The MXL MAC Protocol for HFC Networks

Ran-Fun Chiu
HP Labs
Palo-Alto, CA

Reuven Cohen
HP Israel Science Center
Technion 32000, Israel

Michael Chen and Bob Hutchinson
HP Video Communications Division
Cupertino, CA

Keywords: ALOHA, Cable-Modem, MAC, MXL, tree algorithm

Abstract

The paper describes the upstream MAC protocol employed by the HP cable-modem for hybrid-fiber-coax networks. The protocol, called MXL, divides the upstream channel into two logical channels: a contention channel and a reserved channel. Cable-modem stations that need to send data have to reserve sufficient bandwidth in the reserved channel. In order to send their reservation request, the stations have to compete for access on the contention channel. The paper discusses possible contention algorithms and compares their performance results.

1 Introduction

It is widely recognized that hybrid-fiber-coax (HFC) networks will play an important role in providing economical service access for residential subscribers. The large excess bandwidth available on these networks can provide the communication infrastructure for home interactive services like video on demand, Web surfing and video game playing.

In a typical HFC network, cable-modems are connected to a common head-end by a tree-and-branch transmission network. The network is divided into a fiber domain which extends from the head-end to a neighborhood, and a coax domain which connects the homes to the fiber. Fiber nodes at the edges of the fiber and coax domains transform the optical signal to an electrical signal. The maximum CATV length is approximately 50 miles. The homes are located only at the last 20 percents of this distance.

Assuming a total bandwidth of at least 750 MHz on the coaxial cable links, a possible spectrum allocation is as follows. A 400 MHz band from 50 MHz to 450 MHz is used to carry conventional downstream (from the head-end to the homes) analog TV channels. A 40 MHz band between 5 and 45 MHz can be divided into multiple upstream (from the homes to the CATV head-end) digital channels of 1-2 MHz. A larger number of 6 MHz downstream digital channels can be included in the 300 MHz band between 450 and 750 MHz. Using appropriate modulation techniques, each of the upstream channels is capable of carrying 2-10 Mb/s, whereas each of the downstream channels is capable of carrying 30 Mb/s.

The upstream channels are shared by the cable-modem stations using a multiple access control (MAC) protocol. Unlike in traditional local area networks, the stations cannot directly listen to the upstream transmissions of other stations, and cannot detect collisions. This is due to the CATV tree-and-branch physical architecture. On the other hand, the head-end station can play a major role in coordinating the upstream transmissions stations.

This paper describes the upstream MAC (Medium Access Control) protocol employed by Hewlett-Packard MXL (multimedia transmission link) cable-modems. MXL divides the upstream channel into two logical channels: a contention channel and a reserved channel. Stations that need to send data have to reserve sufficient bandwidth in the reserved channel. In order to send their reservation request, the stations have to compete for access on the contention channel, according to the rules dictated by the MAC protocol. The paper discusses possible contention algorithms and compares their performance results.

The rest of the paper is organized as follows. Section 2 describes the main design concepts of MXL. In Section 3, the implementation of the ALOHA algorithm in the contention channel is discussed and studied. ALOHA is a simple algorithm that yields a maximum throughput of 0.37 on the contention sub-channel, and a total throughput of 0.8. In Section 4, the tree algorithm with some of its variations is presented. Unlike ALOHA, the tree algorithm serves the contending stations in a first come first served manner, and yields a smaller delay standard deviation. Section 5 concludes the paper.

2 The MXL Topology and Upstream Access Control

The HP cable-modem protocol, called MXL (multimedia transmission link), associates a downstream 6 MHz channel with an upstream 2 MHz channel. Using a 6 bits/symbol 64QAM modulation scheme, which carries one bit for control and 5 bits for payload from

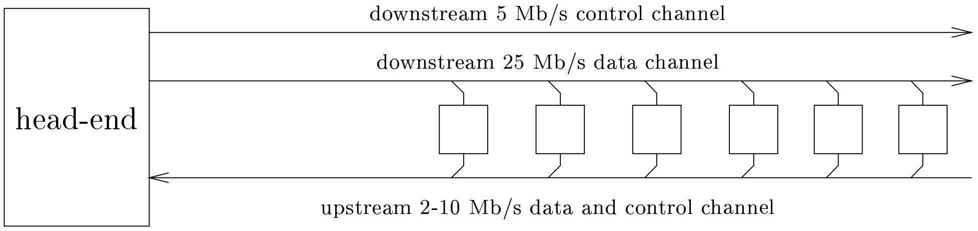

downstream 5 Mb/s control channel
downstream 25 Mb/s data channel
head-end
upstream 2-10 Mb/s data and control channel

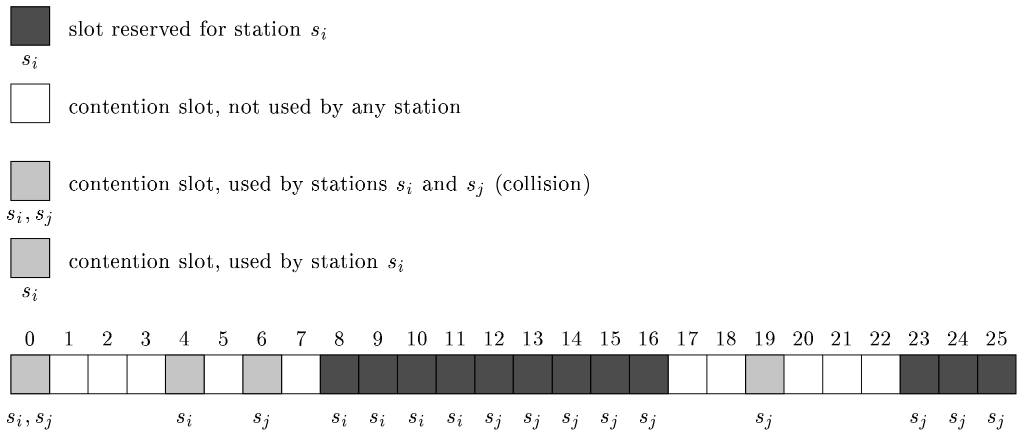
Figure 1: MXL Network Topology

each symbol, the 6 MHz downstream channel is divided into a 25 Mb/s downstream data channel and a 5 Mb/s downstream control channel pair. The 2 MHz upstream channel is shared by all the cable-modems (stations) for delivering data and control packets to the head-end in a rate of 2-10 Mb/s (Figure 1), depending on the upstream channel modulation scheme.

