Improved High Maximum Stable Throughput FCFS Tree Algorithms with Interference Cancellation

G.T. Peeters
gino.peeters@ua.ac.be

B. Van Houdt
benny.vanhoudt@ua.ac.be

University of Antwerp - IBBT
Middelheimlaan 1
B-2020 Antwerp, Belgium

ABSTRACT

Tree algorithms form a well researched class of collision resolution algorithms for solving multiple access control problems. Signal interference cancellation, which allows one to recover additional information from otherwise lost collision signals, has recently been combined with tree algorithms, providing substantially higher maximum stable throughputs (MSTs), up to 0.693 (Yu and Giannakis, IEEE Transactions on Information Theory, Vol. 53(12), 2007). We propose two novel First-Come-First-Served tree algorithms, the operation of which is similar to the well-known tree splitting algorithms, that exploit interference cancellation and derive their MST. Furthermore, these algorithms are also designed such that, at all times, it suffices to store only one signal.

Categories and Subject Descriptors

C.4 [**Performance Of Systems**]: Performance attributes; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network communications*

General Terms

Algorithms, Performance

Keywords

Interference cancellation, random-access, maximum stable throughput.

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 DVB-RCS satellite networks and hybrid-fiber-coaxial (HFC) networks (i.e., DOCSIS 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]. These algorithms were the first to have a provable MST above zero. Afterward 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 packet for the first time. The CRA corresponds to the algorithm all users must use to resolve collisions (i.e., simultaneous transmissions). The easiest and cheapest CAP is *free access*, meaning new packets may be transmitted without any further delay (on the slot boundaries). Other important CAPs include *blocked* (or gated) and *windowed* (or grouped) access.

When blocked (or gated) access is used, an initial collision of n stations causes all subsequent new arrivals to postpone their first transmission attempt until the n initial stations have resolved their collision. The time elapsed from the initial collision until the point where the n stations have transmitted successfully is called the collision resolution period (CRP). Suppose that m new packets are generated during the CRP. Then, a new CRP starts (with m participants) when the previous CRP (with n stations involved) ends. In short, when the blocked access mode is used new arrivals are blocked until the CRP during which they arrived has ended. They will participate in the next CRP.

Finally, windowed (or grouped) access works as follows. Suppose that the random access scheme is activated at time $k = 0$. The unit of time is defined as the length of a slot, so that the i-th transmission slot is the time interval $(i, i+1]$. A second time increment α_0 is chosen and the i-th arrival window is defined as the time interval $(i\alpha_0, i\alpha_0 + \alpha_0]$ (α_0 is not necessarily an integer value). The first transmission rule used by this algorithm is as follows: transmit a new packet that arrived during the i-th arrival window in the first "utilizable" slot following the CRP that resolves the packets belonging to the $(i-1)$-th arrival window. The modifier "utilizable" reflects the fact that the CRP of the $(i-1)$-th arrival window might end before the i-th arrival window itself has ended. If so, a number of transmission slots is skipped until the i-th arrival window ends. One could improve the algorithm by shortening the i-th arrival window. This complicates the analysis and has no influence on the maximum stable throughput [1].

When a binary tree algorithm [3, 12] is used as the CRA in combination with windowed access, a collision of n packets belonging to the same window of length α_0 will cause the set of colliding packets to split into two groups by partitioning the length α_0 arrival window into two windows of length $p\alpha_0$ and $(1-p)\alpha_0$ (with $0 < p < 1$), respectively. The first group of colliding packets then consists of all the packets that arrived during the first (length $p\alpha_0$) window, while the remaining packets that arrived in the second (size $(1-p)\alpha_0$) window join the second group. Stations joining the first group retransmit in the next slot and resolve a possible collision recursively, while the packets of the second group must wait until the first group is completely resolved before applying the same algorithm. Typically, p is chosen to be 0.5, however, in some cases an unequal splitting may provide a slightly higher MST. Notice, using this CRA algorithm with windowed access guarantees that the packets are received in a First-Come-First-Serve (FCFS) order. As explained further on, when we combine this algorithm with an interference cancellation mechanism the FCFS order is not preserved.

Two important improvements have been made to the algorithm above. First, if a first group is empty (and thus resolved in one slot), we can immediately split the second group as it is certain to hold a collision. Second, if the first group, corresponding to some length $2^{-i}\alpha_0$ window, holds a collision, we know nothing about the size of the second group, as such the window of the second group is postponed to the next CRP and all the subsequent initial size α_0 windows are shifted forward by $2^{-i}\alpha_0$. When combined with an interference cancellation mechanism, this second observation no longer applies.

