IEEE
COMMUNICATIONS
SURVEYS

A Review of Contention Resolution Algorithms for IEEE 802.14 Networks

Nada Golmie, Yves Saintillan, and David H. Su
National Institute of Standards and Technology

Abstract

Bidirectional Cable TV networks using hybrid fiber coaxial (HFC) systems are good examples of broadcast environments where a contention resolution algorithm is needed in order to allocate the multiaccess medium (in this case the upstream link) among the various nodes. Recent activities of the IEEE 802.14 Working Group aimed at defining the physical and medium access control (MAC) layer protocols for HFC cable networks have focused on the study and evaluation of several contention resolution solutions for inclusion in the MAC protocol specifications. In this article several contention resolution algorithms considered by the IEEE 802.14 group are reviewed. Different implementations for several well known contention resolution algorithms such as tree-based and p-persistence are presented. Their performance is evaluated in the HFC context with respect to upstream channel allocation, roundtrip delay, various traffic types, and number of stations in the network. Simulation results for configurations and scenarios of interest are also presented.

Emerging residential cable networks with burst speeds up to 40 Mb/s — more than 2000 times faster than an ordinary dial-up modem — will most likely provide the next-generation data communication services, including Internet access to homes and small businesses. This giant leap forward with higher bandwidth and better quality of TV signals is the direct result of introducing the HFC technology in the cable plant. CATV operators have already replaced long chains of copper wiring and amplifiers by fiber-cable and electro-optical converters/multiplexors called fiber nodes. Today most of the major multiple systems operators are either upgrading their cable plants or rolling out a new broadband infrastructure. In 1994 the IEEE 802.14 Working Group was formed to define a physical layer (PHY) as well as a MAC layer for bidirectional data transfer over an HFC-CATV network. The draft specifications, as of the writing of this article, support mainly ATM cell transfers, thus maintaining the ability to provide high quality of service (QoS) and accommodate current and future CATV applications along with television broadcast, namely, video on demand, interactive computer games, and video telephony.

The 802.14 Working Group spent considerable time on the selection of contention resolution algorithms. The two major contenders were tree-based and p-persistence algorithms. After significant deliberations, the group selected the tree-based algorithm mainly because it generates lower access delay variations, which is critical to supporting QoS. During the course of the debate proponents of each camp produced enhancements, many of which were incorporated into the selected tree algorithm. As an unbiased third party, the National Institute of Standards and Technology (NIST) conducted extensive evaluations of the various proposals. The results of these evaluations are presented here. Our goal in this article is to review contention resolution algorithms (CRAs) considered by the IEEE 802.14 Working Group, and summarize the performance results that compared the alternatives and helped make decisions regarding the choice of CRA. We present a wide range of algorithms and features and guide the reader through the selection process by presenting the advantages and disadvantages of each as discovered via extensive simulations. This article's main contribution is to inform researchers of the state of the art in this area and open the window for further improvements in future implementations.

The rest of the article is structured as follows. In the next section, we describe the HFC environment and give details on the MAC operation; in the section that follows we focus on the main elements that constitute a contention resolution mechanism and discuss their relative performance. We then study a number of factors such as bandwidth allocation, round trip delays, traffic types, and number of stations that may have a significant impact on performance. In the final section we offer conclusions.

HFC MAC Protocol Overview

CATV networks are characterized by a tree and branch topology. At the root of the tree, a base station or headend controls the traffic, as shown in Figure 1. The bandwidth is divided into several channels; some are dedicated to downstream communication (from the headend to the stations) while others are for upstream transmission (from the stations to the headend).

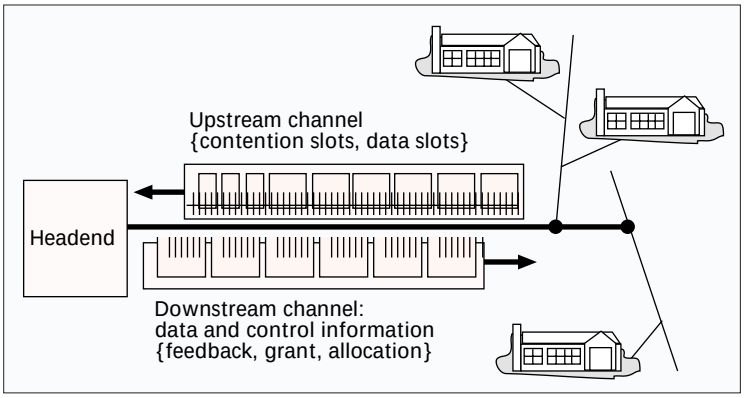

Figure 1. *CATV network structure.*

The upstream data streams are segmented into fixed-size slots, or minislots. Several minislots can be concatenated in order to form a data slot while one contention slot maps into one minislot. Data slots containing subscriber data packets are assigned to stations by reservation. The IEEE 802.14 draft specifications assume that data traffic is carried via ATM cells. The MAC layer prepends an octet to each ATM cell in order to form a MAC packet data unit (PDU) that is carried in a data slot. Contention slots carry stations' request for bandwidth in "shared" or contention mode. Since more than one station can transmit a request at the same time, resulting in a collision, a contention resolution algorithm must be implemented as part of the MAC protocol. The downstream data streams are segmented into a fixed slot size of MAC PDUs. The MAC reservation cycle, or request/feedback cycle, is defined as the time elapsed between request transmission and feedback reception at the station that is farthest from the headend. This insures that all stations have the same opportunity of transmission in any given slot and prevents unfairness due to the relative location of the stations with respect to the headend. A bandwidth allocation algorithm running at the headend controls the number of contention slots and data slots contained in a MAC reservation cycle. The headend also decides on the distribution and location of the contention slots and data slots on the upstream channel. Contention slots can be either grouped together in clusters or distributed over the request/feedback interval. Unless specified otherwise, we assume that minislots are grouped at the beginning of each request/feedback interval. The information about the contention/data slot location is sent to the stations on the downstream channel.

The basic MAC operation is as follows. Upon the arrival of the first data packet, a station sends a request packet in a contention slot that conforms to the First Transmission Rule (FTR). (FTR, which governs the access of newcomer stations, is discussed in greater detail in the next section.) Then the station waits for the request feedback from the headend. If more than one station sends a request in the same contention slot, a collision results. Requests are then retransmitted according to a collision resolution algorithm. In the next section different algorithms are presented and evaluated. It must be stressed, however, that the characteristics of HFC networks impose a number of constraints. Stations cannot monitor collisions because modem receivers and transmitters are tuned to different frequencies for the downstream and upstream channels. Feedback about collisions must be provided by the headend either explicitly or implicitly. The algorithm must also take into account the delay before a station receives the feedback information in order to insure the best utilization of the network resources. In case of a successful request transmission, the station waits for a data grant in order to send its data. At this point if the station has additional requests for bandwidth it may choose to bypass the contention process and use the Extended Bandwidth Request field available in the MAC PDU. This is known as "piggybacking." In order to allocate data slots, the headend uses schemes such as first come first served (FCFS) or round robin (RR); we use RR in our simulations. It is assumed that the headend can deal with priority traffic for both contention resolution and bandwidth allocation. Similarly, it is able to distinguish between different connections and provide QoS.

In this study, a generic MAC model conforming to the above description is implemented to evaluate the various contention resolution algorithms. The default parameters used in the simulations are set according to Table 1 and, unless mentioned otherwise, the cluster mode ternary-tree algorithm is used to resolve collisions. The traffic type used, unless specified otherwise, is based on 48-byte packets generated according to a Poisson distribution with a mean arrival rate of λ (λ is a function of the offered load and the total number of stations).

