0.481219
3	0.455546	0.503665	0.509799
4	0.488476	0.519728	0.523989
5	0.506464	0.529281	0.532518
10	0.538895	0.548256	0.549692
100	0.564490	0.565255	0.565381
10000	0.567116	0.567124	0.567125
∞	0.567143	0.567143	0.567143

TABLE IV
MAXIMUM STABLE THROUGHPUT OF TREE ALGORITHMS WITH g CONTROL CHANNELS (NOT COUNTING OVERHEAD).

all the skipped collisions, which results in

$$\bar{M}^{(g)}_{TF/TF} = \begin{bmatrix} \chi_2(B^{(g)}_{TF/BF1}E) & \chi_2(B^{(g)}_{TF/BF2}) \\ \chi_2((B^{(2)} - P^{(2)})E) & \chi_2(P^{(2)}) \end{bmatrix}, \tag{29}$$

which similar to Section IV-D2, can be reduced to a size $d-1$ matrix.

Table IV holds the MST for each of the algorithms when varying the number of control channels. In practice, the g channels require some fraction r of the length of a slot, implying that the data throughput is actually $(1+r)$ times smaller. Also note that all MSTs converge to the same value as g grows to infinity. This value corresponds to the upper bound for free access algorithms introduced by Kelly [24].

H. Tree algorithms on a channel with variable length packets

In this section we consider a system with variable length packets, where l_k denotes the probability that a packet is $k \geq 1$ slots long. As in [25], [26] we assume that the channel becomes reserved for the remainder of a multislot packet if it succeeds in transmitting its first slot successfully. Thus, all new arrivals that occur during a packet transmission only transmit in the slot following the last slot of the successful transmission. As in [25], all new arrivals that occurred during a successful transmission are resolved first and separately (with some possible even newer arrivals), which is easily established by allowing backlogged users to decrease their counter only upon seeing idle slots. In [26], the new arrivals are not resolved separately[4]. When the mean packet length is large (> 10) the approach considered here is slightly superior [25], otherwise [26] typically prevails.

It is worth noting that the model above can be regarded as a channel with a collision detection and a carrier sensing mechanism (CSMA-CD) in case a single slot corresponds to the time needed to detect an idle or busy channel, while the packet lengths are multiples of one time slot (see [10]).

To analyze this basic q-ary protocol with variable length packets via a branching process, it suffices to remark that the success slots now also induce some offspring. More specifically, the expectation matrix $M^{(q)}_{var}$ is of size d and is identical to $\chi_1(B^{(q)}E)$, except that entry i on the first row now holds the probability that i arrivals occur during a successful transmission:

$$(M^{(q)}_{var})_{1,i} = \sum_{k=1}^{\infty} l_k \exp(-\lambda k)(\lambda k)^i / i!. \tag{30}$$

[4]No exact results are provided for the MST in this case, only lower and upper bounds are established.

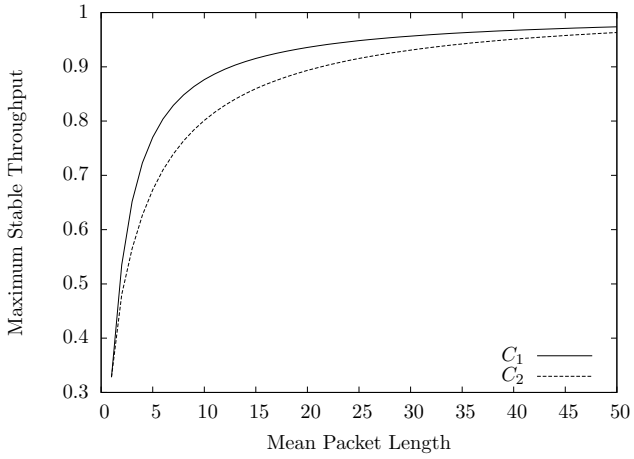

Fig. 8. Maximum stable throughput on a channel with variable length packets using fair coins ($p_1 = \ldots = p_q = 1/q$).

Figure 8 depicts the MST as a function of the mean packet length for $q = 2$ (the results for larger q values are analogue), in case of the C_1 curve all packets have the same length between 1 and 50, while for the C_2 curve packets have either length 1 or 100, where the probabilities are tuned to have the correct mean. These results coincide with [10], [25, Figure 1].