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 a successive interference cancellation (SIC) mechanism, was designed and shown to achieve an MST as high as 0.693 [15, 16]. SICTA uses a blocked access CAP and requires a (theoretically) unbounded amount of memory for storing signals (actually, SICTA with windowed access performs optimal when $\alpha_0 = \infty$, which corresponds to blocked access). The interference cancellation mechanism works as follows. 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. Thus, when combined with a tree algorithm, interference cancellation offers the possibility to obtain the signal of the second group by canceling the signal of the first group from the joint signal, removing the need to assign a slot to the second group, thereby significantly improving the channel throughput.

In [10] we introduced a novel tree algorithm using SIC for the *free* access CAP that requires the storage of at most one signal at a time, achieving minimal memory requirements. The MST of this algorithm was proven to be 0.5698, using tree-like processes [2]. Remark, random access algorithm with free access typically achieve the lowest MST. A disadvantage of this algorithm is that packets are not necessarily received in an FCFS order, while this order is well known to reduce the variance of the delay. Using the same memory limitations (algorithms for more memory locations can be constructed similarly), we proposed two novel tree algorithms in [13], which are able to maintain the FCFS order, by using the windowed access CAP. Both algorithms exploit the interference cancellation mechanism combined with the 0.4871 FCFS algorithm [1] (the 0.4878 algorithm was obtained from this algorithm by optimizing the window length of the first and second group for each level in the tree using dynamic programming techniques). The original 0.4871 algorithm works similar to the basic tree algorithm, except that it implements the two important improvements mentioned above. As indicated above, using SIC, the combined signal of any second group can be deduced from the first group and the joint signal. This allowed us to improve the 0.4871 MST using the following two observations: (i) when a collision splits into two successes, we can recover the second success from the cancellation operation and therefore we gain a single slot, (ii) we still have some knowledge about the groups that are postponed to the next CRP (due to a collision in their corresponding first group), allowing us to skip the last postponed windows in case they are empty. Other information, additionally provided by SIC, is hard, if not impossible, to exploit as this would often violate the FCFS order. These algorithms were shown to have an MST of 0.6048 and 0.6173, respectively.

In this paper, we propose two additional tree algorithms, which operate under the same constraints: single signal memory and FCFS order preservation. Both algorithms achieve a higher MST, by utilizing the windowed access variant of [10], adapted such that packets arrive in FCFS order. The paper is structured as follows. In Section 2, we start by briefly discussing the SICTA algorithm, which makes use of SIC and a (theoretically) unbounded amount of signal memory, to achieve an MST of 0.693, without maintaining the FCFS order. In Section 3 we continue by describing and analyzing the windowed access variant of [10], with minimal signal memory requirements, however, still without the guarantee that the packets are received in an FCFS order. In Section 4, we will discuss how this algorithm can be modified to maintain the FCFS order, with a significant penalty in terms of the MST. In Sections 5 and 6, we propose two methods to reduce this performance penalty. The resulting algorithms will be shown to have an MST of 0.6327 and 0.6334 respectively.

2. THE 0.693 SICTA ALGORITHM

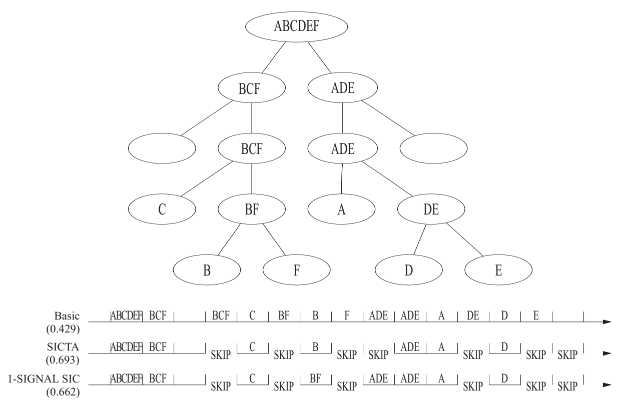

Figure 1: ***Illustration of the collision resolution in a single CRP for the basic tree, the SICTA and a single signal memory SIC algorithm. The slots are transmitted in a depth-first-search traversal of the collision tree, except for the slots which can be skipped using interference cancellation. The MST of each algorithm is indicated, given the optimal windowed or blocked CAP.***

The 0.693 SICTA algorithm [15, 16] is conceptually identical to the basic tree algorithm, applied to windowed or blocked access. As illustrated by Figure 1, each CRP is resolved, by recursively splitting the collision tree. At each splitting stage, the colliding terminals decide to choose either the left of right branch, based on a (pseudo) random decision, for instance by generating a random number, or using the subsequent bits of the arrival timestamp. In this way, eventually every packet is transmitted in a separate slot, solving the initial collision.

