# A Protocol and Correctness Proofs for Real-Time High-Performance Broadcast Networks

Jean-François Hermant[*]

LIX, École Polytechnique
F-91128 Palaiseau Cedex, France

hermant@lix.polytechnique.fr

Gérard Le Lann

INRIA, Projet Reflecs, BP 105
F-78153 Le Chesnay Cedex, France

Gerard.Le_Lann@inria.fr

**Abstract**

*Novel real-time applications require high-performance real-time distributed systems, and therefore high-performance real-time networks. We examine a Hard Real-Time Distributed Multiaccess problem which arises with such application problems. We present a solution, based on broadcast LANs or busses, such as Gigabit Ethernets or busses internal to ATM nodes, associated with a deterministic Ethernet-like protocol called CSMA/Deadline Driven Collision Resolution. We give an analysis of balanced $m$-ary tree algorithms which are used by CSMA/DDCR, and derive feasibility conditions for the HRTDM problem.*

## 1 Introduction

Novel real-time applications require high-performance real-time distributed systems, and therefore high-performance real-time networks. We consider a particular class of such networks, those based on broadcast media. Examples of such media are physically dispersed local area networks (LANs), such as Gigabit Ethernets, and busses internal to switch fabrics, such as busses internal to ATM nodes.

Assuming that an application problem can be unambiguously and completely specified, how can we tell whether a COTS (commercial off-the-shelf) product is or is not a "solution" for that problem? Under current practice, decisions to include specific COTS products - such as ATM switches or Gigabit Ethernets in our case - in installed systems are made without any proof of "correctness". More generally, lack of correctness proofs vis-à-vis system design and/or physical dimensioning decisions seems to be a major cause of project setbacks or operational failures [1, 2].

Proof-based system engineering (SE) methods serve to eliminate such "fuzzy" decisions. The basics of a proof-based SE method can be found in [3, 4]. The generic problem considered in this paper, referred to as the Hard Real-Time Distributed Multiaccess (HRTDM) problem, is specified following this proof-based SE method (section 2).

We then present a solution based on a protocol, called CSMA/DDCR (which stands for Carrier Sense Multi Access/Deadline Driven Collision Resolution), which belongs to the class of balanced $m$-ary tree protocols (section 3).

Any provably correct or optimal solution to the HRTDM problem that would be based on balanced $m$-ary tree protocols rests on establishing exact solutions for the following subproblems:

- $P_1$: express a computable tight upper bound on worst-case deterministic searches for a balanced $m$-ary tree,
- $P_2$: express a computable tight upper bound on worst-case deterministic searches over multiple consecutive balanced $m$-ary trees.

We present an exact solution for $P_1$, and an exact asymptotic solution for $P_2$ (section 4). With such solutions at hand, one can establish provably correct and computable conditions under which any user-defined instantiation of HRTDM is feasible with CSMA/DDCR.

Finally, we discuss why the CSMA/DDCR protocol is directly applicable to two emerging technologies of particular significance, namely ATM switches and Gigabit Ethernets (section 5).

## 2 The HRTDM problem

### 2.1 What is proof-based system engineering

The essential goal pursued with proof-based SE is as follows: starting from some initial description of an application problem (end users/customers requirements and assumptions), to produce a global modular implementable specification - denoted $[S]$ in the

[*]This work has been conducted while J.-F. Hermant was with INRIA's Projet Reflecs.

sequel - of a computing system $S$, along with proofs that system design and system dimensioning decisions made to arrive at specification $[S]$ do satisfy $<X>$, the specification of the computer science problem "hidden" within the application problem. With proofs, it is possible to ensure that future behavior of $S$ is the desired behavior, before implementation or fielding is undertaken. Any module of $[S]$ is a specification of a physical module of $S$, which may end up being a COTS product or a customized/proprietary product.

Novel applications that have combined high-performance, real-time distributed requirements will be deployed in very diverse environments. Future run-time assumptions should then be as general as possible. Furthermore, real world customers express descriptions of such requirements and assumptions that usually are incomplete and ambiguous. How can then a designer, a researcher, a provider of some COTS technology, claim that he/she has a "solution"? Such claims can be substantiated by resorting to proof-based SE, one of the emerging disciplines of interest to the IEEE Computer Society and to INCOSE [5, 6].

A proof-based SE method - the TRDF method - can be used to rigorously bridge the gap between applications and COTS products [3, 4]. That method has been applied to a number of real world problems. Examples are Modular Avionics (French DARPA and Dassault Aviation), safety related control operations in Nuclear Power Plants (Institut de Protection et de Sûreté Nucléaire, French Atomic Energy Authority), the analysis of the Ariane 5 Flight 501 failure [7]. The TRDF method is currently being applied to Air Traffic Control and critical On-Line Transactional applications.

The term "specification" refers to any complete and unambiguous statement - in some human language, in some formalized notation, in some formal language. $<X>$ is a pair $\{< p.X >, < m.X >\}$, where $p.X$ is a set of properties (requirements) and $m.X$ is a set of models (assumptions). Properties as well as models have well defined semantics, such as those commonly considered in computer science [8, 9, 10].

Examples of high-performance real-time applications considered in this paper are distributed interactive multimedia, videoconferencing, on-line transactions (e.g., stock markets), surveillance (e.g., air traffic control).

Space limitations prohibit us from showing how to apply the TRDF method so as to, first, rigorously specify any such application and, second, derive from it the specification of a subproblem, which is the HRTDM problem considered in this paper.

### 2.2 The HRTDM problem

Transit times of messages flowing between an application task and a network module are inevitably variable. This is due, in particular, to software and hardware layers sitting in between an application layer on the one hand and a network layer on the other hand (e.g., calls to operating systems, use of diverse scheduling policies for servicing waiting queues, etc.). Consequently, even for those messages generated by application tasks that are activated periodically, one must abandon the idea that messages are submitted periodically to a network module. Hence, we consider unimodal arbitrary models.

