FCFS tree algorithms with interference cancellation and single signal memory requirements

FCFS Tree Algorithms with Interference Cancellation and Single Signal Memory Requirements

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

Abstract

Tree algorithms are a well studied class of collision resolution algorithms for solving multiple access control problems. 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 (MST). We propose two novel First-Come-First-Served tree algorithms, the operation of which is similar to the well-known 0.4871 *FCFS tree algorithm, that exploit interference cancellation and derive their MST. Both these algorithms are also designed such that, at all times, at most one signal must be stored.*

Keywords: Random-access, Maximum stable throughput, Interference cancellation

1. Introduction

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

Tree algorithms also strongly outperform the class of backoff algorithms (including the BEB) in terms of their maximum stable throughput (MST) [1]. In the standard information theoretical setting, the MST is defined as the highest possible (Poisson) input rate for which a packet has a finite delay with probability one. The first tree algorithms were independently developed in the late 1970s by Capetanakis [3] and Tsybakov, Mikhailov and Vvedenskaya [11]. These algorithms were the first to have a provable MST above zero. Afterwards new tree algorithms were developed with MSTs as high as 0.4878 using the standard information theoretical multiple access model [1, 4, 10].

A random access protocol consists of two components: the channel access protocol (CAP) and the collision resolution algorithm (CRA). The CAP specifies the rules that users need to follow when transmitting a new packet for the first time. The CRA informs the users about the algorithm used to resolve collisions (i.e., simultaneous transmissions). The easiest CAP is *free access*, meaning new packets may be transmitted without further delay. Other important CAPs include *blocked* (or gated) and *windowed* (or grouped) access.

When blocked (or gated) access is used, an initial collision of n stations causes all 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 gated) 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.

When a binary tree algorithm is used in combination with windowed access, a collision of n packets belonging to the same window of length α_0 will cause them to split into 2 groups. Packets that arrived during the first, resp. second, half of the window join the first, resp. second, group. Stations joining the first group retransmit in the next slot (which therefore corresponds to a size $\alpha_0/2$ window) 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. Notice, using this CRA algorithm with windowed access guarantees that the packets are received in a First-Come-First-Serve (FCFS) order. Two important improvements have been made to this scheme. 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 $2^{-i}\alpha_0$.

The 0.4878 MST realized under the standard information theoretical model, has been exceeded in various manners by introducing additional mechanisms not available under the standard model, such as energy measurement techniques to determine the collision multiplicity [8] and an additional control field/bit with separate feedback [7]. Recently, the SICTA algorithm which uses successive interference cancellation (SIC) mechanisms, was designed and shown to achieve an MST as high as 0.693 [12]. 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 cancelling 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 [9] 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]. In this paper we propose and analyze two novel tree algorithms that both have the same memory limitations as in [9], i.e., only a single (collision) signal can be stored at any given time. Both use a windowed access mechanism and the packets are received in a FCFS order. If the single memory location contains a signal at time k and the slot $(k, k+1]$ corresponds to some arrival window $(T(k), T(k)+2^{-i}\alpha_0]$, the stored signal will correspond to the collision signal of the arrival window $(T(k), T(k)+2^{-(i-1)}]$. The first algorithm only cancels successful transmissions from the stored signal (if any) to gather information about the second group and achieves an MST of 0.6048. The second algorithm also cancels the collision signals and uses part of the information revealed by these cancellations to further increase the MST to 0.6173. It is not hard to further improve this MST by dropping the FCFS requirement.

The paper is structured as follows. In Section 2 we start by briefly discussing the well known FCFS tree algorithm with an MST of 0.4871 [1] (the 0.4878 algorithm is obtained from this algorithm by allowing a minor difference between the window length of the first and second group). Section 3 discusses and analyzes the FCFS tree algorithm with success cancellation, while in Section 4 the improved scheme that also cancels collisions is introduced and analyzed.