In MXL, a slot is a time slice from the upstream channel bandwidth. Its length is represented by a fixed number of downstream control channel mini-slots. This number is programmable by the MXL head-end. The length of the slot includes the time to transmit a control packet upstream, and the round-trip propagation delay between the most distanced stations. The latter distance must be smaller than the length of a control upstream packet. The stations are synchronized such that each transmission arrives at the head-end within the timing bounds of a time slot. The timing of the upstream slots is derived by the stations from counter timing data transmitted by the head-end in each mini-slot on the downstream control channel. In addition to providing these synchronization markers, the head-end transmits on the downstream control channel a short status packet (which uses a single mini-slot) every upstream slot time. The purpose of this short packet is to inform the stations of the status of the next upstream slot. An upstream slot can be either reserved or available. A reserved slot can be used by a single station, to which it has been previously assigned by the head-end. An available (contention) slot is open for contention according to the rules dictated by the contention algorithm.

A short upstream data packet, that fits into a slot, is transmitted during a contention time slot with no preceding reservation. For larger packets, an explicit preceding reservation is required. To this end, the station waits for a contention slot on the upstream channel and transmits a short reservation control packet which specifies its identity and the number of slots needed for successive transmission of all its waiting messages. The MXL head-end always returns the acknowledgment for an upstream reservation packet in the next slot on the downstream control channel. The period of time elapsed between transmitting a successful reservation and receiving an acknowledgment is therefore fixed, and will be referred to as an "ACK-window". If a station does not get an acknowledgment during the ACK-window, it assumes a collision has occurred. It then waits some period of time, determined according the contention algorithm, and re-contends using another contention slot.

As already indicated, if the first data packet waiting for upstream transmission is shorter than a slot, it is transmitted in a contention slot with no preceding reservation. Though it contains no reservation, such a packet needs to be acknowledged by the head-end during the ACK-window. In contrast, packets transmitted in the reserved channel do not have to be



s_i
slot reserved for station s_i
contention slot, not used by any station
s_i, s_j
contention slot, used by stations s_i and s_j (collision)
s_i
contention slot, used by station s_i
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
s_i, s_j s_i s_j s_i s_i s_i s_i s_j s_j s_j s_j s_j s_j s_j s_j s_j

Figure 2: MXL Execution Example

acknowledged by the head-end since they cannot collide with other packets. Such packets can still be lost due to transmission errors, but recovering from such losses is the role of upper layer protocols (like TCP). If a station has more packets queued after a short packet, the upstream contention slot into which the short packet is transmitted will contain a reservation for the remaining packets (piggy-backing).

During the sign-on process, each station empirically determines the number of slots in the ACK-window in the following way. Upon being powered-on, the station waits for an upstream contention slot and transmits a sign-on request control packet to the head-end. It then waits for an acknowledgment from the head-end. If an acknowledgment is not received within one millisecond (an upper bound on the ACK-window) the station waits for another contention slot and retransmits its sign-on request. If an acknowledgment is received, the station considers the period of time between the transmission of the last request and the reception of the acknowledgment as its ACK-window. Note that due to the way the upstream channel is slotted, all the stations have the same ACK-window regardless of their distance from the head-end.

An execution example of the MXL protocol is given in Figure 2, assuming that the ACK-window is of 3 slots. This means that a station that transmits its reservation request successfully on slot i of the upstream channel will get an acknowledgment when slot number $i + 3$ is scheduled on the upstream channel. In the depicted example there are two active stations: s_i and s_j. Both stations transmit during slot 0. Thus, a collision occurs and neither of them get an acknowledgment during slot 3. According to some contention algorithm, to be discussed later, station s_i tries again, in slot 4 say, and succeeds. Thus, it gets an acknowledgment from the head-end on the downstream control channel when slot 7 is scheduled on the upstream channel. The acknowledgment packet has a field called *Reservation Delay Slot Count,* which tells the contending station the number of upstream slots after which the station can start transmitting its allocated quota. In this particular case the head-end has no pending reservation when the request from s_i is received. Thus, assuming that s_i requested 4 slots, the head-end grants s_i 4 slots starting immediately after s_i receives the acknowledgment. Hence the value of the *Reservation Delay Slot Count* field is 0. And indeed, slots 8-11 are used by s_i with no interference. Station s_j transmits its reservation for 5 slots during upstream slot 6. Since there is no collision, s_j receives an acknowledgment during time slot 9. This acknowledgment informs s_j that it is granted the requested number of slots, but the *Reservation Delay Slot Count* field is 2 because the next couple of slots are in the middle of the burst reserved for s_i. When station s_j finishes transmitting its 5-slot packet(s) the channel is again available for contention. Unlike other schemes (e.g. [6, 13]), MXL does not allow a transmitting station to use the reserved slots in order to send another request. New requests can be sent using contention slots only. This guarantees fairness among stations without complicating the algorithms performed by the head-end or the algorithm performed by the stations.

The MXL actually divides the upstream channel into two logical channels: a contention channel which consists of all the available (contention) upstream slots, and a reserved channel which consists of all the reserved slots. The total throughput S of the upstream channel can be expressed as

$$\frac{\sigma}{\sigma + 1/S'} < S < \frac{\sigma + 1}{\sigma + 1/S'} \tag{1}$$

where σ is the average number of slots in each reservation request and S' is the throughput of the contention channel. The left-hand part of Eq. 1 represent the case where a reservation slot does not contain a small data packet, whereas the right-hand part represents the case where every reservation slot also contains a small data packet.