With the unimodal arbitrary arrival model, an upper bound $a(msg)/w(msg)$ is defined for the arrival density of every message $msg$, with $w(msg)$ being the size of a sliding time window and $a(msg)$ being the highest number of message $msg$ arrivals that can occur within $w(msg)$ time units. The "adversary" embodied in these models is stronger than, e.g., the periodic arrival model.

Many researchers simply assume periodic arrival models (deterministic approaches) or Poisson arrival models (stochastic approaches) all the way through a design tree. This simplifies design work as well as the establishment of feasibility conditions. However, this does not reflect reality and, most often, leads to incorrect (i.e., arbitrarily optimistic) feasibility conditions. Recent research work in this area - see, e.g., [11, 12, 13, 14] for examples - looks promising. Furthermore, there are no restrictions on the relationships that may exist between deadlines and arrival densities.

The HRTDM problem is specified as follows:

<m.HRTDM>:

- Message sources model: set of $z$ sources, denoted $s_i$, $i \in [1, z]$; number of sources is unrestricted.
- Message model: set of messages $MSG$, each message $msg$ of some bit length $l(msg)$; $|MSG|$ and the $l(msg)$'s are unrestricted.
- Mapping model: subset $MSG_k$ of $MSG$ mapped onto source $s_k$ ($MSG$ is partitioned into $z$ subsets); mapping is unrestricted.
- Message arrival model: unimodal arbitrary arrival law for every message, with $a(msg)/w(msg)$ being the upper bound on $msg$'s arrival density.

<p.HRTDM>:

- Safety: successful transmissions of messages over a broadcast medium must be mutually exclusive.

- Timeliness (real-time): message timeliness constraint = strict constant latest deadline for completing transmission, denoted $d(msg)$ for message $msg$.

Such system-wide requirements correspond to what is called end-to-end QoS (quality-of-service) and CoS (class-of-service) in the Telecommunication/Networking community.

To our knowledge, problem <HRTDM> still is open. It is indeed the case that: (i) there is no known algorithmic solution - associated with some architectural solution - that has been proven optimal, (ii) there is no known expression of feasibility conditions (FCs) - established for a provably optimal algorithmic solution - that have been proven necessary and sufficient.

In section 4, we present two problems $P_1$ and $P_2$ that need be solved exactly in order to establish necessary and sufficient FCs for <HRTDM>, given CSMA/DDCR. FCs are an essential tool for an end user or a technology provider who has to assign numerical values to message lengths, to upper bounds of message arrival densities and to message deadlines. By computing the FCs, it is possible to tell whether or not any quantified instantiation of the HRTDM problem is feasible with our solution.

## 3 A Solution: Broadcast Media, Local EDF and the CSMA/DDCR Protocol

### 3.1 Rationale for a solution

<HRTDM> is a design subproblem. No existing COTS product satisfies <HRTDM>. Recall that "having a solution for HRTDM" implies - at least - delivering some FCs (established for some solution that one may want or may not want to reveal). We are not aware of any COTS product that would be accompanied with computable FCs for <HRTDM>.

Our strategy consists in "adding value" to such products. We explicitly consider ATM switches and Gigabit Ethernets. We develop a design solution - a module of [$S$] - which includes a broadcast medium (many such media can be used in parallel), a local scheduling algorithm, and a distributed contention multiaccess protocol.

There are two main reasons why we are interested in broadcast media, especially in passive broadcast media such as busses internal to ATM switches and/or those used with Gigabit Ethernets. First, they are commonplace, and cheap. Second, they can be shared via protocols that have interesting fault-tolerant properties.

There are two main reasons why we are interested in contention-prone multiaccess protocols. One is that theoretical work on distributed multiaccess protocols for broadcast media has established that tree protocols - a particular class of contention-prone protocols - achieve channel utilization ratios that are very close to theoretical upper bounds [15, 16, 17, 18, 19]. The other reason is that the dominant LAN technology (Ethernet) happens to be based on a contention-prone multiaccess protocol, i.e., the well known CSMA-CD protocol. We believe that this dominance is not going to weaken any time soon.

Consequently, we have developed a protocol called CSMA/DDCR - Carrier Sense Multi Access/Deadline Driven Collision Resolution - which is based on deterministic balanced $m$-ary tree searches. The reason why we have chosen to develop that particular deterministic variant of the famous original Ethernet CSMA-CD protocol is that CSMA/DDCR emulates a distributed Non-Preemptive Earliest-Deadline-First (NP-EDF) scheduling algorithm. It has been shown [20, 21] that centralized NP-EDF is an optimal solution for problems equivalent to the centralized variant of <HRTDM>, considering periodic or sporadic message arrival models.

### 3.2 The solution

For the sake of conciseness, we will only give an informal description of the solution, which comprises two scheduling algorithms (local EDF, CSMA/DDCR). In the sequel, we present and analyze the CSMA/DDCR protocol for Ethernet-like networks, i.e., in the presence of destructive message collisions. Recall that broadcast media are physically characterized by a slot time, denoted $x$. A slot time is a time interval large enough to guarantee that a channel state transition triggered by some source at time $t$ is seen by every other source at time $t' < t + x/2$. With CSMA-CD protocols, a channel can enter one out of three states, namely silence, busy, collision. This will be reflected via variable *chstate*.

One obvious difference between Ethernet-like networks and busses internal to ATM switches is physical spanning, which translates into small values for $x$ (1 or a few bit times) in the latter case, i.e., into the possibility of implementing an exclusive-OR logic at the bus level, this yielding non-destructive message collisions. It is reasonably straightforward to derive an analysis of the CSMA/DDCR protocol in the case of ATM switches from the analysis presented below.

Source $s_i$ receives messages taken from set $MSG_i$. Let $T(msg)$ denote the arrival time of $msg$. Source $s_i$ stores these messages in a waiting queue, denoted $Q_i$.

From now on, we will drop index $i$, unless $i$ is needed.

- Local Algorithm LA

Messages in $Q$ are serviced according to EDF. Let $msg^*$ denote the message ranked first in $Q$ at any time, as per LA.

