Real-time communications over broadcast networks: the CSMA-DCR and the DOD-CSMA-CD protocols

Gerard Le Lann, Nicolas Rivierre

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

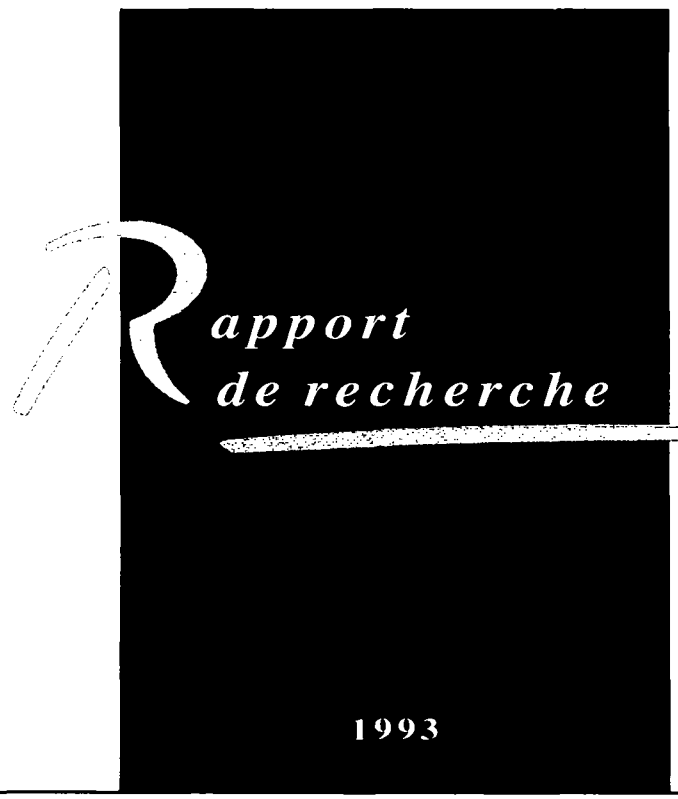
Real-Time Communications over Broadcast Networks: the CSMA-DCR and the DOD-CSMA-CD Protocols

Gérard LE LANN
Nicolas RIVIERRE

N° 1863
Mars 1993

PROGRAMME 1

Architectures parallèles,
bases de données,
réseaux et systèmes distribués



Communications temps réel sur réseaux à diffusion : les protocoles CSMA-DCR et DOD/CSMA-CD

Gérard Le Lann, Nicolas Rivierre

Résumé

Parmi les nombreux protocoles publiés qui «résolvent» le problème des communications temps réel sur les réseaux à diffusion, quelques uns seulement satisfont les contraintes de robustesse et de ponctualité propres aux systèmes temps réel. Parmi ces protocoles, seulement quelques uns sont compatibles avec des normes très répandues, telle la norme ISO/OSI 8802/3 (le «standard Ethernet»).

Dans cet article, on décrit deux protocoles qui satisfont les trois exigences de robustesse, de ponctualité et de conformité à une norme. Basés sur les principes bien connus de compétition et de détection de collision (CSMA-CD), ces deux protocoles fournissent des services déterministes de livraison de messages, en présence non seulement de trafic périodique et sporadique -- hypothèses retenues habituellement-- mais aussi en présence de trafic en rafale ou apériodique.

Le protocole CSMA-DCR (Deterministic Collision Resolution) convient pour gérer des messages temps réel sans échéance ou ayant des échéances implicites, types de messages souvent considérés par les concepteurs de protocoles à passage de jeton ou à scrutation ou de réseaux discrétisés synchrones. Le protocole DOD/CSMA-CD (Deadline Oriented Deterministic) convient pour gérer des messages temps réel auxquels sont associées des échéances strictes.

Cet article a également pour but de démontrer qu'il est possible de résoudre un problème ouvert en informatique répartie temps réel, à savoir l'expression de bornes supérieures certaines sur les temps de réponse, en présence de contrôle décentralisé et de lois d'arrivées apériodiques. Les bornes sur les délais de transmission et sur la durée des époques de résolution de collision sont données pour les deux protocoles. Pour ce faire, on recourt à des résultats récents en théorie de l'ordonnancement et à une technique classique en théorie des jeux (arguments d'adversité).

