Performant Implementations of Tree Collision Resolution Algorithms for CATV Networks

Philippe Jacquet, Paul Mühlethaler, Philippe Robert

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Performant implementations of tree collision resolution algorithms for CATV networks

Philippe Jacquet , Paul Mühlethaler , Philippe Robert

No 4107

Janvier 2001

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4107--FR+ENG

Performant implementations of tree collision resolution algorithms for CATV networks

Philippe Jacquet , Paul Mühlethaler , Philippe Robert

Thème 1 — Réseaux et systèmes
Projet HIPERCOM

Rapport de recherche n ° 4107 — Janvier 2001 — 34 pages

Abstract: In this paper, we analyze optimizations and adptations of the stack (tree) algorithm needed to use it as access scheme for a CATV channel access. In CATV network, we have large round trip delays, therefore it is possible have simultaneous tree collision resolution process. One approach is to handle many independant collision resolution process, another is to interleave all the collision resolution process. We give a detailed analysis of the two schemes concerning average and distribution of access delays. We show that the interleaving technique is both more simple and more efficient. This leads us to define a CATV channel access technique with interleaved collision resolution process on which we had a mechanism to offer reservation for successive slots. We a give a detailed implementation of this protocol as a thorough performance analysis. We deeply investigate the effect on throughput and delays of the end to end propagation delay and of the number of active stations.

Key-words: Medium Access Control scheme, Cable TV network (CATV network), Ethernet, Collision resolution algorithm , tree algorithm, performance evaluation, access delay, delay distribution.

(Résumé : tsvp)

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : 01 39 63 55 11 - International : +33 1 39 63 55 11
Télécopie : (33) 01 39 63 53 30 - International : +33 1 39 63 53 30

Implémentations efficaces de protocoles de résolution de collision en arbre pour réseaux câblés de télévision

Résumé : Dans ce papier, nous présentons des optimizations et des adaptations de l'algorithme de résolution de collision en arbre pour les réseaux câblés de télévision. Dans un réseau câblé de télévision, il y a de longs délais pour savoir si une trame a été envoyée avec succès, donc il est possible d'avoir plusieurs algorithmes de résolution de collision concurrents. Une approche est de gérer plusieurs processus de résolution de collision en arbre indépendants. Une seconde approche est de mélanger les différents processus. Nous décrivons et analysons en détail les deux approches. Nous montrons que la première approche est à la fois plus simple et plus performante. Cela nous conduit à définir un protocole d'accès pour réseau câblé de telévision qui mélange les processus de résolution de collision sur lequel nous offrons un système additionnel de réservation de slot successif. Nous donnons une description et une analyse détaillée de ce protocole. Nous étudions en profondeur l'effet sur le débit et les délais du nombre de stations actives et du délais de propagation de bout en bout.

Mots-clé : Protocoles d'accès multiple, réseau câblé de télévision, Algorithme de résolution de collision, Algorithme en arbre, évaluation de performance, délais d'accès, distribution des délais .

1 Introduction

Cable TV networks are now existing for more than thirty years. These large networks were used in towns to distribute TV channel. In the early 90s the technology started works to use Cable TV to transmit data. Evidently broadcast of data is easy while there are more problems concerning the return path. First the transmission on the return is more difficult due to noise problems, this explain why the return channel generally offers a significantly smaller throughput than the broadcast channel. Secondly the return channel is, of course, a shared medium. Sharing efficiently this medium will be our main interest in this paper. Cable TV networks are metropolitan networks. The topology of a Cable TV network (CATV network) is a tree. At the root one can find the cable TV headend and one finds the user nodes at the leaves of the tree. The headend is an active part of the system which concentrates and redistributes data sent by the nodes. Cable TV networks have a large geographical extension, the headend may be up to 100 km far from a user node in the larger networks. See figure 1. The nodes transmit their packets towards the headend on the return channel. The headend concentrates the packets and broadcasts the packets to the nodes. In the following, we suppose the simplest architecture: the headend station uses a dedicated forward channel on which it echoes all signals received from the node on the return channel. See figure 1.

In paper [10] we have presented a slotted access control scheme designed for CATV network using a tree algorithm as resolution mechanism. The tree algorithm presents the main advantage of being stable, predictable and performant. It has given rise to a large litterature for the past fifteen years [1, 2, 3]. But there are many ways to use tree algorithms as collision resolution scheme in CATV networks. The main reason for this is that one must cope a large round trip feedback time Δ (this delay can be up to 800 μsec). It is peculiar to notice that the physical conditions arising in CATV networks are in some respect the same than the conditions in access for high speed network: the transmission delay is small compared to the propagation delay [11]. Another reason is that one must add a reservation technique to improve the maximum achievable throughput. The presence of such a large feedback time is probably

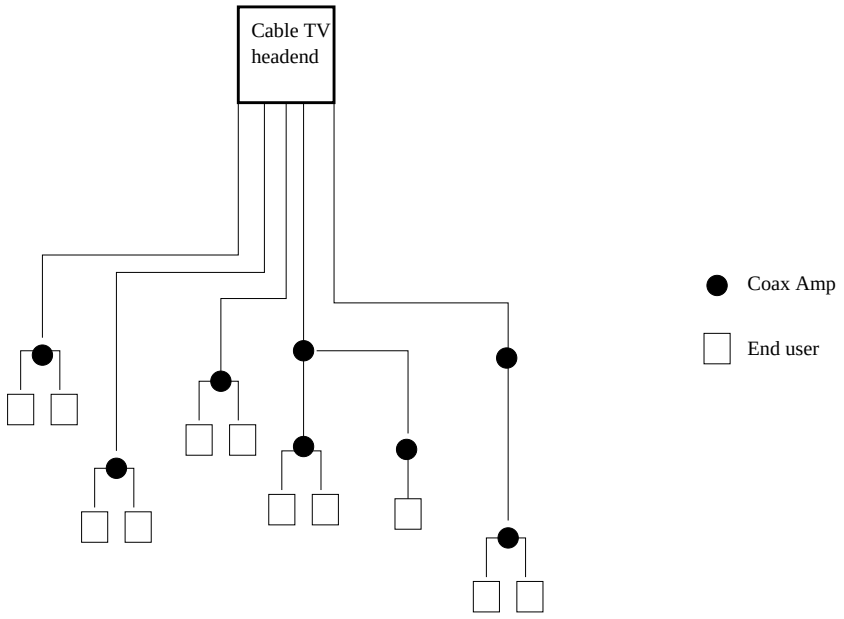

Figure 1: General architecture for CATV.

the newest feature introduced by CATV networks concerning access techniques and tree resolution schemes.

One can identify two main classes of implementations of the tree resolution algorithm:

- the sequential implementations;
- the parallelized implementations.

In our paper [10] we have presented an easy parallelized implementation with reservation of the tree algorithm which does not add any management complexity. It must be noted that it is not necessarily the case for any other parallelized implementation. The paper is organized as followed. In section 2 we describe two different implementations of tree algorithms for large round trip feedback delays. The first scheme *sequential trees* simply handles many parallele tree resolution algorithms. The second scheme *easy parallelized tree*

interleaves the tree resolution process. We give detailed description and performance evaluation of these two schemes. As a result of this analysis we show that the second implementation provides better performance. This leads us to section 3 deals where we describe a CATV channel access technique with interleaved collision resolution process on which we have had extra facilities to offer reservation. a tree resolution algorithm including a reservation scheme. We a give a detailed implementation analysis of this protocol. We deeply investigate the effect on the delays distribution of the end to end propagation delay and of the number of active stations.

This paper has been proposed as a contribution to the IEEE 802.14 commity.

2 Implementation of tree algorithm with large feedback delays

We will use here the slotted access channel that are defined in [10]. In this reference the issues as ranging or synchronization problems which are incurred by using slots are discussed. In this paper the only two assumptions we need

- nodes have slots to send their packets,
- the result of a transmission on a given which is random access for all the nodes in the CATV networks is known a number of slots latter. If the slot has been used by a single node the transmission is a success. If the slot has been used by more than one node the transmission is a collision. If the slot has not been used, the transmission is an idle.

Let us suppose that a slot starts at time t_0, the network nodes can use this slot to send their packet. The result of this transmission will be reported on the slot starting at time $t_0 + \Delta$.

2.1 Obvious sequential tree implementation

We describe the collision resolution process of a transmission starting at time t_0. If a collision occurs on this slot, then at time $t_0 + \Delta$ a collision is reported.

Each colliding users randomly set a counter at an integer value between 0 and $M-1$ inclusively, and follows the algorithm:

1. On slot $t = t_0$ *modulo* Δ if collision is reported then increment C, else (it is a non collision) then decrement C.

2. Retransmit when $C = 0$, and wait for a duration Δ to get the feedback.

If the tree is M-ary then "increment C" means "add $M-1$ to C". Decrement C means "substract 1 to C".