- CSMA/DDCR

Channel sharing between sources works à la CSMA-CD whenever there is no unresolved collision pending (which is not equivalent to having the $Q$'s empty). When a collision is detected and every previous collision has been resolved, every source initiates algorithm CSMA/DDCR shown below. CSMA-CD is also employed internally by CSMA/DDCR, which comprises two inner algorithms, namely TTs (time tree search) and STs (static tree search). Note that LA on the one hand, TTs or STs on the other hand, run in parallel. For instance, while CSMA/DDCR is being run, $Q$ may be updated by LA, because of new incoming message(s). EDF ranking in local $Q$ determines whether a new message should become $msg^*$ (i.e., the current one may have an absolute deadline $DM$ that is not the smallest deadline any longer).

Let $\sharp.Q$ be a reference to local $Q$. For each message $msg$ in $Q$, let $I.msg$ be a data structure which contains a reference to $msg$, denoted $\sharp.msg$, and absolute deadline $DM(msg) = T(msg) + d(msg)$. For any source $s$:

```
CSMA/DDCR ($\sharp.Q$)

% $msg^*$ is the message first in $Q$ %
% while $Q$ empty, $msg^*$ := $nil$ %
begin
  $reft$ := local physical time
  begin loop
    TTs ($reft$, $\sharp.Q$)
    if $out = false$ then
      $reft := reft + \theta(c)$
    else
      begin
        attempt transmit $msg^*$ à la CSMA-CD
        if $chstate = collision$ then
          $reft$ := local physical time
        fi
      end
    fi
  end loop
end
```

Boolean $out$ is maintained by TTs, with $out = true$ meaning "time tree search is over, one message at least has been transmitted", $out = false$ meaning "time tree search is over, no message has been transmitted".

Variable $reft$ stands for reference time. Every source maintains $reft$ locally. Variable $reft$ is always set to local physical time whenever CSMA/DDCR is started. Whenever TTs sets boolean $out$ to $false$ (i.e., whenever $m$ consecutive empty slots are heard while running TTs), this is an indication that pending messages have "large" absolute deadlines (which prohibited them from entering that search). In order to avoid lengthy channel idleness, we may want to "reduce" their absolute deadlines. This is called the "compressed time" mode. When operating under this mode, $\theta(c)$ is assigned some positive value. $\theta(c)$ is any linear fonction of $c$. When "compressed time" mode is not on, $\theta(c)$ is set to value 0. Constant $c$ is a time interval, the size of a deadline equivalence class (see further).

Each time TTs is over and boolean $out$ is set to $false$, a source does $reft := reft + \theta(c)$. This may be repeated consecutively many times. Assignment $reft$ := local physical time is effected whenever a message is successfully transmitted during a time tree search, or whenever a static tree search is completed (see further).

Obviously, $\theta(c)$ determines a tradeoff between reducing potential channel idleness and potentially increasing the number of deadline inversions (or vice-versa).

Note that CSMA/DDCR is run even though local $Q$ is empty. Also, if a message is waiting in $Q$ at the end of some execution of TTs, its transmission is attempted, à la CSMA-CD, when that time tree search is over. If a collision ensues, CSMA/DDCR is invoked again.

```
TTs ($reft$, $\sharp.Q$)

begin
  % $msg^*$ is the message first in $Q$ %
  % $msg^*$ := $nil$ if $Q$ empty %
  if $msg^* \neq nil$ then
    $time\text{-}index^*$:= $f(reft, I.msg^*)$
  fi
  if $time\text{-}index^* > F - 1$ then
    $msg^*$ := $nil$
  fi
  if $msg^* = nil$ then
    $time\text{-}index^*$ := $nil$
  fi
  $m\text{-}ts$(($\sharp.msg^*$), $time\text{-}index^*$)
  % $m\text{-}ts$ serves to conduct a $m$-ary tree search %
  % when $msg^*$ is successfully transmitted,
    $\sharp.msg^*$ is suppressed from $Q$,
    and next message in $Q$ becomes $msg^*$ %
    % whenever a collision is detected while searching
    a time tree leaf, STs($\sharp.msg^*$) is invoked %
    $out := true$ or $false$
    % depending on the outcome of
    the time tree search %
end
```

A source must run TTs even if it does not have to or cannot transmit a message during that time tree search. Integer $F$ is the number of leaves in a time tree (some integer power of $m$), which determines $cF$, the "scheduling horizon". Function $f$ is detailed further. Each execution of algorithm TTs, as well as STs, involves a balanced $m$-ary tree search, denoted *m-ts*. Note that when STs is invoked by TTs, the static tree root has already been "searched".

*Principles of m-ary tree search m-ts.*

Let $N$ be a power of $m$, which is the number of some tree leaves, numbered from 0 to $N-1$ from left to right. First time there is a collision, search proceeds by "examining" the leftmost subtree (out of $m$). Only those sources that have an index (see further) which belongs to that subtree allow themselves to remain active, i.e., to persist attempting transmitting. If only one source remains active, its message is successfully transmitted. If there is a collision again, the splitting process is repeated (the leftmost subtree of the subtree is "examined"). And so on. When a subtree $\sigma$ of height $h$ has been fully searched (silence or one message transmission), the search process is "reversed", i.e., the subtree (of height greater than or equal to $h$) adjacent to $\sigma$ is then searched.

Therefore, in one execution of TTs, a source transmits at least all those local messages whose absolute deadlines are smaller than $t + cF$, $t$ being the time when CSMA/DDCR is started.

While running *m-ts*, a collision may be detected on a time tree leaf. This is an indication that $s > 1$ sources tried to transmit $s$ messages using the same time-index. This is similar to trying to schedule several tasks of equal absolute deadlines with centralized EDF. Some tie breaking algorithm must be resorted to. Static tree searches are the rule here. Algorithm STs is immediately run whenever a time tree leaf collision occurs. Only when STs is over is the time tree search resumed.