Most of the architectures for HFC networks use the concept of separating the upstream channel into a contention logical channel and a reserved logical channel, and employing a contention resolution algorithm in the contention channel. However, the MXL is distinguishable from other architectures by its following properties:

- A packet is transmitted as one unit, using a burst of reserved slots. Thus, fragmentation of packets at the stations and re-assembly at the head-end are avoided.
- Another side-effect of the previous property is that the head-end allocation mechanism is simple. The head-end needs to maintain only a local counter whose value indicates the slot distance to the next available slot. When a request for ρ slots is accepted, the value of the pointer is returned to the requesting station, and the counter is incremented by ρ. This simple allocation method reduces the head-end processing time, and therefore the ACK-window duration, to a minimum. Consequently, stations are informed as early as possible of the results of their last contention, and the throughput of the channel increases.
- By preventing stations from sending new reservation requests in the reserved channel starvation is avoided, and fairness is achieved to some extent, without complicating the head-end allocation algorithm.

3 Running ALOHA in the Contention Channel

ALOHA is probably the simplest algorithm for controlling the access to the contention logical channel. According to this algorithm, upon entering the contention state a station awaits the first available slot and transmits its reservation. The station then waits for an acknowledge form the head-end. If an ACK is not received during the ACK-window, the station waits for some random number of contention slots before retransmitting its reservation. With Poisson arrival and an infinite set of stations, the expected number of successful transmissions in the reservation logical channel is known to be

$$P_s \approx G(n)e^{-G(n)}$$

where $G(n)$ is the expected number of attempted transmissions in a slot when n stations are in the contending state. The maximum of the function $G(n)e^{-G(n)}$, occurs at $G(n) = 1$, is $1/e = 0.367$. This is known as the maximum throughput of slotted ALOHA. Achieving this maximum is not straightforward, however, because n is unknown to the contending stations. Several schemes have been proposed in order to let the contending stations estimate n and determine when to attempt transmissions. Rivest's pseudo-Bayesian algorithm [1] is a relatively simple and efficient one. The problem is that this scheme requires that every contending station will get a feedback for every slot in the contention logical channel: namely whether this slot was idle, success or a collision. MXL provides a much limited feedback: only a station that transmits during a contention slot knows whether this slot contains a collision or a successful transmission.

A scheme that can be easily implemented with such a limited feedback is the Ethernet's *truncated binary exponential back-off (BEB) algorithm.* According to this scheme, if a reservation has been transmitted unsuccessfully i times, then the station will wait $j-1$ contention slots and re-transmits its reservation in the j'th slot, where j is uniformly distributed in $[1 \cdots 2^{\max(i,M)}]$. M is chosen such that $2^{(M-1)}$ is an upper bound on the number of stations connected to each upstream channel. Figure 3 shows the performance of the upstream channel when ALOHA is used in conjunction with the BEB as an access control algorithm for the reservation logical channel. The figure shows the delay and average delay variance versus input load for 4, 32 and 128 stations. In the graph, as well as in those presented later, it is assumed that a slot is 64-byte long, and that 30.4% of the packets are 2-slot long, 8.3% of the packets are 3-slot long, 8% of the packets are 4-slot long, 10% of the packets are 10-slot long, 25% of the packets are 18-slot long, and 18.3% of the packets are 24-slot long. The average packet length is therefore 11 slots, and the *maximum* throughput for a large number of stations becomes, according to Eq. 1 $S = 11/(11 + e) = 0.801$ (note that since no packet is of one slot, the right-hand part of Eq. 1 does not apply). This packet length distribution, determined based on [7], does not necessarily reflect the precise traffic of the future home users.

Figure 4 shows slot statistics for the 128 stations run. Figure 4(a) shows the percentage R of reserved slots, the percentage E of empty slots, the percentage S of successful slots, and the percentage C of collision slots. As the load increases, the bandwidth of the contention logical channel $(E + S + C)$ decreases whereas the bandwidth of the reserved logical channel (R) increases. Figure 4(b) shows the function $S/(E + S + C)$, which indicates the throughput of the contention logical channel. The maximum throughput, 0.30, is smaller than the maximum theoretic throughput of $1/e = 0.367$, because BEB does not guarantee $G(n) = 1$. This maximum is achieved for input load of 0.76, which is the maximum throughput of the protocol for 128 stations and the considered packet length distribution.

The rationale behind the BEB algorithm is that each contending station makes a binary "search" for the number n of contending stations, and determines its access time accordingly. In [10], two drawbacks of this algorithm are indicated. Firstly, it is shown that BEB is rather slow at adapting to transient overload condition, because it acts globally like a linear search.

