Stack Algorithms in Implicit Framing, Free Access and Blocked Access for CATV Networks

Cédric Adjih, Philippe Jacquet, Paul Mühlethaler

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

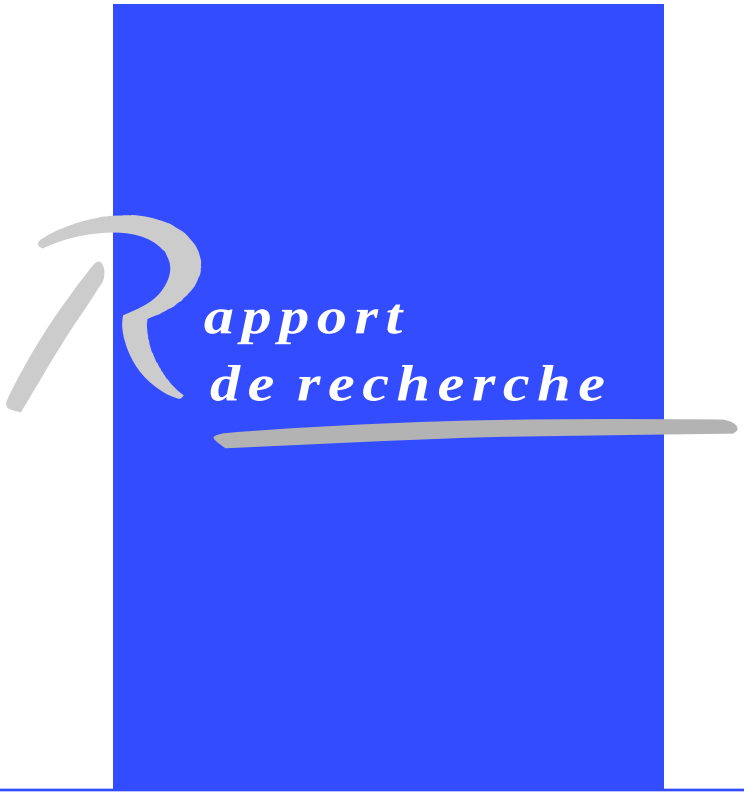
Stack Algorithms in Implicit Framing, Free Access and Blocked Access for CATV Networks

Cédric Adjih , Philippe Jacquet , Paul Mühlethaler

No 4129

Mars 2001

THÈME 1



ISSN 0249-6399 ISRN INRIA/RR--4129--FR+ENG

Stack Algorithms in Implicit Framing, Free Access and Blocked Access for CATV Networks

Cédric Adjih , Philippe Jacquet , Paul Mühlethaler

Thème 1 — Réseaux et systèmes
Projet HIPERCOM

Rapport de recherche n ° 4129 — Mars 2001 — 19 pages

Abstract: In this paper we present in the framework of access protocols for Cable TV (CATV) networks performant and easy implementation of stack algorithms with an implicit framing scheme. We use a stack algorithm with a single interleaved process in a free or blocked access mode. We show that the blocked access can be implemented to have a deterministic behaviour. We show that a dynamic tuning of the access persistence parameter R improves the performance both in terms of throughput and delays.

Key-words: Cable TV network (CATV network), upstream channel, framing, stack algorithm, free access, blocked access, performance evaluation, access delay.

(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

Algorithmes en Arbre à Accès Libre et Bloqué pour Réseaux Câblés de Télévision

Résumé : Ce papier propose dans le cadre de la conception de protcoles d'accès pour réseaux câblés de télévision des implémentations simples et performantes d'un algorithme en pile avec un "framing" implicite. Nous utilisons un algorithme en pile qui mélange les résolutions de collision en un seule "session", cet algorithme peut être à arrivée bloquée ou à arrivée libre. Nous montrons que l'accès bloqué peut avoir un comportement déterministe. Nous montrons également que l'ajustement dynamique du paramètre de persistence d'accès R peut améliorer les performances à la fois en terme de débit et de délais.

Mots-clé : Réseau câblé de télévision (CATV network), canal montant, "framing", algorithme en pile, accès libre, accès bloqué, évaluation de performance, délais d'accès.

