RC 20466 (90479) 5/21/1996 Revised 6/20/96
Computer Science/Mathematics 23 pages

IBM Research Report

msSTART: A Random Access Algorithm for the IEEE 802.14 HFC Network

Chatschik Bisdikian

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York 10598

Bill McNeil, Rob Norman, and Ray Zeisz

IBM Corporation
Networking Hardware Division
3039 Cornwallis Road
Research Triangle Park, NC 27709

msSTART: A Random Access Algorithm for the IEEE 802.14 HFC Network

Chatschik Bisdikian
IBM Research Division
T. J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532, USA
bisdik@watson.ibm.com

Bill McNeil and Rob Norman
IBM Corporation
Networking Hardware Division
3039 Cornwallis Road
RTP, NC, 27709, USA
{wmcneil,norman}@vnet.ibm.com

Abstract: *In this article, we introduce the* n-ary Multi-slot Stack Random Access Algorithm *(msSTART) for use in an IEEE802.14 hybrid fiber/coaxial (HFC) network. The msSTART random access algorithm (RAA) is an evolution of the stack-based START algorithm. It is an easy to implement, high-efficiency, free-access, robust, stable RAA optimized to operate in an environment where contention slots could dynamically change. Through a series of simulation studies, we demonstrate the performance advantages of msSTART by comparing its throughput/delay characteristics to those of other popular RAAs, including ALOHA and tree-search RAAs. For the latter RAAs, we also provide a unified stack representation of their operation.*

1 Introduction

The increasing appetite of people for residential interactive broadband services like high-speed Internet access and Web browsing, remote login and telecommuting, video-on-demand services, remote learning, remote shopping, distributed video-game playing would only be possible to satisfy if the 30 kb/s (or, so) barrier of today's fastest telephone modems is surpussed tenfold, a hundredfold, or even more.

Of all the possible comminications means available today to residential customers, the *hybrid fiber/coaxial* (HFC) cable TV (CaTV) infrastructure, [5], that is currently aggresively pursued by the CaTV operators, appears as the most promising for supporting interactive services in the near and remote future. Already a large number of field trials has proven that the HFC technology can indeed support such services with relatively minimum extra cost [12].

In an HFC network, long coaxial trunks, with their abundance of uni-directional signal amplifiers, are replace with highly reliable, low attenuation fiber cables. Thus many CaTV operators are "urged" to upgrade their plants to HFC due to its lower maintenance cost and the wider bandwidth that can be supported by HFC networks. During this upgrade, to facilitate two-way communications, the CaTV operators need also replace their uni-directional amplifiers with bi-directonial ones, a relative small cost considered that most of the cost is the labor needed to upgrade the all-coaxial plants to HFC.

For data transmissions over regular telephone lines, one needs a telephone modem. Similarly, to transmit and receive digital data over an HFC network a *cable modem* is needed. The desire for CaTV based services is underscored by a recent industry study that projects an exlossive increase of the cable modem market –about 100% or more growth per year until the end of this centrury [6].

In order though to spearhead their widespread acceptance, cable modems should become widely available to consumers, low cost, and extremely easy to use, all of which are features that are enjoyed by (most) telephone modems. To achieve this, there is the undeniable need to develop interoperable systems, i.e., standardized HFC network communication protocols and interfaces, and cable modem units. The IEEE 802.14 working group (WG) is working toward the development of *physical* and *medium access* (MAC) layer communication protocols for HFC networks.[1] Our interest in this paper is in the MAC protocol and in particular in one aspect of it that relates to the sharing of communications resources among uncoordinated stations (the cable modems).

We will start our presentation with a brief review of an HFC network and of the class of MAC protocol considered here; for additional information on the area see [3, 5, 4] and references therein.[2] We will then focus on a *random access algorithm* that has been developed to operate efficiently within an HFC environment.

[1]The IEEE802.14 WG maintains a Web page at URL "*http://www.walkingdog.com/catv.html.*"

[2]An ever increasing list of cable modem and HFC activities can be found through searches on the Web using keywords like "cable modem," or "HFC."

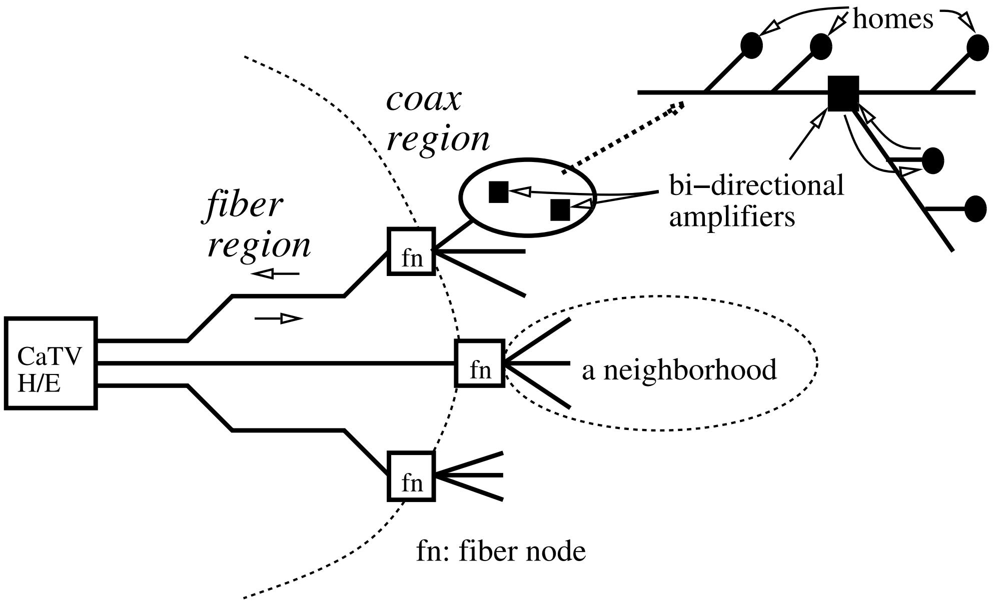

Figure 1: An HFC tree-and-branch network

2 An HFC network summary

The HFC CaTV network is a tree-and-branch network, see figure 1. A *headend* (H/E) station is located at the root of the tree and the leaves are at subscribers' (customers') homes. In an HFC network, most of the "tree-trunk(s)" are replaced with high reliability, high bandwidth, low noise, fiber optic cables. Only the *last-mile* distribution network to the individual homes remains coaxial. *Fiber nodes* located at the boundaries of the *fiber* and *coaxial regions* are responsible for converting optical signals to electromagnetic for *downstream* (H/E-to-station) transmissions, and vice versa for *upstream* (station-to-H/E) transmissions. The maximum end-to-end, from the H/E to a home, distance in an HFC network is about 80 km, while each fiber node serves a neighborhood of the order of a thousand homes.

To support two-way communications, the traditionally CaTV uni-directional signal amplifiers are being replaced with bi-directional ones. In an HFC network, the low signal attenuation on the long fiber cables implies that only a relatively small number of amplifiers are needed, which considerably decreases the cost of preparing an HFC network for two-way communications.

Of the CaTV frequencies available at the subscriber side, the frequency range between 5 and 42 MHz will be used for upstream transmissions, while the range between 54 and 750 MHz (up to GHz in future systems) will be used for downstream transmissions. To maintain compatibility with current analog and digital broadcast transmissions, the new "switched" downstream transmissions will most likely occupy the frequencies above 450 MHz. The actual transmissions will occur within small channels, in the order of a MHz on the upstream, and 6 or 8 MHz on the dowstream. Each upstream channel will support about 3 Mb/s (link speed), while each downstream channel will support about 30 Mb/s.

Since stations on a two-way capable HFC network transmit and receive on totally different frequencies, they are not able to listen upstream transmissions from other stations. The latter is further enforced through the use of directional taps at the receive and transmit sites at each station. This is done, because, a CaTV network is a shared medium network, and as such it has so designed as to discourage listening transmissions from other stations for increased system security and privacy.

Hence, to coordinate station transmissions efficiently, especially on the upstream, the (occasional) use of a random access algorithm (RAA) appears to be a necessity. The use of random access is dictated by the need to coordinate efficiently a large number of stations most of which are expected to be silent at any single moment, that have no means to coordinate their transmission attempts directly with each other.

Even in such a "hostile" shared medium environment, a robust MAC protocol that flexibly combines *contention-based* and *reservation-based* transmissions could be developed to support various types of traffic with diverse quality-of-service (QoS) requirements. MLAP is such a MAC protocol [3]. MLAP is an ATM friendly MAC protocol that is currently under consideration by the IEEE 802.14 WG. In MLAP, random access is used both during power-on of a station, and during an "initial" transmission by a station following a long period of silence. In the latter case, a successful contention-based transmission could trigger a series of reservation-based transmissions, based on requests for more transmissions that are *piggy-backed* on each transmission.