In the sequential tree resolution, each collision is resolved *via* the classic way, *i.e.* each step of the tree algorithm follows after the feedback of the previous step (*i.e.* after Δ). For example if users a and b collide at time t_0, then at time $t = t_0 + \Delta$, they both receive feedback of their collision. We assume that a selects $C = 0$ and b selects $C = 1$. User a will transmit at $t = t_0 + \Delta$ as b will transmit at $t = t_0 + 2\Delta$ after the feedback concerning the second transmission attempt of a (it will a sucess).

If Δ is large several collision resolution can take place each of them mapping on on different slots, *modulo* Δ.

2.2 Easy parallelized tree implementation

A transmission (first attempt to send a packet) starts at time t_0. If a collision occurs on this slot, then at time $t_0 + \Delta$ a collision is reported. Each colliding users randomly set a counter at an integer value between 0 and $M-1$ inclusively, and follows the algorithm:

1. If slot starting t is a collision then increment C, if it is a non collision then decrement C.

2. retransmit when $C = 0$, and wait for Δ feedback.

First of all one remarks that the implementation is in definitive easier than with sequential approach, because the condition $t = t_0$ *modulo* Δ is removed.

Notice that the following algorithm is rigorously equivalent:

1. If slot t is a collision then add M to C,

2. After each slot decrement C.

3. retransmit when $C = 0$, and wait for Δ feedback.

Before going further, one must be convinced that this implementation really provide a tree collision resolution. Indeed, colliders which have been involved in two collisions occurring on two different slots, for example at t_0 and $t_0 + 1$, will never mixe on the future. Hint: if a and b retransmit at time t, *i.e.* their counter C reach zero at time t, then by backward tracking, their counter C have been identical on every slot in the past and must have been initialized at the same time.

In fact the parallelized implementation sees the same contention events than the sequential implementation, but with a different chronology. In the parallelized implementation, the resolution of the tree is made in a width first approach. In this case, when a collision occurs for example between user a and b at time t_0, then at time $t_0 + \Delta$ a select $C = 0$ and b selects $C = 1$. Therefore (provided no other activity on the channel) a successfully transmit on slot $t_0 + \Delta$ and b transmits on slot $t_0 + \Delta + 1$. User b has gained access $\Delta - 1$ time before sequential implementation.

In the example given below, the collision was resolved in an optimal way (no more collision after the initial collision). In case of a longer tree resolution, the gain in access delay between sequential and parallelized would have been greater.

As with sequential implementation one can define free-access or blocked access tree algorithm in (easy) parallelized implementations. There are also other possible parallelized implementations of tree algorithms, but some of them lead to non-trivial slot allocation.

2.3 Performance results

The results which are presented in this section are obtained via analytical means. The methodology is borrowed from [5, 4, 6, 7, 9].

2.3.1 Models

We assume that all contention packets are one slot long and that the feedback delay Δ is 40 slots. We omit the data part of the packets which should be

transmitted in reserved periods in the channel (actually reserved by the contention packets). We assume a Poisson arrival of λ packet per contention slot and the node population is (potentially) infinite. The tree algorithms are free access M-ary splitting algorithms.

In the sequel we address the access delay distribution of each contention packet, *i.e.* the time between the first transmission attempt of the packet and its successful transmission.

2.3.2 Average access delay

In figures 2 and 3 we display the average access delays versus the input load λ per contention slot. For any given splitting argument M, both sequential and parallelized implementation of M-ary tree algorithm are stable under the same conditions, *i.e.* stable for $\lambda < \lambda_{\max}$. The table for $\lambda_{\max}$ is given in table 1 below.

We immediately notice that $\lambda_{\max}$ addresses only the input load *per contention slot*. If the data part of the packet is $\mathcal{L}$ larger than the contention slot and is transmitted in reserved period (reserved via the contention procedure), then the admissible total input load is much higher. Table 1 shows example of admissible loads $C_{\max}$ for different values of $\mathcal{L}$.

M	$\lambda_{\max}$	$C_{\max}$: $\mathcal{L}=8$	$C_{\max}$: $\mathcal{L}=16$
2	0.360177	0.818296	0.900069
3	0.401599	0.842988	0.914806
4	0.399293	0.841713	0.914054
5	0.387241	0.834866	0.910002
6	0.373354	0.826580	0.905058
7	0.359731	0.818008	0.899894

Table 1: maximum stable throughput for tree algorithms

Notice that the difference in $\lambda_{\max}$ between $M = 3$ and $M = 4$ are actually negligible, and lead to negligible access delays discrepancies in sequential implementation. In parallelized implementation there is still a difference between $M = 3$ and $M = 4$ because for in a same tree resolution the 4-ary takes less collision than the 3-ary, and the number of collisions per packet is the main

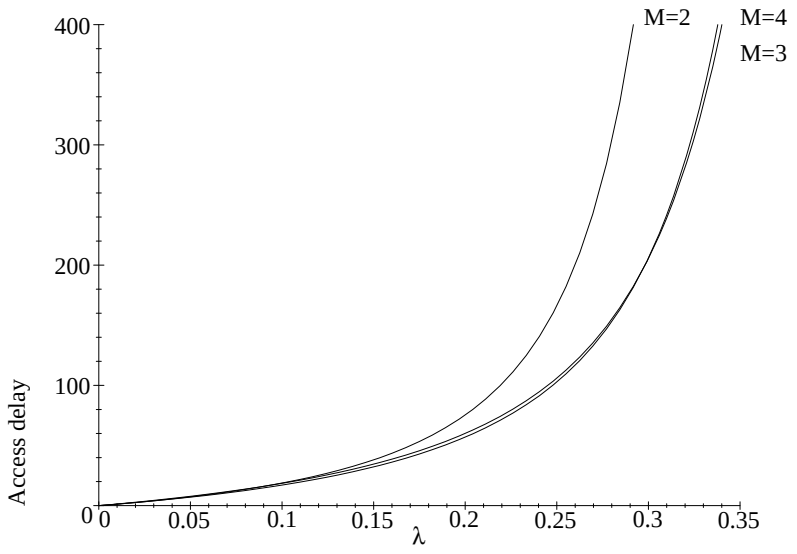

Figure 2: Average access delay versus λ packet per contention slot, sequential M-ary tree algorithm with $M = 2$, 3 and 4. Feedback time $\Delta = 40$ contention slots.

contributor to access delay in parallelized implementations. However 4-ary protocol will diverge slightly earlier than 3-ary when λ increases.

2.3.3 Access delay distributions

In this section we display some results about the distribution of access delay. In particular, for a given number $p < 1$ we display the critical delay D *i.e.* the largest value D such that the probability to have an access delay smaller than D is kept smaller than p. We will take $p = .7$, .8, 0.9, .95 and .99.

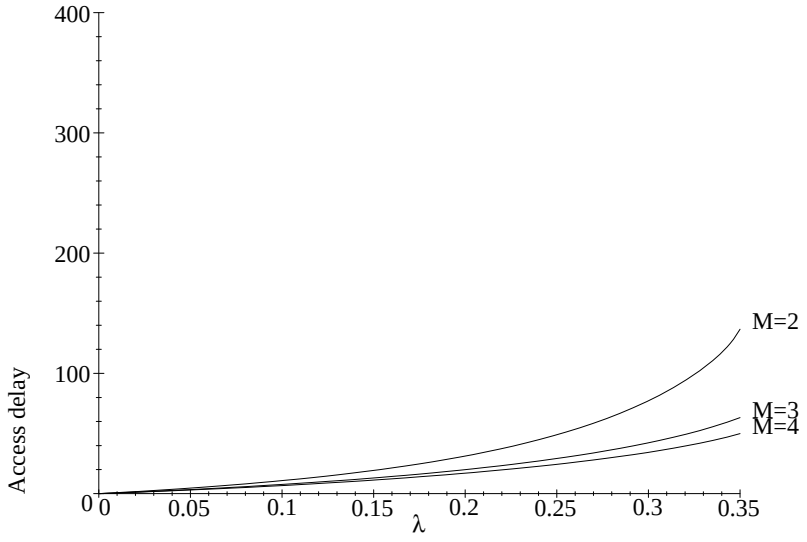

Figure 3: Average access delay versus λ packet per contention slot, easy parallelized M-ary tree algorithm with $M = 2$, 3 and 4. Feedback time $\Delta = 40$ contention slots.

2.3.4 $M = 2$ binary tree

The binary tree algorithm remains stable up to $\lambda = .360177\ldots$ packet per contention slot.

On the next figure, one gives critical access delay in the same condition but with the easy parallelized 2-ary tree algorithm.