Simulation parameter	Values
Number of active stations	200
Distance from furthest station to headend	80 km
Downstream data transmission rate	30 Mb/s
Upstream data transmission rates channels	3 Mb/s
Propagation delay	5 μs/km for coax and fiber
Length of simulation run	30 sec
Length of run prior to gathering statistics	3 sec
Guardband and preamble between stations transmissions	Duration of 5 bytes
Data slot size	64 bytes
Contention slot size	16 bytes
DS/CS size ratio	4:1
MAC reservation cycle	1.536 ms (36 minislots)
Maximum request size per contention slot	32
Bandwidth allocation scheme	Fixed ratio (12 CS, 6 DS)
Collision resolution scheme	Ternary-tree, T_{bound} access
Headend processing delay	0 ms

Table 1. *MAC default parameters.*

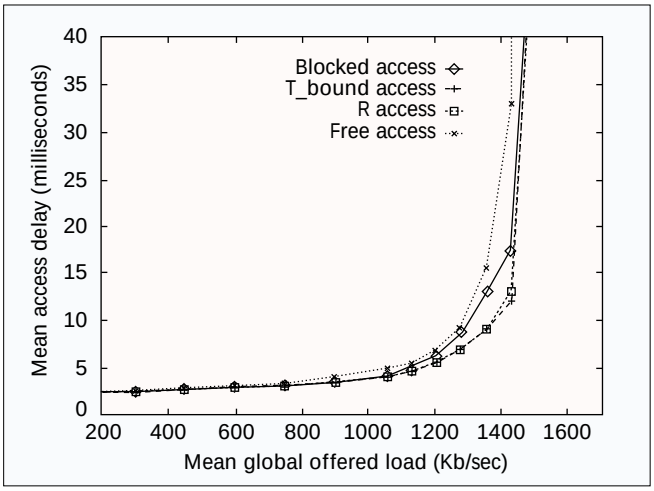

Figure 2. *FTR: mean access delay vs. offered load.*

CONTENTION RESOLUTION

Contention resolution algorithms were subject to much interest in the early 1970s for usage in packet radio transmission, and especially during the development of the ALOHANET project [2]. Since then, much research has been devoted to devising efficient contention resolution mechanisms for multiaccess media for local area networks (LANs), metropolitan area networks (MANs), satellite networks, and radio networks.

The many strategies that have been developed to solve the generic problem of having two transmitters send packets simultaneously can be divided into two basic paradigms. One is the "free-for-all," similar to the one used for the ALOHANET, in which nodes attempt to retransmit collided messages hoping for no interference from other nodes [3]. A variation of this technique adapted to HFC networks, known as p-persistence, associates with each slot a transmission probability, *p*, usually controlled by the headend [10]. Hence collided requests are retransmitted with a probability *p*. This process is repeated until a request is successfully received at the headend. The other paradigm consists of splitting collided nodes into a tree structure. In this tree-based mechanism [4], all nodes involved in a collision split into a number of subsets. The first subset transmits first, followed by the second subset, then the remaining subsets. The chances of future collisions are reduced by forcing stations that collided in the same slot (assuming a slotted frame structure) to retransmit their requests in different slots in the future (thus distributing the number of contending stations over several slots).

These two basic schemes have evolved to adapt to various network environments and constraints. Unlike carrier sense multiple access techniques, which are used where the ratio of propagation delay to packet transmission time is relatively small ($<< 1$) and the stations can monitor the transmission channel, reservation (request/grant) techniques are used for HFC networks. Round trip delays typically range between 0.8 to 2 ms, including propagation time for distances up to 80 km. Also in order to further increase the throughput of the upstream channel where data slot intervals are rather long (in the order of 170 μs for a 3 Mb/s channel), short packets (the ratio of reservation to data packets is typically 1/4) are used to reserve long noncontending slots for sending data. Thus, only short slots are wasted by idles or collisions leading to a better overall channel efficiency.

The key functions provided by a CRA consist of:

- Controlling the transmission of new requests by means of the FTR
- Giving collision feedback
- Managing retransmissions

The FTR regulates when a newcomer station is allowed to send its first request on the upstream link. A feedback message informing the station about the status of its reception at the headend is associated with each contention slot. The status of a contention slot can be empty, successful, or collided, depending on whether there is zero, one, or more stations transmitting in it. Finally, a mechanism is needed to resolve collisions and control the retransmission of request packets. This process is best visualized by means of a stack and is designated as stack management.

In the following sections several solutions for FTR and the stack management are presented and compared. Some were considered during the IEEE 802.14 standardization process. Others appeared in the literature. Simulation results are presented. For each test scenario we use different assumptions in order to stress the impact of each solution on the overall system performance.

FIRST TRANSMISSION RULE

In this section we present different strategies to control the admission of newcomer stations. These strategies, also called first transmission rules (FTRs), define in which minislot a station with a new packet is allowed to send a request. FTRs can be classified as follows:

- *Free Access:* the first transmissions of requests are allowed to take place on the same minislots used to retransmit collided requests. Newcomer requests are mixed with "old" or retransmitted requests.
- *Blocked Access:* new requests are not allowed on the minislots used to resolve current collisions. This is illustrated by a contention interval that is split into two regions. One is reserved for ongoing collision resolution and another, denoted as newcomer minislot region, is open for newcomer requests.

Furthermore, different strategies can be applied to each mode. For example, the Free Access mode can be either persistent or nonpersistent. In this section we consider only the nonpersistent Free Access strategy, but several Blocked Access mechanisms considered by the IEEE 802.14 are reviewed.

In the nonpersistent Blocked Access (simply called Blocked Access) all newcomer stations are allowed in the current newcomer region. In case of an opening, even if only one contention minislot is available for newcomer contention, all backlogged stations with new requests transmit in that minislot.

In the Blocked Access using persistence *R*, only a portion

Algorithm	Load (% cap.)	Max	Mean	σ
T_bound access	40	7	2.211	1.695
	50	8	2.277	1.651
	60	8	2.275	1.649
Blocked access	40	13	2.541	1.973
	50	33	3.589	3.049
	60	38	4.000	3.664
R access	40	8	2.217	1.704
	50	7	2.283	1.657
	60	8	2.277	1.649
Free access	40	6	2.230	1.643
	50	8	2.343	1.530
	60	8	2.345	1.526

Table 2. *FTR: collision multiplicity.*

$MS(j+1)/R$ of newcomer stations are allowed to transmit in the newcomer region of the current $j+1$ cycle, where $MS(j)$ is the number of contention minislots in the region and R is a range parameter computed at the headend. R determines the range over which a station can randomly select a slot of the newcomer region. If the random number chosen is greater than MS, the station repeats the process the next cycle. This method, also known as Soft Blocking or R Unblocking [11, 12], was voted on in November 1996. In this article, it is referred to as R Access and R is computed according to the following:

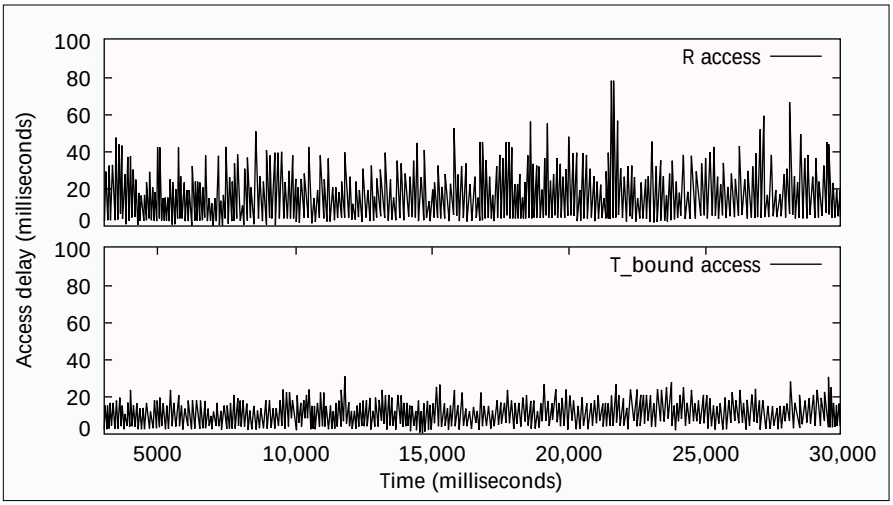

Figure 3. *FTR: access delay vs. time.*

$$R(j+1)=\max\left\{\min\left[n, R(j)-MS(j)+col(j)*\frac{(e-1)}{(e-2)}+\frac{MS(j)}{e}\right], MS(j+1)\right\} \quad (1)$$

where $e = 2.718$, n is the number of stations, and $col(j)$ is the number of collided minislots in the jth cycle. This computation of R is based on the approximation of the traffic load used for the stable ALOHA (also called adaptive p-persistent).