Bien que cela ne soit pas montré dans l'article, ces deux protocoles peuvent être généralisés et utilisés sur des réseaux à diffusion «à haute vitesse» (dont la vitesse est de plusieurs ordres de grandeur supérieure à celle d'Ethernet). Ils peuvent bien évidemment être implantés efficacement sur des réseaux «lents» ou «petits», tels ceux utilisés dans les systèmes de transport (automobiles, trains, avions, véhicules spatiaux par exemple) ou dans les systèmes critiques statiques (centrales nucléaires ou usines chimiques par exemple).

Real-time communications over broadcast networks : the CSMA-DCR and the DOD/CSMA-CD protocols

Abstract

Among the numerous published protocols that «solve» the problem of achieving real-time communications over broadcast networks, only a few are capable of meeting the robustness and the timeliness constraints proper to real-time systems. Among these protocols, only a few are compatible with widely used standards, such as ISO/OSI 8802/3, also known as the «Ethernet standard».

This paper serves two purposes. First, it contains a description of two protocols which meet the triple requirement of robustness, timeliness and compatibility with a standard. Based on well-known contention/collision detection principles (CSMA-CD), these two protocols implement deterministic message delivery services not only in the presence of periodic and sporadic traffic -- assumptions usually considered -- but also in the presence of bursty or aperiodic traffic.

CSMA-DCR (Deterministic Collision Resolution) is a protocol that is appropriate for handling real-time messages that have no deadlines or that have implicit deadlines, as often considered by designers of token passing/polling protocols or time-slotted synchronous networks. DOD/CSMA-CD (Deadline Oriented Deterministic) is a protocol that is appropriate for handling real-time messages tagged with strict arbitrary deadlines.

Second, this paper demonstrates that it is possible to solve an open problem in real-time distributed computing, that is how to express guaranteed upper bounds on response times, in the presence of decentralized control and aperiodic arrival laws (traffic). Bounds on message transmission delays and on collision resolution latency are given for both protocols. In order to do so, we make use of results recently established in Scheduling Theory and combine them with a classical technique in Game Theory (adversary arguments).

Although this is not shown in the paper, both protocols can be generalized and used over «high-speed» broadcast networks (orders of magnitude faster than Ethernet). Obviously, they can be efficiently implemented over «slow» or «small» networks, such as those used in transportation systems (e.g., cars, trains, aircrafts, spaceships) or in static critical systems (e.g., nuclear plants, chemical plants).

1. INTRODUCTION

In a non real-time setting, communications over a (distributed) broadcast channel are accomplished via what is commonly called a multiaccess protocol (the MAC sublayer in the ISO/ OSI reference model). The merits of a multiaccess protocol are usually assessed in terms of average access delay or in terms of achievable average throughput or channel utilization ratio under given load conditions.

In a real-time context, a multiaccess protocol is usually assessed by its ability to transmit messages «in time». Hence, in this case, a multiaccess protocol should more appropriately be viewed as a (distributed) «real-time» scheduling algorithm. As will be seen, the scheduling algorithms we are interested in belong to the «on-line» category. Furthermore, the timeliness constraints considered correspond to what is often called «hard» or «strict» deadlines. The decision to investigate the applicability of distributed on-line algorithms to the «hard/strict deadline» scheduling problem was made for three reasons.

First, we had observed that most results published by proponents of «off-line» approaches (e.g. pre-computed fixed priorities or pre-computed schedules) either are based on unrealistic models/ assumptions or are unapplicable to distributed systems. For example, it is postulated that most tasks are triggered periodically. Periodic tasks only should meet «hard» or «strict» deadlines. It is postulated that aperiodic tasks are bound to meet «soft» deadlines, i.e. no deadline constraint is to be provably satisfied for such tasks. This is a fallacious exposition of the real problems as they arise in real systems. Our research work is targeted at so-called «critical» real-time distributed (RTD) systems. Off-line approaches are based on full or quasi-full global knowledge or clairvoyance assumptions. Consequently, the timeliness proofs given under such approaches are conditioned on the probability that clairvoyance assumptions correctly reflect real (future) operational conditions. The least is assumed at design time, the better, in the case of critical RTD systems. Thus our belief that systems are more «predictable» (assumptions, and therefore proofs, are more «valid») when their behavior is determined by on-line algorithms.