In this paper, we will describe MLAP's RAA component, named *multi-slot n-ary STack AlgoRiThm* (msSTART), particularly as it pertains to its ingenious realization of the original START, [10], concepts in a communications environment where the number of contention opportunities is allowed to change dynamically. msSTART is an evolution of the START RAA and inherits its simple implementation, its high performance and stable operation. The latter point cannot be sufficiently underscored, as some of the most popular RAAs, the purely acknowledgment based ALOHA family of RAAs, including the exponential back-off algorithm and the carrier sense multiple access, are unstable [1]. Instability here means that eventually no new traffic arrival rate can be (theoretically) sustained by the RAA, no matter how small the traffic arrival rate is [2].

Our focus will be mostly on coordinating station transmissions on the upstream. Downstream transmissions are from the single H/E to the various stations, and hence coordinating this transmissions is not very hard. For simplicity, we will concentrate on stations that all use the same upstream channel and the same downstream channel. As part of MLAP, on the downstream channel, the H/E provides control information to the stations *on when*, *in what mode* (contention-based, or reservation-based) and *for how long* stations can transit on the upstream channel. In our presentation of msSTART, we will simply assume that stations know when to send their contention-based transmission. Also, for each upstream contention based transmission, the H/E broadcasts on the downstream feedback information regarding this transmission as soon as possible.

3 The msSTART RAA

In MLAP, (upstream) channel transmission time is divided into fixed size intervals called *blocks*. Under the guidance of the H/E, stations transmit in H/E defined time periods (referred to as *slots*) within each block. Some of these slots are *reserved slots*, while others are *contention slots*. In reserved slots, only a specific station can transmit; hence, reserved slots are always collision-free slots. In contention slots, all the stations in a well defined group[3] of stations can transmit; hence, collisions can occur during a contention slot. In this paper, we will focus only on the contention slots and the RAA that can be used to resolve possible collisions.

Contention slots could be small "minislots" not capable of carrying payload, or large "regular" slots that can also carry payload(s). In this paper, we will not distinguish between minislots and regular slots since msSTART does not distinguish between them either, even though MLAP does distinguish between them. Hence, in the sequel, we will simply call any type of a slot in which a collision can occur a contention slot (CS).

For the implementation of msSTART, each station needs to maintain an msSTART *counter* for each packet that it wants to transmit on a CS. The msSTART counter exists only while a station has something to transmit, i.e., while the station is *active*. While it is *inactive*, a station does not need to do anything regarding msSTART.

Following transmission attempts in the CSs of a block, and after an active station – referred to as the *tagged* station– receives feedback information regarding these transmissions, the tagged station updates its msSTART counter; the update equations will be presented shortly. The size of a block in MLAP is sufficiently long to guarantee (among other things) that feedback information from transmissions in CSs in a block is available to the stations prior to the appearance of CSs in the next block. Note that given the physical size of an HFC network a typical block size is of the order of a few milliseconds.

If the msSTART counter indication is smaller than the number of CSs in a block, the tagged station will transmit in the block, otherwise it will refrain from transmitting. In the latter case, the station will continue updating its msSTART counter until a future block at which the "transmission condition" will hold true.

In particular, suppose that in block i the msSTART counter value at the tagged station is COUNT_i ($\text{COUNT}_i \geq 0$).[4] COUNT_i includes the counter updates due to feedback information from transmissions in CSs in block $(i-1)$, but no feedback information from transmissions in CSs in block i. Let also M_i denote the number of CSs in block i; the CSs are numbered as $0, 1, \ldots, (M_i - 1)$. The number of CSs in a block is decided by the H/E in an effort to maintain a reasonable balance between traffic demand for contention-based and reservation-based transmissions. How this is done is immaterial to this study and we will not discuss it further.

If in block i, $M_i > \text{COUNT}_i$, then the station transmits in the CS number COUNT_i of block i. On the other hand, if $\text{COUNT}_i \geq M_i$, then the station refrains from transmission in block i. The station continues to update its counter until a future block j for which $M_j > \text{COUNT}_j$

[3]The "default" station group is the collection of all the stations in the system.

[4]The block (or, time) index i is *not* traced by any station. It is used merely to facilitate our presentation.

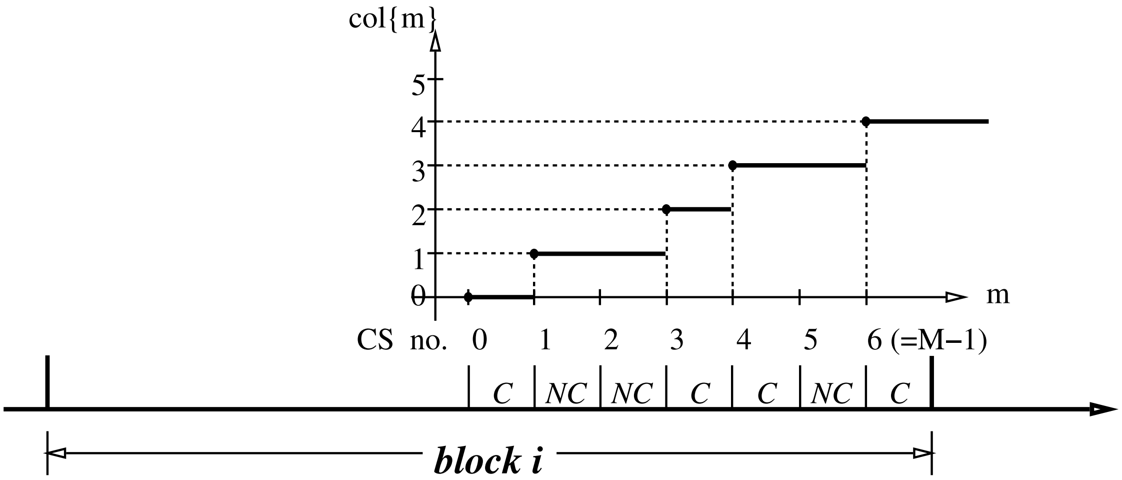

Figure 2: The $col\{m\}$ function

holds true. At that time, the station will transmit in CS number COUNT_j of block j.

3.1 The update of an msSTART counter

A (tagged) station that has new traffic to transmit in block i (with its M_i CSs), initializes its msSTART counter to the value $\text{COUNT}_i = random\{0, \ldots, M_i - 1\}$.[5] Here, we assume that the new traffic is generated prior to the first CS of block i. Hence, to remove any ubiquity, we assume that only new arrivals in block $(i-1)$ are allowed to start their contention-based transmissions during block i, although, this restriction could be somewhat relaxed.

For block i, let the *counting* function $col\{m\}$ denote the number of collisions in CSs number $0, \ldots, \min\{m-1, M_i-1\}$, where $col\{0\} \stackrel{\text{def}}{=} 0$. For each CS in block i, $col\{m\}$ counts the number of collisions that occurred in the CSs of block i that are located "to the left" of the CS number m. For $m > M_i$, $col\{m\} = col\{M_i\}$, which is nothing more than the total number of collisions in block i. Figure 2 shows an example of $col\{m\}$ for a block with $M = 7$ CSs and with four collision (C) slots that have occurred in CSs number 0, 3, 4, and 6.

In block i, suppose that for our tagged station COUNT_i is less than M_i ($\text{COUNT}_i < M_i$), i.e., the tagged station transmits in CS number $m = \text{COUNT}_i$ of block i. After the tagged station receives feedback information regarding CS transmissions in block i, the station updates its msSTART counter as follows:

$$\text{COUNT}_{i+1} = n * col\{\text{COUNT}_i\} + random\{0, \ldots, n-1\}; \tag{1}$$

where n, called the *splitting parameter*, is an integer ≥ 2 to be appropriately selected for system performance optimization [9, 10].

[5]The function $random\{\cdot\}$ returns a randomly selected member of its arguments.

- when an inactive station becomes active, initialize COUNT $= random\{0, \ldots, M-1\}$, where M is the number of CSs in the block following the activation of the station;
- if (COUNT $< M$) then
  - station transmits in CS number COUNT;
  - upon receiving feedback information:
    - if (*did_not_collide*) then
      done;
    - elseif (*collide*)
      set: COUNT $\leftarrow n * col\{\text{COUNT}\} + random\{0, \ldots, n-1\}$;
- elseif (COUNT $\geq M$) then
  - station does not transmit in current block;
  - upon receiving feedback information:
    - set: COUNT $\leftarrow n * col\{M\} + \text{COUNT} - M$;

Table 1: The msSTART counter update algorithm

On the other hand, if the station did not transmit in block i, i.e., $\text{COUNT}_i \geq M_i$, then the station updates its msSTART counter as follows:

$$\text{COUNT}_{i+1} = n * col\{\text{COUNT}_i\} + (\text{COUNT}_i - M_i) = n * col\{M_i\} + (\text{COUNT}_i - M_i), \qquad (2)$$

where, as noted earlier, $col\{M_i\}$ is nothing more than the total number of collisions that occurred during block i.

Finally, if $\text{COUNT}_{i+1} < M_{i+1}$, the station transmits in CS number COUNT_{i+1} of block $(i+1)$.

3.2 The msSTART algorithm

Table 1 summarizes the counter update algorithm for msSTART. Since the block index i is not needed to be traced by a station, in table 1, we have dropped the index i from the algorithm.