Function $f$ that appears in TTs can be any function that computes an integer ranging between 0 and $F-1$. However, as our goal is to emulate distributed NP-EDF, we must avoid contention between messages that have drastically different absolute deadlines DM (deadline inversions result into non optimal schedules). Of course, deadline inversions may occur because the channel is a non preemptable resource. However, such inversions are unavoidable, no matter which protocol is used. Ideally, only those messages that have absolute deadlines belonging to the same deadline equivalence class should enter competition. In our work, we considered $f(reft, I.msg^*) = max\{\lfloor (DM(msg^*) - (\alpha + reft))/c \rfloor, f^* + 1\}$.

Constant $\alpha$ is a tunable parameter that serves to allow messages enter a time tree search before it is "too late" (the duration of a static tree search may be greater than c). Integer $f^*$ is the highest value of those time tree leaves searched last (reset to $-1$ when TTs is exited). The max function over integer $f^* + 1$ serves to guarantee that no source servicing a "late" message will ever compute a negative time index or a positive time index that has been searched already, which would mean that its ("late") $msg^*$ message - as well as subsequent messages in $Q$ - will be put to wait until a new TTs is started. The effect of the max function over $f^* + 1$ is to guarantee that a "late" message is always processed as soon as possible, i.e., right upon arrival.

Let us now examine STs. Let $q$ be some power of $m$ that is higher than or equal to $z$. Static trees have $q$ leaves, numbered 0 to $q-1$, from left to right. Let $q'$ be some subset of $[0, q-1]$ (not all $q$ integers need be allocated to sources). Set $q'$ is partitioned into exactly $z$ subsets. Integers belonging to the subset allocated to source $s_i$ are called $s_i$'s static indices. Let $\nu_i$ be the number of static indices allocated to $s_i$. These indices are locally ranked by increasing values. Static tree search STs is initiated by every source, using first static index, whenever there is a time tree leaf collision. Procedure *m-ts* is also run by STs. At any time, a source knows which of its static indices it has used for its most recent message transmission. Next index in the ranking is used to keep conducting *m-ts*. In one execution of STs, source $s_i$ may transmit up to $\nu_i$ messages. Variable $reft$ is updated by STs, upon completion, by doing $reft :=$ local physical time.

Note that $F$ and $q$ can be different. Also, a time tree and a static tree may have different branching degrees $m$.

## 4 Analysis, Proofs and Feasibility Conditions

Proving that a protocol is a correct solution to a real-time problem consists in establishing feasibility conditions (FCs). Given that CSMA/DDCR makes use of $m$-ary tree searches, problems $P_1$ and $P_2$ presented below need be solved in order to express the desired FCs.

Let $\xi_k^t$ denote the worst-case search time for isolating $k$ leaves in a $t$-leaf balanced $m$-ary tree. Search times are expressed in numbers of tree nodes visited (collision slots) or empty channel slots (searches of subtrees containing no active source). Successful transmissions do not contribute to search times. The physical time duration that corresponds to $\xi_k^t$ simply is $x\,\xi_k^t$.

### 4.1 Problem $\mathbf{P_1}$

$\mathcal{P}_1$: Express a computable tight upper bound on worst-case searches for a $t$-leaf balanced $m$-ary tree.

In the sequel, we give an exact expression of $\xi_k^t$. Furthermore, we establish the asymptotic expression of $\xi_k^t$, denoted $\tilde{\xi}_k^t$, and prove that $\tilde{\xi}_k^t$ is a tight upper bound on worst-case search time for isolating $k$ leaves in a $t$-leaf balanced $m$-ary tree.

Let us consider a $t$-leaf balanced $m$-ary tree, $t = m^n$, $m \in \mathbb{N}^* \setminus \{1\}$, $n \in \mathbb{N}^*$. $\xi_k^t$ is the highest of search times computed over binomial$(t, k) = t!/k!/(t-k)!$ ways of choosing $k$ leaves from $t$ distinct leaves. $\xi_k^t$ satisfies the following recursive equation:

$$\xi_k^t = \begin{cases} 1 + \max\limits_{\substack{k_1+\cdots+k_m = k,\\ (k_1,\ldots,k_m) \in [0, t/m]^m.}} \left\{\xi_{k_1}^{t/m} + \cdots + \xi_{k_m}^{t/m}\right\} & \text{if } k \in [2, t], \\ 0 & \text{if } k = 1, \\ 1 & \text{if } k = 0. \end{cases} \tag{1}$$

When a collision occurs, the set of indices allocated to the message sources allowed to remain active is split into $m$ sets of equal size. Thus, we have:

$$\xi_k^t = 1 + \max_{\substack{k_1+\cdots+k_m = k,\\ (k_1,\ldots,k_m) \in [0, t/m]^m.}} \left\{\xi_{k_1}^{t/m} + \cdots + \xi_{k_m}^{t/m}\right\},\ k \in [2, t]$$

(collision slots).

When no collision occurs, the set of indices allocated to the message sources allowed to remain active contains at most one active source. Thus, we have:

$$\xi_k^t = \begin{cases} 0 \text{ if } k = 1 \text{ (successful transmission)}, \\ 1 \text{ if } k = 0 \text{ (empty channel slot)}. \end{cases}$$

Solving Eq. 1 is far from being trivial. In [22], we prove by induction on $t$ that the $\xi_k^t$ function also satisfies the following divide-and-conquer recursive equations:

$$\xi_{2p}^t = \begin{cases} 1 + \sum\limits_{i=0}^{m-1} \xi_{2\left\lfloor \frac{min(p, t/m)+i}{m} \right\rfloor}^{t/m} - 2max(0, p - t/m) \\ \text{if } p \in [1, \lfloor t/2 \rfloor], \\ \\ 1 \\ \text{if } p = 0. \end{cases} \tag{2}$$

$$\xi_{2p+1}^t = \xi_{2p}^t - 1,\ p \in [0, \lceil t/2 \rceil - 1].$$
$$t = m^n,\ m \in \mathbb{N}^* \setminus \{1\},\ n \in \mathbb{N}^* \setminus \{1\}. \tag{3}$$