SICTA achieves its high MST, by combining SIC with an (theoretically) unlimited amount of memory to store the signals of previous collisions. A motivating example consists of a collision of two packets, which requires only the transmission of the initial collision, and one single packet; the other packet can be recovered by SIC. The same reasoning can be applied to the entire collision tree; indeed, the signal of every right branch can be obtained by subtracting the signal of the corresponding left branch from the common parent node (the signal of which may also be the result of an interference cancellation operation, hence the name successive interference cancellation). A (theoretically) unlimited amount of signal memory is required, since a collision may require an unbounded number of splitting decisions; during this process, each decoded right branch, which cannot be resolved immediately, has to be stored for later conflict resolution.

If we ignore the root node, only half the number of slots are required to resolve a conflict, compared to the basic tree algorithm with blocked access. As a result, for blocked access an MST as high as 0.693 can be obtained (which is twice the MST of 0.3465 achieved by the basic tree algorithm with blocked access, the 0.429 mentioned in Figure 1 is the MST of the windowed access CAP with an optimal α_0). When combined with windowed access, the larger the initial window is chosen, the higher the achievable MST becomes, as illustrated by Figure 2. Effectively, SICTA performs optimal when the initial window is infinitely large, which corresponds to blocked access.

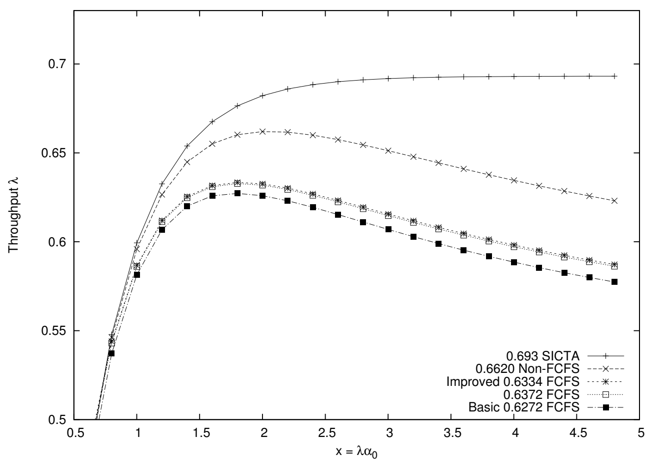

Figure 2: ***Throughput for all algorithms, for the windowed access CAP. For the algorithms with additional parameters β_0 and p, the optimal value was chosen.***

3. A 0.6620 NON-FCFS SPLITTING ALGORITHM

In an earlier paper [10], we designed a variant of the SICTA algorithm, which only requires a single signal memory location. Similar to SICTA, the combination of the basic tree algorithm with successive interference cancellation allowed us to skip some slots part of the conflict resolution tree. Our approach consisted in storing the signal of the parent node in the single memory location. By subtracting the signal of the left branch, SIC revealed the signal information of the right branch. The single signal memory requirements allowed to skip this right set, whenever there was a success (or an empty slot) detected in either (or both) the left or right branch. Otherwise, this right branch was retransmitted after resolving the left branch in order to recover the joint signal of the right branch.

We also established an expression for the mean time L_N required to resolve a conflict of size N, in case of blocked access. $L_0 = L_1 = 1$, while L_N obeys the following recursion:

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

where $\delta_{N,2} = 1$ if $N = 2$ and zero otherwise Indeed, we need one slot for the initial transmission, while the operation of the basic CTMV algorithm can be recognized in the recursive expression for the mean time required to solve the left and right branch, for all possible splitting combinations. The last five terms represent the slots that SIC allows us to skip, that is, whenever either the left or right branch consists of zero or one packet, we gain a single slot. The very last term is required to correct the case where a collision of two splits into one-one, for which we still need a single slot.

Using a similar methodology as the one provided by Mathys and Flajolet [9] for analysis of the basic and modified CTMV algorithm, we also determined a closed-form formula for L_N:

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

In this case, $p = 0.5$ is optimal.