Furthermore, it can be demonstrated that «off-line» algorithms or methods are based on premises that are in contradiction with the definition of distribution. Hence, «off-line» algorithms or methods are no more than empirical solutions to the «strict deadline» scheduling problem as it arises in distributed systems.

Second, with RTD systems, two essential properties, namely serializability (of concurrent tasks) and dependability rest on on-line algorithms. It looked suspicious to us that the third major property, namely timeliness, could be obtained by resorting to an off-line approach.

The third reason simply is that we found it challenging to demonstrate that, contrary to widespread belief, it is indeed possible to establish guaranteed timeliness properties «in spite of» using on-line schedulers and «in spite of» arbitrary arrival laws.

With critical RTD systems, proofs are mandatory. With few exceptions so far, formal approaches have not been satisfactory. The main reason is that reality (e.g., unpredictable load conditions or arrival laws, arbitrary physical deadlines, physically distributed architectures, occurrence of

failures, asynchronous concurrency) is too much abstracted away, thus making the proofs barely usable in a real setting. Our approach is based on exploiting results in Scheduling Theory and techniques in Game Theory. More precisely, we seek to express upper bounds on message latency, using scheduling algorithms that are known to have specific properties. For example, one algorithm considered has been shown to be optimal in (conventional) centralized systems. Any feasible task/deadline set is correctly scheduled (i.e., timeliness constraints are always satisfied) by an optimal algorithm. Hence, any such algorithm is a correct and proven solution whenever feasibility conditions are satisfied. Some of the work reported consists in deriving a distributed version of this optimal scheduling algorithm. Feasibility conditions make it possible for a user or a designer to check a priori whether some (possibly «worst-case») task/deadline set will always be correctly scheduled. The presentation of the feasibility conditions for the problems and solutions considered here will be given in a forthcoming paper. Some results on such conditions have already been established for the periodic and the sporadic cases [GR93].

This work can be viewed as a means to «bridge the gap» existing between the real physical computing systems and the (so far) simple machine abstractions currently accommodated with formal approaches. This work can thus be useful in deploying formal approaches much more widely than has been the case so far.

2. THE PROBLEMS UNDER STUDY

2.1. Two distributed scheduling problems

In the context of this paper, tasks are messages. Under its most general expression, the message scheduling problem consists in considering messages that carry different «values» (i.e., importance to users), such values possibly being time varying. The corresponding scheduling problem is that of maximizing the returned value for any given message/deadline/value scenario. In this paper, we consider two simpler problems, where messages do not carry distinct values.

However, we investigate the hard real-time communication problem under the rather unconventional and difficult assumption that times of message arrivals are unknown, that is under the assumption that messages are aperiodic, which is a non-assumption.

Let us recall that a sporadic message is an aperiodic message that is characterized by a non-zero lower bound b on its release/submission cycle (at least b time units elapse between two consecutive arrivals). An aperiodic message is characterized by b=0 i.e., there is no upper bound on the number of messages that may be released/submitted at any point in time.

The first problem studied, referred to as Ω_1, addressed section 4, corresponds to the case where messages do not carry explicit deadlines. Bounds on message latency are equivalent to the relative deadlines that will be provably met for any possible pattern of message arrivals. One particular way of exploiting these results is to consider that relative deadlines are values of message cycles, taking the conventional view that message arrival laws are periodic. Of course, bounds obtained are equally valid for sporadic and aperiodic arrival laws.

The second problem studied, referred to as Ω_2, addressed section 5, is the more difficult problem

where messages carry strict arbitrary deadlines, that are revealed only at times of message arrivals.