Note that msSTART requires only a C/NC (Collision/NoCollision) feedback information for its operation. Also, the function $col\{m\}$ can either be determined by the station itself, or provided to the station (by the H/E) as part of the feedback information.

3.3 An msSTART example

The operation of msSTART can be better visualized with the help of a *virtual stack*, or, simply, the *stack*, with levels indexed as $0, 1, 2, \ldots$. All stations with $\text{COUNT}_i = m$ "reside," or "placed," at stack level m. Figure 3 illustrates the operation of msSTART with the help of the stack.[6] Note that the first M_i levels of the stack coincide with the CSs of block i.

The stack is a visual tool for representing the global operation of the system. We need to emphasize that the stack is a creation of human imagination. It is used merely for illustration purposes, and it is not part of msSTART or realized by any station in the system. Later, we will also visualize other popular RAAs using the stack tool.

As an example, let us consider a newcomer (a tagged station that has just become active), that, initially, randomly chooses to transmit in CS no. 2 of block $(i-1)$, i.e., its msSTART counter initializes to the value $\text{COUNT}_{i-1} = 2$; recall that the time index is not really traced by any station. Following a collision in CS no. 2, the tagged station randomly selects a number between 0 and 2 $(= 3 - 1 = n - 1)$; suppose that it selects an 1. Since in block $(i-1)$, there is one more collision to the left of CS no. 2, i.e., in CSs no. 0 and 1 of block $(i-1)$ there is only one C slot, the tagged station updates its counter as follows:

$$\text{COUNT}_i = n * col\{\text{COUNT}_{i-1}\} + random\{0, \ldots, n-1\} = 3 * 1 + 1 = 4. \tag{3}$$

The number of CSs in block i is $M_i = 4 \geq 4 = \text{COUNT}_i$, and, hence, our tagged station does not transmit in block i. Yet, it still acts upon receipt of feedback information from CS transmissions in block i. Since in block i, there exists only one C slot, the tagged station updates its counter as follows:

$$\text{COUNT}_{i+1} = n * col\{M_i\} + (\text{COUNT}_i - M_i) = 3 * 1 + (4 - 4) = 3. \tag{4}$$

Since, $M_{i+1} = 5 > 3 = \text{COUNT}_{i+1}$, the tagged station transmits in CS no. 3 of block $(i+1)$. If CS no. 3 in block $(i+1)$ is an NC slot, the station is done, otherwise it continues as before with the collision resolution.

It should be clear from figure 3 that each collided group of stations splits into n subgroups. The stations that did not transmit allow just enough space, by moving up and down the stack levels, so as not to interfere with the collision resolution of any group of stations that is known to have already collided.

The stack level pointed by the arrow tagged "max_COUNT" is not an integral part of msSTART. In our example, it simply represents the lowest stack level that is guaranteed to be empty, in other words, no station resides at or above the max_COUNT level. If needed, an observer that continuously monitors the operation of the system, e.g., the H/E, can easily determine the value of "max_COUNT" at each block.

The selection of $n = 3$ for our example is not arbitrary. It has been shown analytically (for the single CS case) [9, 10], and our simulation studies confirm this, that the operation of msSTART is optimized when $n = 3$.

[6] The CSs are only logically located at the right side, i.e., the end, of a block. At the H/E's discretion, CSs can be physically located anywhere in a block.

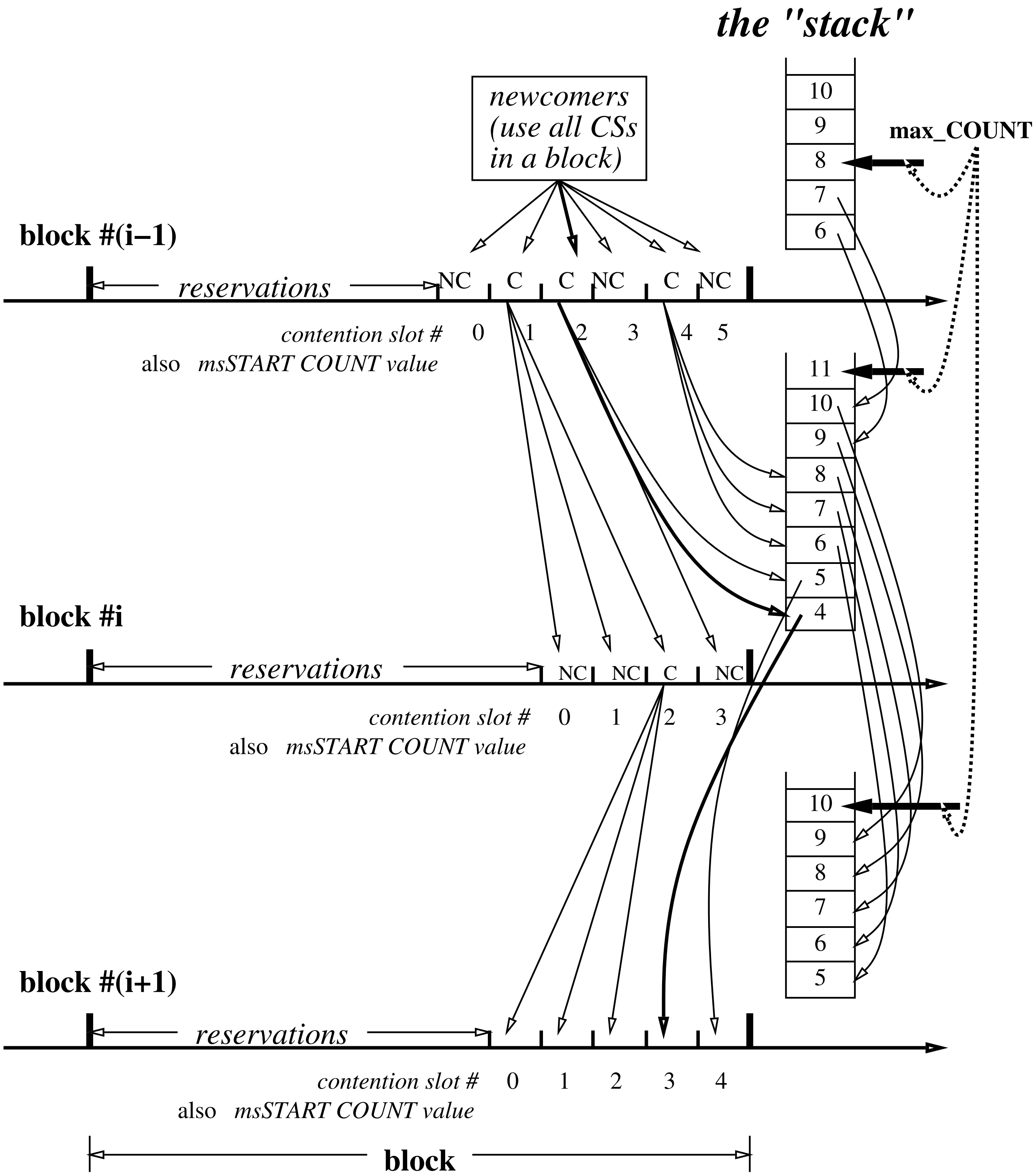

Figure 3: Visualizing the operation of the msSTART via the help of the *stack*; $n = 3$

4 Performance evaluation

Through a series of simulation studies, we have evaluated the performance of the msSTART. We have further compared its performance to that of other popular RAAs.

In our performance analysis, we focus exclusively on the subscriber (station) side. In particular, we are only interested in how the station reacts to the changing CS environment. We are not interested in the relation between the number of CSs in a block and the number of reservations accumulated at the H/E. We believe, that no matter what the number of CSs per block is, the RAA used by the subscriber must always respond efficiently and smoothly.

We start, with the description of the RAAs used for our comparative studies.

4.1 Other popular RAAs

We have considered (stable) ALOHA-type and tree-search RAAs. For a review of RAAs and literature on this topic see [1, 2, 11].

According to the ALOHA RAA all stations, newly activated and old ones, use a (common) transmission probability to decide whether they should transmit in the CSs of a block or not. Two flavors of the ALOHA RAA are provided:

1. The *ideal* ALOHA: In block i, all stations learn from a "genie" the total number of backlogged stations B_i; B_i also includes the newcomers. All stations transmit, in any randomly selected CS of block i, with probability

$$p_i = f(B_i) = \min\{1, \frac{M_i}{B_i}\}. \tag{5}$$

   Clearly, this algorithm is non-implementable since it is impossible to know B_i with certainty. Nevertheless, this case is very important because it provides a lower bound on the performance of any ALOHA-type system.

2. The *implementable* ALOHA: Similar to the previous case, but now an estimate $\widehat{B}_i$ of the backlog B_i is used instead. $\widehat{B}_i$ is updated recursively at each block based on feedback information received from CS transmissions in the block. Our update equation for the backlog is based on the one given in [8] and it represents a "multi-slot" generalization of the recursion shown in [1, 2], which was developed for Poisson traffic:

$$\widehat{B}_{i+1} = M_i e^{-1} + \max\{\,0,\ \widehat{B}_i - M_i + \frac{e-1}{e-2} col\{M_i\}\,\}, \tag{6}$$

   where, $col\{M_i\}$ represents the total number of collisions that occur in block i.