For a windowed access system, given Poisson arrivals with rate λ and an initial window of size α_0, the average length $E\{k\}$ of a CRP can be calculated as follows:

$$E\{k\} = \sum_{N=0}^{\infty} L_N \frac{(\lambda\alpha_0)^N e^{-\lambda\alpha_0}}{N!}.$$

In order to have a stable system, the average length $E\{k\}$ of a CRP must be less than the average distance that the starting point of the allocation window advances between two (initial) windows, which equals α_0. By multiplying both sides with λ, we can rewrite this as

$$\lambda < \frac{\lambda\alpha_0}{E\{k\}},$$

where the right hand side of this equation is a function f of $\lambda\alpha_0$. By numerically maximizing this function, denoting x_{max} as the point in which the maximum is reached and $f(x_{max})$ as its maximum value, we obtain the highest possible maximum stable throughput λ_{max} by setting $\alpha_0 = x_{max}/f(x_{max})$. For the function above, as shown by Figure 2, the maximum is reached in $\lambda\alpha_0 = 2.0596$, for $p = 0.5$, resulting in $\alpha_0 = 3.1109$ and $\lambda_{max} = 0.6620$. These values have also been confirmed using simulation experiments.

4. A BASIC 0.6272 FCFS SPLITTING ALGORITHM

The previous algorithm did not maintain the FCFS order in case the interference cancellation detects a success in a right branch, while the left branch still contains a conflict that has to be resolved. As such, a straightforward way to obtain an FCFS order, is by refraining from skipping the right branch in such cases.

In order to analyze this modified algorithm, we need to determine the mean time $\bar{L}_N$ required to resolve a conflict of size N. Clearly, $\bar{L}_0 = \bar{L}_1 = 1$ and $\bar{L}_N$ obeys the following recursion:

$$\begin{aligned}\bar{L}_N = 1 + \sum_{i=0}^{N} \binom{N}{i} p^i (1-p)^{N-i} \left(\bar{L}_i + \bar{L}_{N-i}\right) \\ -p^N - (1-p)^N - Np(1-p)^{N-1}.\end{aligned}$$

This recursion is similar to Equation (1), except for the last two terms. These terms correspond to the slots which were revealed by SIC, but not in an FCFS order; now SIC only allows us to skip a slot if all the colliding packets are either in the left or right branch, or when the left branch contains exactly one packet. In order to generate numerical results, it suffices to have this recursive expression. However, a closed form expression can be obtained analogue to the analysis of the basic and modified CTMV algorithm by Mathys and Flajolet [9]. Define $L(z) = \sum_{N=0}^{\infty} \bar{L}_N z^N/N!$, then we find the following functional equation for $L(z)$:

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

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

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

Equating the coefficients of z^k on both sides implies

$$\begin{aligned}&L^*_k(1 - p^k - (1-p)^k) = \\ &\quad (-1)^k/k! \left(k(1-p) - p^k - (1-p)^k + kp^k\right),\end{aligned}$$

for $k \geq 2$. This finally results in

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

Similar to previous section, by numerically maximizing the function $f(\lambda\alpha_0) = \frac{\lambda\alpha_0}{E\{k\}}$, we obtain the highest possible maximum stable throughput for $p = 0.5$ in $\lambda\alpha_0 = 1.7674$, resulting in $\alpha_0 = 2.8230$ and $\lambda_{max} = 0.6260$. Setting $p = 0.5$ is however no longer optimal. More specifically, the optimal MST is found by choosing $p = 0.471$, obtaining a maximum in $\lambda\alpha_0 = 1.7850$, which results in $\alpha_0 = 2.8458$ and $\lambda_{max} = 0.6272$ (see also Figure 2). These values have also been confirmed using simulation experiments.

An intuitive explanation for having an optimal $p < 0.5$ can be given as follows: consider a collision that splits in $1 - \mathit{coll}$ or $\mathit{coll} - 1$. The first option requires one slot, as the collision signal *coll* can be recovered using SIC. The second option however, requires two slots, due to FCFS constraints. As such, a slightly higher probability of having the first option, i.e., choosing $p < 0.5$, improves the MST.

5. A 0.6327 FCFS SPLITTING ALGORITHM WITH POSTPONED SUCCESSES

The previous algorithm revealed a performance loss, caused by the fact that we ignore the knowledge about successes in the right branch, when the left branch contains a collision. Indeed, to preserve the FCFS order, these packets cannot be considered as received at this point. However, we can still exploit this knowledge to improve the MST. We will first do so if the ignored success is contained in the very last slot of the collision resolution tree. The motivation behind this approach is to combine this last slot with a (small) part of length $\beta_0 \leq \alpha_0$ of the next CRP, in the hope that this part of the next CRP contains no arrivals. If this is the case, we have successfully solved a (slightly) larger interval at no additional cost (see Figure 3). Moreover, in case this additional interval is non empty (and has size $\beta_0 = \alpha_0$), no additional slots are required if during the first split operation, we choose the splitting such that the left branch is precisely the success of the previous CRP, as demonstrated in Figure 4. When $\beta_0 < \alpha_0$, there is in general no guaranteed gain as the arrival windows get shifted by β_0. Since a higher probability of having this next CRP empty will increase the effects of this modification, there is a trade-off to