Of course, the major difficulty encountered in trying to solve these two distributed scheduling problems is that of establishing the expression of upper bounds on successful message transmission latency. In other words, such bounds would measure how reactive a distributed channel can be, on a per message basis. We show that, contrary to conventional approaches, it is not necessary to assume full or nearly full a priori knowledge of future arrival scenarios to express (and predict) such upper bounds.

As will be seen, the protocols described in the sequel have been designed taking into account the fact that scheduling over a broadcast channel is non-preemptive (preemption would be interesting only in the case of slow networks or/and long messages but is never used in reality). This is an important observation, as it tells us that a great many conventional scheduling algorithms/ methods proposed for scheduling tasks over (preemptable) processors are not applicable as such to our problems (e.g. Highest Priority First, with priorities computed off-line using Rate Monotonic (RM) or Deadline Monotonic (DM) methods). This is somewhat ironic given the fact that some current mainstream approaches to «hard» real-time issues precisely are based on pre-computed priorities/preemption combinations.

Actually, the above observation simply illustrates (i) the obvious fact that we have not seen yet a correct expression of the RM or DM approaches in the case of distributed systems, (ii) the less obvious fact that we will never see one (there cannot be a fixed priority-based solution to the distributed scheduling problem, for the reason that such a solution would be based on assumptions that would inevitably contradict the concept of distribution [LL77, L78]). To be more specific, adaptations or extensions of the RM/DM approaches to distributed systems (e.g., [LS86], [SM89], [SSL89]) can only yield empirical or approximate solutions. With such approaches, it is impossible to solve the problems we consider, because it is impossible to express guaranteed upper bounds on (aperiodic) message latency.

2.2. The channel model

Consider a channel with M message sources connected to it. Taking the conventional off-line view, absolute or relative times of message arrivals is advance knowledge. This permits the definition of a cyclic time frame. Static synchronous time division multiplexing access (SS-TDMA), pre-computed schedules (e.g. [K et al. 89]) and pre-planned polling (e.g., Fieldbus) are examples of such approaches. Under our approach, times of message arrivals are arbitrary. In other words, we can reason about arbitrarily dense message arrivals.

Let m be the physical transmission time of a message. The physical transmission time strictly needed for k messages will be denoted $T_k = m_1 + ... + m_k$. As with every broadcast channel, a channel slot time, denoted s, is defined. For example, s ranges between 25 µs and 50 µs (approximately) with the Ethernet or ISO/OSI 8802/3 standards. Our algorithms are extensions of these standards. They are meant to handle message collisions deterministically.

We are interested in analyzing how «reactive» a channel can be, depending on the scheduling

algorithm used, when being submitted a given message scenario. In order to provably solve problems Ω_1 and Ω_2, as well as to measure the reactivity induced by our algorithms, we need to express B, the upper bound on successful transmission latency, for any aperiodic message, i.e. the worst-case time elapsed between its arrival and end of its transmission.

With problem Ω_1, B determines what would be the guaranteed message cycles (periodic arrivals) under a conventional approach. With problem Ω_2, message deadlines are prescribed a priori. Bound B serves also the purpose of deriving feasibility conditions for any scenario, i.e. for any combination of message arrival timings and explicit deadlines.

As will be seen, bound B is expressed for any given source, denoted i, and for any given message ranked r^{th} in source i's message waiting queue. In the sequel, we give the general expression of bound B(i,r). We follow the convention that a message leaves a waiting queue only after it is completely transmitted.

Besides bound B(i,r), one might also want to know the cost (or the overhead) incurred with our solutions, as well as the corresponding channel efficiency. Such measures are given. They are interesting, as there is widespread belief that on-line algorithms can only be inefficient. Channel efficiency will be denoted Δ. More precisely, we denote $\Delta(i,r)$ the channel efficiency achieved whenever it takes B(i,r) to transmit a given message. Quite naturally, $\Delta(i,r)$ is defined as follows :

$$\Delta(i, r) = T_{n(r)} / B(i, r) \quad ,$$

where $T_{n(r)}$ is the time strictly needed to transmit that sequence of n(r) messages which yields B(i,r). The overhead incurred, denoted $\Gamma(i,r)$, is 1 - $\Delta(i,r)$.