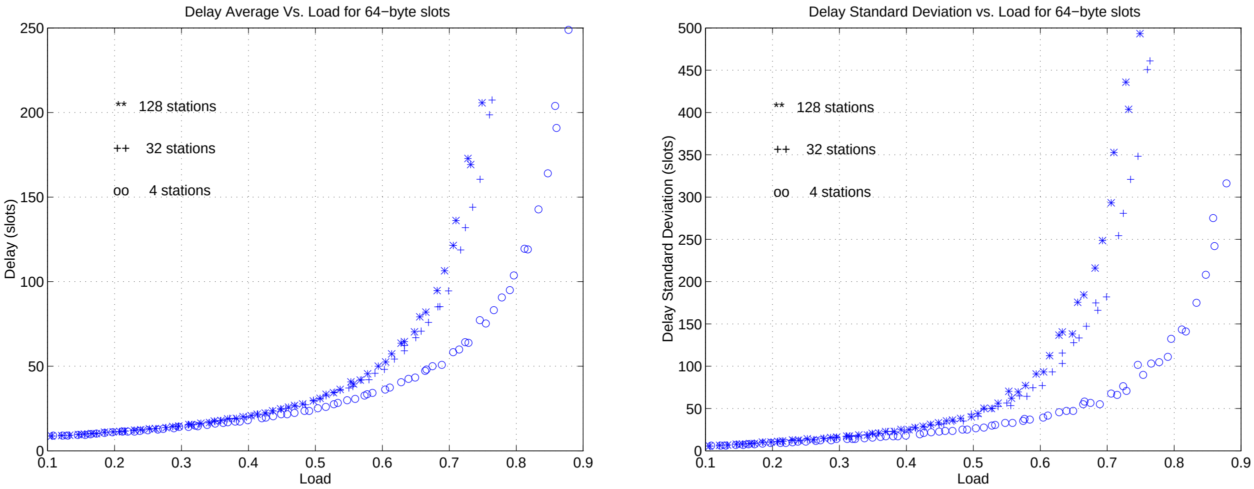


Figure 3: The Performance of MXL With the Binary Exponential Back-off Algorithm

If m stations are involved in a collision, the value L_m representing the average number of back-off slots until the first successful transmission growths linearly with m ($L_m \approx 2/9 \cdot m$). Secondly, BEB suffers from the so-called *capture effect,* which increases the unfairness and delay variance. To understand the capture effect consider two stations s_i and s_j, contending in the contention channel. Suppose that s_i is the first to succeed. If immediately after sending its reservation request s_i has another reservation to make, it has a better chance to acquire the contention channel in the second time before s_j makes it for the first time. This happens despite of the fact that MXL does not allow to transmit a reservation in a reserved slot, but only in a contention slot. The reason is that after its first successful transmission, s_i resets its collision counter to 0, thus significantly decreasing the expected time interval before its next contention. This suggests that BEB enforces last come first served rather than first come first served access policy, which significantly increases the access delay variance.

The performance of current TCP implementations, like Tahoe or Reno, is rarely affected by the MAC layer access delay variance. The reason is that in these implementations the estimates of the round trip time and variance are computed using a coarse (500ms) grained timer. Consequently, the computed round trip time and variance are much larger than the actual values. Newer TCP proposals, like Vegas [3], that use a timer with a finer grain in order to increase the TCP throughput by having a tighter estimate of the round trip delay, will be more sensitive to the MAC layer delay variance.

A possible way to reduce the delay variance of the truncated binary exponential back-off is to let the stations adjust the value M dynamically, according to the exact number of signed-on stations. The idea is that instead of using a fixed value of $M = \lceil log_2 N \rceil + 1$, where N is an upper bound on the number of signed-on stations, the head-end will occasionally broadcast

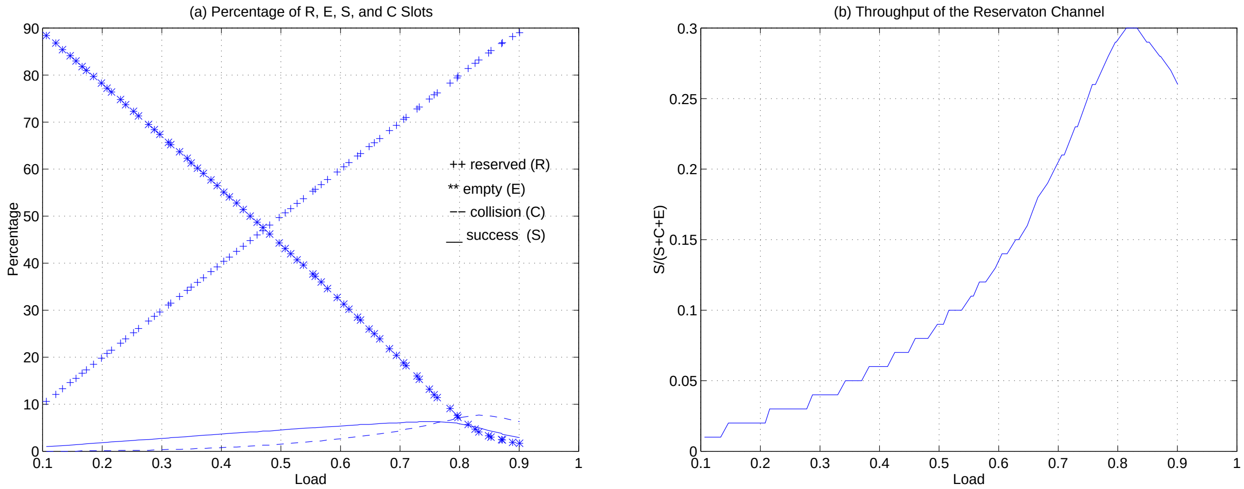


Figure 4: Slot Distribution for 128 Stations

on the control channel a short packet indicating the actual number of signed-on stations. All the stations will record the announced number and use it instead of N in order to determine the value of M. The effect of this improvement is shown in Figure 5 for 4 and 32 stations. The curves labeled 'static M' represent the cases where M is chosen in advance to be 8, assuming the maximum number of signed-on stations is restricted to $N = 128$. The curves labeled 'dynamic' represent the cases where 4 and 32 stations are signed-on, and M is chosen to be $log_2(4)+1 = 3$ and $log_2(32)+1 = 6$ respectively, rather than 8. This modification has a little effect on the average delay. However, as Figure 5(a) and (b) indicate, when the channel load is moderate or heavy the standard deviation and the maximum delay are significantly reduced.

4 The Tree Algorithm

If decreasing the delay variance is of importance, an access control scheme that serves the stations in a first come first served manner should be employed. Such a scheme is the tree algorithm, originally suggested by Capetanakis in 1977 [4]. The tree algorithm is based on the observation that a contention among several active stations is resolved only if the contending stations are somehow subdivided into groups such that each group contains at most one active station. In a channel with 0 propagation delay, the simplest form of the tree algorithm is as follows. When a collision occurs in some slot, all the involved stations split into two subsets. The first subset transmits in the next slot whereas the second subset transmits only after the first subset is finished. All the stations not involved in the collision must wait until both sets are finished. If a subset contains more than one node, another collision occurs. Then, this set splits again into two subsets and the algorithm continues