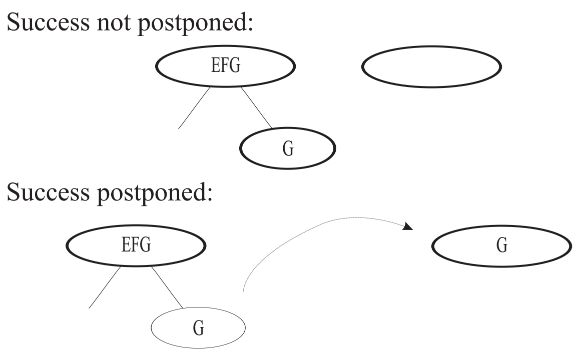

Figure 3: ***This figure illustrates that postponing the last success to the next CRP will gain an additional slot, when no new arrivals occur. Here, G is postponed and combined with the empty slot of the following short CRP. Without postponing, this would require two slots: one for the success and one for detecting the empty CRP (slots drawn using hairlines are not transmitted).***

be made when selecting β_0: a shorter arrival window will result in a higher probability of having an empty CRP, while the interval that is resolved becomes smaller.

There is some resemblance between this idea and the approach taken by Vvedenskaya [14]; there, the idea was that a collision in a very small interval consists most likely of only two packets. Therefore, whenever the splitting process revealed the first packet, the transmission of the other packet was immediately combined with a new CRP. However, the positive impact on the MST was minimal, for two reasons: first, the occurrence of a collision in a very small interval is rare. Second, whenever this recombination caused an additional conflict, for example because there were three arrivals in the original collision slot, this did result in a performance penalty.

To analyze the performance of this algorithm, we need to distinguish between two types of CRPs: standard CRPs (Type 0) which have an initial arrival window of size α_0, and short CRPs (Type 1), which combine a success postponed from the previous CRP with a new arrival window of size β_0. The probability $p_{0,1}$ of having a standard CRP followed by a short CRP is given by:

$$p_{0,1} = \sum_{i=1}^{\infty}(1-(1+pR_i)e^{-pR_i})(1-p)R_i e^{-(1-p)R_i},$$

where $R_i = \lambda\alpha_0(1-p)^{i-1}$. Indeed, if the initial arrival window ends with a collision (an $\alpha_0 p(1-p)^{i-1}$ interval), followed by a success (corresponding to an $\alpha_0(1-p)^i$ interval for some $i > 0$), the transmission of this success packet is postponed and combined with the following, short CRP.

Similarly, the probability $p_{1,1}$ of having a short CRP followed by another short CRP is given by:

$$p_{1,1} = \sum_{i=1}^{\infty}(1-(1+pR'_i)e^{-pR'_i})(1-p)R'_i e^{-(1-p)R'_i},$$

where $R'_i = \lambda\beta_0(1-p)^{i-1}$. Notice, a short CRP always has an initial window larger than β_0, however, the first right branch is always of size β_0.

To determine the probability that an arbitrary CRP is standard or short, we set up a two-state discrete time Markov

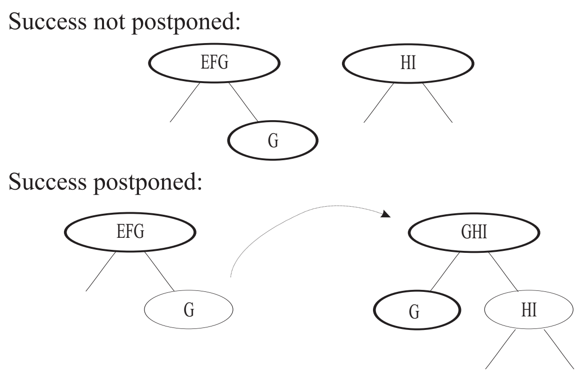

Figure 4: ***This figure illustrates that postponing the last success to the next CRP will not cost us an additional slot, when new arrivals occur. In this example, packet G is postponed, and combined with H and I. Only the slots GHI and G have to be transmitted, to have the information about slots G and HI; the signals in the remaining slots, which are drawn using hairlines, can be recovered from this using SIC mechanism. If G was not postponed, two slots were also required to retrieve the same information.***

chain with transition matrix

$$P = \begin{bmatrix} p_{0,1} & 1-p_{0,1} \\ 1-p_{1,1} & p_{1,1} \end{bmatrix}.$$