I. Tree algorithms on a channel with capture

In this section we consider a channel that allows capture to occur. We consider both the feedback with and without capture model of [12], where a group of dominating users (DG users) generates packets according to a Poisson process with rate λ_D and a group of nondominating users (NDG users) with rate λ_{ND}. The only difference with the basic q-ary protocol occurs when a collision takes place that consists of a single packet from a DG user, as well as $k > 0$ packets belonging to k NDG users; as the DG packet is captured, meaning it is still received correctly. As in [12], two types of feedback are considered: feedback with capture (FWC) and feedback without capture (FWOC). The FWC feedback allows the receiver to indicate that a capture event has occurred, therefore the $k > 0$ NDG packets are retransmitted in the next slot (together with possible new arrivals). In the FWOC case, the receiver uses a different signal for the success of a DG and NDG user, but is no longer able to determine whether any NDG users transmitted simultaneously in case of a DG success. Thus, after a DG success, the possibly empty set of NDG users is resolved first. We assume that both types of users use the same splitting probabilities, though this assumption can be relaxed easily.

To determine the throughput of the FWC model, we will use branches of type (i_{DG}, i_{NDG}) for slots that hold i_{DG} DG and i_{NDG} NDG packets. Hence, the expectation matrix is of

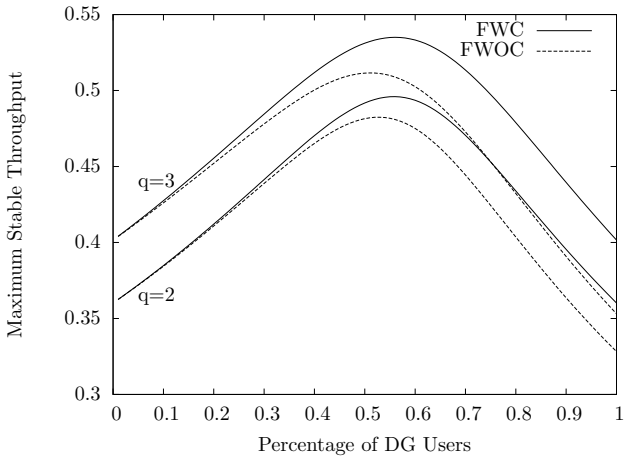

Fig. 9. Maximum stable throughput on a channel with capture using fair coins ($p_1 = \ldots = p_q = 1/q$).

size $(d+1)^2$. To construct it, we define the size $d+1$ matrices $B^{(q)}_{e,k}$, for $k = 1$ to q, as

$$(B^{(q)}_{e,k})_{i,j} = \begin{cases} C^i_j p^j_k (1-p_k)^{i-j} & i \geq 0, i \geq j \\ 0 & otherwise. \end{cases} \quad (31)$$

Entry (i, j) holds the probability that the k-th group holds j packets if a set of $i \geq 0$ packets is split. As the splitting events are all independent, the matrix B_{FWC} is defined[5] as $\sum_{k=1}^{q}(B^{(q)}_{e,k} \otimes B^{(q)}_{e,k})$, except that row $(0, 0)$, $(0, 1)$ and $(1, 0)$ is zero, as such slots generate no offspring, while the only nonzero entry on row $(1, i_{NDG})$, for $i_{NDG} > 0$, is entry $(0, i_{NDG})$, the value of which is one due to the capture event. The number of DG and NDG arrivals is also independent, therefore

$$M^{(q)}_{FWC} = B_{FWC}(E_D \otimes E_{ND}), \quad (32)$$

where E_x is identical to E with $b_i = \exp(-\lambda_x)(\lambda_x)^i/i!$ for $x = D$ and ND.

The branching process of the FWOC model is nearly identical, except that a $(1, 0)$ slot now also has some potential offspring as the receiver resolves the empty NDG group (together with some new arrivals) first. Hence, $M^{(q)}_{FWOC} = B_{FWOC}(E_D \otimes E_{ND})$ where B_{FWOC} is identical to B_{FWC}, except that its $(1, 0)$-th row now holds a 1 in position $(0, 0)$.

Figure 9 depicts the MST for $q = 2, 3$ as a function of the percentage of DG users (i.e., $\lambda_D/(\lambda_D + \lambda_{ND})$) for both the FWC and FWOC model. For both models, slightly higher throughputs are achieved when the DG group is somewhat larger than the NDG group, while the ternary scheme remains optimal (higher q values are not shown for clearness). For the FWOC model, the MST degrades quickly when most of the users are DG. This is as expected, as the NDG group hiding behind a DG success is often empty and therefore it is better not to resolve this set separately. In case all users become DG users, an MST of 0.328226 is attained, which corresponds to the MST of the variable packet length algorithm of [10], [25] and Section IV-H in case all packets have length 1.