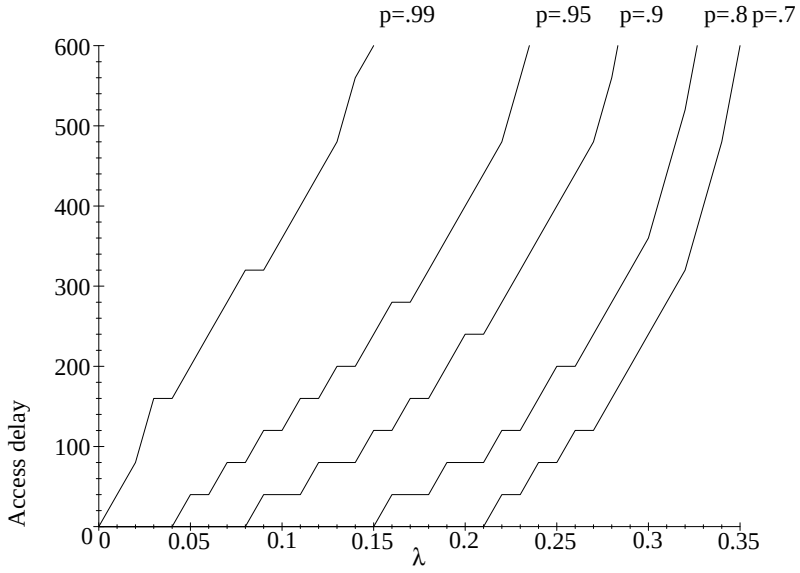

Figure 4: Critical access delay in contention slots for probability $p = .7$, .8,0.9, .95 and .99, versus λ packet per contention slot,sequential 2-ary tree algorithm. Feedback time $\Delta = 40$ contention slots.

2.3.5 $M = 3$ ternary tree

The ternary tree algorithm remains stable up to $\lambda = .401599\ldots$ packet per contention slot.

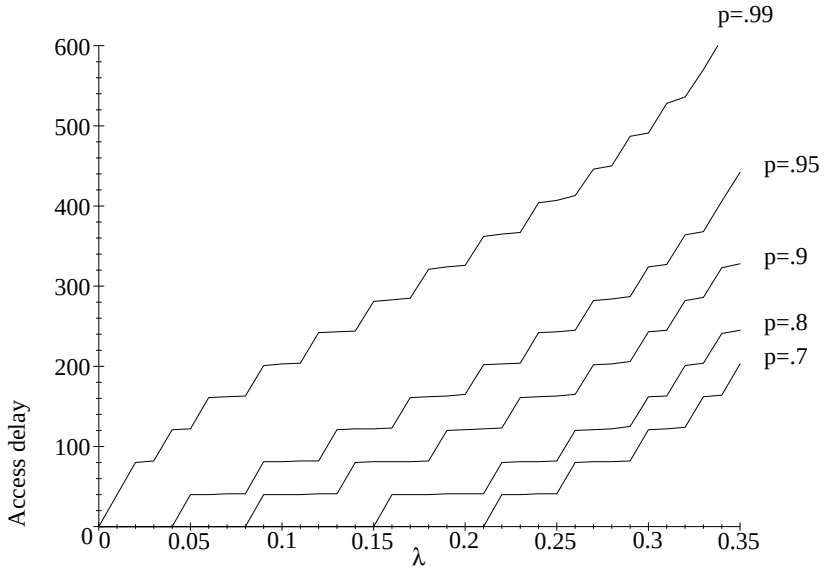

Figure 5: Critical access delay in contention slots for probability $p = .7$, .8,0.9, .95 and .99, versus λ packet per contention slot,easy parallelized 2-ary tree algorithm. Feedback time $\Delta = 40$ contention slots.

2.3.6 $M = 4$ quaternary tree

The quaternary tree algorithm remains stable up to $\lambda = .399293\ldots$ packet per contention slot.

3 Tree algorithm with reservation scheme

3.1 Tree algorithm in CATV context

The implementation for CATV differs with the basic implementation analyzed in the previous section in the fact that packets are reserved by request packets and request packet solve contentions via the parallelized tree algorithm. The algorithms for CATV have been described in [10]. The request packet contains the reserved field: *i.e* the length of the data to be transmitted during the reserved burst. The reserved queue is filled with the requests which are successfully transmitted. Each node monitor the size L of the reserved queue

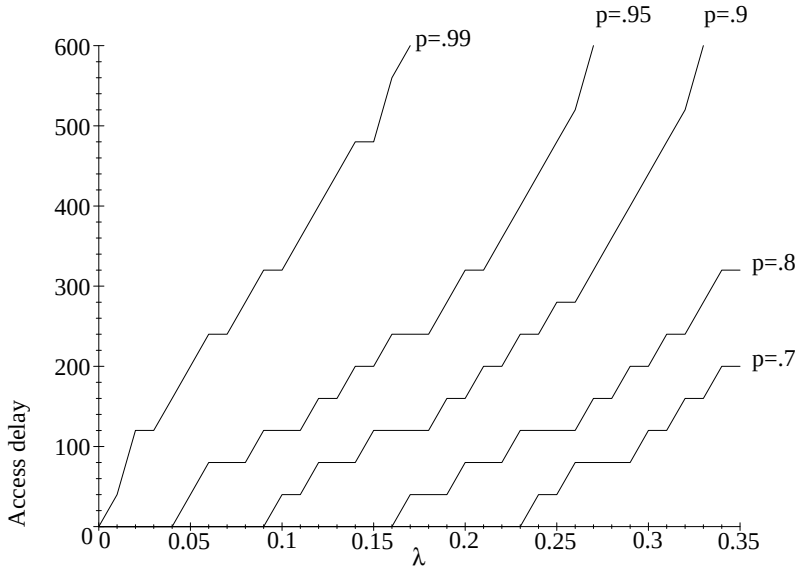

Figure 6: Critical access delay in contention slots for probability $p = .7$, .8, 0.9, .95 and .99, versus λ packet per contention slot, sequential 3-ary tree algorithm. Feedback time $\Delta = 40$ contention slots.

simply by adding to it the reserved field by each request successfully detected on the feedback, and by decrementing it with time.

In the following section we display the performance of the parallelized tree implementation under the CATV traffic model.

The algorithm augmented of reservation is as follow (reserved field and L are managed in multiple of contention slots):

1. if slot t is a collision then add M to C;
2. if slot t is a success then add the reserved field to L;
3. after each slot decrement C if $L = 0$, decrement L otherwise;
4. retransmit when $C = 0$, and wait for Δ feedback.

Notice a minor variation within clause 2 which can be modified into "if slot t is a success then add 1 to C and add the reserved field to L". This version is more tractable to analysis [8] when the packet size is variable and random.

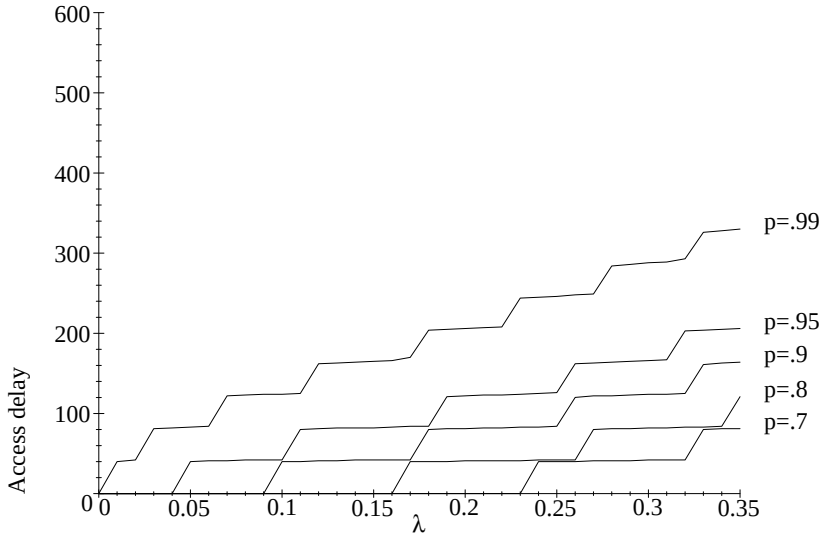

Figure 7: Critical access delay in contention slots for probability $p = .7$, .8, 0.9, .95 and .99, versus λ packet per contention slot, easy parallelized 3-ary tree algorithm. Feedback time $\Delta = 40$ contention slots.

The results which are presented throughout this section are obtained via an event driven simulator. The packets are generated according to a Poisson process.The size of the packets follows a random distribution which is given in table 2. Each node is modeled via a FIFO queue where locally generated packets are stored. The size of the node buffer is set to 20 packets. In fact queues in node actually become existent only when the load is very close or exceeds protocol capacity C_{max}.

packet length	probability
512	0.6
1024	0.06
2048	0.04
4096	0.02
8192	0.25
12144	0.03

Table 2: Packet length distribution

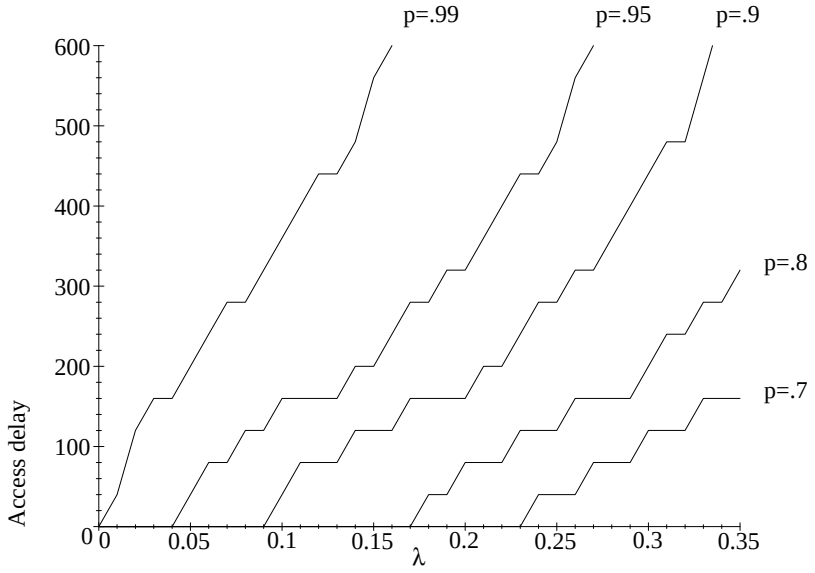

Figure 8: Critical access delay in contention slots for probability $p = .7$, .8, 0.9, .95 and .99, versus λ packet per contention slot, sequential 4-ary tree algorithm. Feedback time $\Delta = 40$ contention slots.