The invariant vector $\pi = (\pi(0), \pi(1))$ of this transition matrix P gives us the probabilities $\pi(0)$, $\pi(1)$ that a given CRP is standard or short respectively, by:

$$\pi(0) = \frac{1-p_{1,1}}{1-p_{1,1}+p_{0,1}}, \pi(1) = \frac{p_{0,1}}{1-p_{1,1}+p_{0,1}}.$$

Next, we need to determine the mean time $\dot{L}_N$ and $\ddot{L}_N$, in time slots, required to resolve a conflict of size N, for both a standard CRP and a short CRP. Before we proceed, we must define to which CRP a slot belongs, since a short CRP is started before the previous CRP has finished completely. That is, when a short CRP is started, it takes one or two slots before the postponed arrival of the previous CRP is transmitted successfully. Only the last of these possible two slots will be associated with the previous CRP. More precisely, in case there is no new arrival in a short CRP, a single slot is transmitted (containing the postponed success of the previous CRP): this slot is considered part of the previous CRP. Otherwise, the second slot of this short CRP will contain the postponed success and thus is associated with the previous CRP, while the first slot, containing also the new arrivals, is part of the current, short CRP.

Using these definitions, we now argue that $\dot{L}_N = \ddot{L}_N = \bar{L}_N$ for $N > 0$, with $\bar{L}_N$ expressed by Equation (2), while $\dot{L}_0 = \bar{L}_0 = 1$ and $\ddot{L}_0 = 0$. Clearly, for a standard CRP with no postponed success, this follows readily. When a success in a standard CRP is postponed to the following short CRP, we count one of the slots in the subsequent short CRP as part of the standard CRP, while saving the postponed slot in the standard CRP, resulting in $\dot{L}_N = \bar{L}_n$. For a short CRP with no new arrivals, the only slot that is transmitted after the start of the CRP, is part of the previous CRP, as defined previously, so $\ddot{L}_0 = 0$. For a short CRP with new arrivals and no postponed success, we see that the root of the collision tree for this CRP (including the previously postponed success) minus its left branch (containing only the postponed success), results in the right branch slot, containing only the new arrivals, as illustrated by Figure 4. From these three slots, the root is effectively transmitted; the left branch slot is also transmitted, however it is counted as belonging to the previous CRP. In this manner, the right branch, which is obtained using SIC, induces a collision resolution tree, identical to a standard CRP, with the same number of slots. For a short CRP with a postponed success, we can use a similar argument as for the standard CRP. Thus, we obtain $\ddot{L}_N = \bar{L}_N$ for $N > 0$.

Given Poisson arrivals with rate λ, an initial window of size α_0 and β_0 for standard and short CRPs respectively, the average length $E\{k\}$ of a CRP can be calculated as follows:

$$E\{k\} = \sum_{N=0}^{\infty} \frac{\dot{L}_N \pi(0)(\lambda\alpha_0)^N e^{-\lambda\alpha_0} + \ddot{L}_N \pi(1)(\lambda\beta_0)^N e^{-\lambda\beta_0}}{N!}.$$

The average distance $E\{s\}$ that the starting point of the allocation window advances between two (initial) R windows equals:

$$E\{s\} = \pi(0)\alpha_0 + \pi(1)\beta_0.$$

Similar to the previous algorithms, we have a stable system if

$$\lambda < \frac{\lambda E\{s\}}{E\{k\}},$$

where the right hand side of this equation is a function f of $\lambda\alpha_0$ and $\lambda\beta_0$. By numerically maximizing this function, denoting (x_{max}, y_{max}) as the point in which the maximum is reached and $f(x_{max}, y_{max})$ as the maximum value, we obtain the highest possible maximum stable throughput λ_{max} for $p = 0.5$ by setting $\alpha_0 = x_{max}/f(x_{max}, y_{max})$ and $\beta_0 = y_{max}/f(x_{max}, y_{max})$. For the function above, the maximum is reached in $(\lambda\alpha_0, \lambda\beta_0) = (1.8016, 1.0156)$, resulting in $\alpha_0 = 2.8517$,$\beta_0 = 1.6078$ and $\lambda_{max} = 0.6317$.

As with the previous algorithm, the optimal p is however not located in $p = 1/2$. The optimal MST corresponds to setting $p = 0.474$, obtaining a maximum in $(\lambda\alpha_0, \lambda\beta_0) = (1.8170, 1.1017)$ as illustrated by Figure 5, which results in $\alpha_0 = 2.8717$, $\beta_0 = 1.6067$ and $\lambda_{max} = 0.6327$. These values have also been confirmed using simulation experiments. Intuitively, a similar argument as in Section 4 applies, to explain that the optimal $p < 0.5$. However, as the penalty for splitting *coll* $- 1$ is now slightly reduced, a slightly less asymmetric splitting is required, implying that the optimal p exceeds 0.471.