[5] $\otimes$ denotes the Kronecker matrix product

k	1-by-1	2-by-2
1	0.5671432904	-
2	1.0750631447	0.91409902869
3	1.6185228340	1.48541522999
4	2.1908056123	2.11203466481
5	2.7862417308	2.74084924902
6	3.4007493405	3.39247963770
7	4.0313395238	4.05656034133
8	4.6757552422	4.73068257264
9	5.3322428980	5.41956847542
10	5.9994066059	6.11213027150

TABLE V
THE MST FOR THE COORDINATED SPLITTING ALGORITHM ON A CHANNEL WITH MULTIPLE RECEPTION CAPABILITIES.

J. Tree algorithms with coordinated splitting

In this section we study the MST of a multiple access algorithm under free access on a channel with multiple reception capabilities where the users that collide are allowed to communicate amongst each other. Hence, new arrivals still transmit immediately and the retransmitting users are unaware of the arrival times of the new packets.

A first possibility exists in achieving a collision free retransmission by splitting every collision of size i in i groups holding one packet. The branching process used for a channel that is able to resolve any collision of k or less packets, associates a type $c > k$ branch with each slot holding c packets. Hence, a type $c > k$ branch induces c slots each holding one retransmitted packet and possibly some new packets. Thus the expectation matrix becomes:

$$M_{cf} = \chi_{k+1}(BE), \quad (33)$$

where B is a size $d + 1$ matrix with entry (i, j) equal to i if $j = 1$ and zero otherwise. The resulting MSTs are presented in Table V. Notice, the value for $k = 1$ coincides with the well known upper bound of F. Kelly [24], as collision free retransmissions are optimal in case $k = 1$. For $k > 1$, retransmitting the packets one-by-one may not be optimal as a slot may be under utilized if no new arrivals are added. Table V also holds the results if the collisions are retransmitted two-by-two (where the last set contains only one packet if an odd number of packets was involved in a collision). It indicates that slightly higher MSTs can be realized for larger k. Retransmitting in groups of three packets did not further improve the MSTs for $k \leq 10$. The values in Table V are below the more general bounds presented in [21].

V. LIMITATIONS AND OTHER PERFORMANCE MEASURES

A. Limitations

The previous section demonstrates that the branching process technique often allows one to determine the MST with very little effort. There are however also a number of interesting cases that seem less suitable for the branching process technique.

a) Markovian components: A first set of cases are those where some of the model components have a Markovian nature: Markovian arrivals [16], [17], Markovian capture [27], Markovian noise [28], etc. In order to use the branching

process technique for such a system, one typically needs to know the probability that the Markovian component is in any particular state at the end of the CRI of the first group, as this influences the type of the second branch. These probabilities are however not readily available, except for some exceptional cases. For instance, if the underlying Markov chain of the Markovian noise has two states, such that there is never an error in state 1 and always an error in state 2. In this case, the state at the end of the CRI must be state 1, as an error is perceived as a collision and a CRI never ends with a collision.

b) Carrier sense mechanism: In the previous section we demonstrated that the MST of a channel with collision detection can be analyzed using a simple branching process. A carrier sense mechanism allows one to shorten the length of the idle slots to ℓ_c instead of the length of the collision slots. The problem that arises now is that the CRI of the left branch may end with an idle slot, implying that fewer new arrivals become part of the right branch. Hence, in order to determine the MST, we need the probability that a CRI of i users ends with an idle slot (under free access). Remark, in the slightly modified algorithm of [10], a CRI always ends in an idle slot and therefore we can determine the MST using a branching process as discussed in IV-H. Even for the original algorithm, we can still determine lower and upper bounds on the MST by bounding these probabilities. For instance the obvious upper bound of 1 indicates that the MST is below the MST of the channel with a collision detection mechanism (Section IV-D), when the time needed to detect an idle coincides with the time required for detecting a collision.

c) Erasers: When a successful or collision slot is incorrectly interpreted by the receiver as an idle slot, a channel eraser is said to occur [8], [20]. The users involved in this erasure are clearly aware of its occurrence, while the remaining users consider the current group as empty, i.e., resolved. There are two main approaches to address these erasures: either we consider the packets of the users involved in the erasure event as lost [8], or we retransmit them in the next time slot (that is, we use the persist strategy of [20]). The first case causes no problems when setting up a branching process, however, in the second case we need to know the probability that the CRI of a left child ends with an erasure of j packets, given that the CRI started with i packets (under free access).