   Once again, all stations transmit, in any randomly selected CS of block i, with probability

$$p_i = f(\widehat{B}_i) = \min\{1, \frac{M_i}{\widehat{B}_i}\}. \tag{7}$$

Contrary to msSTART and the ALOHA protocols, in the tree-search RAAs newcomers are blocked, i.e., wait until the current collision resolution "cycle" finishes. Collision resolution is achieved by having collided station groups randomly split in halves, and let each subgroup continue in turn with its own collision resolution. When a collision resolution cycle ends, all waiting (newcomer) stations are allowed to transmit, and a new collision resolution cycle starts again.

4.2 Stack visualizations for the other popular RAAs

Before proceeding with our numerical results, we would like to show the *stack visualizations* of the RAAs considered here for the single CS per block case; based on what is shown next and figure 3, the multi-slot cases are direct extensions of their single CS counterparts.

Each RAA is described by two sets of rules:

1. the *first transmission rule*, which regulates when newcomers are allowed their first transmission; and

2. the *collision resolution rules*, which dictates how stations resolve collided transmissions.

Thus, in our stack visualization, we will show where in the stack the newly activated stations are placed, and how stations relocate themselves in the various stack levels following a collision.

This "unified" view of the various RAAs exaggerates their similarities as well as their differences. It has also provided us with a simple mechanism to simulate efficiently all of them, since we did not have to rewrite each simulator from scratch, but simply tune the behavior of the stack to the peculiarities of each RAA. Just like msSTART, all the RAAs that have a stack representation can be implemented with the help of a single counter.

4.2.1 msSTART RAA

As a reference point, we start with the msSTART algorithm, whose stack visualization is that of figure 3 under the condition that each block has exactly one CS. The corresponding stack visualization is shown in figure 4. The "max_COUNT" level in figure 3 has been replaced by the "guaranteed empty level" tag, just to stress the point that the operation of the algorithm guarantees that there are no stations residing at this or higher levels.

Following an NC CS, all stations go down by one stack level.

4.2.2 Tree-search RAA

Although, the "guaranteed empty" level in msSTART plays no real role for the algorithm, let us suppose momentarily that instead of placing the newcomers at level 0, they are placed at the "guaranteed empty" level instead, which has been appropriately renamed to the "newcomers level" in figure 5. After a little thought, it should be clear that the newcomers will get their first opportunity to transmit only after the current collision resolution ends. In other words, the stack visualization in 5 is *equivalent* to the blocked access binary tree-search RAA!

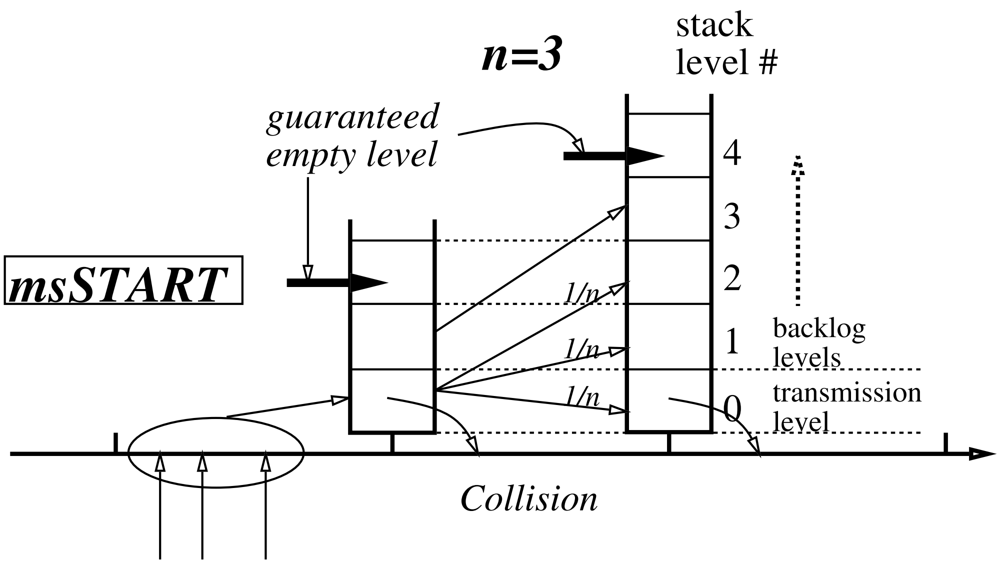

Figure 4: The stack visualization of msSTART RAA for the case of one CS per block

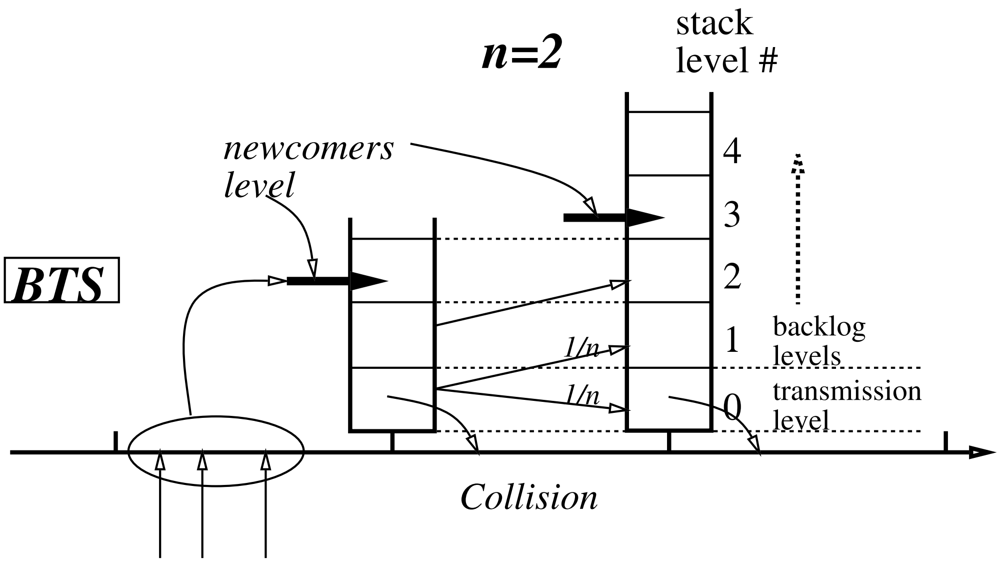

Figure 5: The stack visualization of the binary tree-search RAA for the case of one CS per block

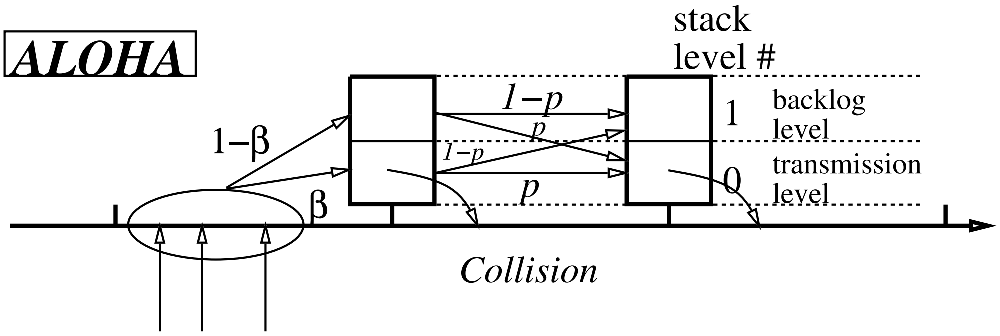

Figure 6: The stack visualization of an ALOHA-type RAA for the case of one CS per block

Newcomers do not really have to know the "newcomers level" to join the system. Since, all these stations will be blocked and remain inactive anyway, they can simply wait until the H/E notifies them that they are finally allowed to transmit.[7] The H/E does that by waiting until the end of the current collision resolution, or, equivalently, until the arrow pointing at the newcomers level falls to level 0. Since it continuously monitors the system, the H/E is capable of recreating the movement of the "newcomers level" arrow, hence, learn about the end of a collision resolution, and then simply signal the waiting newcomers that they can start their transmissions.

The actual collision resolution proceeds similarly to msSTART, where now each collided group splits in the first two cells of the stack, while other stations move up and down the stack in an effort to "avoid" on-going collision resolutions.

Following an NC CS, all stations go down by one stack level.

4.2.3 ALOHA RAA

Finally, in figure 6, we show the stack visualization of an ALOHA-type RAA. Here, we show the case of an ALOHA RAA that uses a transmission probability to regulate transmissions at each CS.

Contrary to the previous cases, the ALOHA stack RAA has only two levels, the transmission level no. 0, and the backlog level no. 1. At each block, station placement to any one of these levels is regulated by two probabilities: the *admission probability* β that regulates newcomers, and the *transmission probability* p that regulates who among all backlogged stations transmits. These probabilities may be fixed, or they may change from block to block.