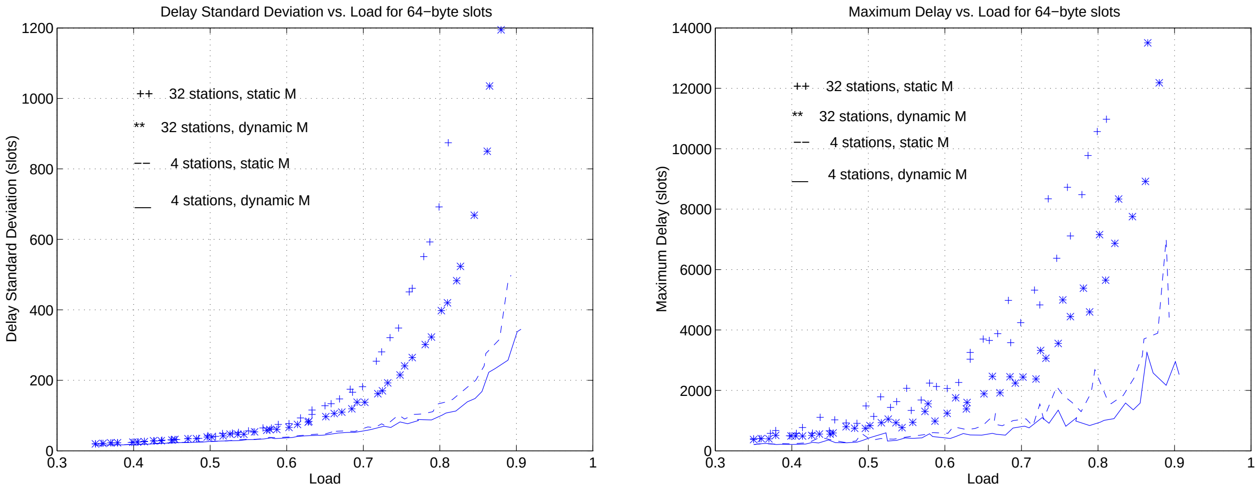


Figure 5: Dynamic vs. Static M

recursively. This procedure can be modeled as a binary tree. Each leave of the tree represents a successful transmission or an empty slot or a collision. When a collision occurs in slot i, the involved stations split into two subsets, represented by two children nodes of node i. More details about this algorithm and its many variations can be found in [1, 11].

In the considered cable-modem network, one must address several issues in order to implement the tree algorithm. First, the non zero propagation delay between the time a set of stations transmits into a slot and the time a feedback from the head-end is received. During this time interval, of Δ slots say (the ACK-window), stations competing for access to the tree must hold. A solution to this problem, suggested in the past in the context of satellite channels, is to view the contention channel as Δ independent sub-channels and to run a separate copy of the tree algorithm in each one. Consequently, a slot belonging to a given tree is defined only after the feedback from the previous slot of the same tree is known.

Another issue that must be addressed is the limited feedback provided by the channel to the stations, which is insufficient for the tree algorithm. Recall that in the tree algorithm after a tree node splits, the second subset is supposed to transmit its reservation only after the first subset is finished. Thus, every station involved in a collision must get a feedback from the channel not only for the slots during which it transmits its reservation request, but also for all the slots used by stations in the other subsets[1]. This problem can be solved using the status packets transmitted by the head-end in mini-slots of the downstream control channel once every upstream slot time. Recall that the purpose of these short packets is to inform

[1] If such a feedback is provided, each station belonging to the second subset following a collision need a single counter in order to keep track of its position in the tree. The counter is initialized to 1. It is incremented by 1 after each collision and decremented by 1 after each non-collision (empty or successful) slot. When it reaches 0, the station should contend again.

the stations whether the next upstream slot is reserved or available for contention, namely whether it belongs to the contention logical channel or to the reserved logical channel. In order to let the stations execute the tree algorithm, the head-end, that has all the needed information, will indicate to which tree and to which node in that tree every contention slot belongs. The tree description is needed because there are several trees running simultaneously in the contention channel. The slot description tells the stations contending on each tree which of them can transmit its reservation in the next slot assigned to the tree. Contending slots are assigned by the head-end to the trees in a round robin manner. The nodes of each tree are traversed according to a depth first search (DFS) order.

In the following the performance of several variations of the tree algorithm in the MXL is presented and discussed. These variations differentiate from each other in one or more of the following design options:

Access scheme: The access scheme specifies when a station with a reservation request can join its tree. There are three main options as follows (see [11] for more details). In the simplest variation, called *free access,* a station is allowed to join its tree in the middle of an epoc. In the second variation, called *gated access,* a station is allowed to join its tree only in the beginning of a new epoc, namely in the slot associated with the root of the tree. In the last variation, called *window access,* a stations is allowed to join its tree only in the beginning of a new epoc. However, unlike in gated access, only stations whose reservation requests arrive during a specific time period (the window) on the arrival time axis, are allowed to join the next epoc of the tree. The other stations must wait for the beginning of subsequent epocs. This scheme can be viewed as laying windows on the arrival time axis sequentially with the left boundary of the current window coinciding with the right boundary of the previous window. The purpose of the window access is to obtain the “correct” number of active stations, which is somewhat larger than 1, at the beginning of each epoc. It is an alternative to Capetanakis's “dynamic” tree algorithm [4].

Tree selection: In a system with multiple trees there are several options for determining to which tree each station should join. The first option, referred to as *fixed assignment,* is that upon signing-on each station is allocated by the head-end a fixed tree to which it joins whenever it has a new reservation to send. Under the second option, referred to as *dynamic assignment,* a station with a new reservation joins to a random tree. Under the third option, referred to as *free assignment,* each station joins the *first* available tree, where tree availability is determined according to the access scheme, as described above.

Level skipping: In the tree algorithm, when a collision is followed by an idle slot, the next slot must be a collision as well. To avoid wasting this slot, one can simply skip the next slot and proceed directly to the next level of the tree. This option is referred to as “level skipping”. It may lead to a deadlock if an empty slot is incorrectly perceived as a collision, in which case the algorithm continues splitting indefinitely without any success. However, in the considered network the number of signed-on stations is known to the head-end which determines how to split the tree. Thus, the head-end knows what the maximum depth of the tree should be, and when this maximum is reached the head-end will stop skipping levels.