6. AN IMPROVED 0.6334 FCFS SPLITTING ALGORITHM WITH POSTPONED SUCCESSES

The modification introduced in the previous algorithm can also be applied in some more cases. Consider the last success of the CRP and suppose it has a conflict in its corresponding left branch. Previously, the modification was only applied when such an event occurred at the very end of the initial arrival window. However, if there is no other arrival between the last success and the end of the initial arrival window, the modification can also be applied. In such cases, one or more empty slots may follow (in depth-first-search order) the last success in the collision tree.

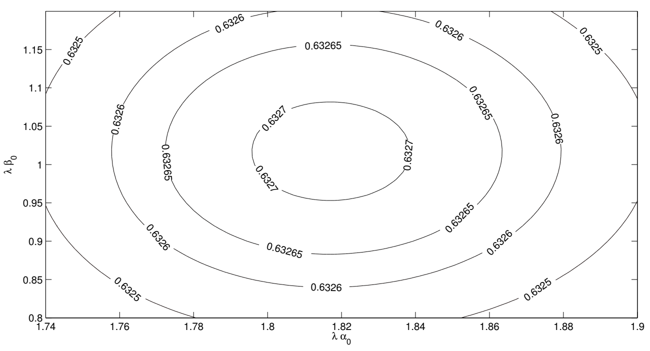

Figure 5: ***Illustration of the optimal point*** $(\lambda\alpha_0, \lambda\beta_0) = (1.8170, 1.1017)$***, for the*** 0.6327 ***algorithm, for*** $p = 0.474$***. The throughput for the various combinations are indicated.***

The analysis of this new algorithm is more involved than the previous one, as the probabilities $p_{i,j}$ that a Type i CRP is followed by a Type j CRP ($i, j \in \{0, 1\}$) are much harder to determine. First, we find all possible tuples of the form (C, S, R), with C, S and R representing fractions of the initial window. C and S correspond to intervals of a left and right child of the same parent node, while R is the fraction of the initial arrival window located right of the S window. Notice, each tuple can be associated with a possible short CRP following the current CRP; for example, starting in a standard CRP, if we have a collision in an interval of length $C\alpha_0$, a success in an interval of length $S\alpha_0$ and no arrival in the remaining interval $R\alpha_0$. The set of all tuples can be constructed on a per level base: define S_i as the set of tuples (C, S, R), with the corresponding parent node of C and S at depth i in the collision tree. Then, for $i = 0$, we have

$$\mathcal{S}_0 = \{(p, 1-p, 0)\}.$$

For $i > 0$, we can have the success S in either the main left or the main right branch; for the main left branch (fraction p), the entire right branch (fraction $1 - p$) has to be empty, while when the success occurs in the right branch, there is no additional requirement for the left branch. This yields that S_i can be constructed recursively as follows:

$$\begin{aligned}\mathcal{S}_i = \{&(pC, pS, pR + (1-p)) | (C, S, R) \in \mathcal{S}_{i-1}\} \\ &\cup \{((1-p)C, (1-p)S, (1-p)R) | (C, S, R) \in \mathcal{S}_{i-1}\}.\end{aligned}$$

For $p = 0.5$, we can provide simple closed expressions for the tuples belonging to $\mathcal{S}_i$:

$$\mathcal{S}_i = \{(2^{-i-1}, 2^{-i-1}, 2^{-i}k) | k = 0, \ldots, 2^i - 1\}.$$

The probability $p_{0,1}$ of having a standard CRP followed by a short CRP now becomes:

$$p_{0,1} = \sum_{(C,S,R) \in \{\mathcal{S}_0, \mathcal{S}_1 \ldots\}} (1 - (1 + \lambda\alpha_0 C)e^{-\lambda\alpha_0 C})\lambda\alpha_0 S e^{-\lambda\alpha_0 S} e^{-\lambda\alpha_0 R}.$$

Similarly, the probability $p_{1,1}$ of having a short CRP followed by a short CRP is given by:

$$p_{1,1} = \sum_{(C,S,R) \in \{\mathcal{S}_0, \mathcal{S}_1 \ldots\}} (1 - (1 + \lambda\beta_0 C)e^{-\lambda\beta_0 C})\lambda\beta_0 S e^{-\lambda\beta_0 S} e^{-\lambda\beta_0 R}.$$