2. The 0.4871 FCFS splitting algorithm

The celebrated 0.4871 FCFS splitting algorithm [1] determines the users who are allowed to transmit next based on the arrival time of their packet ready for transmission. At any time k, three values are used: $T(k)$, $\alpha(k)$ and $\sigma(k)$. $T(k)$ indicates the start of the current allocation interval, users who generated their packet during this interval are allowed to transmit at time k. All packets generated before time $T(k)$ have been transmitted successfully. The length of the allocation interval is $\alpha(k)$. For all *initial windows* it is set equal to the minimum of α_0, a protocol parameter, and $k - T(k)$ (such that the allocation window does not exceed time k). The last value $\sigma(k)$ indicates whether the allocation window is a left (L) or a right (R) branch in the splitting tree, where for an initial window we set $\sigma(k)$ equal to R.

When $\sigma(k) = R$, the packets generated in the time interval $(T(k)+\alpha(k), k]$, which we call the *waiting window*, must wait for the next initial window. If there is no collision in an R window, a new initial window is started, otherwise the R window w is split into an L and R window both

having half the size of w and the L window becomes the allocation window. When $\sigma(k) = L$, the interval $(T(k), k]$ consists of the allocation window $(T(k), T(k) + \alpha(k)]$, its corresponding R window $(T(k) + \alpha(k), T(k) + 2\alpha(k)]$ and the *waiting window* $(T(k) + 2\alpha(k), k]$, holding arrivals that must wait for the next initial window. Thus, at all times, there is at most one R window. When an L window holds a collision it will split in a smaller L and R window, while the corresponding larger R window becomes part of the waiting window. The logic behind this approach is that when an L window holds a collision, we have no information at all about its corresponding R window (which was part of the same collision), therefore there is no use in treating it separately.

The operation of this algorithm, which is illustrated in Figure 1, can be summarized as follows. If the current slot at time $k-1$ holds a collision, we have

$$\begin{aligned} T(k) &= T(k-1), \\ \alpha(k) &= \frac{\alpha(k-1)}{2}, \\ \sigma(k) &= L. \end{aligned} \tag{1}$$

If it holds a success and $\sigma(k-1) = L$, we continue with the R window

$$\begin{aligned} T(k) &= T(k-1) + \alpha(k-1), \\ \alpha(k) &= \alpha(k-1), \\ \sigma(k) &= R. \end{aligned} \tag{2}$$

When it is empty and $\sigma(k-1) = L$, we know that the R window must hold a collision, so this window is split immediately

$$\begin{aligned} T(k) &= T(k-1) + \alpha(k-1), \\ \alpha(k) &= \frac{\alpha(k-1)}{2}, \\ \sigma(k) &= L. \end{aligned} \tag{3}$$

While if there was a success (or empty slot) with $\sigma(k-1) = R$, we initiate a new initial window:

$$\begin{aligned} T(k) &= T(k-1) + \alpha(k-1), \\ \alpha(k) &= \min(\alpha_0, k - T(k)), \\ \sigma(k) &= R. \end{aligned} \tag{4}$$

This algorithm can be adapted such that all initial windows have a length of α_0, by leaving the channel idle during the interval $(k, T(k) + \alpha_0]$ in case $T(k) + \alpha_0 > k$, i.e., $k - T(k) < \alpha_0$. The maximum stable throughput is not affected by this modification and simplifies its analysis. The same simplifications will be used for the analysis of the new FCFS algorithms.

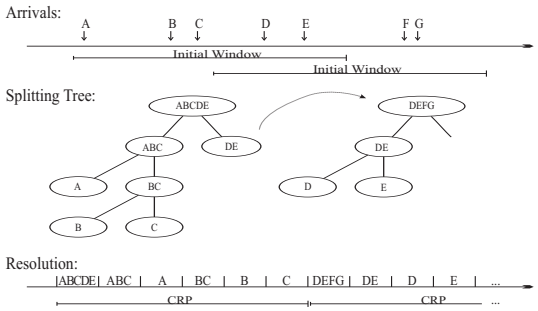

Figure 1. Illustration of the FCFS algorithm

3. A 0.6048 FCFS tree algorithm with success cancellation

The first of the two proposed algorithms will exploit the interference cancellation mechanism by retrieving the content of the R window, whenever its corresponding L windows holds a success. Thus, the single memory location always stores the signal of the last unsuccessful transmission. If the content of the R window does not hold a collision, we can immediately skip the R window as well, meaning (2) changes to