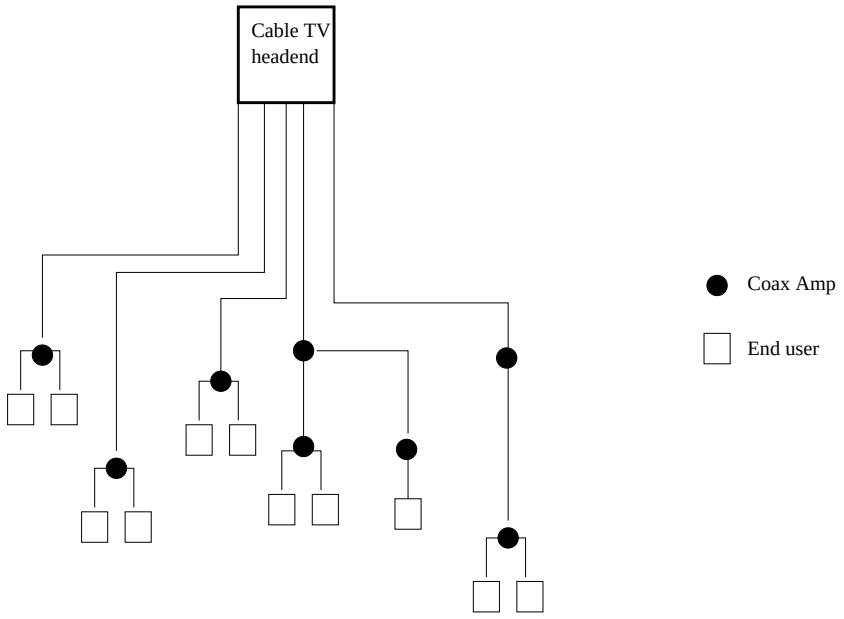


Figure 1: General architecture for CATV.

1 Introduction

A CATV network physically looks like a tree made of coaxial cables and usually covers an area of several blocks of houses. The end-users are connected to the external nodes of the tree. For transmission of interactive data CATV networks have usually two types of channels: upstream channels and downstream channel. Upstream channels are used to send data from an end user to the headend as downstream channels are used to send data from the headend to an end user. The distribution system relies on the head-end which is located at the root of the tree. See figure 1.

The upstream channel is obviously a shared channel since all the end-users may use it to send data. Of course the data is successfully sent if only one end-user transmits at the same time (this "time" takes into account propagation delay) otherwise it is a collision. Therefore an efficient access protocol is needed to share this bandwidth. The design of access channel scheme for CATV upstream channel is discussed in [1] and further background for CATV networks

is given. In particular ranging and synchronization schemes are discussed. In [1] we use a tree or stack algorithm to rule the contention access of CATV upstream channel. In [2] further work on implementation and performance analysis of tree algorithm in the framework of CATV network is presented. Sharing a CATV upstream channel requires in addition to a random access scheme a reservation scheme. The sharing between the random access scheme and the reservation scheme is the result of a framing protocol. In [3] we discuss different framing protocols. Except for experts in random access scheme references [1, 2, 3] will be necessary to fully understand the work presented in this paper. In this paper we use the single interleaved stack algorithm in free and blocked access which is described in [2] and we will consider an implicit framing [3]. We show that the blocked access can be implemented to have a deterministic behaviour. We also show that dynamic tuning of persistence access parameter R improves the performance both in term of throughput and delays. This paper is organized as it follows. Section 2 gives a short remainder for stack free and blocked access algorithms and provides two important properties of these algorithms. Section 3 reports simulation results for stack algorithms both for free access or blocked access. In the second part of section 3 we study how the dynamic tuning of persistence access parameter R improves the performance in terms of throughput and delays.

This paper has been proposed as a contribution to the IEEE 802.14 commitee.
[1]

2 Stack algorithms

We focus on single interleave stack algorithms, with the following general description:

- *A0*) the head-end maintains a virtual stack of contention groups and a virtual stack of successful requests. Each group contains end-users which have collided together in their last transmission attempt.

[1] Actually a comon time can be defined if the nodes are synchronized [1]

- *A1*) the algorithm receives feedback: collisions, and successul requests. It pushes corresponding contention groups and successful requests stored in the stacks.
- *A2*) the algorithm allocates the set of slots, using the requests or contentions groups from the stacks:
  - *B1*) if it is request, then data minislots are allocated
  - *B2*) if it is a contention group, then contention minislots are granted with an identifier of the dequeued group.
  - *B3*) on some occasions contention minislots are granted for newcomers, with a default (zero) group identifier. Such a new group will be called a *root group*.