For a $m$-leaf balanced $m$-ary tree, we know by definition that:

$$\xi_0^m = 1;\ \xi_{2p}^m = 1 + m - 2p,\ p \in [1, \lfloor m/2 \rfloor], \text{ and}$$
$$\xi_{2p+1}^m = \xi_{2p}^m - 1,\ p \in [0, \lceil m/2 \rceil - 1]. \tag{4}$$

Let us now concentrate on the $\xi_{2p}^t$ function. We get the following equations from Eq. 2 and Eq. 4:

$$\xi_2^t = m\,log_m(t) - 1,$$
$$t = m^n,\ m \in \mathbb{N}^* \setminus \{1\},\ n \in \mathbb{N}^*. \tag{5}$$

$$\xi_{2t/m}^t = \frac{t-1}{m-1} + \left(t - \frac{2t}{m}\right),$$
$$t = m^n,\ m \in \mathbb{N}^* \setminus \{1\},\ n \in \mathbb{N}^*. \tag{6}$$

$$\xi_t^t = \frac{t-1}{m-1},$$
$$t = m^n,\ m \in \mathbb{N}^* \setminus \{1\},\ n \in \mathbb{N}^*. \tag{7}$$

$$\xi_{2p+2}^t - \xi_{2p}^t = m(log_m(t) - \lfloor log_m(mp) \rfloor) - 2,$$
$$p \in [1, \lfloor t/2 \rfloor - 1],$$
$$t = m^n,\ m \in \mathbb{N}^* \setminus \{1\},\ n \in \mathbb{N}^* \setminus \{1\}. \tag{8}$$

Eq. 5 gives the worst-case search time for isolating 2 leaves in a $t$-leaf balanced $m$-ary tree. Eq. 6 gives the worst-case search time for isolating $2t/m$ leaves in a $t$-leaf balanced $m$-ary tree. Eq. 7 gives the worst-case search time for isolating $t$ leaves in a $t$-leaf balanced $m$-ary tree. Eq. 8 gives the "derivative" of the $\xi_{2p}^t$ function. From these equations, we get the following closed form of the $\xi_{2p}^t$ function:

$$\xi_{2p}^t = \begin{cases} \frac{m^{\lceil log_m(mp) \rceil} - 1}{m-1} + mp\left\lfloor log_m\left(\frac{t}{mp}\right)\right\rfloor + (m-2)\,p \\ \text{if } p \in [1, \lfloor t/2 \rfloor], \\ \\ 1 \\ \text{if } p = 0. \end{cases} \tag{9}$$

Finally, we derive the following closed form of the $\xi_k^t$ function from Eq. 3 and Eq. 9:

$$\xi_k^t = \begin{cases} \frac{m^{\lceil log_m(m\lfloor \frac{k}{2} \rfloor)\rceil}-1}{m-1} + m\lfloor \frac{k}{2} \rfloor \left\lfloor log_m\left(\frac{t}{m\lfloor \frac{k}{2} \rfloor}\right)\right\rfloor - (k - m\lfloor \frac{k}{2} \rfloor) \\ \text{if } k \in [2, t], \\ \\ 0 \text{ if } k = 1 \\ 1 \text{ if } k = 0. \end{cases}$$

$$t = m^n,\ m \in \mathbb{N}^* \setminus \{1\},\ n \in \mathbb{N}^*. \tag{10}$$

Let us now examine the $\tilde{\xi}_k^t$ asymptotic function of the $\xi_k^t$ function, which is derived from the closed form of the $\xi_k^t$ function:

$$\tilde{\xi}_k^t = \frac{m\,\frac{k}{2} - 1}{m-1} + m\,\frac{k}{2}\, log_m\left(\frac{2t}{k}\right) - k,$$
$$k = 2\,m^i,\ i \in [0, \lfloor log_m(t/2) \rfloor]. \tag{11}$$

In [22], we prove the following:

- $\tilde{\xi}_k^t = \frac{m\,\frac{k}{2} - 1}{m-1} + m\,\frac{k}{2}\, log_m\left(\frac{2t}{k}\right) - k,\ k \in [2, t].$
- By construction (see Eq. 11), $\tilde{\xi}_k^t$ is a tight upper bound on the $\xi_k^t$ function over interval $[2, 2t/m]$. Furthermore, we establish the following properties:

$$Max_{k\in[2,2t/m]}\left\{\tilde{\xi}_k^t - \xi_k^t\right\} = Max_{k\in[2t/m^2,2t/m]}\left\{\tilde{\xi}_k^t - \xi_k^t\right\}, \tag{12}$$

$$Max_{k\in[2,2t/m]}\left\{\tilde{\xi}_k^t - \xi_k^t\right\} \leq \left(\frac{m^{\frac{1}{m-1}}}{e\, ln(m)} - \frac{1}{m-1}\right) t, \tag{13}$$

$$Max_{k\in[2,2t/m]}\left\{\tilde{\xi}_k^t - \xi_k^t\right\} \leq \left(\frac{\sqrt{\sqrt{3}}}{2\, e\, ln(3)} - \frac{1}{8}\right) t \leq 9.54\%\, t. \tag{14}$$

Eq. 14 helps quantify the tightness of $\tilde{\xi}_k^t$ over interval $[2, 2t/m]$. Indeed, the difference between $\tilde{\xi}_k^t$ and $\xi_k^t$ over interval $[2, 2t/m]$ is small. Over interval $[2t/m, t]$, the $\tilde{\xi}_k^t$ function is not needed, given that the $\xi_k^t$ function satisfies the following equation:

$$\xi_k^t = \xi_{2t/m}^t - \left(k - \frac{2t}{m}\right) = \frac{m\,t - 1}{m-1} - k. \tag{15}$$

The $\tilde{\xi}_k^t$ asymptotic function is concave. An example is shown in fig. 1.