Splitting scheme: When two or more stations transmit their reservation requests in the same node of a tree, a collision occurs and none of the involved station gets an acknowledgment from the head-end. In such a case the stations should be split between the right and the left sub-trees of that node according to the splitting scheme. If the window access scheme is employed, the stations must split according to the arrival time of their reservation requests [1, 11]. If free access or gated access are employed, the stations can use either a pseudo-random bit generator or their deterministic addresses. The later option is more efficient because it decreases the expected time needed to solve a collision. We shall compare between two kinds of addresses: the 48-bit MAC addresses the stations use for their regular communication and short 8-bit labels assigned by the head-end to every station during the sign-on process.

The maximum throughput of the 'standard' tree algorithm, with a single tree, gated access, no level-skipping and random splitting, for a channel with an infinite number of stations is 0.346. When level skipping is employed, the maximum throughput becomes 0.375. If window access is used instead of free access, the throughput increases to 0.43 without level-skipping and to 0.46 with level-skipping. However, in the following study window access is not considered from the following three reasons. First, window access requires the stations to get a feedback about *every* contending slot, rather than for only those slots during which they actually transmit. Second, window access increases the throughput only if the stations can predict the expected number of active stations during each time interval. This is the case under the traditional theoretic models (e.g. [1, 4, 11], where the system has an infinite set of stations, each new packet arrives at a new station, the size of the packets is equal to the slot size, and packets arrive at each station according to independent Poisson processes. None of these assumptions holds in a real network [9, 12]. Finally, increasing the throughput of the contention channel from 0.375 to 0.46 has a little effect on the overall throughput of the upstream channel. E.g. if the average frame length is 11 slots, then according to Eq. 1 the overall throughput of the upstream channel increases by from 0.8 to 0.835.

In the following discussion it is assumed that the round trip delay is $\Delta = 5$ slots. Thus, the head-end needs to maintain 5 contention resolution trees. Figure 6 shows the performance of two variations of the tree algorithm versus ALOHA for 128 stations. In both variations the access scheme is gated access and the tree selection is fixed. In one variation level skipping is used and the stations are assigned short labels for enhanced splitting. In the other variation, level skipping is not used and splitting is performed using the long MAC addresses. Figure 6(a) shows that for light loads (< 0.6) the average delay of ALOHA is smaller than of the tree algorithm. The reason is that for such loads the number of stations contending for access at the same time is small (usually less than 3). Thus, the tree algorithm has no advantage and the number of collisions is roughly the same. On the other hand, after each collision the stations performing the tree algorithm must wait more time than the stations performing the ALOHA algorithm. In the tree algorithm the average delay of a station before contending again after the first collision with another active station, assuming there are no more contending stations, is $\frac{3}{2}\Delta + \frac{\sigma}{4}$, where Δ is the number of trees and σ is the average length of a packet. This is because with probability 0.5 after the first collision the station falls into the first subset, waits Δ slots and tries again; with probability 0.25 both stations fall into the second subset and therefore each of them waits 2Δ slots before re-contending; and with probability 0.25 the considered station falls into the second subset whereas the other station falls into the first subset, in which case the considered station waits $\Delta + \sigma$ slots for the second station to contend again and to transmit its frame, and additional Δ slots awaiting its contention tree. In ALOHA, on the other hand, the average delay in this case is only $\Delta + 0.5$, because with probability 0.5 the station waits Δ slots, and with the same probability its waits $\Delta + 1$ slots.

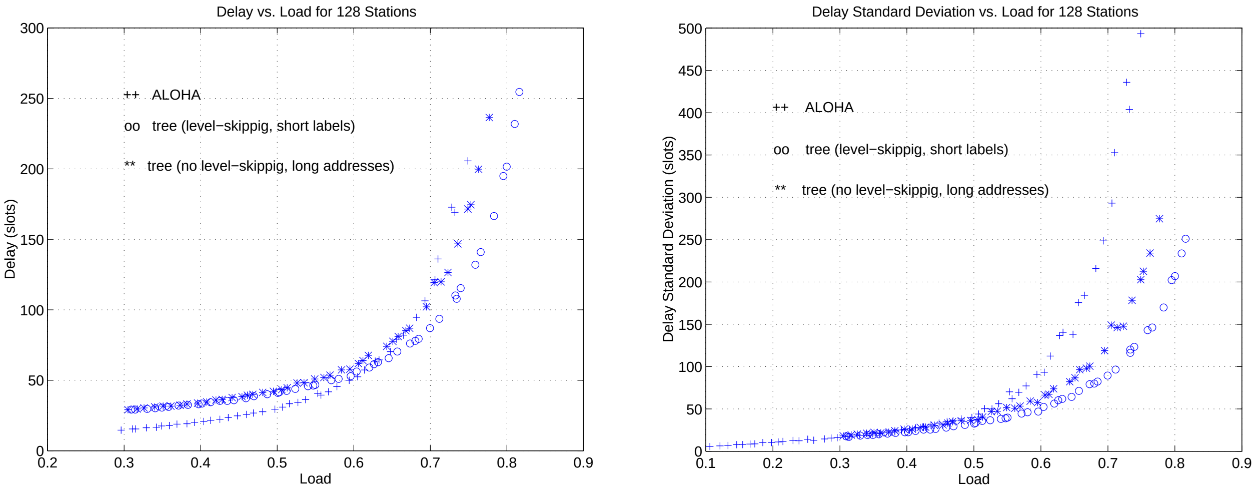


Figure 6: ALOHA vs. the Tree Algorithm

For heavier loads, both variations of the tree algorithm yields lower delay than ALOHA. However, the most important advantage of the tree algorithm is the significant decrease in the delay standard deviation. For instance, for a load of 0.6 the delay standard deviation decreases from 100 to 50 slots and for a load of 0.7 it decreases from 250 to 100 slots.

Figure 7 shows the effect of the access scheme and tree selection method on the performance of the tree algorithm. All the tree algorithm variations considered for this figure use level skipping and short labels. For one variation the access scheme is gated access and the tree selection method is fixed. For another variation the access scheme is free access and the tree selection method is fixed. The third variation uses free access scheme and free tree selection. Since the splitting scheme is based on deterministic addresses and since the head-end informs the stations which group of addresses is allowed to access each node of the tree, the graph shows no difference between free access and gated access. In particular, even under free access a station joining a tree might have to skip several nodes of the tree before being able to transmit its reservation for the first time (successfully or unsuccessfully).