Many different policies can be implemented within this framework:

Stack management	LIFO	FIFO
Data versus contention slots	priority to data	stacks merged
Newcomer access	free access	blocked access
Newcomers persistence	1-persistence	using persistence R

We have implemented two types of stack management:

- a conventional Last-In-First-Out stack algorithm : LIFO contention group stack, LIFO data stack, priority to data.
- a FIFO stack algorithm: data and contention group stacks merged, FIFO stack.

A more detailed description of the implementation of the FIFO algorithm is given in [4]. For these two algorithms, simulations were done, experimenting with free access versus blocked access, and persistence versus non-persistence (1-persistence).

We first present noticeable properties of the algorithms in the next section, and after these, results of the simulations.

2.1 Properties of single interleave stack algorithm

In this section we show important properties of single interleave algorithms, namely a bound on the number of parallel collision resolution trees, and the possibility of implementing deterministic blocked algorithms.

2.1.1 Bound on the number of parallel collision resolution trees

In this section we investigate a property which is common to every single interleave stack policy, whatever it is free access or blocked access. This property is interesting because it shows that single interleave stack algorithms in implicit framing automatically self stabilize with implicit and efficient equivalent of several interleaves which provide an optimal utilization of the bandwidth.

Collision Resolution Trees (CRT). Except for root groups, each group called by the HE, *i.e.* queued in the virtual contention stack comes from the ternary split of an older group which have given rise to a collision. This is true even in free access since the groups augmented of newcomers are still originated from the ternary split of an older group. Therefore we can formally backward link the groups between them in order to obtain *trees* of groups, each tree terminating on a single root group. We call such structure a Collision Resolution Tree (CRT). A new CRT is open each time a root group is called by the HE. A CRT is closed when its last group has been called and has given "No Collision" feedback. A CRT which has not yet been closed is said to be in processing.

With the single interleave stack algorithm, the CRTs play the role of "implicit" interleaves since several CRT can simultaneously be in processing. We defer to appendix the proof of the following important property:

Property 1 *Let D be the maximum delay between grant and feedback for minislots, In the implicit framing there are at most D parallel Collision Resolution Trees simultaneously in processing*

2.2 The blocked access, deterministic implementation

We assume here that a blocked access stack algorithm is used, with any type of contention/data stacks (LIFO, FIFO, merged, *etc.*).

In blocked access, new comers never interfere with currently queued groups. Thus an end user with a request never tried before will transmit it for the first time in a root group. In the simulations within we assume that such end user transmits in the next available root group.

Since each group contains end-user which have collided together for the same number of times, it is possible to implement a *deterministic* version of blocked access. For this special case we assume that there are N connected end-user each of them having a ternary key of length K distinct from the keys of the other users. Clearly $K \approx \frac{\log N}{\log 3}$. With this hypothesis it is impossible to have two or more end-user having successively collided more than K times. Furthermore the number of groups (or minislots) needed to solve an initial collision, *i.e.* the number of nodes in the corresponding CRT, cannot exceed $\frac{3^{K+1}-1}{2}$.

Remark: Minislots can be interleaved with data slots, and we have to consider both slot types in order to derive bounds on actual access delays. Furthermore the contention stack can be exhausted ("empty") even with still open CRTs since all the remaining groups may be waiting for transmission feedback. It means that new comers could start a new CRT before any former CRT be completely resolved.

We show below a very interesting property which states that deterministic single interleave blocked access stack algorithms provide bounded access delays.

Property 2 *Let L be the maximum data length reserved per single request (including concatenation and piggybacked extensions) and D be the propagation delay, both expressed in minislot unit. The maximum access delay with deterministic single interleave blocked access stack algorithm in implicit framing is $D(K+1) \times (\frac{3^{K+1}+1}{2} + N \times L)$ minislots.*

If we assume that $K = \lceil \frac{\log N}{\log 3} \rceil$, then the maximum delay becomes: $D(\lceil \frac{\log N}{\log 3} \rceil + 1) \times (\frac{9}{2}N + NL)$

Proof: We first to prove that $D \times (3^{K+1} - 1 + NL)$ is the maximum delay between two consecutive occurrence of the event "stack empty". To this end we refer to property 1 which states that there are at most D CRTs in simultaneous processing at any time. Since each of its CRT cannot contains more than

$3^{K+1} - 1$ groups and cannot give rise to more than NL data slots, the stack will necessary be exhausted before $D \times (3^{K+1} - 1 + NL)$ minislot time units.