The main parameter of the protocol is the size of the contention slot. We simulate two contention slot sizes: 64 bit and 128 bit. The the request packet which fits the contention slot must contain a source ID, a reservation field and a CRC field. Notice that the source ID can be replaced by any smaller pseudorandom sequence which will suffice for the transmitter to identify its own request packet with low error probability.

	case 1	case 2
contention slot	64	128

Figure 10 shows the difference in mean access delay for traffic scenario type 1 with a feedback time of 160μsec corresponding to a distance to head-end of 80 km. Case 1. Notice that the difference of throughput is negligible, because the average packet size (368.1 bytes) is important compared to contention slot size (8 bytes). The optimal is still $M = 4$ but the discrepancies are not very significant.

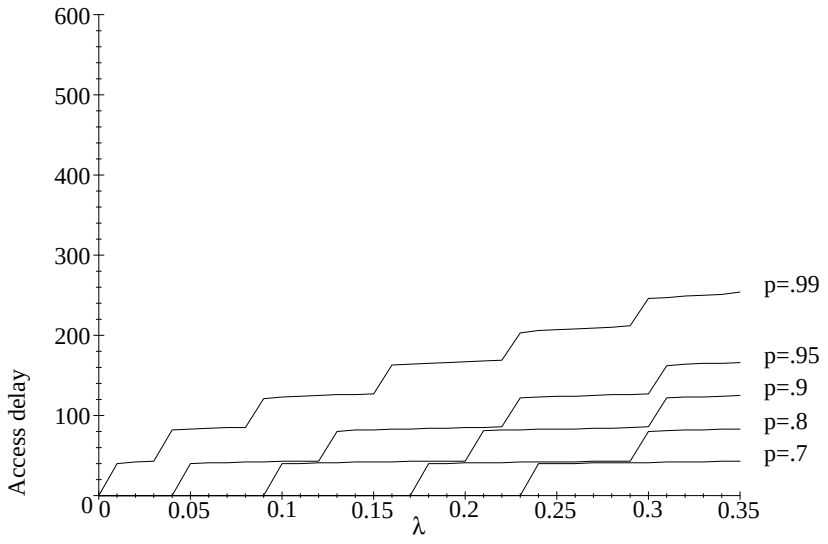

Figure 9: Critical access delay in contention slots for probability $p = .7$, .8, 0.9, .95 and .99, versus λ packet per contention slot, easy parallelized 4-ary tree algorithm. Feedback time $\Delta = 40$ contention slots.

3.2 Access delays for varying number of nodes

This section analyzes the impact of the number of nodes on the access delays. As shown with the analytic model, the tree algorithm is stable under a (potentially) infinite population. This property is sometimes called "channel transparency". Consequently, provided that the offered load remains the same and stays under protocol capacity, the access delay tends to a finite distribution when the number of active user indefinitely increases. This property is well illustrated by simulations.

Figures 11 and 12 show the mean access delay for different size N of active population. Quantity N varies from 5 to 200. Notice the relative fast convergence of the average access delays when the offered load is below the protocol capacity (around 0.95).

Figure 13 shows the standard deviation of access delays for N varying from 5 to 200.

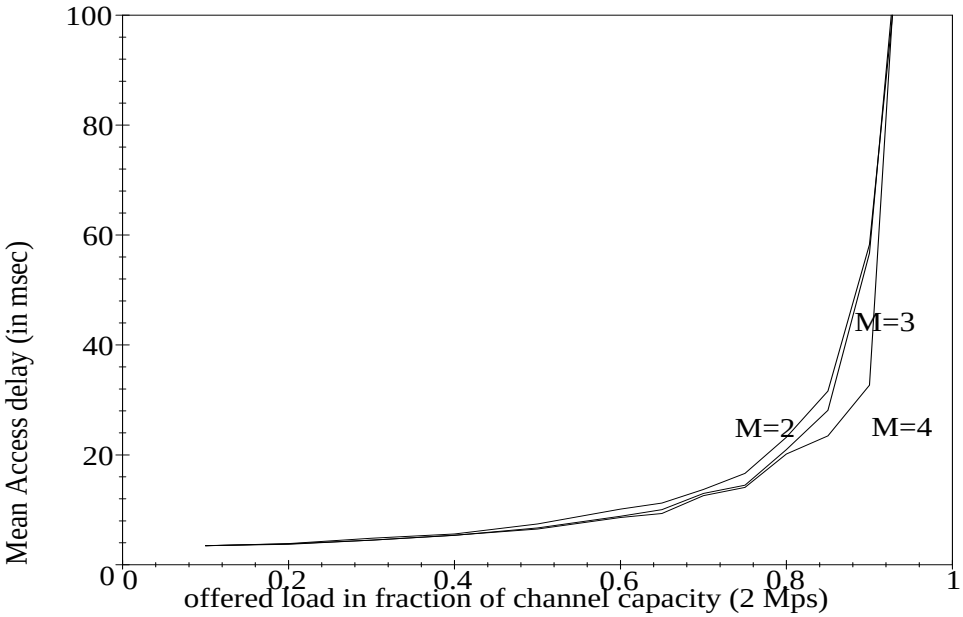

Figure 10: Mean access delay versus offered load, with parallelized M-ary tree algorithm with $M = 2$, 3 and 4. Case 1, Feedback time $\Delta = 800\mu$sec (25 contention slots).

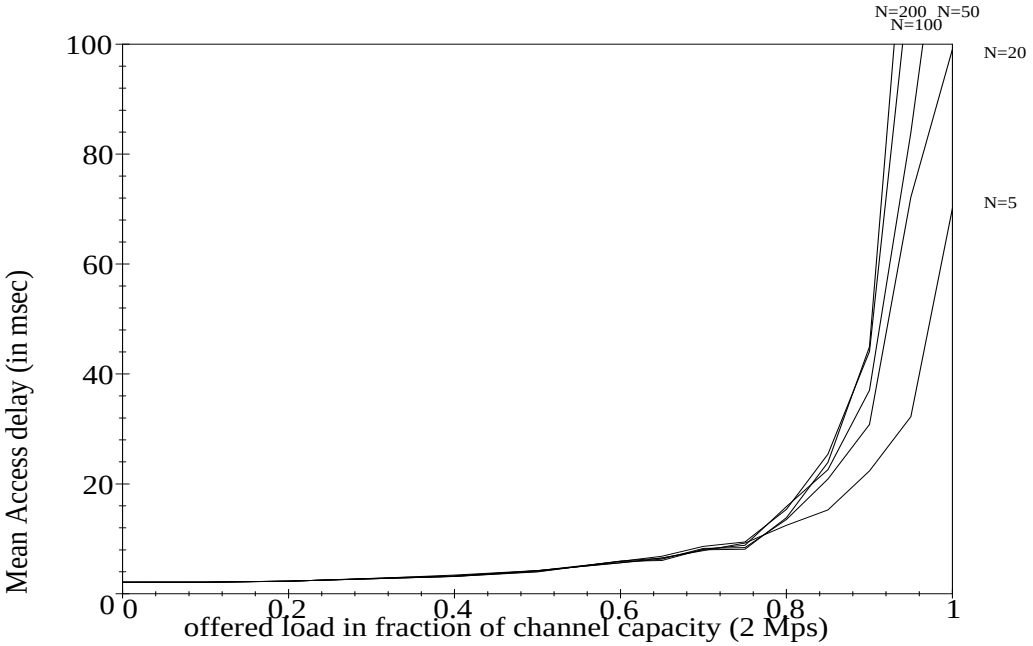

Figure 11: Mean access delay versus offered load, with parallelized 4-ary tree algorithm with N connected users $N = 5$, $N = 20$, $N = 50$, $N = 100$ and $N = 200$. Case 1, Feedback time $\Delta = 160\mu$sec (5 contention slots).