For the admission probability β, the following two cases have been considered in the literature:

1. *immediate access*: $\beta = 1$, i.e., each newly activated station is allowed to transmit immediately following its activation; and

[7] Since we are dealing with a blocked-access algorithm, following this first transmission, any new newcomers will again wait until the collision resulted (if any) from this initial transmission is first resolved.

2. *delayed access*: $\beta = p$, i.e., each newly activated station is treated as a backlogged station and it is allowed transmission only with probability p.

For the stable ALOHA RAAs studied here, only the delayed access first transmission rule is considered, i.e., $\beta = p$, since system stability has been guaranteed only for this case.

4.3 The simulation model and scenarios

For our performance results, we have used the infinite Poisson station (IPS) model, which is the standard model for the analysis of RAAs. According to the IPS model, the system is comprised of a large number of stations that cumulatively generate packet traffic according to a Poisson process.

As before, we consider that the (upstream transmission) time axis is comprised of fixed size intervals, called the *blocks*, and within each such block there exists a random number M of contention slots (CSs) that the stations use for their contention-based transmissions. It is assumed that the size of the block is sufficiently long so that a station receives feedback information regarding its contention-based transmissions in a block prior to the appearance of CSs in the following block. In other words, it is assumed that at each block all stations have the necessary information to execute one more step of the RAA considered.

Varying the parameter λ of the Poisson process, which represents the packet generation rate, or *offered load* due to newly generated traffic, in packets per block, we have studied the delay behavior of msSTART for four different CS scenarios:

1. each block has only a single CS, i.e., $M = 1$ for all blocks;
2. each block has exactly 2 CSs, i.e., $M = 2$ for all blocks;
3. even numbered blocks have 2 CSs, while odd numbered blocks have 8 CSs; and
4. each block has either 4, 10, or 16 CSs, randomly selected.

The first scenario represents the classical case studied for practically any RAA known in the literature, and is used as a reference point. The second scenario is an extension of the first one, and its objective is to study how gracefully does msSTART adopt to this situation.

The third scenario is an artificially constructed one, yet, highly motivated by the real operation of the system. We have created scenario 3 to study the performance of msSTART under a very severe situation where the number of CSs continuously changes from one block to the next. Although, this situation is not expected to occur during regular system operation (rather, we expect the fluctuation in the number of CSs to be much slower), we have chosen this scenario to study the robustness and responsiveness of msSTART in this extreme situation. Finally, scenario 4 is a more reasonable version of the previous extreme case. At the same time, we also study possible limiting behavior of the system as the number of CSs per block starts to increase.

The performance metrics derived from the simulations is the average delay $E(D)$ (in number of blocks) and the throughput of the system. These metrics are organized in a series of delay vs. offered load curves to be presented shortly.

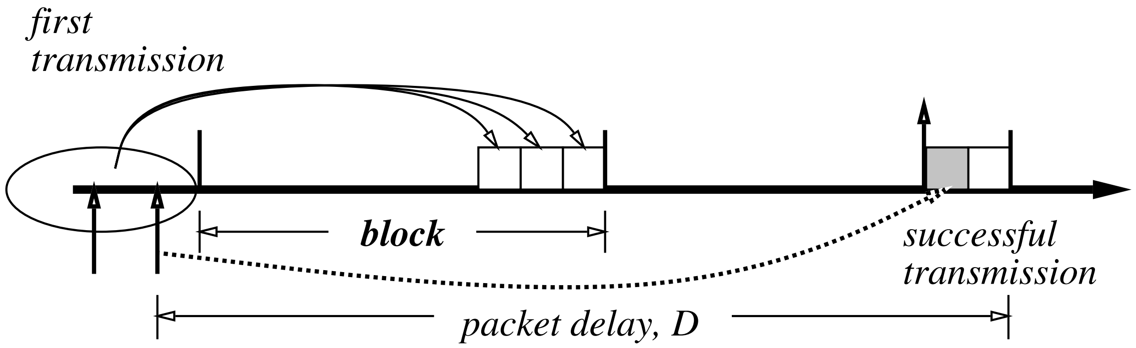

Figure 7: Delay definition

The delay D counts the time elapsed from the time the packet is generated by the Poisson source, until the end of the block in which it is successfully transmitted, see figure 7. We have chosen to register the delay in this manner so as to disassociate our results from the actual size of a contention slot and its location within a block. Without really providing more information, the latter elements may unnecessarily complicate our analysis. The integer part of the delay D represents the number of algorithmic steps that the station performed until it transmitted its packet.

For the delay results shown next, depending on the offered load λ, the simulations lasted until the delays from anywhere of at least 250,000 up to at least 1,550,000 successfully transmitted packets were registered. In order to avoid biasing due to transients and due to "forgotten" packets in the system, we extended our simulations until the first time, beyond their scheduled termination point, that the system becomes empty.

For scenario 1 and for a given λ, we have also obtained one more performance metric. We have constructed the *collision multiplicity* histogram of each RAA, i.e., we have obtained the distribution of the number of simultaneous transmissions that occur in a randomly selected slot. Since, in this case, individual probabilities are sought, instead of mere averages, to increase the reliability of our results, we have averaged out the simulation results obtained from 6 independent simulation runs for each RAA considered. Each simulation run lasted until at least 2,050,000 successfully transmitted packets were observed.

4.4 Numerical results

We start with some analytical results borrowed from [2] and [9]. In particular, table 2 shows the maximum (stable) throughput, under the IPS model, for the various RAAs considered here. All these RAAs operate with a C/NC feedback information. The last entry in table 2 has obtained through simulations. It represents a case like the ideal ALOHA, where now following a collision of k packets, the splitting parameter n is a function of k, e.g., $n = k$, thus, always providing the best possible splitting opportunity. This RAA is not implementable,

RAA	throughput	comments
binary tree-search	0.3465	blocked access
msSTART ($n = 2$)	0.3601	free access
ternary tree-search	0.3661	blocked access
stable "ALOHA"	0.3678	any known stabilized ALOHA
msSTART ($n = 4$)	0.3992	free access
msSTART ($n = 3$)	0.4015	free access
msSTART*	0.407	optimal splitting

Table 2: Throughput for various RAAs under the IPS model; scenario 1

but it provides for valuable limiting performance. As it is apparent from table 2, the case where the splitting parameter n remains fixed to the value 3 provides performance which is practically identical to that of the ideal msSTART. In the sequel, we will consider only the $n = 3$ case.

In figure 8, we compare the offered load due to newly generated packets, λ, vs. average delay, $E(D)$, trade-offs for the RAAs mentioned earlier, when there is always a single CS block. It it clear from figure 8 that msSTART provides the best throughput/delay trade-off from all the implementable RAAs. For light loads the ideal ALOHA provides better delay, because it continuously adjusts itself for the best possible transmission probability, something that it cannot happen in reality. The implementable (stable) ALOHA achieves higher delays than msSTART for all arrival rates. Both versions of ALOHA reach their stability capabilities (under the IPS model) when $\lambda = e^{-1}$, see table 2.

For scenario 1, we have also derived the collision multiplicity histogram for the various RAAs when $\lambda = 0.30$ packets per block. In an HFC environment, collision multiplicity is of interest since excessive signal power due to a high multiplicity collisions may cause unpredictable non-linearities on the HFC network that could disturb unrelated transmissions in adjacent upstream channels.

We present our collision multiplicity results in two graphs, one linear, see figure 9, and one logarithmic, see figure 10; each figure is better suited for showing the same histogram at different ranges. From the graphs, it is clear that at $\lambda = 0.30$ the ideal ALOHA has the best behavior, in the sense that the collision multiplicities are smaller than in the other RAAs. Among the implementable RAAs, msSTART has the best behavior, while the binary tree-search has the worst, about 0.11% of the CSs contain collisions of more than 10 packets. This behavior is attributed to the lower throughput of the binary tree-search, see table 2. Also, the tree-search algorithms are blocked-access algorithms, hence, it is relatively quite probable that the first transmission of a collision resolution cycle results to the collision of a large number of anxiously waiting newcomers.