Another performance measure of interest is the highest incoming message throughput that can be sustained by a source, with no deadline being missed. More precisely, we define $\Phi(i,r)$ as the upper bound on the timely influx density, i.e. the density of message arrivals such that deadlines are satisfied for all messages up to rank r in source i's waiting queue.

3. OFF-LINE VERSUS ON-LINE APPROACHES

3.1. Clairvoyance assumptions considered harmful

The various approaches to solving Ω_1 and Ω_2 which can be envisioned differ essentially by the amount of advance knowledge, called clairvoyance, which is assumed to be made available at design time. Clairvoyance relates to fault and load patterns. We ignore here the issues raised with the existence of faults. (It is clear, however, that an on-line algorithm is more fault-tolerant than any algorithm based on off-line computations).

Clairvoyance consists in assuming advance knowledge of future scenarios. Most often, a scenario comprises the following variables :

- K_0 : the actual length of every message that can be submitted,
- K_1 : the exact number of messages generated by every source (or an upper bound),
- K_2 : the actual times of message submissions or arrivals.

SS-TDMA, pre-computed schedules, polling, are examples of protocols based on clairvoyant load assumptions. It is important to understand that, under a «clairvoyant» approach, knowledge K_0, K_1 and K_2 is needed to design a solution. Such a solution consists in pre-allocating channel slots or turns to message sources. From K_0, K_1 and K_2, one can compute B. Hence, the feasibility of a scenario can be checked a priori. Clearly, the value computed for B is guaranteed only for the scenario considered. Would operational conditions violate the clairvoyance assumptions, there is no guarantee anymore. Furthermore, a new design need be undertaken would K_0 or K_1 or K_2 be modified.

Clairvoyance based designs can yield proofs only when considering periodic traffic (handled efficiently in many cases) and sporadic traffic (handled rather inefficiently in most cases). Such designs are implicitly based on the belief that external event arrival laws (i.e., those observed from the outside of a system) that may have particular properties (e.g., periodicity) are kept unchanged by the computation steps that transform those external events into internal events, namely message submissions over some (internal) shared communication channel. Furthermore, under an SS-TDMA or a pre-computed schedule approach, it is often assumed that arrival times (from the outside, over the channel) are such that mutual exclusion among messages is naturally obtained over the channel. This very strong hypothesis is what we call the perfect isochrony assumption. For the vast majority of real world computing systems, such an assumption simply is unrealistic. This very peculiar view can only apply to systems where internal resource contention or internal queueing phenomena (other than at the channel level) cannot develop. Such an assumption is valid only with sequential automata, not with distributed systems.

With distributed systems, resource contention and internal queueing phenomena are unavoidable. It is impossible to predict every possible future system history, even less at which time would some particular future global state be instantiated. This is one of the reasons why approaches based on off-line computations make no sense in the case of distributed systems. In particular, even if external laws are assumed to have particular properties (e.g., periodicity), this is of no help to solve the «hard deadline» multiaccess channel problem. Nevertheless, it can be easily observed that the term «distributed system» is commonly used by a number of authors, even when inappropriate. For example, Mars [K et al.89] is not a distributed system. Mars is a wait-state free synchronous shared bus system based on dedicated processors. The design of Mars and other similar systems is based on the assumption that there is a perfect match between times of sensor reads and times of message arrivals over the internal channel (the perfect isochrony assumption). It is easy to demonstrate that the design of such systems is strictly equivalent to the design of a conventional (centralized) system. The issue is not just a matter of definitions. What can be demonstrated has profound practical consequences. The design approach adopted for Mars or similar systems simply cannot be used for real distributed systems.

Being concerned with real world distributed systems, we have to solve the real problem, that is how to prove timing properties under aperiodic arrivals.