This bound is first the maximum time for a random end-user with a new request to find the next root group to transmit, *i.e* when the stack will return to empty state. Second, when added of D minislots it is also the maximum time between two consecutive retransmissions of the same request.

Since there are a maximum of K retransmissions, adding this maximum delay, we obtain the expression of the maximum access delay.

3 Simulations results

3.1 Simulations framework

The simulations use the parameters of table 1.

Simulations parameter	Values
Number of active stations	200
Distance from stations to headend	200 km (ranging)
Downstream data transmission rate	infinite
Upstream data transmission rates	3 Mbits/s
Propagation delay	5 μs/km
Length of simulation	30 seconds
Guardband and pre-amble	duration of 5 bytes
Minislot size	16 bytes
Headend processing delay	0 ms
Headend processing scheme	feedback and allocation at each minislot

Table 1: Network Simulation Configuration Parameters

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.

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

Additionaly packets are send in ATM cells (4 minislots for a 48 bytes payload), and neither piggy-backing nor data concatenation are used.

3.2 Results for free and blocked access

The results providind the mean access delay versus the channel load are given in figure 2 and 3.

The results show that the performance of different implicit framing with 1-persistence are close to each other. For delay and standard deviation, only one algorithm, free entry FIFO stack, that performs slightly better than the others. Two reasons explain this better performance: first the fact that free entry algorithms should lower collision multiplicities than blocked algorithms (with no persistence), second the fact that the simulated LIFO algorithms give priority to data, which may delays collision resolution.

3.3 Influence of the entry persistence R for free and blocked access

In our implementation, we used a simple scheme for R:

- R is initialized to zero
- each time a contention minislot is allocated for a root group R is decreased.
- each time data minislots for one request are allocated, R is increased.

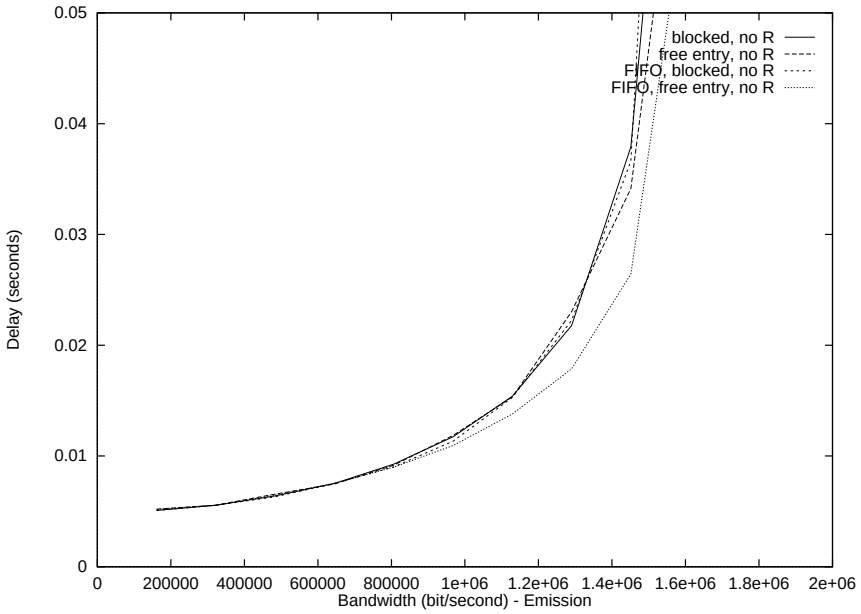


Figure 2: Average delay versus input load.

The rationale is that for each successful request there should roughly be one contention minislot. Figure 4 gives an idea of how R evolves in the minislot stream.

Notice that within this scheme, as shown on the above example, if one station receives a new packet during the long stream of data minislots (from $R = 1$ to $R = n$), then it will make a request on one of the n following contention minislots (from the one marked $R = n$ to the first marked $R = 1$) with uniform probability $1/n$: allocating contention minislot one-by-one (implicit framing), is this consistent with allocating contention minislot by blocks (here n), with a simple value.