As an example for free versus gated access, consider an execution example of the tree algorithm with fixed trees, short labels and level skipping. This execution is depicted in Figure 8(a) for free access. Suppose that two stations whose labels are 100 and 110 join tree #1. When the head-end announces the root, both stations transmit and a collision occurs. When the head-end announces the next node of this tree, it informs that only the stations whose label prefix is 0 can transmit. Thus, both stations remain silent. The next node of the tree should be assigned to prefix 1, but it is skipped due to level skipping. Thus, the head-end continues with the prefix 10. Suppose that a new station whose label is 111 joins the tree in the meantime. In the 10 node of the tree, station 100 can successfully transmit

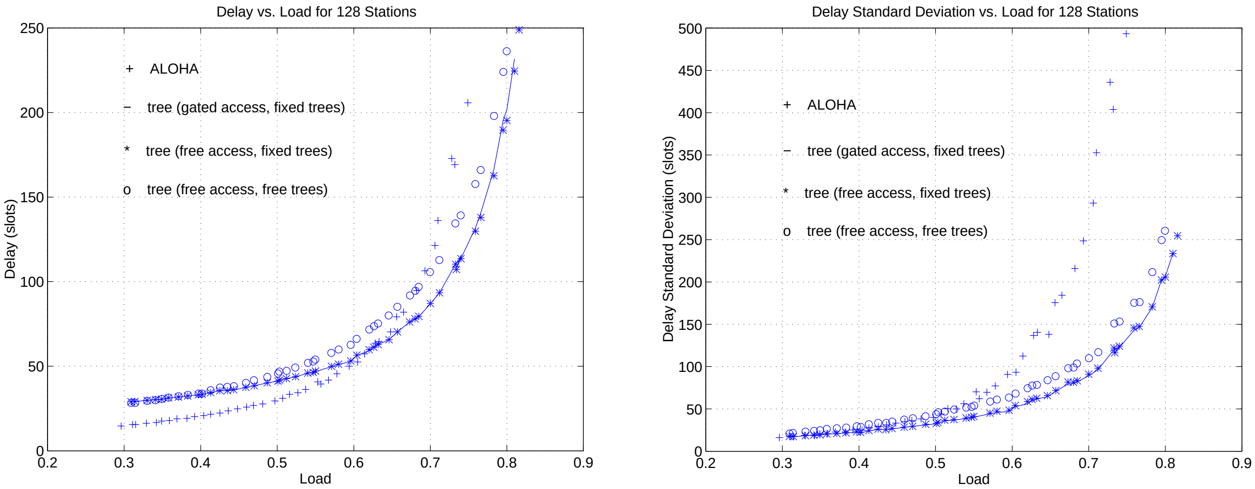
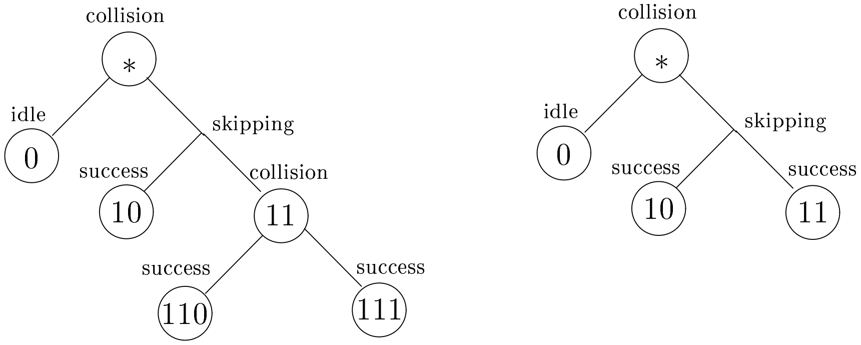


Figure 7: The Tree Algorithm: free vs. gated access

its reservation. The next node, assigned by the head-end to the prefix 11, is a collision. The subsequent two slots will be successfully used by stations 110 and 111. Then, the current epoc of tree #1 is finished, and a new one will start in the next slot of this tree. Note that under gated access station 111 is not allowed to join this epoc, and the tree is traversed as in Figure 8(b).

From Figure 7 follows that assigning trees to stations is a better strategy than letting each station join the first available tree. The reason is that fixed assignment has the effect of spreading the load to the various trees more evenly, like when window access is used, or when the "dynamic" tree algorithm is employed [4, 5]. For example, consider Figure 9 where $\Delta = 5$ trees are held by the head-end. From slot 8 there is a burst of 13 slots belonging to the reserved logical channel and used by one or more stations. Suppose that the next two slots define the roots of tree #4 and tree #5. Under free selection of trees, all the stations that get a packet for transmission during the long time interval between slot 6 and slot 20 will transmit in slot 21, and none of them in slot 22. Thus, both slots will be probably wasted. Under fixed selection of trees only 1/5 of the stations that get a packet for transmission during time interval [6-20] will transmit in slot 21, and another 1/5 will transmit in slot 22. The remaining stations will wait until the root of the other three trees are scheduled. Thus, the effect of long bursts on the expected number of contending stations in the next root node decreases.

So far it has been assumed that the round trip delay between the MXL stations and the head-end is $\Delta = 5$ slots, and therefore 5 trees are needed in order to compensate for this delay. If more trees are needed, either due to a larger propagation delay, or because the head-end needs more time to process a request before sending an acknowledgment (e.g. because



(a) under free access

(a) under gated access

Figure 8: Tree Algorithm Execution Example

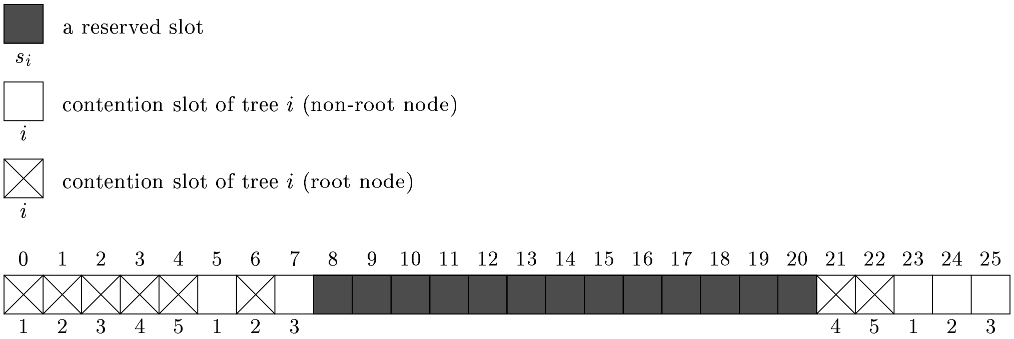


Figure 9: The Effect of Free Assignment

of forward error correction code processing), the average delay increases whereas the delay standard deviation is much less affected. Figure 10 shows the case where stations can get acknowledgments only after $\Delta = 21$ slots.