To complement the last two figures, for scenario 1 and $\lambda = 0.30$, table 3 shows the portion α of time for which there exists at least one packet waiting to be transmitted in the system, i.e., α is the probability that the system is *active*. For scenario 1, in an ideal system, in

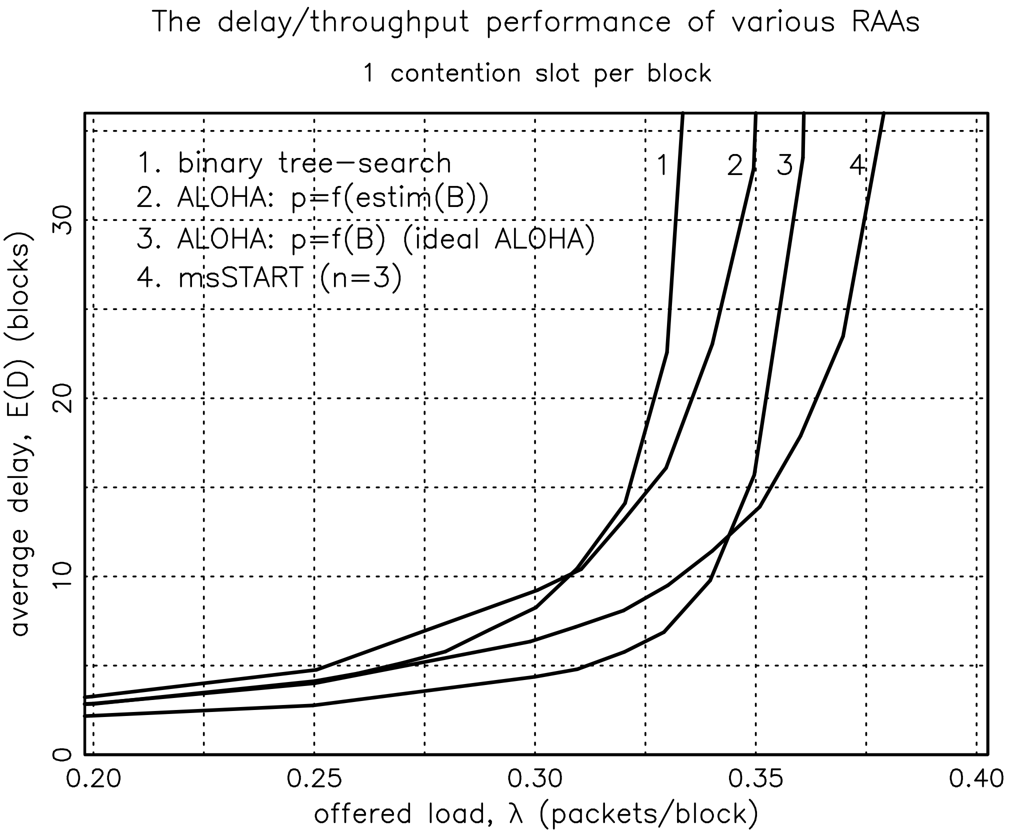

Figure 8: Performance comparison of msSTART with other popular RAAs; scenario 1

which no collisions occur, α equals λ. Hence, the *distance* $d = |\alpha - \lambda|$ could be used as a metric of the speed of the chosen RAA: the smaller the d, the better. Once again, among the imlpementable RAAs, msSTART is the "fastest" RAA. Though, it is interesting to note that the slowest RAA is the implementable ALOHA, signifying that for this RAA it may take longer times to resolve collisions than with the other RAAs. Since figure 8 shows that, at $\lambda = 0.30$, the implementable ALOHA has lower delay than the tree-search algorithm, table 3 implies that, for the given λ, the implementable ALOHA should exhibit higher delay variance and higher delay percentiles than a tree-search algorithm.

In figure 11, we compare the offered load vs. delay trade-offs for the RAAs mentioned earlier, when there is always two CSs per block. In this figure, we show two flavors of the tree-

RAA	binary tree-search	ALOHA (implementable)	ALOHA (ideal)	msSTART
α	0.5516	0.5953	0.4201	0.5340

Table 3: The probability a of a nonempty system for scenario 1, when $\lambda = 0.30$

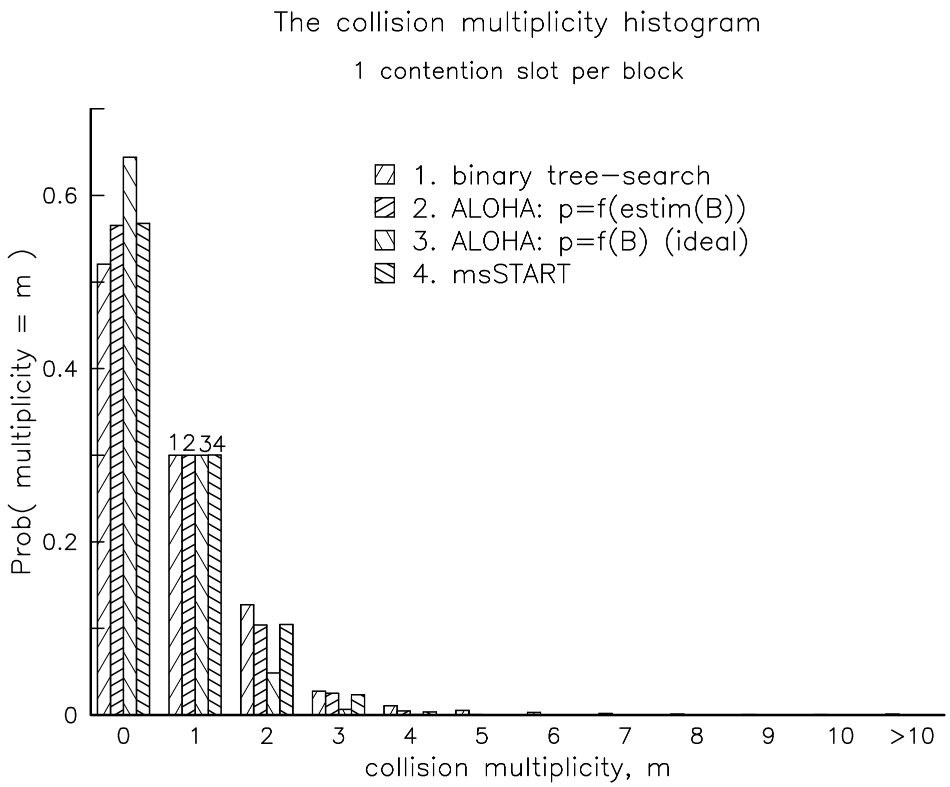

Figure 9: The collision multiplicity histogram for scenario 1 and $\lambda = 0.30$; linear bar graph

search and msSTART algorithms. In their *interleaved* versions (dashed curves 1a and 4a), two independent copies of the corresponding single CS RAA are used. Newcomers randomly select one or the other of the CSs and perform their collision resolution on the corresponding CS at each block. In the interleaved case, the stations, once they select a CS, act as if there is only one CS available per block. In the "non-interleaved" versions (solid curves 1 and 4), stations utilize both CSs for their collision resolution. The two flavors of these RAAs have the same maximum throughput, but as it can be easily seen the non-interleaved versions have superior delay performance, since they utilize better the combination of both CSs simultaneously.

In this case, (non-interleaved) msSTART is also very competitive with respect to delay with the ideal stable ALOHA for any arrival rate. All RAAs studied have maximum stable throughput which is 2 times the their throughputs achieved in the single CS case.

Next, in figure 12, we compare the offered load vs. delay trade-off for the case where the number of CSs in each block varies. We either have 2 or 8 CSs in alternate blocks, and hence the average number of CSs per block is 5. Here, we study only two types of RAAs: (a) the msSTART, and (b) the ALOHA RAAs. It may be possible to adapt other RAAs in this environment, but this is beyond the scope of this paper. In any case, as the previous studies have established, the free-access msSTART provides better performance than the

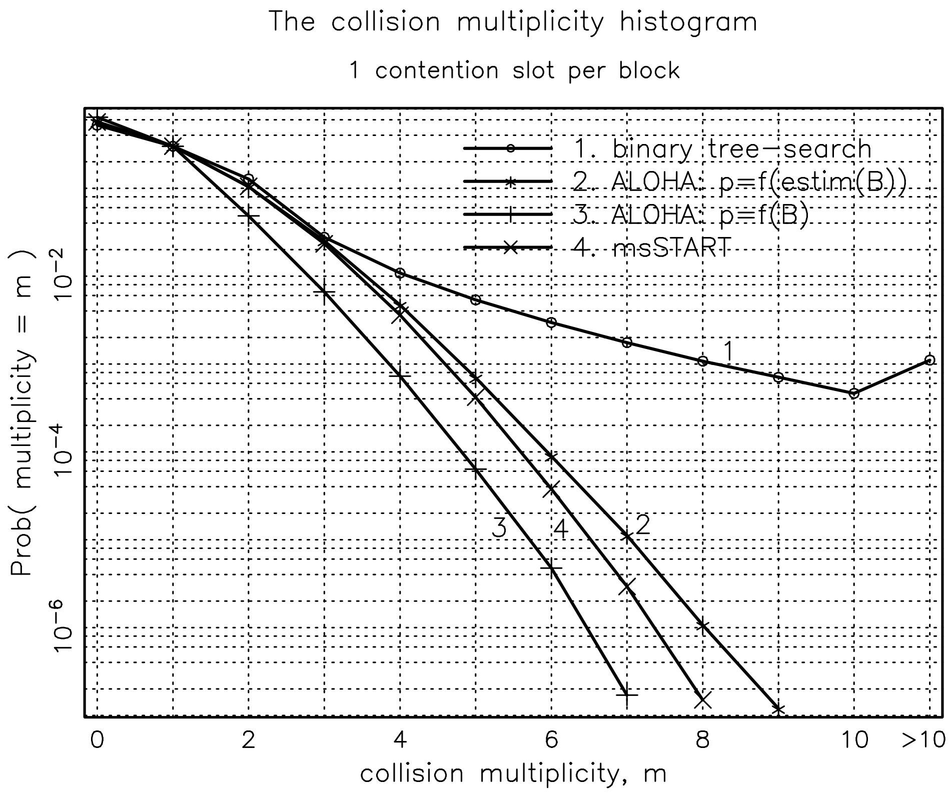

Figure 10: The collision multiplicity histogram for scenario 1 and $\lambda = 0.30$; logarithmic graph

blocked-access tree-search RAA studied, a characteristic that we believe will persist even if/when a variable multi-slot version of the tree-search RAA is developed.