The different stack implementations perform still closely with entry persistence; the difference is now that the blocked versions are more efficient than free versions. For a direct comparison of the consequences of implementations

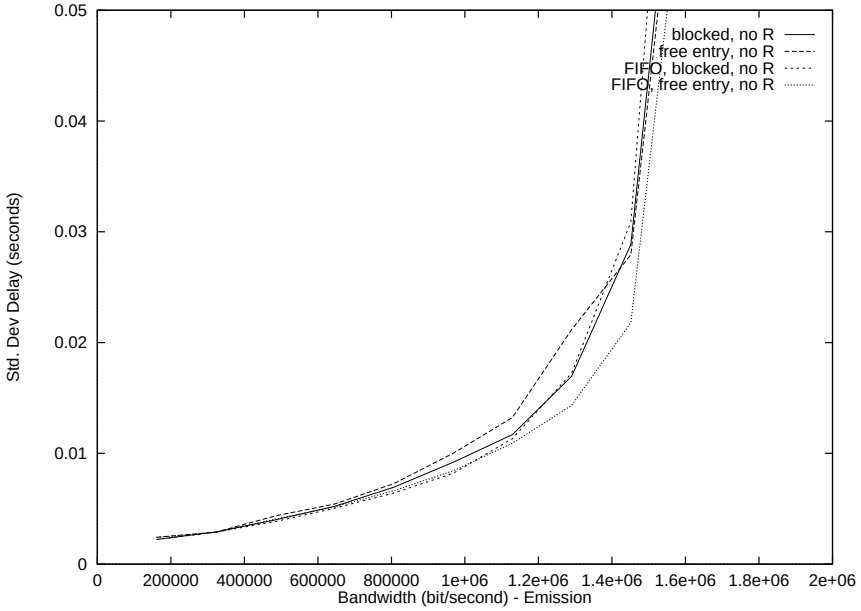


Figure 3: Standard deviation of the delay versus input load.

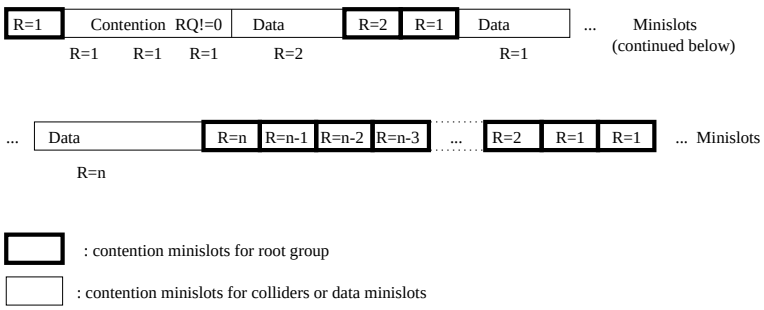


Figure 4: Evolution of R with minislot stream.

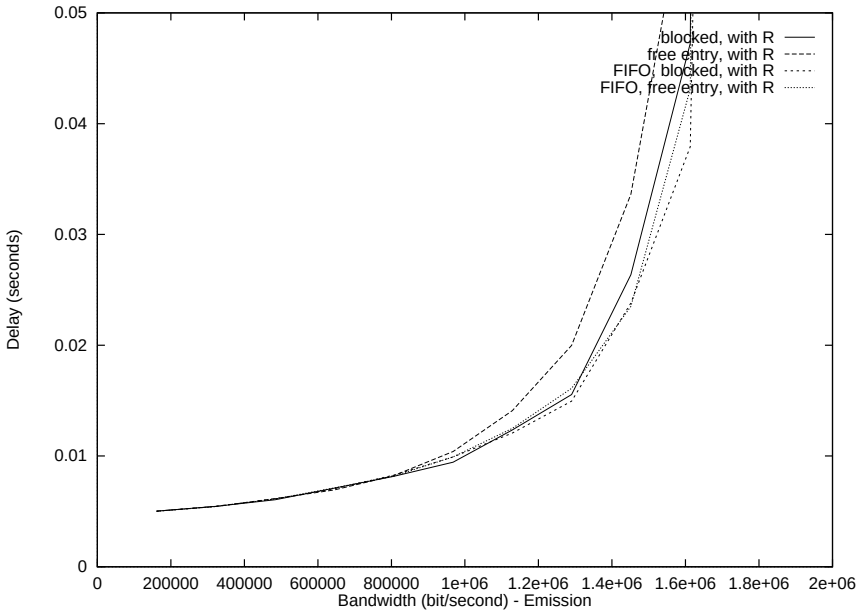


Figure 5: Average delay versus input load with persistence.

with and without entry-persistence, both versions are displayed on the same figure, see figures 7 and 8.