On the other hand the Blocked Access using T_{bound} was proposed by Bisdikian in [9] and adopted by the committee in November 1997. It is based on a time moving window and emulates a first-in-first-out (FIFO) access ordering that is known to be optimal [3]. Every cycle, the headend sets a value T_{bound}. Only if its message is generated prior to T_{bound}, than a station is allowed to transmit its request in the newcomer region (it selects a minislot randomly). Thus the newcomer admittance becomes deterministic and is no longer a pure random process. T_{bound} can be computed from R used previously according to:

$$T_{bound}(j+1)=T_{bound}(j)+\frac{MS(j)}{R+1}*\left(T_{current}-T_{bound}(j)\right) \quad (2)$$

where $T_{current}$ is the time at which R is computed. This method, called T_{bound} Access, significantly reduces the delay variability.

To compare the relative performance of the nonpersistent Free Access with the variants of the Blocked Access defined above, we look at the the mean access delay and collision multiplicity for different offered loads, and the access delay versus time for a particular load. The mean access delay is defined as the average transmission time for a packet from the moment it is generated until it is received at the headend. This includes, in addition to all queuing delays at the MAC layer, the time to transmit a request and resolve collisions. Collision multiplicity is another significant performance measure of the FTR. It is defined as the number of stations that collide in one slot. The offered load represents the total packets generated by all stations in Kb/s.

In Fig. 2 we plot the mean access delay vs offered load for all four admittance policies described above. Starting from 20 percent of the channel capacity (600 Kb/s) the access delay for the Free Access is slightly above the access delay for the other mechanisms. At 47.5 percent capacity (1425 Kb/s) the difference in mean access delay is 15 ms between the Free Access and the Blocked Access and 20 ms between the Free Access and the T_{bound} Access. Since in Free Access requests from newcomer stations can mix with collided stations, the collision resolution is less effective than in the case of Blocked Access, which results in higher mean access delays at higher loads. The Blocked Access method was mainly introduced in order to resolve collisions more efficiently. However, Blocked Access starts to exhibit problems starting from 40 percent (1200 Kb/s) where its mean access delay is higher than the delays for R Access and T_{bound} Access (5 ms difference at 45 percent). This is mainly due to a high collision multiplicity factor. With Blocked Access all backlogged stations are let into the system at once. As the load is higher, the number of contention slots reserved for newcomer access is relatively small, which may cause a large number of stations (maximum of 33 at 50 percent from Table 2) to collide in one slot. The maximum values for the Blocked Access collision multiplicity are much higher than for all other methods: we measure 13, 33, and 38 at 40, 50, and 60 percent of capacity, respectively, compared to maximums of 6, 8, and 8 at the same loads for Free Access (Table 2). R Access and T_{bound} Access, while controlling a station's entry, have a relatively low collision multiplicity, since in both cases the newcomers are let into the system gradually. The maximum, mean, and standard deviations of collision multiplicity are similar to those for Free Access. Although the mean access delays for R Access and T_{bound} Access are almost identical, there is a significant difference in terms of access delay between the two methods (Fig. 3). The FIFO ordering of T_{bound} Access reduces the difference between the minimum and maximum access delays or the delay variation. A relatively low access delay variability generally translates into a low delay jitter for higher layer applications. This may be critical for some ATM applications with stringent constraints on cell delay variations.

Stack Management

Often in the literature CRAs are defined in terms of operations performed on a stack [5]. The idea is based on managing a stack of collided and newcomer stations where each stack level corresponds to a different group of stations ranked from top to bottom. Note that the stack in this case is only a visualization and needs not be implemented at the station or the headend [7]. In this section we study the various stack implementations available for CRAs. We consider the ternary-tree LIFO, ternary-tree FIFO, and p-persistent algorithms with respect to the relative performance of their stack management. For ternary-tree algorithms, colliding requests are split into three groups and each group is transmitted in a different minislot. The process repeats until all collisions are resolved. For this analysis we consider that each group of requests that results in a split occupies a level of the stack. We will look at the various ways for a group to enter and

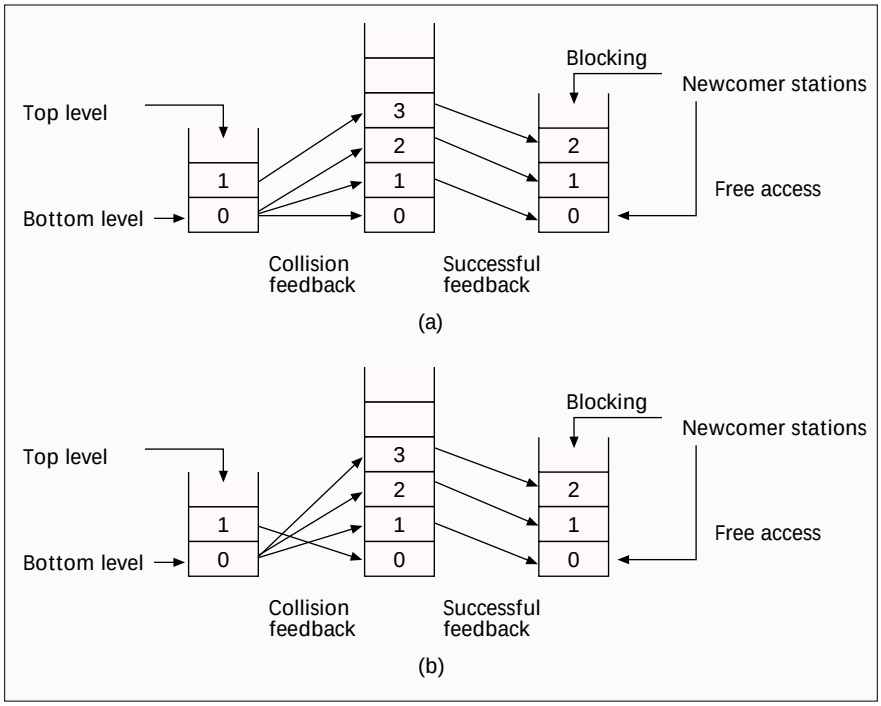

Figure 4. *Ternary-tree stack visualization: (a) LIFO stack ordering; (b) FIFO stack ordering.*

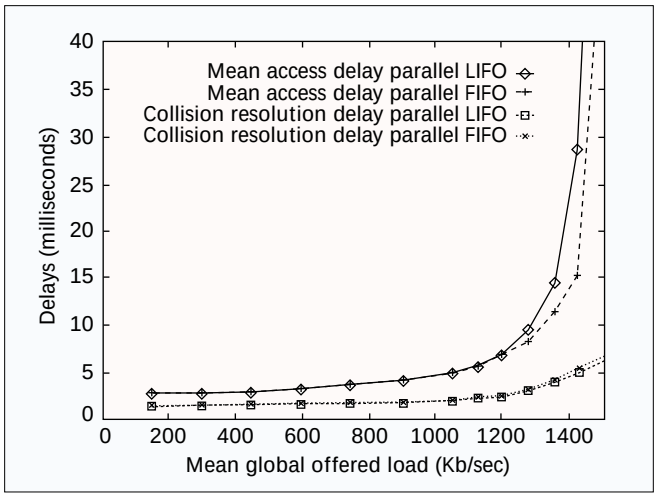

Figure 5. *Tree stack: mean access/collision resolution delay vs. offered load.*

leave the stack.