Once again, it is clear that msSTART achieves higher throughput, and has a delay that is competitive to that of the ideal ALOHA for any offered load for which ALOHA is stable. Also, all RAAs achieve throughputs that are pretty close to 5 (the average number of CSs per block) times their throughputs achieved in the single CS case.

Finally, in 13, we compare the offered load vs. delay trade-off for the case where the number of CSs in each block varies randomly. Each block has either 4, 10, or 16 CSs, and hence the average number of CSs per block is 10. Note that all three RAAs appear to achieve maximum throughputs that is close to 10 times their throughput in the single CS case, and, hence, we draw the same conclusions as before.

Based on the results shown in figures 8 to 13, the following interesting observation can be made. As the (average) number of CSs per block increases, the observed delay curves flatten, and become less distinguishable among the various RAAs, at least through most of their corresponding stability regions. At the same time, as the offered load approaches closer to the maximum throughput, the delay curves shoot up more and more sharply. Figure 14 demonstrates our observation for the msSTART and ideal ALOHA RAAs.

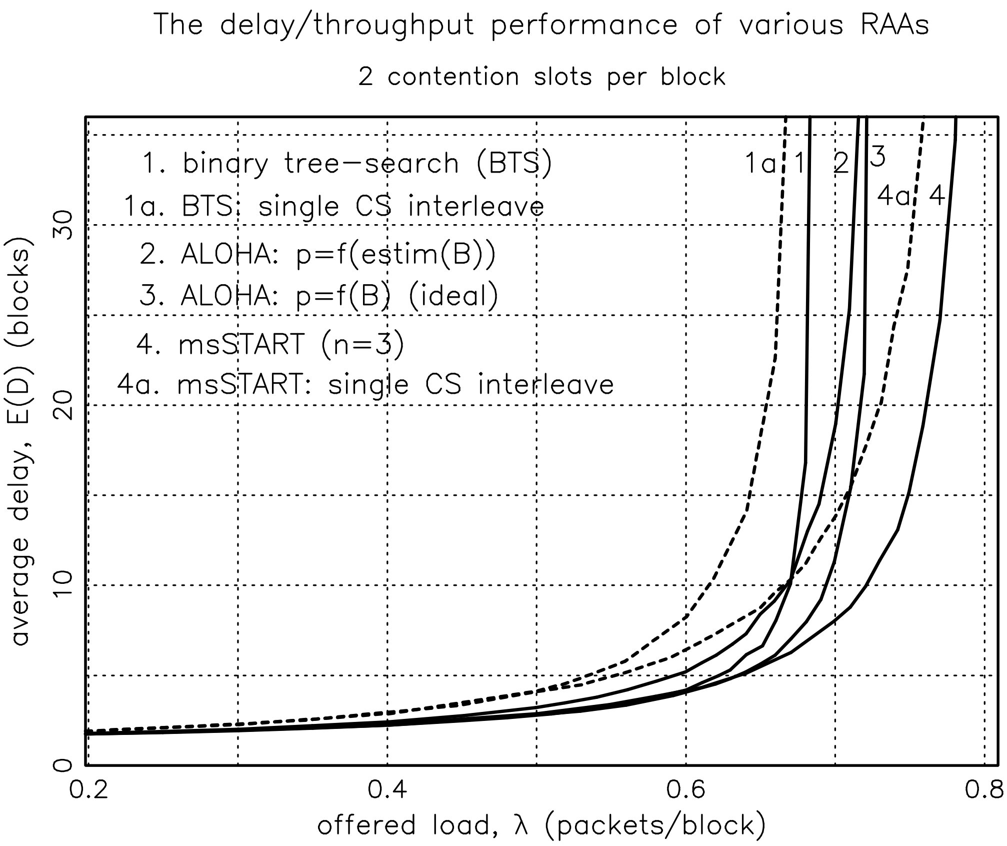

Figure 11: Performance comparison of msSTART with other popular RAAs; scenario 2

Based on the previous observation, we may go as far as to conjecture the following statement, the proof of which should rely on the *law of large numbers* [7]: As the number of CSs per block increases to infinity, the delay/throughput curves of the RAAs studied here remain flat at a height of 1.5 blocks for any λ between 0 and a point ever closer to the maximum stable throughput of the RAA; moreover, the maximum throughput is proportional to the average number of CSs per block, at least in the case of a fixed number of CSs per block. Hence, due to its higher throughput, as the offered load increases msSTART will eventually consistently outperform any of the other RAAs, even though all RAAs could have comparable delay characteristics at lower loads.

Note that as the (average) number of CSs per block increases, the sudden delay increase close to the system stability point requires us to perform longer simulation runs to derive reasonably reliable results, since, in this case, the sensitivity of the delay to the offered load becomes very high. This is the reason for the somewhat jerky curves at high loads in figures 12 and 13.

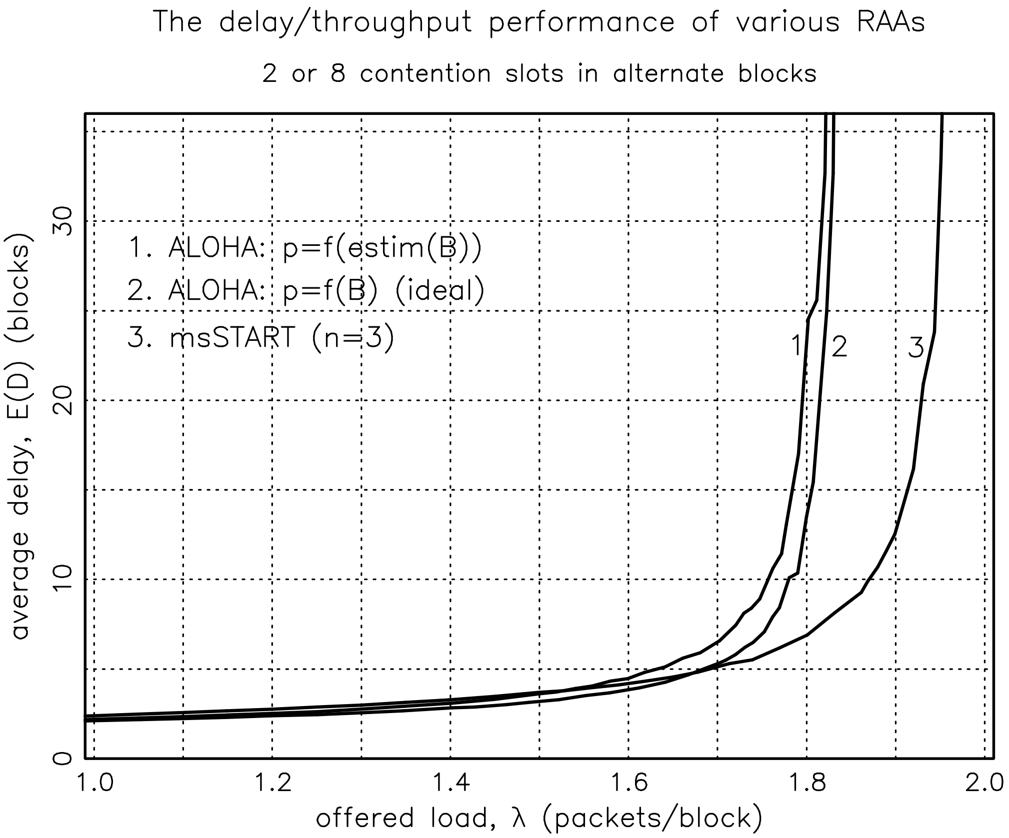

Figure 12: Performance comparison of msSTART with other popular RAAs; scenario 3

5 Concluding remarks

We have presented the msSTART RAA. The msSTART RAA is an evolution of the stack-based START algorithm. It is an easy to implement, high-efficiency, free-access, robust, stable RAA optimized to operate with a dynamically changing number of contention slots. The latter is achieved by disassociating the msSTART counter update algorithm from the number of CSs in the upcoming block. Thus, msSTART adjusts effortlessly to any variations in the number of CSs from block to block.