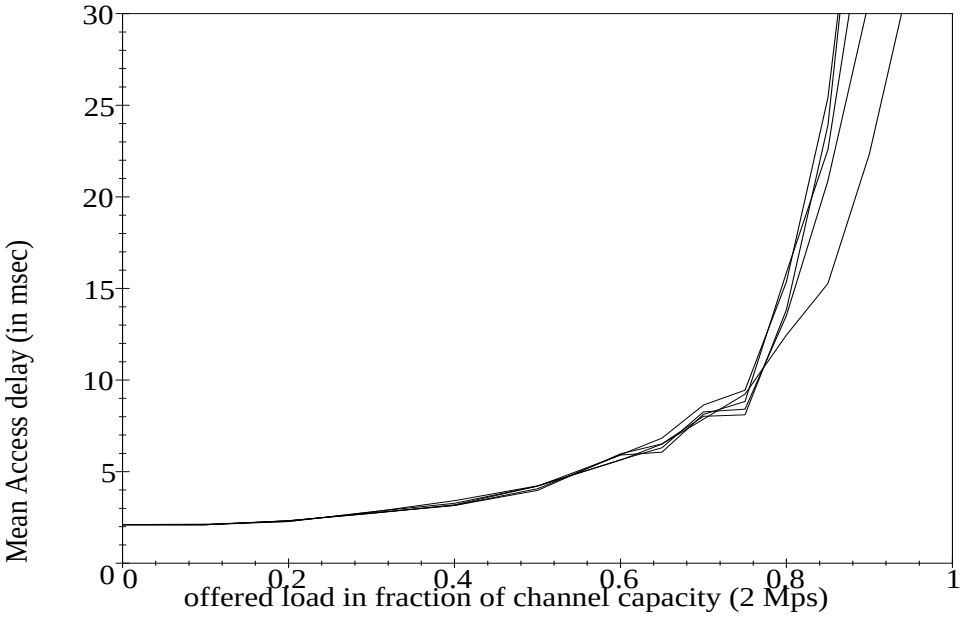

Figure 12: Mean access delay versus offered load, with parallelized 4-ary tree algorithm with N connected users $N = 5$, $N = 20$, $N = 50$, $N = 100$ and $N = 200$. Case 1, Feedback time $\Delta = 160\mu$sec (5 contention slots). Magnified part of previous figure.

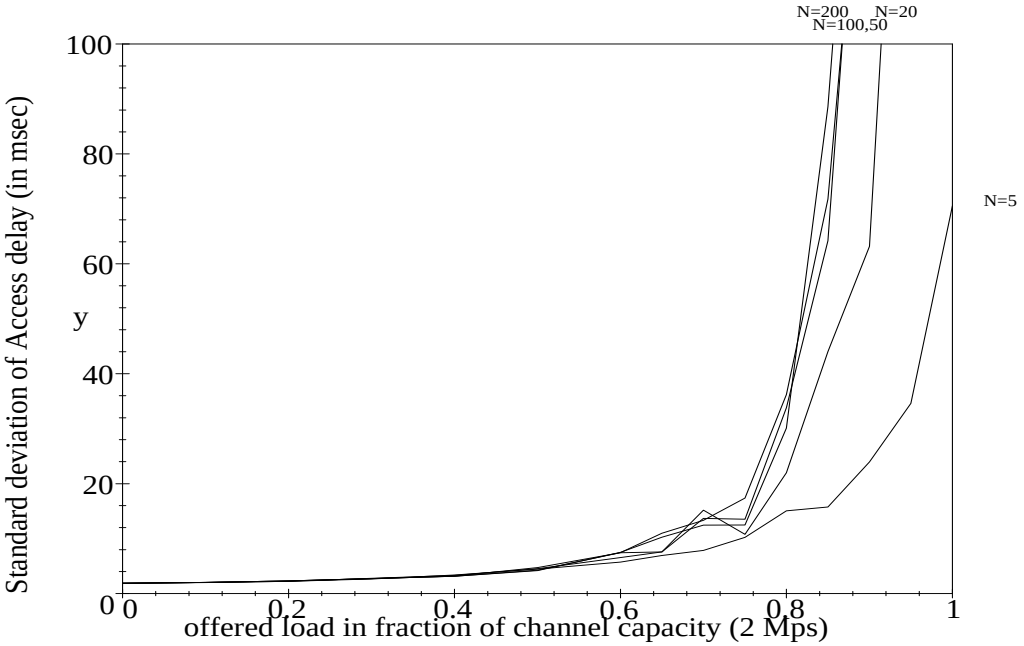

Figure 13: Standard deviation of access delay versus offered load, with parallelized 4-ary tree algorithm with N connected users $N = 5$, $N = 20$, $N = 50$, $N = 100$ and $N = 200$. Case 1, Feedback time $\Delta = 160\mu$sec (5 contention slots).

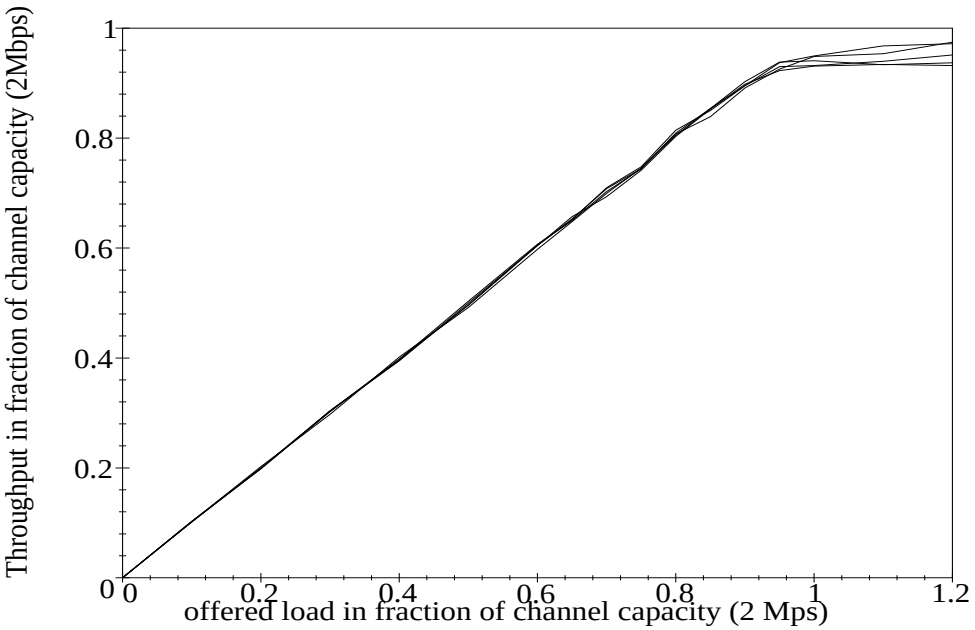

Figure 14: Throughput versus offered load, with parallelized 4-ary tree algorithm with N connected users $N = 5$, $N = 20$, $N = 50$, $N = 100$ and $N = 200$. Case 1, Feedback time $\Delta = 160\mu$sec (5 contention slots).

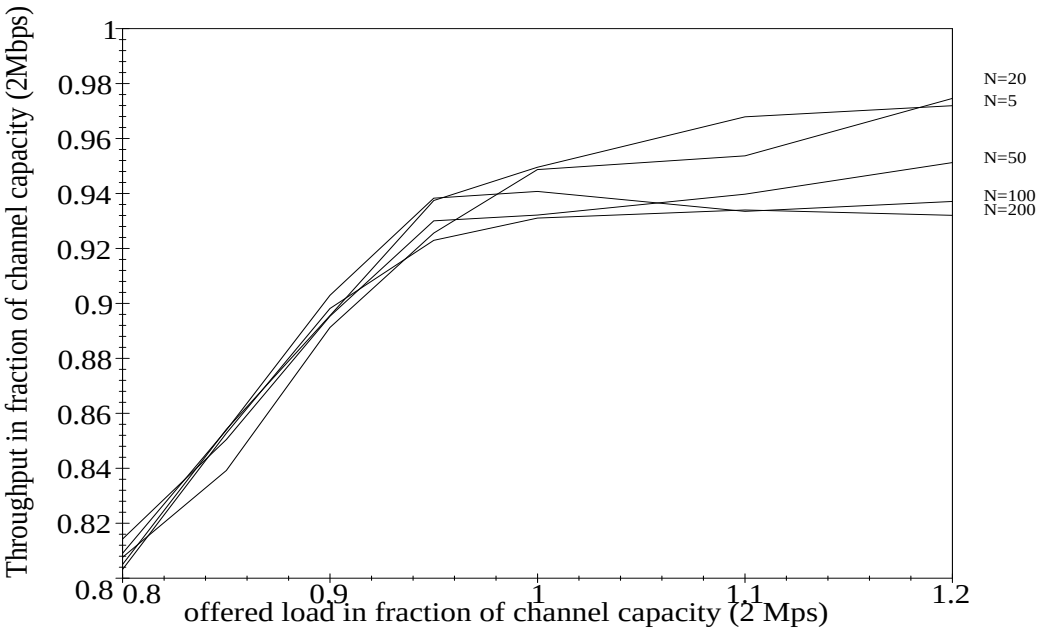

Figure 15: Throughput versus offered load, with parallelized 4-ary tree algorithm with N connected users $N = 5$, $N = 20$, $N = 50$, $N = 100$ and $N = 200$. Case 1, Feedback time $\Delta = 160\mu$sec (5 contention slots). Magnified part of previous figure.