Assuming that the next group of stations that are allowed to transmit occupies the bottom level, collided stations entering the stack may either go on top or at the bottom of the stack, thus achieving a FIFO or a LIFO ordering, respectively. In Fig. 4 a stack visualization for the ternary-tree algorithm is shown. Figure 4(a) illustrates a LIFO ordering. Stations at level 0 transmit their request at the next contention opportunity. In case of collision feedback, the group of station splits into three subgroups at levels 0, 1, and 2. Stations previously at level 1 are pushed two levels up. On the other hand, in case of a successful feedback, all stations are pushed down one level. Newcomer stations entering the system for the first time can either be placed at the top or bottom level depending on the FTR. For Blocked Access Mode, stations are put at the highest level of the stack, while for Free Access Mode, stations start at level 0. Figure 4(b) depicts a FIFO stack ordering. In case of successful feedback the stack dynamics are similar to LIFO ordering, as shown in Fig. 4(a). However, in case of collision feedback, the stations at level 0 go on top while stations previously at level 1 go to level 0, thus achieving a time-based sorting. In Fig. 5, we compare the stack management with Free Access Mode for both LIFO and FIFO policies. LIFO ordering has higher access delays than FIFO starting from 40 percent of capacity (15 ms and 30 ms for FIFO and LIFO, respectively, at 47.5 percent capacity). Note that this behavior is not due to differences in the mean collision resolution delay, which is shown to be the same, but to high access delay variations of the LIFO ordering (Fig. 5).

Although the stack is a good visual aid for understanding the dynamics of splitting algorithms and usually associated with tree-based contention resolution mechanisms, the idea can be applied to p-persistence schemes as well [5]. The result is a two-level stack, as shown in Fig. 6. Based on a collision feedback, stations at levels 0 and 1 remain at the same level with a probability p and $1 - p$, respectively. On the other hand, stations at levels 0 and 1 change levels with probability $1 - p$ and p, respectively. After a successful feedback, stations at level 1 either go to level 0 with probability p or remain at level 1 with probability $1 - p$. In Fig. 7 we compare the mean access and collision resolution delays of the p-persistence and the ternary-tree implementation. For the ternary-tree we use T_{bound} Access for newcomer stations. As the offered load increases, the mean collision resolution for the ternary-tree is kept constant, while it is always increasing for the p-persistence. This is a direct result of keeping the transmission of collided requests and newcomer requests in separate contention intervals for the ternary-tree T_{bound} Access, as described previously. Starting from 30 percent of capacity, the mean access delay for the p-persistence is higher than for the ternary-tree: the difference is almost 10 ms at 45 percent of capacity. Both mean access delay curves converge after 45 percent of capacity since the maximum throughput is at 50 percent. Figure 8 shows that the percentage of collisions (relative to the total number of requests) is almost 30 percent higher with p-persistence than with ternary-tree starting from 40 percent of capacity. In Fig. 9 we plot the cumulative density function for the tree-based and p-persistence with 50 percent applied load. We observe that for the tree-based the probability that the access delay is less than 20 ms is almost 100 percent, while it is close to 75 percent for the p-persistence. This result is expected, considering the random nature

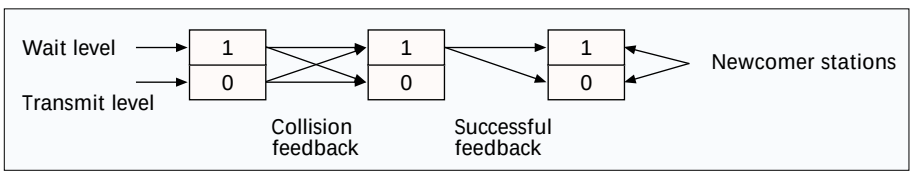

Figure 6. *p-persistence stack visualization.*

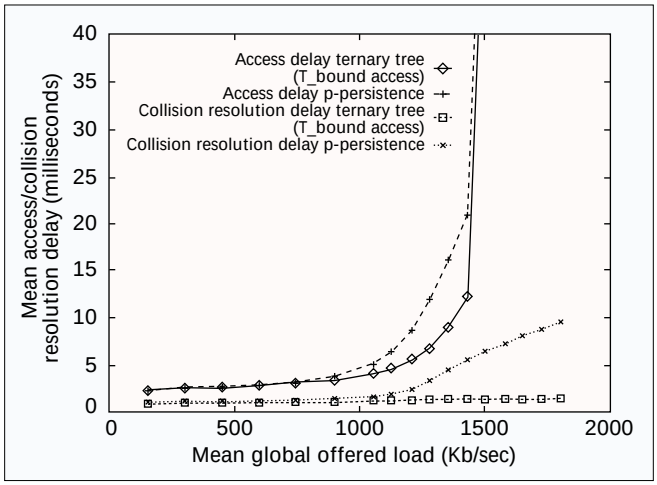

Figure 7. *Ternary-tree vs. p-persistence: mean access/collision resolution delay vs. offered load.*

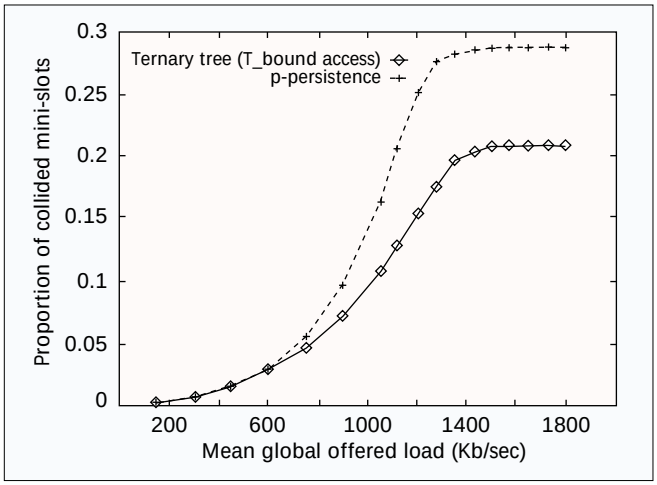

Figure 8. *Tree vs. p-persistence: percentage of collision vs. offered load.*

of the p-persistence. This result was crucial in the decision of the IEEE 802.14 group to adopt the ternary-tree algorithm for contention resolution.

TERNARY-TREE IMPLEMENTATIONS

In this section we focus on the ternary-tree contention algorithms and present three different implementations that could be used together with a LIFO or a FIFO stack management scheme.

After the station sends its request it waits for feedback from the headend. In case of collision feedback, the station enters the contention resolution cycle and activates its resolution state machine. The complexity of the state machine depends on the algorithm considered. Splitting algorithms make use of the type and frequency of the feedback in order to develop more intelligent (but also more complex) collision resolution state machines. In Fig. 10 the dynamics of three different implementations of the ternary-tree are presented over an interval of 12 contention slots. Note that data slots are omitted from the picture for a better understanding of the feedback dynamics. Both the sequential and parallel implementations, shown in Figs. 10(a) and (b), respectively, use feedback information given on each slot. However, they differ in the number of feedback messages a station is required to monitor after a collision has occurred. In the sequential implementation [6], the station monitors the feedback message every interval equivalent to a round trip delay δ to the headend. When the station counter reaches 0 the station is allowed to transmit its request. If a collision occurs and the station counter is equal to 0, the station selects a random value for its counter between 0 and 2. Otherwise, the counter is incremented by two for every collision feedback. The station decrements its collision counter by one for an empty or successful feedback. In the parallel implementation [13, 14], the station's counter dynamics are similar to the sequential case, but the counter is updated every slot rather than every round trip delay. Thus the station needs to monitor the channel for all feedback messages following a collision until the station can retransmit its request. An example is shown in Fig. 10 (a) and (b) where stations A, B, and C collide in slot 0. The feedback is received at the end of slot 2 and station C retransmits in slot 3 in both the sequential and parallel implementations. However, in Fig. 10(b), since stations A and B update their counter every slot, an empty feedback message at the the end of slot 3 causes station B's counter to go to 0 and retransmit in slot 4. In Fig. 10(a), station B's counter is only updated every δ, in this case at 2δ B retransmits in slot 6. As