The msSTART algorithm is part of the MLAP MAC protocol for HFC networks, [3], yet it is not tied to it. It can be utilized on its own or within other MAC protocols that regulate station transmission over shared media. For example, it can be used in wireless networks, where uplink and downlink transmissions can naturally be mapped to the upstream and downstream transmissions considered in msSTART and, more generally, in HFC networks. Also, although in our presentation, we have assumed that the upstream channel consists of fixed size blocks, a close inspection of the operation of msSTART reveals that this is not a requirement for the algorithm. The msSTART algorithm could be made to work also over variable size blocks (or, no blocks at all). The algorithm only needs to be able to distinguish

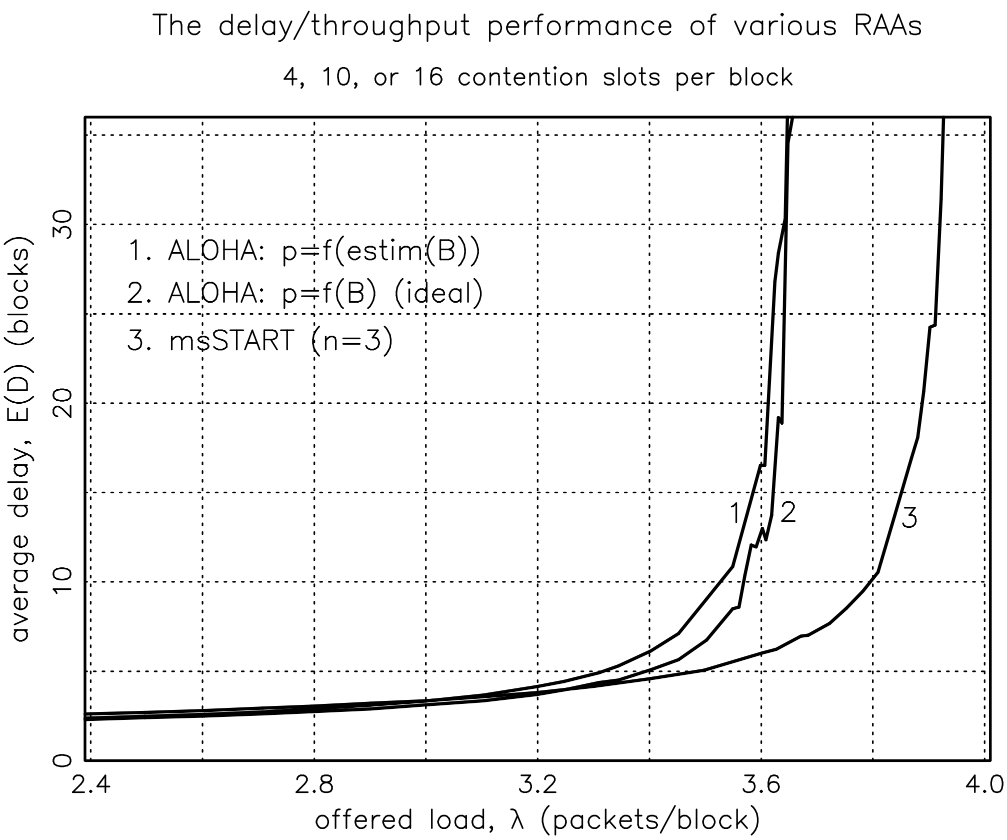

Figure 13: Performance comparison of msSTART with other popular RAAs; scenario 4

the group of CSs in a block from the group of CSs in the following block. Clearly, for the latter there is no need for fixed size blocks.

We have studied via simulations the performance of the msSTART RAA, and compare its performance with that of other popular RAAs under networking scenarios where the number of CSs per block can change dynamically. We have shown that msSTART provides the best delay/throughput trade-off among the RAAs studied without sacrificing in complexity. Although, there exist other much more efficient, even more efficient than msSTART, tree-search RAAs than the ones considered here, [1, 11], we have not considered them in this study because these algorithms are much more complicated in implementing, especially in a multi-slot environment, that renders them as unlikely candidates for adoption by the 802.14 WG.

Our study can easily extend to cover the case where *interleaving* is used to increase the efficiency of the RAA. Interleaving, i.e., the multiplexing of a number of independent collision resolutions is used when the round-trip delay on the network and the time to receive feedback information becomes excessively high. In this case, the use of a single collision resolution instance reduces substantially the network utilization.

Future extensions of this work should be focused on the analytical study of msSTART, as

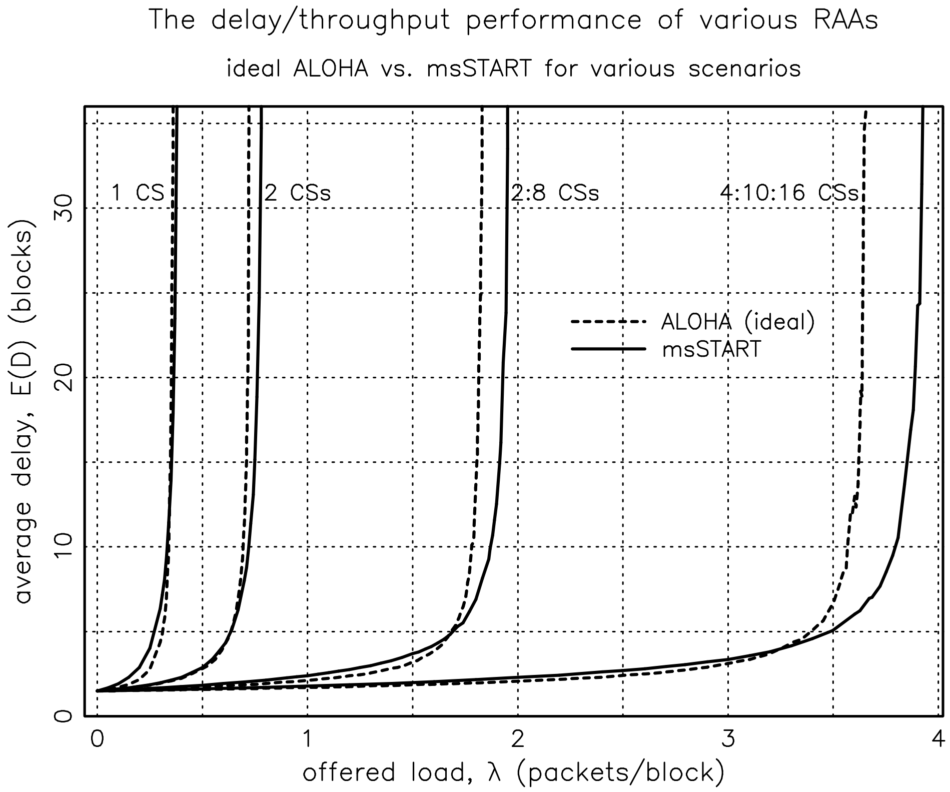

Figure 14: The delay vs. offered load for msSTART and ideal ALOHA; all scenarios

well as the other RAAs, in the multi-slot environment. Also, it will be intersting to establish the validity, or not, of our conjecture in the end of the last chapter.

Finally, as a by-product of our study, we have demonstrated how the class of stack RAAs can be expanded to encompass many known RAAs, including the block-access, tree-search algorithms, and the ALOHA RAAs. This exercise could provide us with the foundations for developing common mechanisms for implementing a large number of RAAs.

References

[1] D. Bertsekas and D. Gallager, *Data Networks, 2nd ed.*, Prentice Hall, 1992.

[2] C. Bisdikian, "A review of random access algorithm," IBM Res. Rep. RC20348, Jan. 1996; see in *IBM Cyberjounral* at URL "*http://www.watson.ibm.com:8080.*"

[3] C. Bisdikian, B. McNeil, R. Norman, R. Zeisz, "MLAP: A MAC level access protocol for the HFC 802.14 network," *IEEE Commun. Mag.*, vol. 34, no. 3, pp. 114-121, March 1996.

[4] "The Cable On-Line access system," IBM Corp., at URL "*http://www.raleigh.ibm.com/cab/cabprod.html.*"

[5] *IEEE Spectrum*, special report on "Digital TV," vol. 32, no. 4, April 1995.

[6] "Internet surfers to fuel cable modem market," *Dataquest Interactive*, April 1996; see at URL "*http://www.dataquest.com/.*"

[7] L. Kleinrock, *Queueing Systems, vol. 1: Theory*, John Wiley & Sons, 1975.

[8] C. C. Lee, D. Lin, R. Citta, "Performance analysis of adaptive random access protocol, part II," contr. no. IEEE802.14-96/071, *IEEE 802.14 working group* (WG) meeting, San Diego, CA, USA, March 1996.

[9] P. Mathys and P. Flajolet, "Q-ary collision resolution algorithms in random-access systems with free or blocked channel access," *IEEE Trans. on Inform. Theory*, vol. 31, no. 2, March 1985

[10] L. Merakos and C. Bisdikian, "Delay analysis of the n-ary stack algorithm for a random access broadcast channel," *IEEE Trans. on Inform. Theory*, vol. 34, no. 5, Sept. 1988.

[11] R. Rom and M. Sidi, *Multiple Access Protcols: Performance and analysis*, Springer-Verlag, 1990.

[12] T. S. Perry, "The trials and travails of interactive TV," *IEEE Spectrum*, vol. 33, no. 4, pp. 23-28, April 1996.