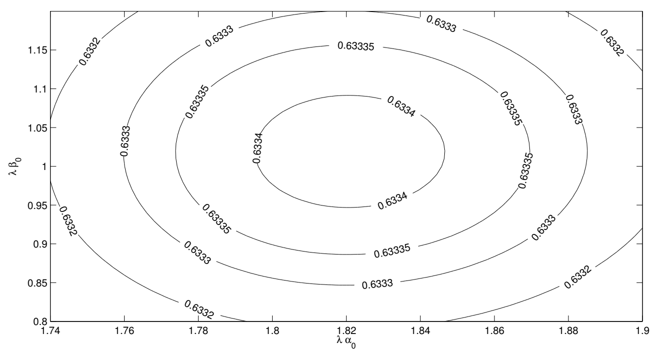

Figure 6: ***Illustration of the optimal point*** $(\lambda\alpha_0, \lambda\beta_0) = (1.8207, 1.1082)$***, for the*** 0.6334 ***algorithm, for*** $p = 0.476$***. The throughput for the various combinations are indicated.***

For $p = 0.5$, this can be simplified to:

$$p_{0,1} = \sum_{i=1}^{\infty}(1-(1+P_i)e^{-P_i})P_i e^{-P_i}\left(\sum_{k=0}^{2^{i-1}-1} e^{-2kP_i}\right),$$

$$p_{1,1} = \sum_{i=1}^{\infty}(1-(1+P_i')e^{-P_i'})P_i e^{-P_i'}\left(\sum_{k=0}^{2^{i-1}-1} e^{-2kP_i'}\right),$$

with $P_i = \lambda\alpha_0 2^{-i}$ and $P_i' = \lambda\beta_0 2^{-i}$. Notice, if we only consider tuples of the form $(C, S, R) \in S_i$ with $R = 0$, we obtain the same expressions as for the previous algorithm.

By numerically maximizing the resulting function, we obtain the highest possible maximum stable throughput λ_{max} of 0.6325 for $p = 0.5$ in $\alpha_0 = 2.8548$ and $\beta_0 = 1.6093$. Choosing $p = 0.476$, we obtain a maximum in $\alpha_0 = 2.8743$ and $\beta_0 = 1.6074$, as demonstrated by Figure 6, resulting in an MST of $\lambda_{max} = 0.6334$. These values have also been confirmed using simulation experiments. Similar to Section 5, the penalty for splitting *coll* $- 1$ is now even further reduced, providing an intuitive explanation for an even more symmetric splitting.

7. CONCLUSION

In this paper, we introduced a number of novel tree algorithms, which use interference cancellation and preserve an FCFS order. These algorithms were designed such that, at all times, at most one signal must be stored. Starting, from a non-FCFS algorithm, we showed how to limit the penalty for making it FCFS, by postponing successes from one CRP to the next CRP. For each algorithm, we showed how to assess its performance in terms of the MST. The highest MST achieved in this manner was 0.6334 using an FCFS algorithm using a single signal memory location.

8. REFERENCES

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

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

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

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

[5] N. Golmie, F. Mouveaux, and D. Su. A comparison of MAC protocols for hybric fiber/coax networks: IEEE 802.14 vs. MCNS. In *Proceedings of the 16th International Conference on Communications*, pages 266–272, Vancouver, Canada, June 1999.

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

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

[8] S. Khanna, S. Sarkar, and I. Shin. An energy measurement based collision resolution protocol. In *Proceedings of the 18-th ITC conference*, Berlin Germany, 2003.

[9] P. Mathys and P. Flajolet. Q-ary collision resolution algorithms in random-access systems with free or blocked channel access. *IEEE Trans. Inf. Theory*, IT-31(2):217–243, 1985.

[10] 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*, 64:1041–1052, 2007.

[11] G. Polyzos and M. Molle. Performance analysis of finite nonhomogeneous population tree conflict resolution algorithms using constant size window access. *IEEE Trans. Commun.*, 35(11):1124–1138, 1987.

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

[13] B. Van Houdt and G. T. Peeters. FCFS tree algorithms with interference cancellation and single signal memory requirements. In *Proceedings of the International Workshop on Multiple Access Communications (MACOM)*, Saint-Petersburg, Russia, 2008.

[14] N. D. Vvedenskaya and M. S. Pinsker. Non-optimality of the part-and-try algorithm. In *Abstracts of the International Workshop on Convolutional Codes and Multiuser Communications*, pages 141–148, Sochi, USSR, 1983.

[15] 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)*, pages 1908–1916, March 2005.

[16] Y. Yu and G. B. Giannakis. High-throughput random access using successive interference cancellation in a tree algorithm. *IEEE Trans. Inf. Theory*, 53(12):4628–4639, Dec. 2007.