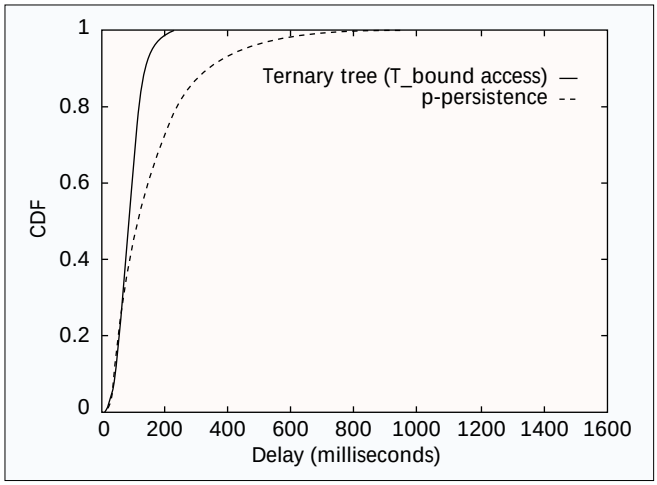

Figure 9. *Tree vs. p-persistence: probability that mean access delay < x ms, offered load = 50 percent of capacity.*

illustrated in this example, the parallel implementation may reduce the request delay by allowing stations to retransmit their collided request earlier than with the sequential implementation. The average request delay in Fig. 10(a) is equal to 6 contention slots

$$\left(\frac{3+6+9}{3}\right)$$

while the average request delay in Fig. 10(b) is equal to 5 slots

$$\left(\frac{3+4+8}{3}\right)$$

But one has to keep in mind that less request delay in the parallel implementation case comes at the cost of extra processing of feedback messages at the station. In addition, the parallel implementation requires a feedback message for each contention slot regardless of whether it is used or not. This constitutes an extra bandwidth overhead in the downstream in order to carry control information for the upstream channel.

The cluster implementation chosen by the IEEE 802.14 is shown in Fig. 10(c). It is analogous to the sequential implementation, since the station needs to tune in for the feedback every round trip delay and not every contention slot as in the parallel implementation shown in Fig. 10(b). There is even "bonus" information contained in the feedback message received at the beginning of each cluster. Note that the

feedback delay in this case accounts for the round trip propagation time plus the transmission time of all the contention slots contained in the cluster. This guarantees that feedback information from transmissions in contention slots in a cluster is available to the stations prior to the occurrence of contention slots in the next cluster. The headend can then use its knowledge of the number of collisions that occurred in the cluster in order to distribute the collided stations in the next cluster. For example, consider cluster 1 in Fig. 10(c) which contains two collided minislots: 0 and 2. All stations involved in a collided minislot (0 and 3 in this case) split into 3 subsets (for the ternary-tree), each choosing a random number between 0 and 2. The stations involved in the first collided slot 0 are dispatched in the first three slots of cluster 2 (slots 5, 6, and 7), those involved in the second collided slot may use the next three slots, and more generally, the stations involved in the ith collided slot, select a subset between (i * n + 1) and ((i + 1) * n) slots. If cluster j contains p contention slots, then the first p subsets are able to retransmit, the ith subset transmitting in the ith slot. The other subsets wait in the stack. In Fig. 10(c), stations collided in slot 2 wait until cluster 3. If c_j new collisions occur in cluster 2, the waiting subsets must be shifted by 3 * $c_j - p$ positions in the stack to make room for the new subsets. We note that clusters of collided minislots are resolved in a LIFO order where more recent clusters preempts older ones. However, within each cluster collided minislots use a FIFO ordering corresponding to their order of occurence in the cluster. The average

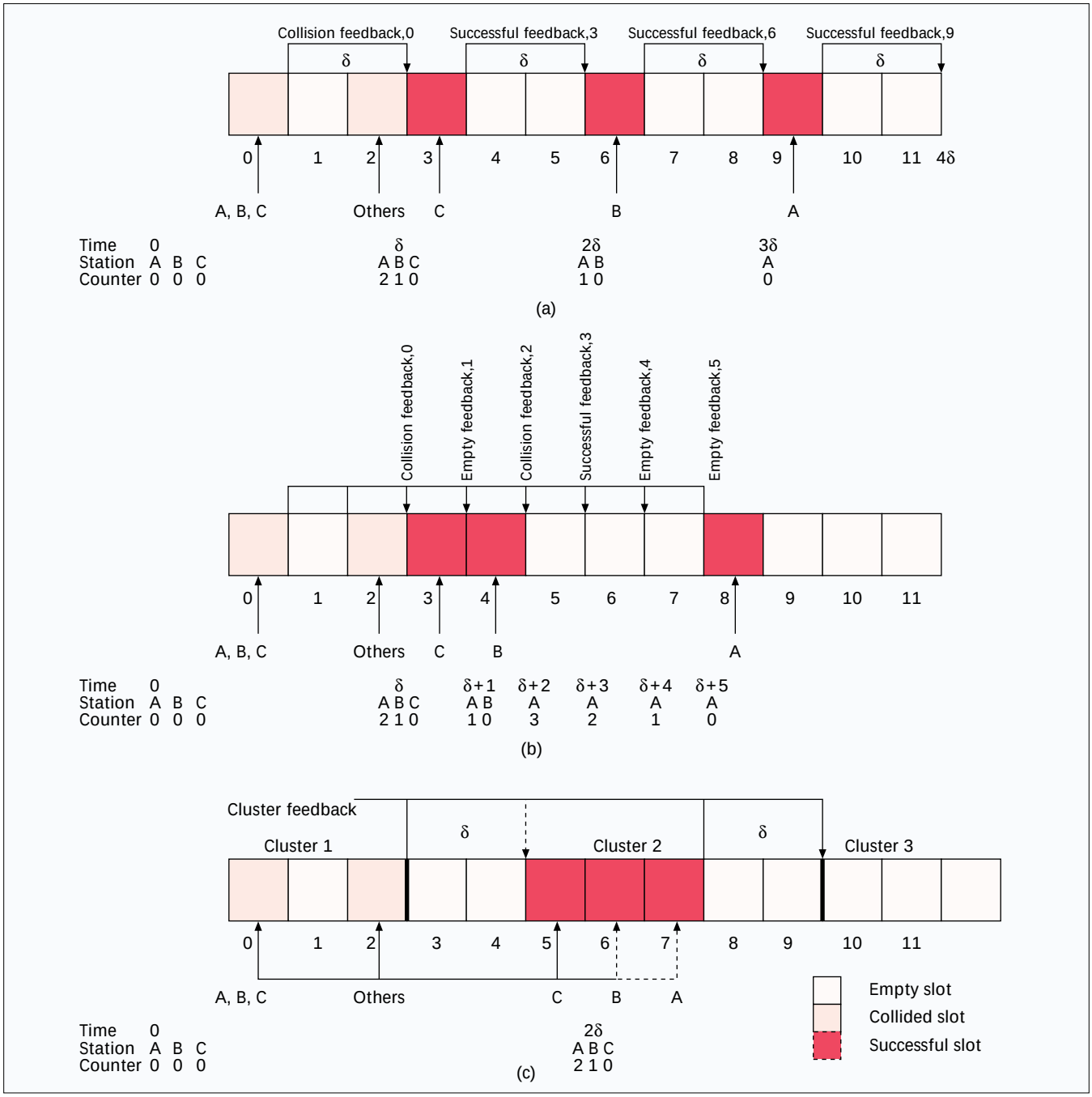

Figure 10. *Tree implementation: (a) sequential implementation; (b) parallel implementation; (c) cluster implementation.*

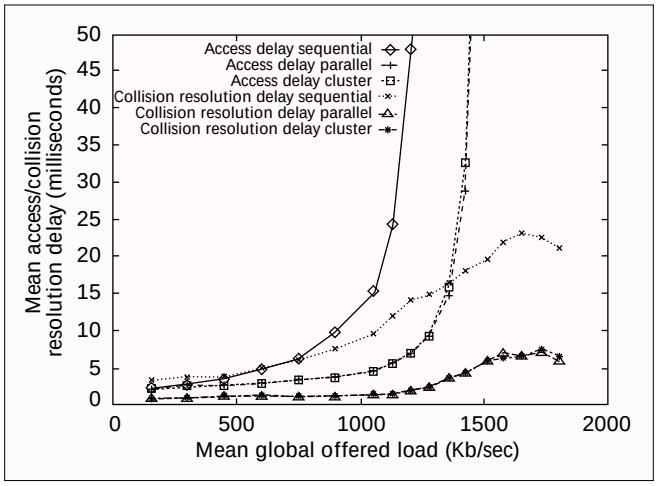

Figure 11. *Ternary-tree implementation: mean access/collision resolution delay vs. offered load.*