Free entry does not benefit much from the persistence entry; obviously, implicit framing acts like a a primitive but efficient persistence. Blocked versions are more sensitive to entry persistence: for instance, if we consider the bandwidth for a mean delay of 0.05, we see that about 10% of bandwidth is gained for FIFO blocked algorithm.

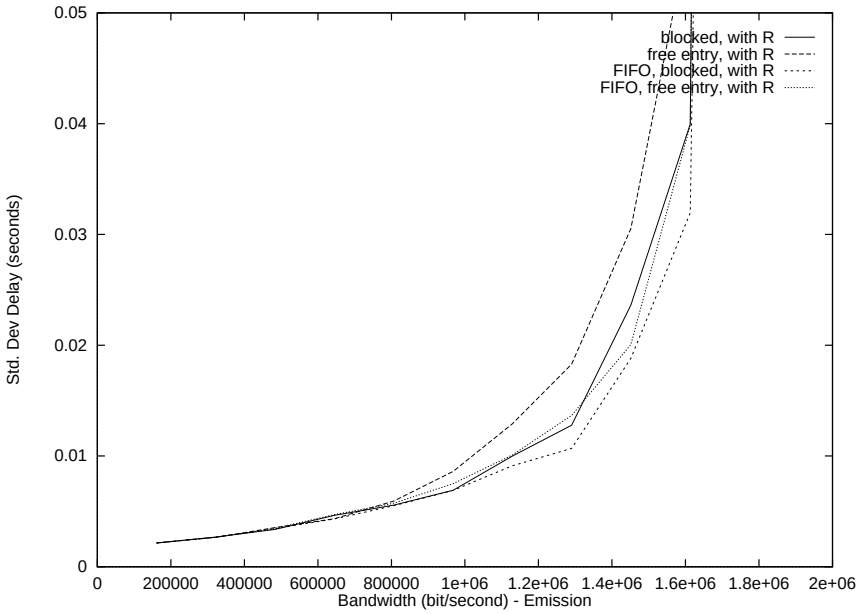


Figure 6: Standard deviation of the delay versus input load with persistence.

4 Conclusion

In this paper, the performance of several implicit framing stack algorithms was investigated ; the different algorithms are found to be generally of comparable performance. A simple computation of entry persistence range R improves access delay even with implicit framing. The use of free access instead of blocked seems to be an efficient way to get some delay improvement without persistence, but some reserves must be done in term of fairness [4]. We believe that the existence of different performant algorithms is important since it would give choices for implementation with regard to other specific metrics.

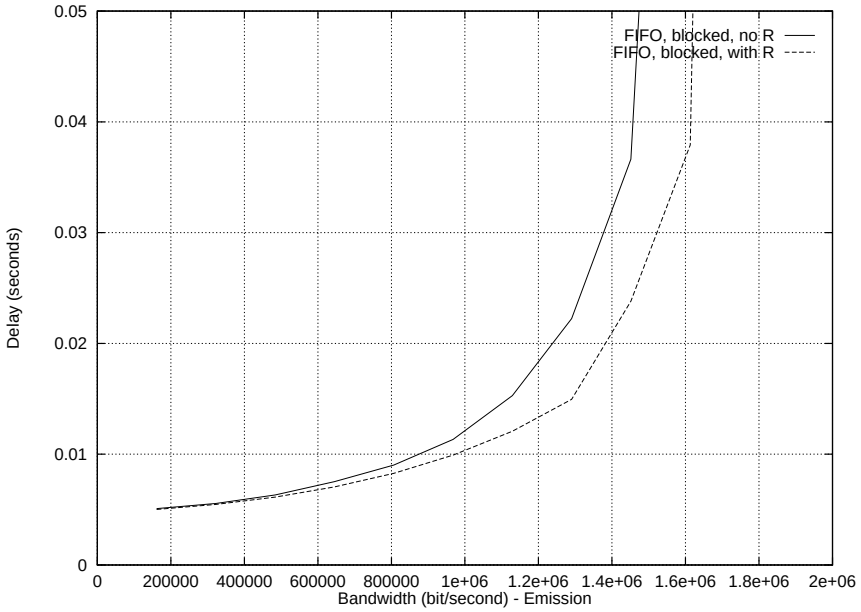


Figure 7: Average delay for the blocked access FIFO algorithm with or whitout persistence versus input load .