It should be noted that the traditional analysis of *free* access algorithms using functional equations cannot be applied directly either as it requires the same probabilities. This is for instance also why only upper and lower bounds were established in [26]. The methodology that relies on tree structured Quasi-Birth-Death Markov chains developed in [16], [17] does not have these limitations, however, the computation times to get even a fairly rough approximation of the MST are considerably higher as one needs to solve a set of nonlinear matrix equations to determine whether an algorithm is stable for a specific arrival rate. This equation is typically solved using a fixed point iteration with linear convergence [29], implying that thousands or more iterations may be required to get a fairly accurate approximation. Furthermore, the effort required to set up these Markov chains is considerably higher in comparison with the simplicity of the branching process methodology (see [11], [17]).

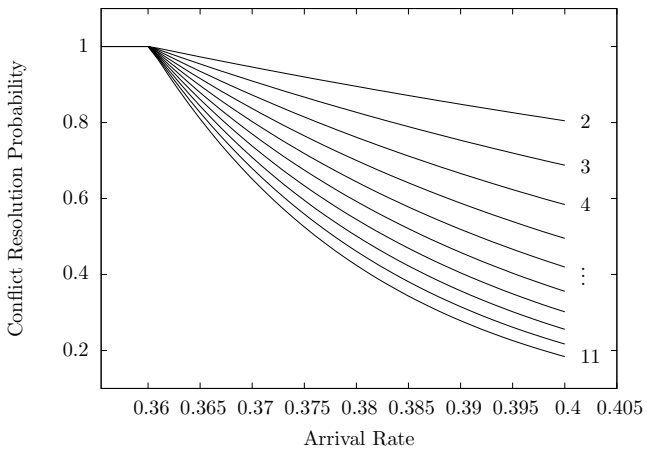

Fig. 10. Conflict resolution probability of a size 2 to 11 conflict as a function of the arrival rate λ for the basic binary tree algorithm.

B. Other performance measures

Apart from computing the MST for a variety of tree algorithms with free access, we can also exploit its corresponding branching process to determine the probability that a size i conflict gets resolved in a finite amount of time when the arrival rate exceeds the MST. These probabilities correspond to the entries of the extinction probability vector of the branching process, as the i-th entry of this vector gives us the probability that a tree rooted by a type i branch dies out in a finite amount of time. For a binary splitting algorithm, this vector is the smallest nonnegative solution to the extinction equation

$$x = a + A(x \otimes x), \tag{34}$$

where the vector a and matrix A were defined in Section III-A. This matrix equation can be solved efficiently using a Newton iteration [30]. In Figure 10 the conflict resolution probability of the basic binary tree algorithm with free access is depicted for size 2 to 11 conflicts. When λ exceeds the MST these probabilities suddenly drop below one and fan out as we move away from the MST. We are not aware of any prior results on these probabilities.

REFERENCES

[1] J. Capetanakis, "Tree algorithms for packet broadcast channels," *IEEE Trans. Inf. Theory*, vol. 25, no. 5, pp. 319–329, 1979.
[2] B. S. Tsybakov and V. Mikhailov, "Free synchronous packet access in a broadcast channel with feedback," *Problemy Peredachi Informatsii*, vol. 14, no. 4, pp. 32–59, 1978.
[3] D. Bertsekas and R. Gallager, *Data Networks*. Prentice-Hall Int., Inc., 1992.
[4] A. Ephremides and B. Hajek, "Information theory and communication networks: an unconsummated union," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2416–2434, October 1998.
[5] M. L. Molle and G. Polyzos, "Conflict resolution algorithms and their performance analysis," University of Toronto, CS93-300, Tech. Rep., 1993.

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

[7] P. Mathys and P. Flajolet, "Q-ary collision resolution algorithms in random-access systems with free or blocked channel access," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 2, pp. 217–243, 1985.

[8] N. Vvedenskaya and B. S. Tsybakov, "Random multiple access of packets to a channel with errors," *Problemy Peredachi Informatsii*, vol. 19, no. 2, pp. 69–84, 1983.

[9] R. M. Liang and H. H. Tan, "On the error analysis of single-channel free-access collision resolution algorithms," in *IEEE Aerospace Conference Proceedings, Vol. 1*, Big Sky, Montana, USA, 2000, pp. 129 – 140.

[10] P. Jacquet and E. Merle, "Analysis of a stack algorithm for CSMA-CD random length packet communication," *IEEE Trans. Inf. Theory*, vol. 36, no. 2, pp. 420–426, 1990.

[11] G. T. Peeters, B. Van Houdt, and C. Blondia, "A multiaccess tree algorithm with free access, interference cancellation and single signal memory requirements," *Performance Evaluation*, vol. 64, pp. 1041–1052, 2007.

[12] M. Sidi and I. Cidon, "Splitting protocols in presence of capture," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 2, pp. 295–301, March 1985.