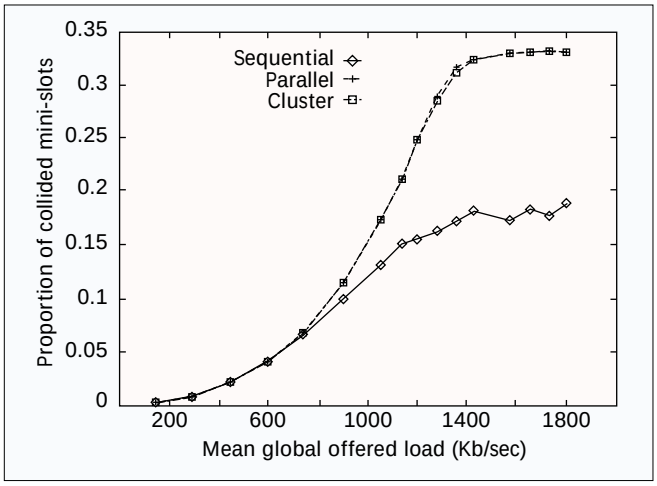

Figure 12. *Ternary-tree implementation: percentage of collisions vs. offered load.*

request delay for the cluster implementation, as shown in Fig. 10(c), is equal to

$$\frac{5+6+7}{3} = 6$$

which is also the result obtained for the sequential implementation. In Fig. 11 we plot the mean access delay and the mean collision resolution delay for the sequential, parallel, and cluster implementations discussed above. A Free Access policy is used for newcomer stations. We define the mean collision resolution delay as the average time required to resolve a collision. It is also the time elapsed from the reception of a collision feedback at the station until the reception of a successful feedback for a given request. First we note that the mean collision resolution delay curves for all three implementations at high loads (> 50 percent capacity) are subject to a slight dip down due to the accumulation of packets in the station's buffer and an increase in the request size. Thus stations are able to request for multiple packets. This keeps the number of collisions from increasing and we see it stabilizing in Fig. 12 when the channel completely saturates at 50 percent of the capacity (which is also the theoretical throughput limit for a fixed allocation of 12 CS and 6 DS). It seems rather counterintuitive that the mean collision resolution delay for the sequential implementation is slightly higher than the mean access delay at low loads (between 5 and 10 percent). This is due to adding the time to make a successful request in one round trip time to the mean collision resolution delay in order to obtain a mean request delay. Note that the mean access delay is the sum of the mean request delay plus the packet transmission time (depending on the bandwidth allocation). As expected, the mean access delay and the mean collision resolution delay for the sequential implementation are much higher at higher loads than the two other methods: the collision resolution delay is 15 ms higher at 50 percent and the mean access delay is 40 ms higher at 40 percent capacity (Fig. 11). However, the percentage of collisions (number of collided slots over the total number of minislots) incurred by the sequential implementation is less than the percentage of collisions for the cluster and parallel methods as seen in Fig. 12. In this case, reducing the number of collisions comes at the cost of increasing the delays. For both the parallel and cluster implementation the results are almost identical in terms of mean access delay, collision resolution delay, and number of collisions (Figs. 11 and 12).

IEEE 802.14 Solution

Having described in detail the various algorithms presented and compared their performance, the rationale for the IEEE 802.14 solution adopted becomes clear. As of the writing of this article, the IEEE 802.14 draft specifications [1] include T_{bound} Access for FTR, and a cluster implementation for the tree-based collision resolution algorithm.

Related Factors and Impact on Performance

So far we have compared various contention resolution features under the same network constraints in order to stress the differences between them. However, overall system performance depends on a number of design choices and specific network conditions. In this section we examine some of those factors and their impact on performance. The experiments are conducted using the tree-based cluster mode with T_{bound} Access for contention resolution.

Bandwidth Allocation

In this section we focus on the bandwidth allocation element of the MAC protocol and study its impact on performance. We look at three different strategies to improve bandwidth, namely, varying the CS/DS ratio over the request/feedback cycle, piggybacking, and increasing the request size.

Variable Slot Allocation — We compare the fixed slot allocation we have been using so far in the experiments (2:1 CS/DS ratio) to a variable slot allocation strategy that we shall define shortly. In the fixed scheme, the number of contention minislots and reservation data slots in each cycle remains the same. In the variable scheme, the headend varies the ratio of contention slots (CSs) to reservation data slots (DSs) from cycle to cycle, based on the traffic in the system. This mechanism is similar to what is presented in [15-17] with some slight modifications. The number of CSs, N_{CS}, contained in each upstream cluster is dynamically adjusted as the headend converts the number of DSs into CSs, N_{DS}, according to the following expression:

$$N_{DS} = \left\lceil \frac{2 * MAX\_DATA}{(2 + m * k)} \right\rceil \quad (3)$$

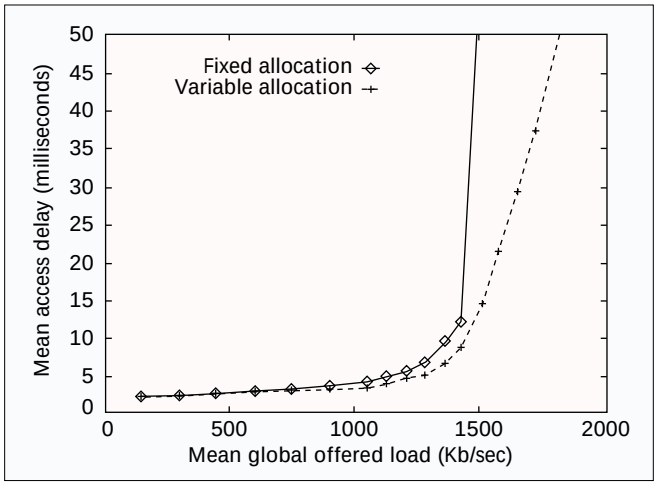

Figure 13. *Variable allocation: mean access delay vs. offered load.*

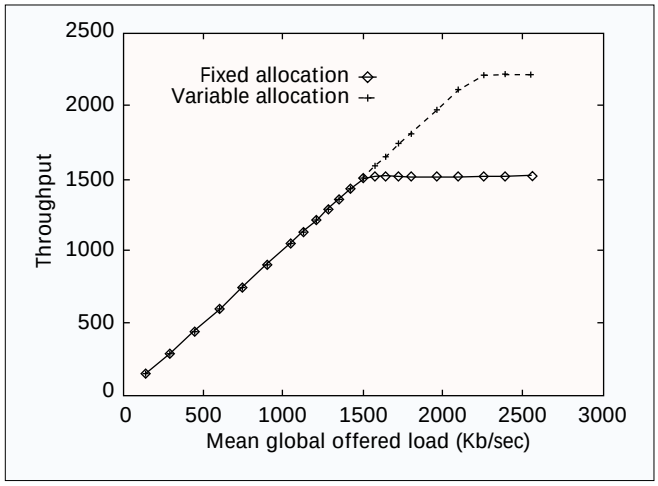

Figure 14. *Variable allocation: throughput vs. offered load.*

where *MAX_DATA* is the maximum number of data slots in a frame, *m* is the number of minislots that a data slot occupies, and *k* is the average number of DSs that can be requested at a time. As a result, N_{CS} can be determined by:

$$N_{CS} = \begin{cases} 0 & \text{if } DQ \geq \alpha * (MAX\_DATA - N_{DS}) \\ m * N_{DS} & \text{else} \end{cases} \quad (4)$$

where *DQ* is the total number of data slots requested but not yet allocated by the headend, and α is a design parameter set to 2.5. In addition, any unused data slots resulting from the above allocation are then converted into contention slots as suggested in [14]. Figures 13 and 14 compare mean access delay and throughput, respectively, for the fixed and variable allocation. The results obtained confirm the analysis given in [16, 17] where it is shown that variable allocation performs better than the fixed ratio policy in terms of lower delays at higher loads and higher throughput. At higher loads, keeping the number of contention slots fixed constitutes a significant bandwidth overhead. Stations may take advantage of requesting for more than one packet at a time, thus reducing the need for contention slots. At low loads, the frequency and location of contention slots could reduce the delays as shown in [18].