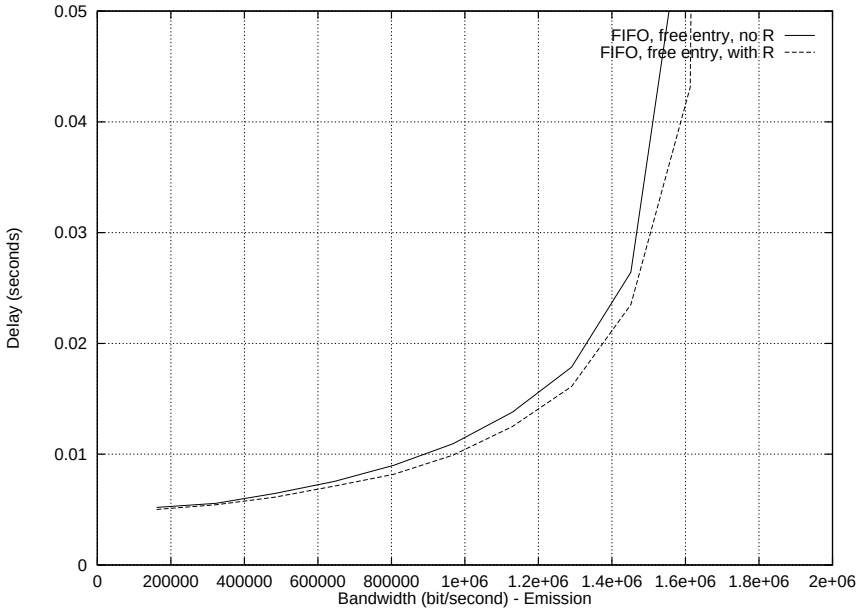


Figure 8: Average delay for the free access FIFO algorithm with or whitout persistence versus input load .

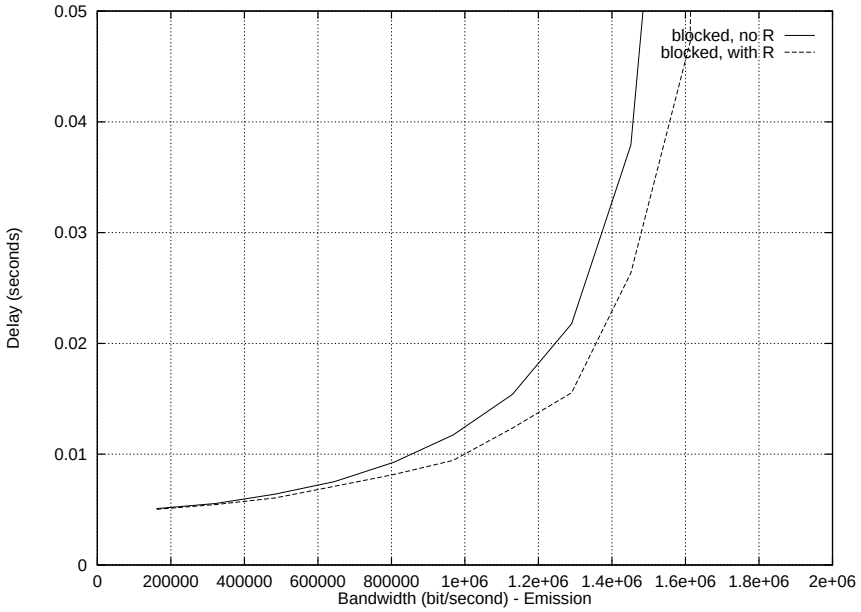


Figure 9: Average delay for the blocked access LIFO algorithm with or whitout persistence versus input load .