$$\begin{aligned} T(k) &= T(k-1) + 2\alpha(k-1), \\ \alpha(k) &= \min(\alpha_0, k - T(k)), \\ \sigma(k) &= R, \end{aligned} \tag{5}$$

when the interference cancellation operation reveals either an empty of successful slot in the R window. Otherwise, the R window holds a collision, meaning we can immediately split it

$$\begin{aligned} T(k) &= T(k-1) + \alpha(k-1), \\ \alpha(k) &= \frac{\alpha(k-1)}{2}, \\ \sigma(k) &= L. \end{aligned} \tag{6}$$

As showed by Figure 2, the only R windows that correspond to an actual transmission on the channel in this adapted algorithm are the initial windows, all other allocation windows correspond to an L window. This property makes it very easy to determine the maximum stable throughput as we shall see below.

We denote the system state at time k as i, if the slot at time k is the i-th slot following the last initial window, for $i \geq 0$. Thus, for $i = 0$, the current allocation window is an R window, while for all other i values, it is an L window. Denote $p_{i,j}$ as the probability that the system state makes a transition from state i at time k to state j at time $k+1$. Assuming Poisson arrivals with rate λ and defining $G_i = 2^{-i}\lambda\alpha_0$, we have

$$p_{0,0} = (1 + G_0)e^{-G_0},$$

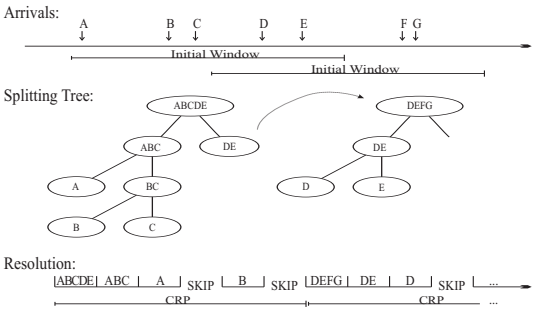

Figure 2. Illustration of the 0.6048 FCFS tree algorithm with success cancellation

as 0 or 1 arrivals in the initial R window results in a new initial window (see (4)). Otherwise, an initial slot is followed by an L slot, thus $p_{0,1} = 1 - p_{0,0}$. For $i > 0$, we have

$$p_{i,0} = \frac{(G_i e^{-G_i})(G_i e^{-G_i})}{(1-(1+G_{i-1})e^{-G_{i-1}})} = \frac{G_i^2}{e^{2G_i} - (1+2G_i)},$$

as the stored signal of the size $2^{-(i-1)}\alpha_0$ slot must hold two packets and one of them is located in the current L window (see (5)). In all other cases, the system state i will change to state $i+1$, meaning $p_{i,i+1} = 1 - p_{i,0}$.

Define a conflict resolution period (CRP) initialized by an (initial) R window as the R window itself followed by all the L windows until the start of the next (initial) R window. Let $E\{k\}$ denote the mean time needed to resolve a CRP (in slots), then due to the possible probabilities presented above, we have

$$E\{k\} = 1 + \sum_{i=1}^{\infty} p_i,$$

where p_i denotes the probability that the CRP lasts more than i slots, i.e., that state i is visited before the CRP ends, meaning $p_i = p_{0,1}\left(\prod_{j=1}^{i-1} p_{j,j+1}\right) = p_{i-1}(1 - p_{i-1,0})$, for $i > 0$. Thus, $E\{k\}$ can be evaluated very easily as the terms of this summation rapidly decrease to zero.