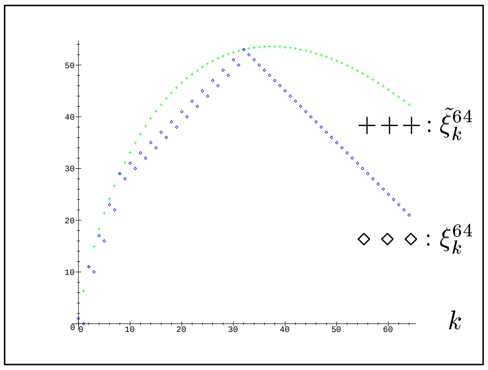

Fig. 1: Worst-case search times for a 64-leaf balanced quaternary tree

Let us illustrate these results by comparing worst-case search times for 64-leaf balanced binary and quaternary trees (fig. 2).

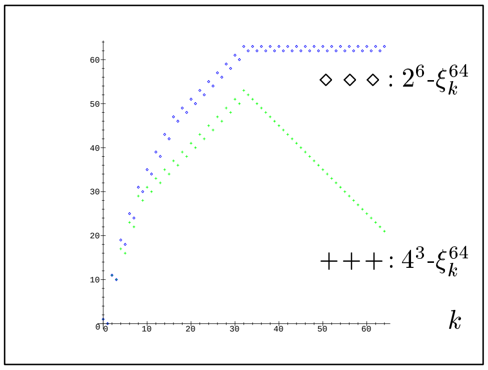

Fig. 2: Worst-case search times for 64-leaf balanced binary and quaternary trees

It is interesting to note that $4^3$-$\xi_k^{64}$, the worst-case search time for isolating $k$ leaves in a 64-leaf balanced quaternary tree, is smaller than or equal to $2^6$-$\xi_k^{64}$, the worst-case search time for isolating $k$ leaves in a 64-leaf balanced binary tree, for all $k$ in $[2, 64]$. Thus, better algorithmic efficiency is achieved with a 64-leaf balanced quaternary tree than with a 64-leaf balanced binary tree. More generally, optimal $m$ is derived from the general expression of $\xi_k^t$.

### 4.2 Problem $\mathbf{P_2}$

$\mathcal{P}_2$: Express a computable tight upper bound on worst-case searches over multiple consecutive $t$-leaf balanced $m$-ary trees.

To solve $\mathcal{P}_2$, we use the $\tilde{\xi}_k^t$ asymptotic function of the $\xi_k^t$ function. $\mathcal{P}_2$ can be re-written as the following optimization problem:

$$\underset{\substack{k_1+\cdots+k_v=u,\\ (k_1,\ldots,k_v)\in[2,t]^v.}}{\text{Maximize}} \quad \{\xi^t_{k_1}+\cdots+\xi^t_{k_v}\}, \qquad (16)$$

where $u$ is the number of messages to be transmitted over $v$ consecutive $t$-leaf balanced $m$-ary trees.

We know that, by construction, the $\tilde{\xi}^t_k$ function is a tight upper bound on the $\xi^t_k$ function. Hence,

$$\underset{\substack{k_1+\cdots+k_v=u,\\ (k_1,\ldots,k_v)\in[2,t]^v.}}{\text{Max}} \{\xi^t_{k_1}+\cdots+\xi^t_{k_v}\} \le \underset{\substack{k_1+\cdots+k_v=u,\\ (k_1,\ldots,k_v)\in[2,t]^v.}}{\text{Max}} \{\tilde{\xi}^t_{k_1}+\cdots+\tilde{\xi}^t_{k_v}\}. \qquad (17)$$

In [22], we prove that:

$$\underset{\substack{k_1+\cdots+k_v=u,\\ (k_1,\ldots,k_v)\in[2,t]^v.}}{\text{Max}} \{\tilde{\xi}^t_{k_1}+\cdots+\tilde{\xi}^t_{k_v}\} = v\,\tilde{\xi}^t_{u/v} = \tilde{\xi}^{tv}_u - \frac{v-1}{m-1}. \qquad (18)$$

Therefore, $\tilde{\xi}^{tv}_u - \frac{v-1}{m-1}$ is the solution to $\mathcal{P}_2$, given that Eq. 17 and Eq. 18 yield the following:

$$\underset{\substack{k_1+\cdots+k_v=u,\\ (k_1,\ldots,k_v)\in[2,t]^v.}}{\text{Max}} \{\xi^t_{k_1}+\cdots+\xi^t_{k_v}\} \le \tilde{\xi}^{tv}_u - \frac{v-1}{m-1}. \qquad (19)$$

## 4.3 Feasibility Conditions

Our goal is to express a computable function, called $B_{DDCR}(s_i, msg)$, an upper bound on successful transmission latency for any given message msg processed by any source $s_i, i \in [1,z]$.

Let $\psi$ be the nominal physical throughput of the communication medium considered (e.g., $10^9$ bit/s for Gigabit Ethernet). Observe that, for any message $msg$, $l(msg)$ is the bit length of a Data Link - Protocol Data Unit, which is encapsulated into a Physical - Protocol Data Unit (Ph-PDU). The physical framing and signalling overhead incurred at the physical layer translates into an actual Ph-PDU bit length $l'(msg) > l(msg)$.

Let us consider some message $\mathcal{M}$ generated by some source $s_i$. Recall that $T(\mathcal{M})$ is the arrival time of $\mathcal{M}$ and $d(\mathcal{M})$ is its relative deadline. Let $I(\mathcal{M})$ be interval $[T(\mathcal{M}), T(\mathcal{M}) + d(\mathcal{M}))$. Establishing worst-case conditions for $\mathcal{M}$ implies expressing two functions:

- $r(\mathcal{M})$, an upper bound on the ranking of $\mathcal{M}$ in queue $Q_i$; in other words, $r(\mathcal{M}) - 1$ is an upper bound on the number of messages of subset $MSG_i$ that will be serviced by source $s_i$ over any interval $I(\mathcal{M})$ before $\mathcal{M}$ is successfully transmitted,
- $u(\mathcal{M})$, an upper bound on the number of messages that will be transmitted by all sources over any time interval $I(\mathcal{M})$.