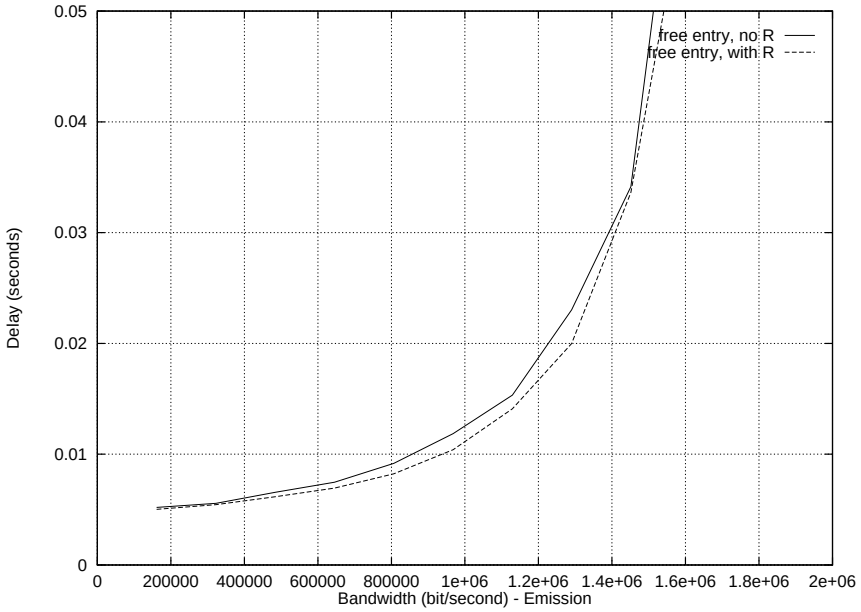


Figure 10: Average delay for the free access LIFO algorithm with or whitout persistence versus input load .

References

[1] Philippe Jacquet, Paul Mühlethaler and Philippe Robert "A slotted Medium Access Control Scheme designed for CATV Network" INRIA Research Report 4106, 17 p., January 2001.

[2] Philippe Jacquet, Paul Mühlethaler and Philippe Robert "Performant implementations of tree collision resolution algorithms for CATV networks" INRIA Research Report 4107,34 p., February 2001.

[3] P. Jacquet, P. Mühlethaler, P. Robert, "Framing protocols on upstream channel in CATV networks: asymptotic delay analysis," INRIA Research Report 22 p., February 2001 to appear.

[4] C. Adjih, P. Jacquet, P. Mühlethaler, "Stack Algorithms at high loads: analysis of unfairness or singular behaviours", INRIA Research Report , 12 p., March 2001 to appear.

A Proof of Property 1

Let us consider the evolution of the stack between two time t_1 and t_2 supposing that the stack is never empty during the interval $[t_1, t_2]$. We denote stack(t) the value of the stack at time t. Let G be the number of granted slot during the interval of observation. Let F the number of feedbacked minislots during the interval and C and NC be the respective number of collision and non-collision among those minislots ($F = C+NC$). We have stack$(t_2) =$ stack$(t_1) - G + 3C$. Therefore stack$(t_2) =$ stack$(t_1) + 2C - NC + F - G$.

If we start from an empty system at $t = t_1$, then we have stack$(t_2) = 2C - NC + G - F$. Each time stack$(t_2) = 0$ a new Collision Resolution Tree (CRT) is open. By now we cancels the condition stack$(t) > 0$ for all $t \in [t_1, t_2]$. If S_o is the number of the CRT opened during the interval $[t_1, t_2]$, then we have stack$(t_2) = S_o + 2C - NC + F - G$.

When a group that a group has been called by the HE and the feedback of its transmission has been received by the end-user, then we say that the group has been seen. It is a property of ternary trees that a CRT terminates when the number of its seen non collision groups equals twice plus one unit the number of its seen collision groups. As long a CRT is not closed the number of its seen collision groups multiplied by two is always greater than or equal to the number of if its seen non collision groups. Therefore, if S_c is the number of closed CRT before time t_2 (starting from an empty system), then $S_c \geq NC - 2C$.

From the above inequality together with the identity stack$(t_2) = S_o + 2C - NC + F - G$, we obtain the fact that the number of CRTs in processing at time t_2, $S_o - S_c$ is smaller than or equal to $G - F$. Since $G \leq F + D$, Property 1.

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