[13] D. Towsley and P. Vales, "Announced arrival random access protocols," *IEEE Trans. Commun.*, vol. COM-35, no. 5, pp. 513–521, May 1987.

[14] N. Likhanov, E. Plotnik, Y. Shavitt, M. Sidi, and B. S. Tsybakov, "Random access algorithms with multiple reception capabilities and n-ary feedback channel," *Problemy Peredachi Informatsii*, vol. 29, no. 1, pp. 82–91, 1993.

[15] C. J. Mode, *Multitype Branching Processes*, R. Bellman, Ed. American Elsevier Publishing Company, Inc., 1971.

[16] B. Van Houdt and C. Blondia, "Stability and performance of stack algorithms for random access communication modeled as a tree structured QBD Markov chain," *Stochastic Models*, vol. 17, no. 3, pp. 247–270, 2001.

[17] ——, "Throughput of Q-ary splitting algorithms for contention resolution in communication networks," *Communications in information and systems*, vol. 4, no. 2, pp. 135–164, 2005.

[18] H. Mohamed and P. Robert, "Dynamic tree algorithms," *CoRR*, vol. abs/0809.3577, 2008.

[19] P. Flajolet and P. Jacquet, "Analytic models for tree communication protocols," INRIA, Tech. Rep. 648, 1987.

[20] I. Cidon and M. Sidi, "Erasures and noise in splitting multiple access algorithms," *IEEE Trans. Inf. Theory*, vol. IT-33, no. 1, pp. 132–143, January 1987.

[21] B. S. Tsybakov, V. Mikhailov, and N. B. Likhanov, "Bounds for packet transmission rate in a random-multiple-access system," *Problemy Peredachi Informatsii*, vol. 19, no. 1, pp. 61–81, 1983.

[22] Y. Yu and 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)*, March 2005, pp. 1908–1916.

[23] ——, "High-throughput random access using successive interference cancellation in a tree algorithm," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4628–4639, Dec. 2007.

[24] F. Kelly, "Stochastic models of computer communication systems," *Journal of the Royal Statistical Society (Series B)*, vol. 47, no. 3, pp. 379–395, 1985.

[25] P. Jacquet and E. Merle, "Analysis of a stack algorithm for random length packet communication," INRIA, Tech. Rep. 831, 1988.

[26] B. S. Tsybakov and S. Fedortsov, "Local-area network with random-multiple-access communications," *Problemy Peredachi Informatsii*, vol. 22, pp. 49–58, 1986.

[27] M. Seri and M. Sidi, "Splitting algorithms in channels with Markovian capture," *European Transactions on Telecommunications and Related Technologies*, vol. 5, no. 1, pp. 19–26, January-February 1994.

[28] I. Kessler and M. Sidi, "Splitting algorithms in noisy channels with memory," *IEEE Trans. Inf. Theory*, vol. IT-35, no. 5, pp. 1034–1043, September 1989.

[29] D. Bini, G. Latouche, and B. Meini, "Solving nonlinear matrix equations arising in tree-like stochastic processes," *Linear Algebra Appl.*, vol. 366, pp. 39–64, 2003.

[30] S. Hautphenne, G. Latouche, and M.-A. Remiche, "Newton's iteration for the extinction probability of a Markovian Binary Tree," *Linear Algebra and its Applications*, vol. 428, pp. 2791–2804, 2008.

Gino T. Peeters received his M.Sc. degree in Computer Science from the University of Antwerp (Belgium) in July 2005. In September 2005, he joined the "Performance Analysis of Telecommunication Systems" research group, at the Mathematics and Computer Science Department of the University of Antwerp. His main research interests include the performance evaluation of telecommunication systems, more specifically medium access control problems and scheduling issues in satellite communication networks.

Benny Van Houdt received his M.Sc. degree in mathematics and computer science, and a PhD in science from the University of Antwerp (Belgium) in July 1997, and May 2001, respectively. From August 1997 until September 2001 he held an Assistant position at the University of Antwerp. Starting from October 2001 onwards he has been a postdoctoral fellow of the FWO-Flanders. In 2007, he became a professor at the Mathematics and Computer Science Department of the University of Antwerp, where he is a leading member of the PATS research group. His main research interest goes to the performance evaluation and stochastic modelling of wired and wireless communication networks and random access systems in particular. Other areas of interest include manufacturing, operating systems, tool development, etc. He has published various papers, containing both theoretical and practical contributions, in a variety of international journals (e.g., IEEE JSAC, Performance Evaluation, Journal of Applied Probability, Stochastic Models, Queueing Systems, etc.) and in conference proceedings (e.g., ACM Sigmetrics, Networking, Globecom, Opticomm, ITC, etc.).