These bounds need be established assuming peak-load (i.e., worst-case) conditions, i.e., assuming that all message arrivals occur at their bounded densities over $I(\mathcal{M})$.

*Bound $r(\mathcal{M})$*

Bound $r(\mathcal{M})$ is easily established by observing that source $s_i$ can service a message $msg$ before servicing $\mathcal{M}$ only if $msg$ has arrived no sooner than $T(\mathcal{M}) - d(msg)$ and no later than $T(\mathcal{M}) + d(\mathcal{M}) - d(msg)$. It follows that:

$$r(\mathcal{M}) = \sum_{m \in MSG_i} \left\lceil \frac{d(\mathcal{M})}{w(m)} \right\rceil a(m) - 1.$$

*Bound $u(\mathcal{M})$*

Bound $u(\mathcal{M})$ is easily established by observing that a source may service a message $msg$ over interval $I(\mathcal{M})$ only if $msg$ has arrived no sooner than $T(\mathcal{M}) - d(msg)$ and no later than $T(\mathcal{M}) + d(\mathcal{M}) - \frac{l'(\mathcal{M})}{\psi}$. It follows that:

$$u(\mathcal{M}) = \sum_{m \in MSG} \left\lceil \frac{d(\mathcal{M}) + d(m) - \frac{l'(\mathcal{M})}{\psi}}{w(m)} \right\rceil a(m).$$

We can now compute $v(\mathcal{M})$, an upper bound on the number of static trees that need be searched to transmit message $\mathcal{M}$ ranked $r(\mathcal{M})^{th}$ in $s_i$'s waiting queue. Given that $\nu_i$ indices are used by source $s_i$ to conduct a complete static $m$-ary tree search STs (see end of section 3), it follows that:

$$v(\mathcal{M}) = 1 + \left\lfloor \frac{r(\mathcal{M})}{\nu_i} \right\rfloor.$$

Upper bound $B_{DDCR}(s_i, \mathcal{M})$ is equal to the sum of:

- the time needed to physically transmit $u(\mathcal{M})$ messages at throughput $\psi$,
- the upper bounds on worst-case search times for isolating:
  - $u(\mathcal{M})$ messages over $v(\mathcal{M})$ consecutive static tree searches; this is obtained by applying the solution to problem $P_2$ (cf. section 4.2)
  - $v(\mathcal{M})$ time tree leaves over interval $I(\mathcal{M})$. It is known (verification is obvious) that having

2 active leaves per (time) tree is the worst-case assignment. Hence, worst-case search will be conducted over $\lceil v(\mathcal{M})/2 \rceil$ consecutive time trees.

That is: $B_{DDCR}(s_i, \mathcal{M}) =$

$$\underbrace{\sum_{m \in MSG} \left\lceil \frac{d(\mathcal{M}) + d(m) - \frac{l'(\mathcal{M})}{\psi}}{w(m)} \right\rceil a(m) \frac{l'(m)}{\psi}}_{\substack{\text{physical transmission time} \\ \text{for the } u(\mathcal{M}) \text{ messages} \\ \text{at throughput } \psi}} + x\mathcal{S},$$

with:

$$\mathcal{S} = \underbrace{v(\mathcal{M})\, \tilde{\xi}^{q}_{u(\mathcal{M})/v(\mathcal{M})}}_{\mathcal{S}_1} + \underbrace{\lceil v(\mathcal{M})/2 \rceil\, \xi^{F}_{2}}_{\mathcal{S}_2}.$$

$\mathcal{S}_1$: upper bound on worst-case search times for isolating $u(\mathcal{M})$ messages over $v(\mathcal{M})$ consecutive static trees.

$\mathcal{S}_2$: upper bound on worst-case search times for isolating $v(\mathcal{M})$ time tree leaves over $\lceil v(\mathcal{M})/2 \rceil$ consecutive time trees.

Hence, feasibility conditions are as follows:

$$\forall s_i,\ i \in [1, z],\ \forall \mathcal{M} \in \mathrm{MSG}_i,\ B_{DDCR}(s_i, \mathcal{M}) \leq d(\mathcal{M}).$$

## 5 Applicability of the CSMA/DDCR Protocol

As defined by the IEEE 802.3z standard Working Group, half duplex Gigabit Ethernet retains the CSMA-CD protocol, the 802.3 and Ethernet frame format, and the 802.3 flow control and managed object specifications [23]. By exercising packet bursting, a source may transmit the first $k$ messages (EDF ranked) waiting in Q, $k > 1$, without relinquishing channel control, for up to 512 bytes at most. This will entail much less deadline inversions than those resulting from using deadline equivalence classes. It is indeed very uncommon for users to define message deadlines with an accuracy smaller than 4.096 microseconds. Therefore, CSMA/DDCR is fully compatible with the half duplex Gigabit Ethernet standard. Similarly, CSMA/DDCR is fully compatible with the emerging CATV Protocol standard (IEEE 802.14), notably with tree-based algorithms [24].

IEEE 802.1Q specifies explicit priorities in 802 network packet headers. With those real-time applications we consider, Classes-of-Service are naturally defined via task deadlines $D$, transformed into message deadlines $d$, which can be passed on to the CSMA/DDCR layer via the standard conformant priority field. Furthermore, IEEE 802.1p specifies the use of priority queuing mechanisms, which paves the way for local EDF queue schedulers.

The fact that standard committees have somewhat recently started paying attention to QoS/CoS considerations is good news to end users and to researchers. In the 80's, we succeeded in transferring an STs-like protocol - called 802.3D or CSMA/DCR [25] - to French customers (e.g., the French Navy) as well as to French system developers/integrators (e.g., Dassault Electronique and APTOR, a CAP-SESA Industrie subsidiary), which have deployed CSMA/DCR-based single and dual bus Ethernets for such various applications as discrete/continuous manufacturing or local area networking (e.g., across the Ariane launchpad in Kourou). Transferring a TTs-like protocol was unthinkable at that time, because of lack of compatibility with existing standards. This is not the case any longer.