Piggybacking — Piggybacking consists of sending requests for additional data transmission along with a data packet without having to go through contention. Note that the first request for data transmission is sent in contention. In Fig. 15 we compare the mean access delay with and without the piggybacking feature. For both cases we use the variable allocation algorithm described above. We note 2–3 ms lower delays when using piggybacking starting from 50 percent capacity.

Request Size — As the offered load is increased, short packets tend to accumulate in the station's buffer. This is true even at lower loads depending on the traffic type. Large packets for some applications may generate several ATM cells at once when segmented into 53-byte chunks. Thus in order to avoid sending several requests in contention and to reduce the access delay, the station can send a request for all the packets in its buffer at once. For the IEEE 802.14 MAC the request size is limited to 32 packets (request field is 8-bit long). In Fig. 16 we compare the mean access delay for different request limits of 1, 2, and 32 with the variable allocation scheme. Since short packets are used in the experiment, the difference in the three curves become significant only at high loads, starting at 45 percent of capacity. For a request size of 1, delays are 3 seconds higher than for request size of 2 at 75 percent capacity. Delays for the request size of 32 are still in the order of milliseconds (~ 100 ms) at the same load.

Number of Stations

An interesting issue for cable operators and subscribers is the number of stations that can share the upstream channel.

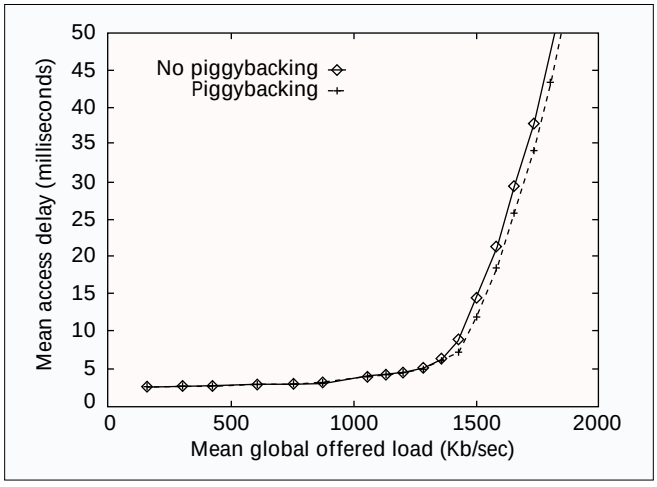

Figure 15. *Piggybacking: mean access delay vs. offered load.*

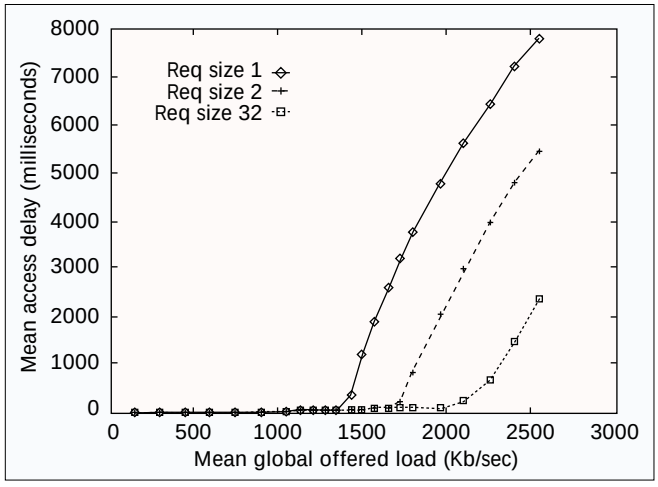

Figure 16. *Request size: mean access delay vs. offered load.*

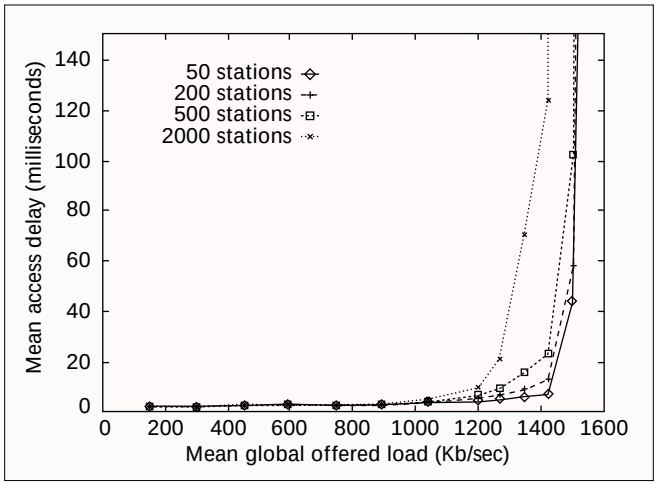

Figure 17. *Varying number of stations: mean access delay vs. offered load.*

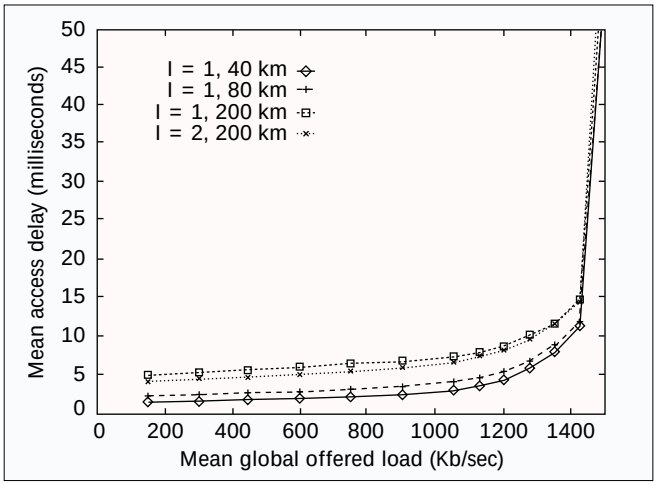

Figure 18. *Varying feedback delays: mean access delay vs. offered load.*

Should there be a limit on how many stations share the same channel? It is definitely a design issue, a trade-off between cost and performance. Adding more stations to the HFC network not only increases the overall load but also the number of contenders. We saw in the previous experiments how to better utilize resources by using a variable allocation scheme, a larger request size, and piggybacking when the number of stations is constant. Although these strategies are scalable to various numbers of stations, there is still a price to pay for increasing the number of subscribers. It may be more economical to the cable operator, but it increases the access delay and lowers the throughput. In Fig. 17 the mean access delay at 47 percent of capacity is close to 7 ms for 50 stations, while it is 11, 23, and 123 ms for 200, 500, and 2000 stations, respectively.

Feedback Delays

Another design parameter of HFC networks is the total length of the cable or the maximum distance between the headend and the furthest station in the network. Increasing this distance will require additional amplification of the signal. In addition it will increase the round trip delay time for the entire distribution network since the feedback delay is set to the round trip delay between the headend and the furthest station. Note that this requirement is necessary in order to insure fairness of access among all stations. For example, in case of a collision feedback, all stations have an equal opportunity for retransmission. It is interesting to note that regardless of the distance used, the maximum throughput is the same. In Fig. 18 all delay curves converge around 50 percent capacity (maximum throughput with fixed allocation). Between 5 and 50 percent capacity, the differences between the delay curves for 40 and 80 km, and 80 and 200 km, are constant and equal to 1 and 2 ms, respectively.