3.2. An approach to reason about the unknown

Our approach is the exact counterpart of clairvoyant approaches. We believe that a design should be conducted, and general properties should be established, under minimally clairvoyant

assumptions. It is possible to prove the existence of timeliness properties by resorting to the following approach, which stresses the importance of algorithmic issues. First, an (on-line) algorithm A has to be selected or designed. Second, an adversary is defined, whose behavior is as unrestricted as desired. The only constraint that cannot be relaxed is that the adversary has to obey algorithm A as well. As will be seen, we explore two algorithms. One is called CSMA-DCR, which stands for Deterministic Collision Resolution CSMA-CD. The other one is called DOD/CSMA-CD, which stands for Deadline Oriented Deterministic CSMA-CD.

Techniques based on adversary arguments are widely used in Game Theory. Combining such techniques with awareness of current state-of-the-art in the areas of Scheduling or/and Algorithms (for the selection of A) is the key to proving the existence of finite time bounds, even when considering unrestricted adversaries.

In this paper, we demonstrate the power of this approach by establishing the expressions of bounds on message latency for any source of messages, analyzed as a player against an adversary referred to as Z_0, which represents the environment and the other message sources, and which is free to generate an infinite amount of messages at any point in time (Z_0 is fully unrestricted)[1]. Of course, those bounds established for Z_0 are also valid for partially restricted adversaries, which we refer to as Z_k, $k > 0$, k being representative of the degree of restriction imposed to the adversary. Nevertheless, better bounds can be established when considering $Z_k \neq Z_0$. This is part of our on-going work. In this paper, we concentrate solely on adversary Z_0, as it is widely believed that bounded response times cannot be guaranteed when considering infinite influx of external events (messages in our case). This is a mistaken belief.

Bounds are established as general functions of variables, denoted $B_{alg}(i,r)$, where alg is the multiaccess/scheduling algorithm used, i is the name of the message source considered and r is the rank (in source i's waiting queue) of the message being considered. These bounds are established without assuming knowledge K_0, K_1, K_2.

The variables needed to express bounds $B_{alg}(i,r)$ are as follows (see annexes 2 and 3, where algorithms and associated variables are presented) :

* for CSMA-DCR and DOD/CSMA-CD :
  - μ, the upper bound on message length,
  - Q, the static binary tree size, determined after Q-1, the greatest index in use,
  - $t_i(1)$, $t_i(2)$,...,$t_i(v(i))$, the v(i) static indices allocated to source i,

* for DOD/CSMA-CD, in addition to the above :
  - c, the duration of a deadline equivalence class,
  - F, the time tree size, in number of leaves (or indices) used,
  - deadlines (relative to unknown times of arrival) of messages generated by source i.

Those functions giving bounds $B_{alg}(i,r)$ being expressed, they can be valued for every imaginable

1. Interestingly enough, these techniques are also receiving increased attention from some of our colleagues working on formal methods [R93, p. 31].

scenario. Similarly, in Mathematics, general formula are established (theorems) and then applied to particular cases. It is important to understand that assumptions w.r.t. load and arrival laws are made a posteriori, only for computing the actual resulting values of $B_{alg}(i,r)$, as well as for checking the feasibility of any given scenario. Note also that considering Z_0 is equivalent to considering all possible (non-countable) arrival timing patterns. As a consequence, we can reason about bounds $B_{alg}(i,r)$ considering global loads, not being forced to enumerate individual loads on a source-by-source basis, as is often the case with clairvoyant approaches. In other words, the power of our proofs is infinitely superior to that of proofs established under clairvoyant assumptions.

One additional variable is needed to express the highest overhead incurred (or the lowest efficiency achieved) by our algorithms, that is μ_0, the smallest possible message size.

3.3. Why CSMA-CD based algorithms

The class of on-line algorithms we have chosen to use to solve problems Ω_1 and Ω_2 is the well-known class of CSMA-CD protocols. We will not bother presenting the well known advantages of such protocols regarding simplicity, robustness, flexibility/adaptability. It is also a well known fact that CSMA-CD currently is the dominating technology as far as local area networking is concerned.

In addition to these advantages, another major reason for choosing CSMA-CD based algorithms is that CSMA-CD is the only existing class of standard protocols that do not use fixed priorities. Contrary to widespread belief, those multiaccess protocols that rest on fixed priorities cannot be considered to solve problem Ω_1 or Ω_2, as fixed priorities come in the way of any on-line scheduling algorithm proven to be appropriate (or, possibly, optimal).