Let $E\{f\}$ be the fraction of the initial size α_0 window that is shifted to the waiting window during the CRP due to possible collisions in the L windows of the CRP. The fraction lost for a collision in the i-th L window is 2^{-i}, yielding

$$E\{f\} = \sum_{i=1}^{\infty} 2^{-i} p_i \frac{1-(1+G_i)e^{-G_i}}{1-(1+G_{i-1})e^{-G_{i-1}}},$$

where the fraction gives the probability that the i-th L window holds a collision given that the $i-1$-th L window held a collision.

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) R windows, which equals $\alpha_0(1 - E\{f\})$. We can rewrite this as

$$\lambda < \frac{(\lambda\alpha_0)(1-E\{f\})}{E\{k\}},$$

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 the 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, the maximum is reached in $\lambda\alpha_0 = 1.613$, resulting in $\alpha_0 = 2.666$ and $\lambda_{max} = 0.6048$ (with $E\{f\} = 0.1355$ and $E\{k\} = 2.3053$). These values have also been confirmed using simulation experiments.

Remark: The following approach is very effective when verifying the stability of this algorithm by simulation. Assume that the start of the current allocation window equals T, meaning all packets generated in $(0, T]$ have been received correctly. Let S be the number of slots used to resolve the interval $(0, T]$, *without* taking into account the idle periods inserted whenever $T(k)$ exceeded k. Then, stability implies that $S - T$ will decrease to minus infinity (due to the recurrent presence of the idle periods), otherwise $S - T$ will increase to plus infinity.

4. A 0.6173 FCFS tree algorithm with success and partial collision cancellation

In the previous section we indicated how to benefit from a successful transmission that follows a collision. When a collision is followed by an empty slot, there is nothing to cancel and we use the standard approach given by (3). However, when a collision c is followed by another collision c_L (in an L window), we can retrieve information about the R window that becomes part of the waiting window due to the c_L collision. That is, we know whether this R window holds zero, one or more packets. Notice, whenever the R window holds a single packet, it can be retrieved from the cancellation operation, however as this would cause the packet to be out-of-order, such packets will be retransmitted as explained below. When the R window holds one or more packets, we cannot store the signal obtained from the cancellation as we use the single memory location to store the c_L signal.

In order to keep the operation of the algorithm simple, we will only exploit the information about the last, if any, R window that became part of the waiting window during a CRP. Thus, during a CRP, we keep track of the size α_R of the last postponed R window and whether this window

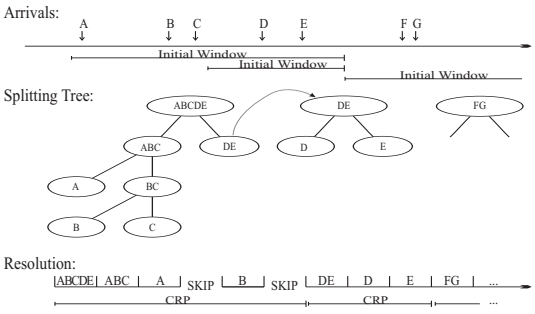

Figure 3. Illustration of the 0.6173 FCFS tree algorithm with success and partial collision cancellation

held one or more packets. When a CRP ends at time k, the algorithm works as follows (see Figure 3):

- If the last postponed R window was nonempty, the next initial window is no longer of length $\min(\alpha_0, k - T(k))$, but has length $\alpha_R < \min(\alpha_0, k - T(k))$. Hence, the new CRP will start with either a successful transmission or a collision (in order to retrieve the collision signal of the last postponed R window).
- If the last postponed R window was empty, we can immediately advance the start $T(k)$ of the allocation window by another α_R as the possible success in this R window was already obtained from the cancellation operation.
- Otherwise, the operation is identical to the algorithm presented in Section 3.