5 Conclusions

The paper has described the upstream MAC protocol employed by HP MXL (multimedia transmission link) cable-modem. It has shown that the MXL upstream channel is divided into two logical channels: a contention channel and a reserved channel. In order to transmit a packet on the upstream reserved channel, an MXL station has first to contend for an access on the contention upstream channel. Two contention algorithms have been discussed: ALOHA with truncated binary exponential back-off and the tree algorithm. It has been shown that the main drawback of ALOHA is that new contending stations are likely to get access before old contending stations. This increases the access delay variance and therefore may increase the time needed for upper layer transport protocols to re-transmit missing

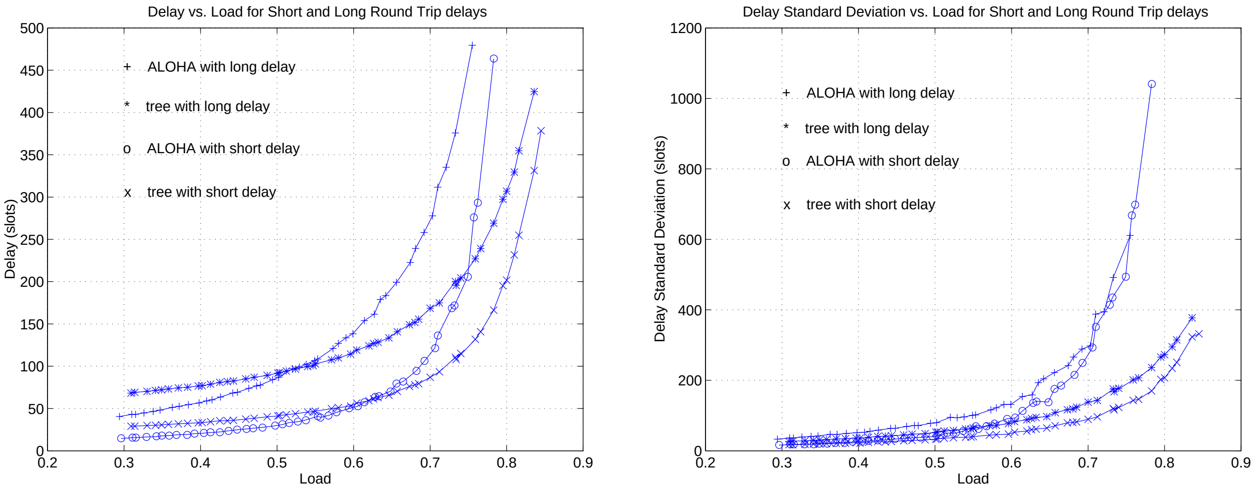


Figure 10: ALOHA vs. Tree for Long Round Trip Delay

packets.

The tree algorithm has been shown to have many variations. Some of them have been considered and studied. It has been shown that in order to implement this algorithm, the head-end needs to maintain Δ trees, where Δ is the number of slots in the ACK-window, and to assign each upstream contention slot to a given node in a given tree. The trees are served in a round robin manner, and traversed in a depth first search (DFS) order. All the considered variations of the tree algorithm have been shown to yield a small throughput advantage and a significant delay variance advantage over ALOHA.

6 References

[1] D. Bertsekas and R. Gallager. *Data Networks.* Prentice Hall, second edition, 1992.

[2] C. Bisdikian, B. McNeil, R. Norman, and R. Zeisz. MLAP: A MAC level access protocol for the HFC 802.14 network. *IEEE Communications Magazine*, 34(3), March 1996.

[3] L. Brakmo, S. O'Malley, and L. peterson. TCP Vegas: New techniques for congestion detection and avoidance. In *SIGCOMM*, pages 24–35, May 1994.

[4] J. Capetanakis. Tree algorithms for packet broadcast channel. *IEEE Transactions on Information Theory*, 25(5):505–515, 1977.

[5] J. Capetanakis. Generalized TDMA: the multi-accessing tree protocol. *IEEE Transactions on Communications*, 27(10):1476–1484, 1979.

[6] J. Dail, M. Dajer, C. Li, P. Magill, C. Siller, K. Sriram, and N. Whitaker. Adaptive digital access protocol: A MAC protocol for multiservice broadband access networks. *IEEE Communications Magazine*, 34(3), March 1996.

[7] R. Gusella. A measurement study of diskless workstation traffic on an ethernet. *IEEE Journal on Selected Areas in Communications*, 38(9), 1990.

[8] M. Johnson. Proof that timing requirements of the fddi token ring protocol are satisfied. *IEEE Transactions on Communications*, 35(6):620–625, June 1987.

[9] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic. *IEEE/ACM Transactions on Networking*, 2(1):1–15, February 1994.

[10] M. Molle. A new logarithmic arbitration method for Ethernet. Technical Report CSRI-298, Computer Systems Research Institute, University of Toronto, 1994.

[11] M. Molle and G. Polyzos. Conflict resolution algorithms and their performance analysis. Technical Report CS93-300, University of California, San Diego, 1993.

[12] V. Paxson and S. Floyd. Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[13] D. Sala and J. Limb. A protocol for efficient transfer of data over a fiber/cable systems. In *INFOCOM*, March 1996.