As previously discussed in the ternary-tree cluster implementation (see subsection "Stack Management"), the length of the cluster is usually equal or greater than the round trip delay plus the transmission delay of the contention slots in the cluster. Thus, if a station used a contention slot in cluster *i*, it gets the feedback before cluster $i + 1$ in case it needs to retransmit its request. Another way to fully utilize the channel and get feedback to the stations at the beginning of each cluster is to split the cycle time into smaller clusters (up to 16) and thus have several collision resolution engines operate in parallel on each cluster. This method is known as interleaving. A station with a request to send can randomly select an interleave. Collisions are resolved for each interleave separately. In Fig. 18 we use an interleave of 2 for a distance of 200 km. The mean access delay with 2 interleaves for 200 km is slightly less than with one interleave ($\sim$ 1 ms).

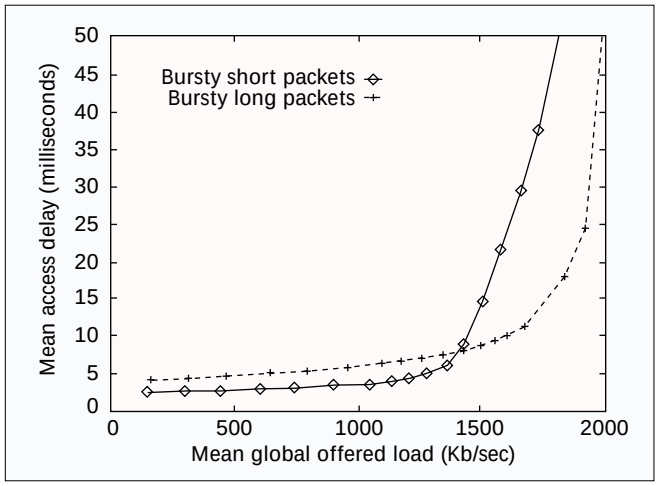

Figure 19. *Traffic types: mean access delay vs. offered load.*

Traffic Types

In this case we consider the traffic type and its impact on performance. The source generating long packets in the experiment results of Fig. 19 is a bursty source with a batch Poisson arrival model. The message size distribution is equal to 64, 128, 256, 512, 1024, and 1518 bytes with a probability of 0.6, 0.06, 0.04, 0.02, 0.25, and 0.03, respectively. The message interarrival time is exponentially distributed with mean $T = 1/\lambda$ where λ varies according to the load. The source generating short packets is the one used in all the other experiments so far (48-byte packets with an arrival rate of λ). For loads less than 40 percent of capacity, delays for longer packet bursts are higher (2 ms) than for shorter packets. This is mainly due to the larger transmission delay incurred by long packets (Fig. 19). At higher loads, shorter packets cause more requests sent in contention resulting in more collisions and less throughput. With large packet sizes, the delays are lower for higher loads (knee of the curve is pushed further to the right). The large average packet size results in larger request sizes so that less contention minislots are needed and more data slots can be allocated.

CONCLUSION

This article gives an overview of contention resolution design considerations for HFC networks and paves the way to further enhancements in protocol development. We study important elements of CRAs such as first transmission rule and stack management. We show that mean access delays for Blocked Access with a FIFO ordering of newcomers are lower than for Free Access. In addition, Blocked Access significantly reduces the delay variation while maintaining a low collision multiplicity. Finally, we note that the p-persistence scheme exhibits higher delays and a larger probability distribution tail than the tree mechanism.

REFERENCES

[1] IEEE 802.14 Working Group, Media Access Control, IEEE Draft Std. 802.14, Draft 3 R3, October 1998.

[2] N. Abramson, "Development of the ALOHANET," *IEEE Trans. on Info. Theory*, vol. IT-31, no. 2, March 1985.

[3] D. Bertsekas and R. Gallager, *Data Networks,* 2nd Ed., Prentice Hall, 1992.

[4] J. Capetanakis, "Tree Algorithms for Packet Broadcast Channels," *IEEE Trans. on Info. Theory*, vol. IT-25, no. 5, Sept. 1979.

[5] C. Bisdikian, "Performance Analysis of the Multi-slot n-ary Stack Random Access Algorithm (msSTART)," contr. no. IEEE802.14-96/117, IEEE 802.14 Working Group (WG) meeting, May 1996.

[6] C. Bisdikian *et al.*, "MLAP: A MAC level access protocol for HFC 802.14 network," *IEEE Commun. Mag.*, vol. 34, no. 3, pp. 114–121, March 1996.

[7] C. Bisdikian, "A review of random access algorithms," *Proc. Int'l Workshop on Mobile Commun.*, pp. 123-127, Thessaloniki, Greece, Sept. 1996.

[8] C. Bisdikian, B. McNeil, and R. Norman, "msSTART: A random access algorithm for the IEEE 802.14 HFC network," *Computer Commun.*, vol. 19, no. 11, pp. 876–887, Sept. 1996.

[9] C. Bisdikian,"Enhancing the Collision Resolution algorithm: Comments to d2R2," contr. no. IEEE802.14- 97/121, IEEE 802.14 Working Group (WG) meeting, Nov. 1997.

[10] R. Citta, C. C. Lee, and D. Lin, "Phase 2 Simulation Results for Adaptive Random Access Protocol," contr. no. IEEE802.14-96/114, IEEE 802.14 Working Group (WG) meeting, May 1996.

[11] R. Citta, J. Xia and D. Lin, "The tree-based Algorithm with Soft Blocking," contr. no. IEEE802.14-96/244, IEEE 802.14 Working Group (WG) meeting, Nov. 1996.

[12] N. Golmie and D. Su, "First Transmission Rule: Unblocking Techniques," contr. no. IEEE802.14-96/245, IEEE 802.14 Working Group (WG) meeting, Nov. 1996.

[13] P. Jacquet, P. M"uhlethaler, P. Robert, "Performant Implementations of Tree Collision Resolution on CATV Network," contr. no. IEEE802.14-96/, IEEE 802.14 Working Group (WG) meeting, April 1996.

[14] D. Sala, J. Limb and S. Khaunte, "Adaptive Control Mechanism for Cable Modems MAC Protocols," *Proc. INFOCOM '98*, San Francisco, March 1998.

[15] K. Sriram, and P. Magill, "Enhanced Throughput Efficiency by Use of Dynamically Variable Request Minis- lots in MAC Protocols for HFC and Wireless Access Networks," *Telecommun. Systems: Special Issue on Multimedia,* vol. 9, nos. 3 & 4, 1998.

[16] K. Sriram, "Performance of MAC protocols for Broadband HFC and Wireless Networks," *Advances in Performance Analysis*, vol. 1, no. 1, pp. 1–37, March 1998.

[17] B. Doshi, S. Dravida, P. Magill, C. Siller, and K. Sriram, "A Broadband Multiple Access Protocol for STM, ATM, and Variable Length Data Services on Hybrid Fiver-Coax Networks," *Bell Labs Technical J. (BLTJ)*, vol. 1, no. 1, pp. 36–65, Summer 1996.

[18] N. Golmie, S. Masson, G. Pieris, and D. Su "A MAC protocols for HFC networks: Design issues and performance evaluation," *Computer Commun.*, vol. 20, 1997, pp. 1042–1050.

BIOGRAPHIES

NADA GOLMIE (nada.golmie@nist.gov) received an M.S. in electrical engineering and computer engineering from Syracuse University, New York, in 1993, and a B.S. in computer engineering from the University of Toledo, Ohio, in 1992. Since 1993, she has been a research engineer in the high-speed networks technologies group at the National Institute of Standards and Technology (NIST). Her interest includes performance evaluation of ATM network protocols, media access control, and HFC network technologies.

YVES SAINTILLAN received an Engineering Diploma in computer science from the National Institute of Applied Sciences (INSA), Lyon, France, in 1995, and an M.S. in computer science from Concordia University, Canada, in 1998. Since 1997 he has been a guest researcher at the at the National Institute of Standards and Technology (NIST). His interests include routing and admission control, congestion control, and performance evaluation of medium access control protocols.

DAVID H. SU received a Ph.D. in computer science from Ohio State University in 1974. He is the manager of the High-Speed Networks Technologies Group at the National Institute of Standards and Technology (NIST). His main research interests are in modeling, testing, and performance measurement of communications protocols.