When analyzing the maximum stable throughput of this algorithm, we refer to a CRP that starts with an allocation interval of size $2^{-j}\alpha_0$ as a type j CRP. To simplify the analysis we will also include the concept of a length zero CRP. More specifically, when the last postponed R window in a CRP was located at level j of the splitting tree, we state that the next CRP will be of type j, irrespective of whether this window held a packet. Hence, the length (that is, the number of slots devoted to this CRP) of a type j CRP, with $j > 0$, equals zero whenever the interference cancellation indicates that the last postponed R window was empty. In this case, the type j CRP will subsequently be followed by a type 0 CRP.

We start by determining the probabilities $q_{i,j}$ that a type i CRP is followed by a type j CRP. Clearly, $q_{i,j} = 0$ if $i \geq j \neq 0$, as a CRP starting with a length $2^{-i}\alpha_0$ allocation window either postpones a smaller R window or none at all. A type $j = 0$ CRP will only follow a type i CRP if none of the L windows holds a collision. All the L windows of a CRP with an initial allocation window of size $\alpha(k)$ that starts at $T(k)$ will be collision free if and only if the intervals $(T(k)+(1-2^{-(s-1)})\alpha(k), T(k)+(1-2^{-s})\alpha(k)]$ hold either zero or one packet for all $s \geq 1$, which occurs with probability $(1+\lambda\alpha(k)/2^s)e^{-\lambda\alpha(k)/2^s}$. In other words,

$$q_{i,0} = \left[\prod_{s=1}^{\infty}(1+\frac{G_i}{2^s})\right]e^{-G_i},$$

as $\alpha(k) = 2^{-i}\alpha_0$ and $\prod_{s\geq 1} e^{G_i/2^s} = e^{-G_i}$. For numerical stability further on, we rewrite this as

$$q_{i,0} = \left[1+\sum_{s=1}^{\infty}\frac{G_i^s}{\prod_{t=1}^{s}(2^t-1)}\right]e^{-G_i}.$$

We also note that

$$q_{i,0} = q_{i+1,0}(1+G_{i+1})e^{-G_{i+1}},$$

as $G_i/2 = G_{i+1}$. In order to express the remaining $q_{i,i+s}$ values, for $s > 0$, we introduce some additional variables.

Let b_i be the probability of having a collision in a length $2^{-(i-1)}\alpha_0$ interval, that is,

$$b_i = 1 - (1+G_{i-1})e^{-G_{i-1}}.$$

Let c_i be the probability of having a collision in an interval of the form $(T(k), T(k)+2^{-i}\alpha_0]$ given that there was a collision in the $(T(k), T(k)+2^{-(i-1)}\alpha_0]$ interval and such that none of the L windows part of $(T(k), T(k)+2^{-i}\alpha_0]$ holds a collision. We can compute c_i as

$$c_i = \frac{q_{i,0} - (1+G_i)e^{-G_i}}{1-(1+G_{i-1})e^{-G_{i-1}}}.$$

To avoid numerical bit cancellation, we will compute c_i as

$$c_i = \frac{\left(\sum_{s=2}^{\infty}\frac{G_i^s}{\prod_{t=1}^{s}(2^t-1)}\right)e^{-G_i}}{\left(\sum_{s=2}^{\infty}\frac{G_{i-1}^s}{s!}\right)e^{-G_{i-1}}}.$$

Finally, recall that $p_{i,i+1}$, as defined in the previous section, equals the probability that a length $2^{-i}\alpha_0$ L window is followed by another L window of size $2^{-(i+1)}\alpha_0$.

We are now in a position to express $q_{i,i+s}$ for $s > 0$. In order for a type i CRP to be followed by a type $i+s$, there has to be a collision in the initial size $2^{-i}\alpha_0$ window, which occurs with probability b_{i+1} and leads to a size $2^{-(i+1)}\alpha_0$ L window. This L window should be followed by a series of $s-1$ L windows of size $2^{-(i+2)}\alpha_0$ to $2^{-(i+s)}\alpha_0$, which occurs with probability $\prod_{k=1}^{s-1} p_{i+k,i+k+1}$. Finally, the size $2^{-(i+s)}\alpha_0$ L window must hold a collision, while none of the subsequent L windows of this CRP holds a collision, an event that takes place with probability c_{i+s}. This yields