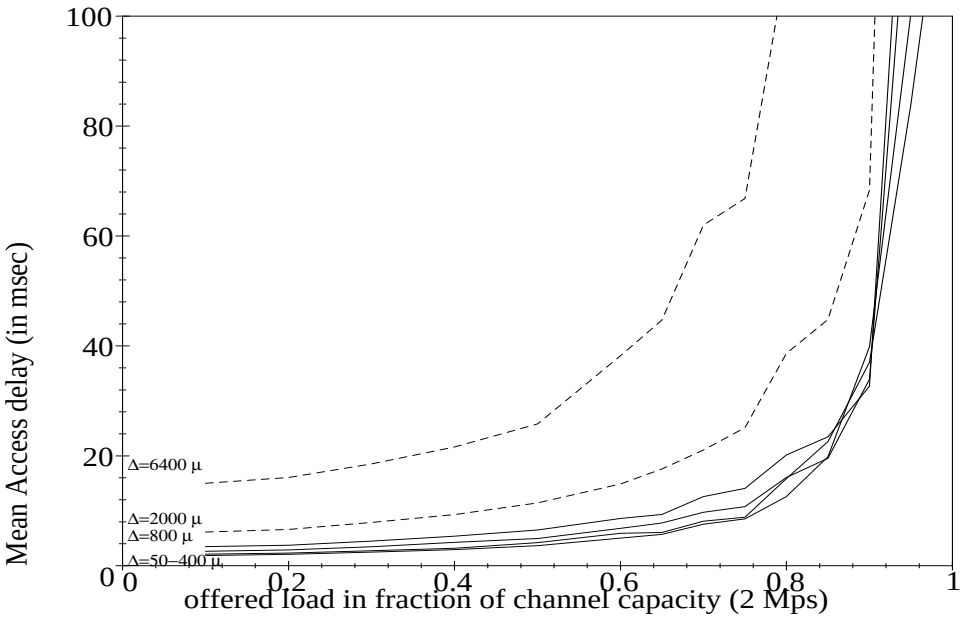

Figure 16: Mean access delay versus offered load, with parallelized 4-ary tree algorithm with $N = 50$ active users and varying feedback times. Case 1, Feedback time $\Delta = 50\mu$sec (5 km), $\Delta = 160\mu$sec (16 km), $\Delta = 400\mu$sec (40 km), $\Delta = 800\mu$sec (80km), $\Delta = 2000\mu$sec and $\Delta = 6400\mu$sec.

Figures 14 and 15 show the throughput delivered versus offered load with varying number of node. The relative stability with respect to the number of nodes illustrate the channel transparency of the protocol.

3.3 Access delay for varying feedback time

We assume that the feedback times varies with the distance to head-end with a propagation time of 5μsec per kilometer (therefore 10μsec per km for the round trip delay). The feedback time can also depend on electronic and/or processing times.

Figures 16 and 17 show the mean access delay for varying feedback time. The feedback time varies between 50μsec and 800μsec, and from 2000μsec to 6400μsec. With 10μsec round trip delay km, this feedback times correspond

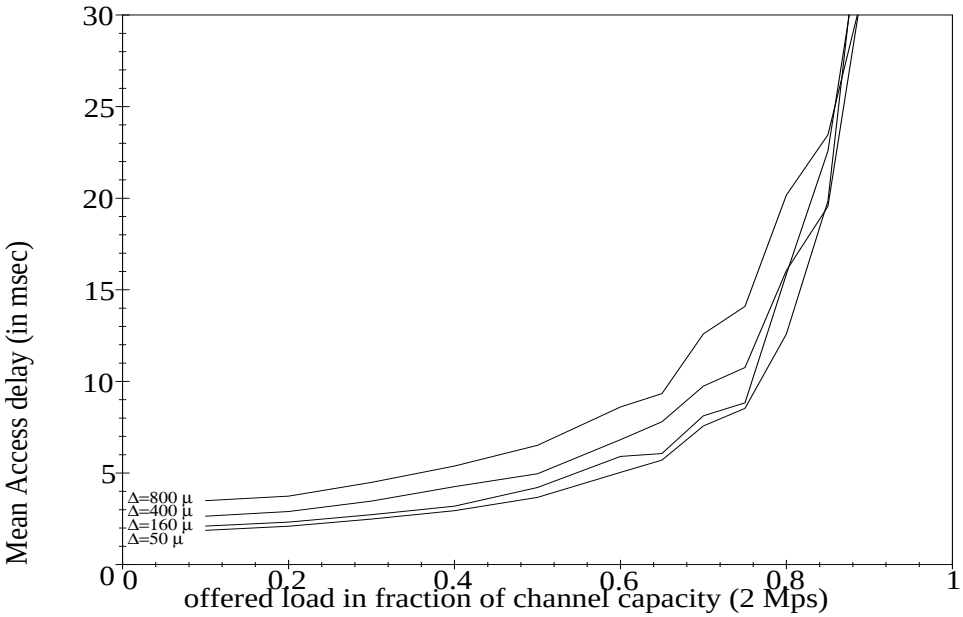

Figure 17: Mean access delay versus offered load, with parallelized 4-ary tree algorithm with $N = 50$ active users and varying feedback times. Case 1, Feedback time $\Delta = 50\mu$sec (5 km), $\Delta = 160\mu$sec (16 km), $\Delta = 400\mu$sec (40 km), $\Delta = 800\mu$sec (80km). Magnified part of previous figure.

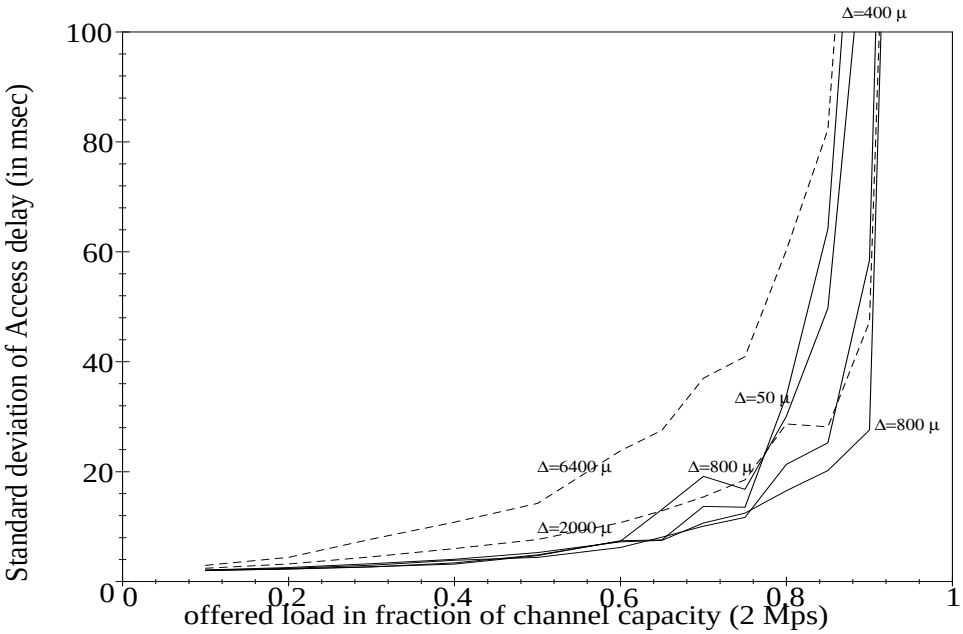

Figure 18: Standard deviation of access delay versus offered load, with parallelized 4-ary tree algorithm with $N = 50$ active users and varying feedback times. Case 1, Feedback time $\Delta = 50\mu$sec (5 km), $\Delta = 160\mu$sec (16 km), $\Delta = 400\mu$sec (40 km), $\Delta = 800\mu$sec (80km), $\Delta = 2000\mu$sec and $\Delta = 6400\mu$sec.