Similarly, ATM cells can be prioritized. Many priority based schemes have been proposed (see [26] for an example). Again, in our case, message deadlines would serve as priorities.

## 6 Conclusions

Novel real-time applications require high-performance real-time distributed systems, and therefore high-performance real-time networks. We have specified a Hard Real-Time Distributed Multiaccess problem which arises with such application problems. We have presented a solution, based on broadcast LANs or busses, such as Gigabit Ethernets and busses internal to ATM nodes, associated with a deterministic Ethernet-like protocol called CSMA/Deadline Driven Collision Resolution. This protocol has been informally described. We have given an analysis of balanced $m$-ary tree algorithms which are used by CSMA/DDCR, and have derived feasibility conditions for the HRTDM problem. Hopefully, the solution presented as well as the feasibility conditions can be useful to those in charge of developing COTS products aimed at supporting novel real-time high-performance applications.

## 7 References

1. Kuhn, D. R., Sources of Failure in the Public Switched Telephone Network, IEEE Computer, April 1997, 31-36.

2. Leveson, N. G., Turner, C., An investigation of the Therac-25 accidents, IEEE Computer, July 1993,

18-41.

3. Le Lann, G., Proof-Based System Engineering and Embedded Systems, School on Embedded Systems, Veldhoven (NL), Nov. 1996, Lecture Notes in Computer Science, Springer Verlag Pub., to appear in 1998, 40 p.

4. Le Lann, G., Proof-Based System Engineering for Computing Systems, $1^{st}$ Joint ESA/INCOSE Conference on Systems Engineering - The Future, IEE/ESA Pub., vol. WPP-130, Nov. 11-13, 1997, 5a.4.1-5a.4.8.

5. White, S. et al., Engineering of Computer-Based Systems: Current Status and Technical Activities, IEEE Computer, June 1995, 100-101.

6. IEEE Transactions on Aerospace and Electronic Systems, vol. 33 (2), April 1997, 577-733.

7. Le Lann, G., An Analysis of the Ariane 5 Flight 501 Failure - A System Engineering Perspective, IEEE Conference on Engineering of Computer-Based Systems, Monterey, CA, March 1997, 339-346.

8. Bernstein, P. A., Hadzilacos, V., Goodman, N., Concurrency Control and Recovery in Database Systems, Addison-Wesley Pub., ISBN 0-201-10715-5, 1987, 370 p.

9. Joseph, M., et al., Real-Time Systems - Specification, Verification and Analysis, Prentice Hall UK Pub., ISBN 0-13-455297-0, 1996, 278 p.

10. Lynch, N.A., Distributed Algorithms, Morgan Kaufmann Pub., ISBN 1-55860-348-4, 1996, 872 p.

11. Leland, W.E. et al., On the Self-Similar Nature of Ethernet Traffic, IEEE/ACM Transactions on Networking, vol. 2(1), Feb. 1994, 1-15.

12. Paxson, V., Floyd, S., Wide Area Traffic: The Failure of Poisson Modeling, IEEE/ACM Transactions on Networking, vol. 3(3), June 1995, 226-244.

13. Tsybakov, B.S., Georganas N.D., On Self-Similar Traffic in ATM Queues: Definitions, Overflow Probability Bound, and Cell Delay Distribution, IEEE/ACM Transactions on Networking, vol. 5(3), June 1997, 397-409.

14. Knightly, E.W., Zhang, Hui, D-BIND: An Accurate Traffic Model for Providing QoS Guarantees to VBR Traffic, IEEE/ACM Transactions on Networking, vol. 5(2), April 1997, 219-231.

15. Kelly, F.P., Stochastic Models of Computer Communication Systems, J. of Royal Statistical Soc., B 45, 1985, 379-395.

16. Gallager, R.G., A Perspective on Multi-Access Channels, Special Issue on Random-Access Communication, IEEE Transactions on Information Theory IT-31, 1985, 124-142.

17. Tsybakov, B.S., Survey of USSR Contributions to Random Multiple-Access Communications, Special Issue on Random-Access Communication, IEEE Transactions on Information Theory IT-31, 1985, 143-165.

18. Mathys, P., Flajolet, P., Q-ary Collision Resolution Algorithms in Random Access Systems with Free or Blocked Channel Access, IEEE Transactions on Information Theory IT-31, 1985, 217-243.

19. Jacquet P., Random Infinite Trees and Supercritical Behavior of Collision Resolution Algorithms, IEEE Transactions on Information Theory IT-39, 1993, 1460-1465.

20. Jeffay, K., Stanat, D.F., Martel, C.U., On Non-Preemptive Scheduling of Periodic and Sporadic Tasks, IEEE Real-Time Systems Symp., Dec. 1991, 129-139.

21. Hermant, J.-F., Leboucher, L., Rivierre, N., Real-Time Fixed and Dynamic Priority Driven Scheduling Algorithms: Theory and Experience, INRIA Research Report 3081, Dec. 1996, 142 p.

22. Hermant, J.-F., Analysis of Real-Time Distributed Scheduling Algorithms, PhD Thesis, University of Paris VI, to appear in 1998.

23. IEEE 802.3z WG, Gigabit Ethernet, ftp://stdsbbs.ieee.org/pub/802_main/802.3/gigabit/ and http://www.gigabit-ethernet.org/.

24. Bisdikian, C., Performance Analysis of the Multislot *n*-ary Stack Random Access Algorithm msSTART, IEEE 802.14-96/117, May 1996.

25. Le Lann, G., Rolin, P., The 802.3D Protocol: A Variation on the IEEE 802.3 Standard for Real-Time LANs, multinational patent, 1984.

26. Chao, J.H., A Novel Architecture for Queue Management in the ATM Network, IEEE Journal on Selected Areas in Communications, vol. 9(7), Sept. 1991, 1110-1118.