$$q_{i,i+s} = b_{i+1}\left(\prod_{k=1}^{s-1} p_{i+k,i+k+1}\right)c_{i+s}, \qquad (7)$$

for $s > 0$. Next, we determine the probability $\pi(i)$ that an arbitrary CRP (including the length zero CRPs) is of type i, by computing the invariant vector of the infinite matrix Q with entry (i,j) equal to $q_{i-1,j-1}$. Let $(\pi(0), \pi(1), \pi(2), \ldots)$ be the stochastic invariant vector of Q (its existence and uniqueness is immediate from the Lemma of Pakes [1] as $q_{i,0}$ increases to one). Then, due to the structure of Q, we have

$$\pi(i) = \sum_{j=0}^{i-1} \pi(j) q_{j,i},$$

for $i > 0$, while $\pi(0)$ is found using the normalization condition $\sum_{i\geq 0} \pi(i) = 1$.

Having found $\pi(i)$, for $i \geq 0$, we continue by computing $E\{k|i\}$ and $E\{f|i\}$, which denote the mean duration of a type i CRP (in slots) and the mean fraction of the initial size $2^{-i}\alpha_0$ allocation window that is postponed due to possible collisions in any of the L windows, respectively. Note, even the last postponed R window is counted by this fraction, irrespective of whether or not it holds a collision. Analogue to the previous algorithm, we find that for $i = 0$

$$E\{k|0\} = \sum_{j=0}^{\infty} \text{Pr[CRP length} > j \text{ slots]} = 1 + \sum_{j=1}^{\infty} p_j,$$

with p_i as defined in the previous section. For $i > 0$, the CRP has a length larger than zero with probability $1-e^{-G_i}$, meaning

$$\begin{aligned} E\{k|i\} &= \sum_{j=0}^{\infty} \text{Pr[CRP length} > j \text{ slots]} \\ &= (1 - e^{-G_i}) + \sum_{j=i}^{\infty} b_{i+1} \left(\prod_{k=1}^{j-i} p_{i+k,i+k+1} \right). \end{aligned}$$

To compute $E\{f|i\}$, we note that every collision in an L window of size $2^{-j}\alpha_0$ causes the loss of a $2^{-j}\alpha_0$ R window, that is, a fraction of 2^{i-j} of the initial $2^{-i}\alpha_0$ window is lost. Hence,

$$E\{f|i\} = \sum_{j=i+1}^{\infty} b_{i+1} \left(\prod_{k=1}^{j-i-1} p_{i+k,i+k+1} \right) b_{j+1} 2^{i-j}.$$

Using $\pi(i)$, we obtain $E\{k\}$ and $E\{s\}$, the mean CRP length (in slots) and the mean distance that the starting point of the allocation window advances, respectively:

$$\begin{aligned} E\{k\} &= \sum_{i\geq 0} \pi(i) E\{k|i\}, \\ E\{s\} &= \sum_{i\geq 0} \pi(i) 2^{-i} (1 - E\{f|i\}) \alpha_0. \end{aligned}$$

Stability is reached if and only if $E\{k\} < E\{s\}$, which can be written as

$$\lambda < (\lambda\alpha_0) \frac{\left(\sum_{i\geq 0} \pi(i) 2^{-i} (1 - E\{f|i\}) \right)}{E\{k\}},$$

where the right hand side is again a function 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 the 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, the maximum is reached in $\lambda\alpha_0 = 1.691$, resulting in $\alpha_0 = 2.7392$ and $\lambda_{max} = 0.6173$ (with $E\{k\} = 1.9835$). These values have also been confirmed using simulation experiments.

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. Inform. Theory*, 25(5):319–329, 1979.

[4] A. Ephremides and B. Hajek. Information theory and communication networks: an unconsummated union. *IEEE Transactions on Information 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 *Proc. of the 16th Int. Conf. on Comm.*, 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. Computers*, 46(4):473–476, 1997.

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

[9] 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.

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

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

[12] 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.