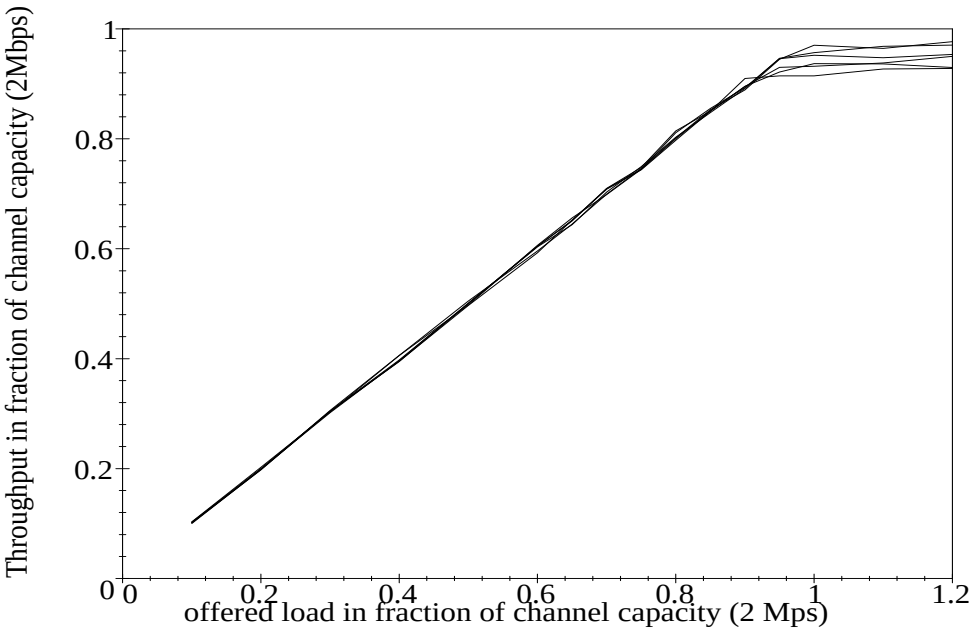

Figure 19: Throughput versus offered load, with parallelized 4-ary tree algorithm with $N = 50$ active users and varying feedback times. Case 1, Feedback time $\Delta = 50\mu$sec (5 km), $\Delta = 160\mu$sec (16 km), $\Delta = 400\mu$sec (40 km), $\Delta = 800\mu$sec (80km), $\Delta = 2000\mu$sec and $\Delta = 6400\mu$sec.

to theoretical distance to head-end varying between 5 km and 640 km. The last figure may look unrealistic, but it maybe reflects the situation when the processing delay on head-end would be added to physical propagation delay. Figure 18 shows the standard deviation of access delay under the same hypothesis.

Figures 19 and 20 show the impact of the feedback time on the channel capacity. The number of user is still 50, and the feedback time varies between 50μsec and 6400μsec. Notice the slight drop of throughput for $\Delta = 6400\mu$sec. It is due to the fact that the 50 nodes cannot fill the total bandwidth because the round trip delay makes it impossible to fill it with only 50 simultaneous requests.

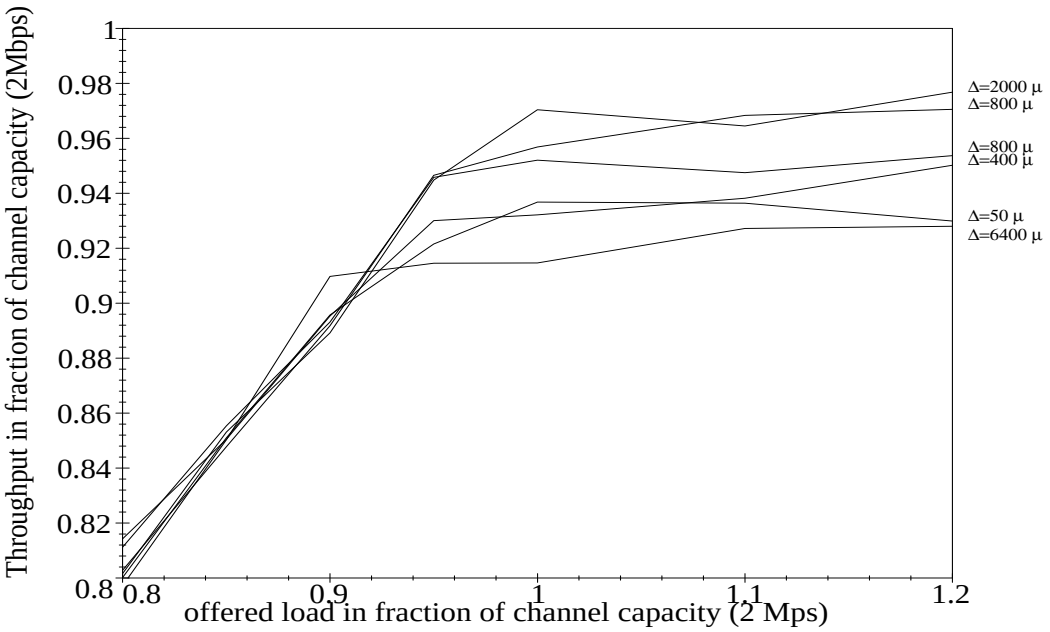

Figure 20: Throughput versus offered load, with parallelized 4-ary tree algorithm with $N = 50$ active users and varying feedback times. Case 1, Feedback time $\Delta = 50\mu$sec (5 km), $\Delta = 160\mu$sec (16 km), $\Delta = 400\mu$sec (40 km), $\Delta = 800\mu$sec (80km), $\Delta = 2000\mu$sec and $\Delta = 6400\mu$sec. Magnified part of previous figure.

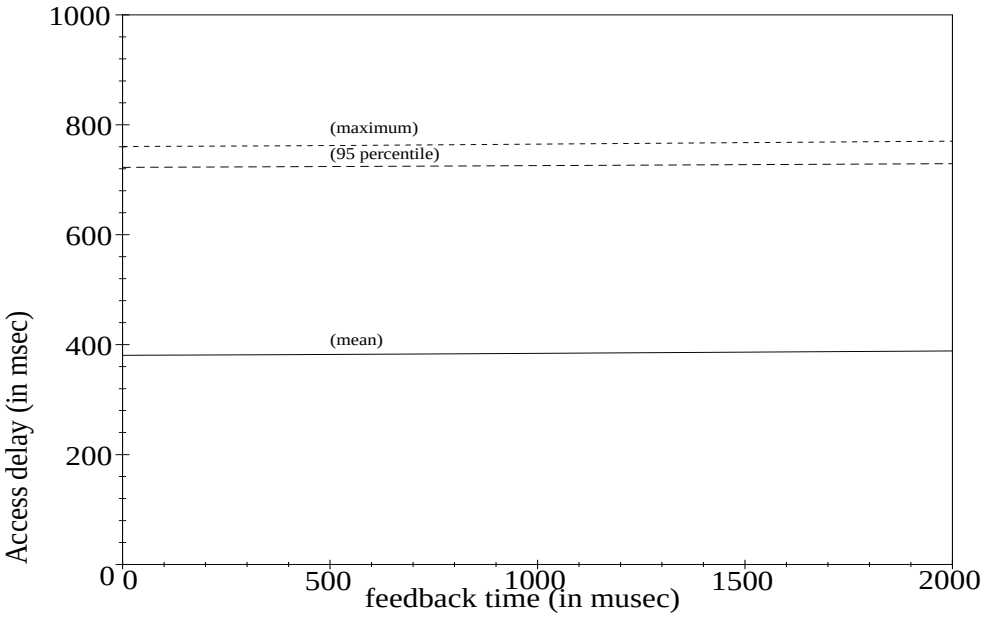

Figure 21: Burst response of the parallelized 4-ary tree algorithm with 2000 colliders versus feedback time. Average transmission delay of 2000 nodes (plain), 95 percentile of transmission delay dashed), maximum transmission delay (dotted).

3.4 Burst response

In the burst scenario one forces the resolution of an initial collision involving exactly 2,000 nodes, each of them with a single pending packet of 64 bytes. It is not necessary to introduce randomness in the date of first request transmission, since the initial collision introduces a negligible cost (see figure 22). Furthermore the simulations are repeated with increasing feedback times, between 0μsec and 2000μsec, in order to check again the stability of the protocol against propagation delay.

Figure 21 shows the distribution of transmission delays after the collision of multiplicity 2,000: the transmission delay averaged over the 2,000 colliders, the 95 percentile transmission delay (time needed to transmit 95% of 2,000 packets, 1800 packets), and the total time needed to transmit the 2,000 packet.

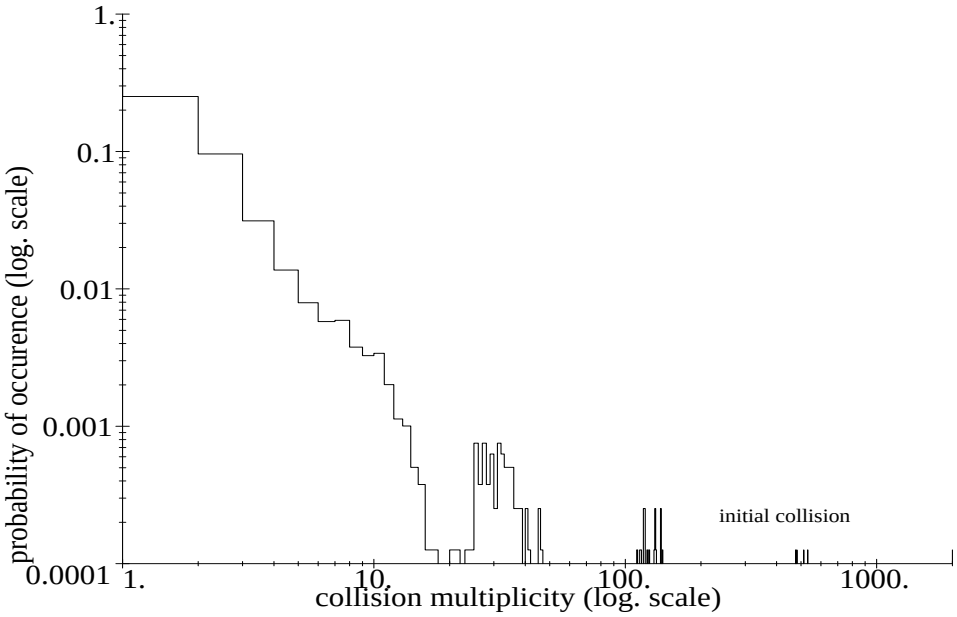

Figure 22: Collision multiplicities in a burst response of the parallelized 4-ary tree algorithm with 2000 colliders.

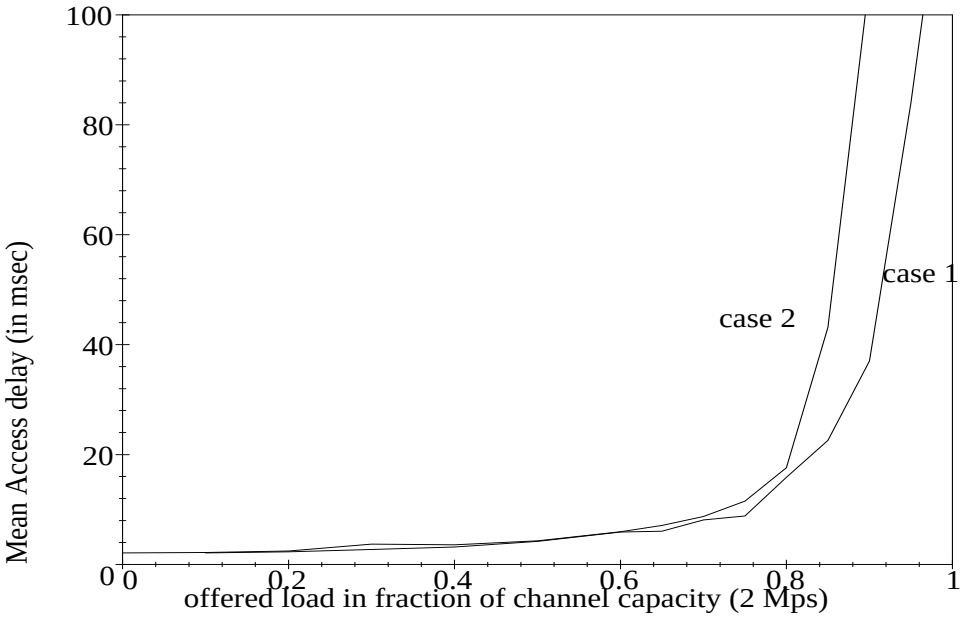

Figure 23: Mean access delay versus offered load, with parallelized 4-ary tree algorithm with $N = 50$ active users. Case 1 and Case 2, Feedback time $\Delta = 160\mu$sec (16 km).

Figure 22 shows an histogram of collision multiplicities on the collision slot during the resolution of the initial collision of size 2,000. Notice that the initial collision is also recorded in the histogram.

3.5 Overhead cost and node speed

The contention slot is the main source of overhead in the protocol. The simulation can be run with various contention slot sizes: for example, case 2 is with a contention slot size of 128 bits. Figure 23 shows the mean access delays of case 1 and case 2.

Simulations can also take into account the node speed when moving packets in its local memory. There is a parameter called minimum latency time R which is the minimum time between the successful transmission of a packet and the availability of the next packet in the local queue. In figure 24 shows

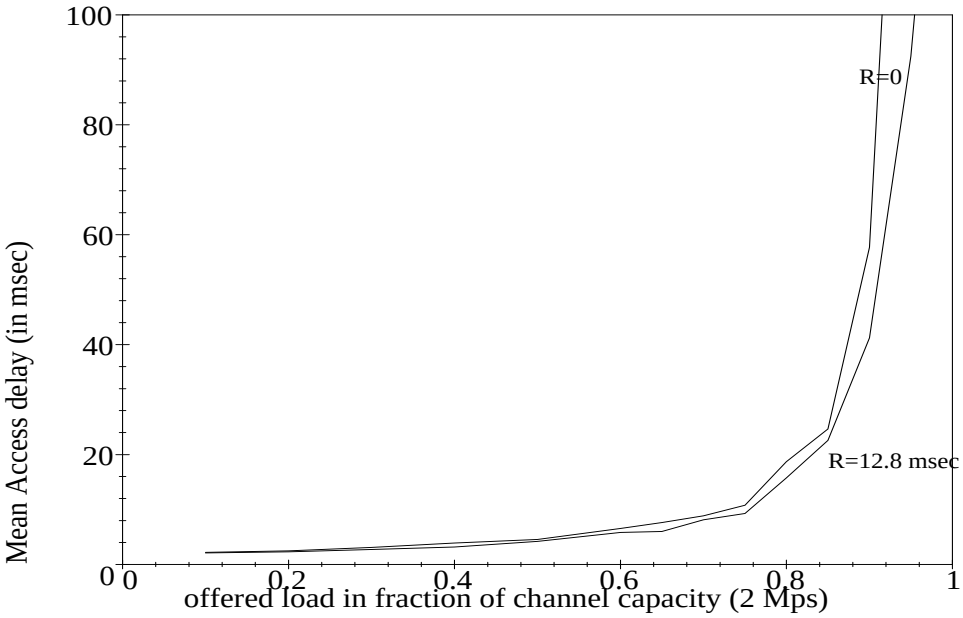

Figure 24: Mean access delay versus offered load, with parallelized 4-ary tree algorithm with $N = 50$ active users. Case 1. Minimum latency time R between two local consecutive packets is respectively equal to 0 and 12.8 msec.

the mean access delay versus offered load respectively for $R = 0$ and $R = 12.8$ msec. Since the tree algorithm leads to negligible queueing in nodes, the parameter R has very little impact on performance. The parameter R should be of the order of the period between two packets generation in order to have noticeable impact. In other word any optimization based on concatenation of requests would not be useful at this stage.

4 Conclusion

In this paper, we have seen that the tree algorithm can be efficiently used as a collision resolution scheme in networks where the feedback delays are large. We have also shown that it is more simple and more efficient to mix the numerous collision resolution process which are generated by the large feedback delays. We have given detailed performance analysis of these access schemes and shown

that if we use an additional reservation scheme, the tree algorithm can provide very good performances as an access protocol to CATV network. We have also shown that these performances are not significantly altered by a large number of active nodes or by by very large feedback delays.The salient improvements brought by parallelized tree algorithms stand in the specific insensitivity to the variation in the number of node and in the low reactivity to feedback time increases (distance to head-end).

References

[1] J. Capetanakis, "Generalized TDMA: The multi-accessing tree protocol," *IEEE Trans. Commun.*, Vol. COM-27, No. 10, pp. 1476-1484, 1979.

[2] J. L. Massey, "Collision Resolution Algorithms and Random-Access Communications", in *Multi-User Communication Systems*, G. Longo Editor, *CISM Courses and Lectures no.* **255**, Springer Verlag, Wien-New York, 1981, 73-137.

[3] B. S. Tsybakov, V. A. Mikhailov, "Free Synchronous Packet Access in a Broadcast Channel with Feedback", *Probl. Inform. Transmission* **14**, 1979, 259-280..

[4] P. Mathys, P. Flajolet, "Q-ary collision resolution algorithms in random-access systems with or blocked channel access," in *IEEE Trans. on Information Theory*, vol IT-31, pp 217-243, 1985.

[5] G. Fayolle, P. Flajolet, M. Hofri, P. Jacquet, "Analysis of a Stack Algorithm for random multiple-access communication," *IEEE Trans. Inform. Theory*, vol IT-31, pp. 244-254, 1985.

[6] L. Merakos, C. Bisdikian, "Delay analysis of the n-ary stack random-access algorithm," in *IEEE Trans. on Information Theory*, vol IT-34, pp 931-942, 1988.

[7] P. Jacquet, P. Mühlethaler, "Minimac : a high speed access protocol based on a free access tree algorithm," INRIA RR-0849, 22 p., 1988.

[8] P. Jacquet, E. Merle, "Analysis of a stack algorithm for CSMA-CD random length packet communication," *IEEE Trans. on Inform. Theory*, vol IT-36, pp. 420-425, 1990.

[9] P. Jacquet, P. Mühlethaler, "MACHNET: a simple access protocol for high speed or long haul communications," *Second IEEE Workshop on Future Trends of Distributed Computing Systems*, pp. 197-206, 1990.

[10] P. Jacquet, P. Mühlethaler, P. Robert, "A slotted multiple access scheme designed for CATV netwok," INRIA Research report to appear 2001.

[11] P. Mühlethaler "Protocoles d'accès pour réseaux à haut débit". Novembre 1989. Thèse Paris Dauphine.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399