With CSMA-CD protocols, collisions must be resolved. However, under particular circumstances, that match many realistic scenarios, collisions cannot occur in fact. Recall that μ_0 is the smallest possible message size. Consider a conventional solution (SSTDMA, polling) under some feasible load and under the perfect isochrony assumptions. If $\mu_0 \geq s$, then no collisions can occur. Consequently, under such assumptions, that are commonplace with conventional off-line solutions, a CSMA-CD protocol behaves exactly like a conventional solution. The rationale for choosing an ad-hoc or customized solution that is no more satisfactory than a standard one should thus be carefully examined.

For all other cases, that is $\mu_0 < s$, or $\mu_0 \geq s$ but no perfect isochrony assumed, collisions can occur. With CSMA-CD, having $\mu_0 < s$ yields poor efficiency in general (padding comes into play). Nevertheless, CSMA-CD still is applicable. Messages are automatically (and artificially) expanded, if needed, so that $\mu_0 \geq s$ holds true.

The essential algorithmic ingredient used in CSMA-DCR and in DOD/CSMA-CD is binary tree search. Before embarking on describing how bounds can be derived, it is important to understand how collisions are deterministically resolved when using binary tree search. We refer the reader to annex 2 for this purpose.

Bounds $B_{alg}(i,r)$ achieved by our algorithms can be compared with those obtained under an off-line approach, provided them too are proven to hold for aperiodic traffic. Such a detailed comparison is left for a future paper. The former bounds should normally be worse than the latter, as our solutions are designed under limited advance knowledge. Limitation stems from :

- our explicit rejection of clairvoyance,
- the distributed nature of problems Ω_1 and Ω_2 ; in a distributed system, a local instantiation of an on-line scheduler is provided with local knowledge only (a subset of all messages waiting to be scheduled).

It turns out that it is not always the case that algorithms based on off-line computations perform better than on-line algorithms.

4. SOLUTION, BOUNDS AND PERFORMANCE FOR PROBLEM Ω_1

The major objective here is to give the expression of $B_{alg}(i,r)$, considering messages that have no strict deadline explicitly declared.

Knowledge of these bounds makes it possible for a designer/user to derive the (relative) message deadlines that are guaranteed. Taking the conventional off-line/rate-monotonic view that relative deadlines correspond to message periods, these bounds are the lower bounds of the periods that will be satisfied for every scenario being provably feasible. However, as observed before, and contrary to those results established under the restrictive periodic/off-line approach, our results apply to every possible case of arrival laws.

The algorithm selected to solve Ω_1 is CSMA-DCR. A description is given in annex 2. Again, the major objective of this work is not to describe a deterministic «Ethernet» protocol that has been published before (see references in annex 2). Rather, we seek to demonstrate the power of adversary arguments in solving open problems in the area of «hard real-time» or «reactive» computing. Consequently, the description given annex 2 is essentially meant to provide the reader with a flavor of what CSMA-DCR is. The focus really is on the fact that binary search is the basic algorithmic ingredient. Therefore, it is those essential properties germane to binary trees that really matter. All properties P used throughout this paper can be found in annex 1.

4.1. Strategy of adversary Z_0

Knowing source i's indices $t_i(1)$, $t_i(2)$,...,$t_i(v(i))$, and knowing that r consecutive indices need be used by source i to process a message ranked r^{th} in source i's waiting queue, adversary Z_0 will generate a collision before or at the same time when source i first attempts transmitting, this depending on the values of source i's indices. More specifically, the adversary will seek to generate the longest sequence of r consecutive intervals, transmitting its own messages, each interval being delimited by source i's indices. How can we find the expression of this longest sequence, denoted $a_i(r)$?

Consider a waiting queue containing x messages and let m_0 be the message ranked first. Given the convention adopted (see section 2.2) that a message being transmitted is ranked first in a waiting queue until fully transmitted, it is easy to see that the largest sojourn time for a message ranked x^{th} upon arrival corresponds to an earliest arrival time equal to end of transmission of m_0.

Consequently, we have to consider intervals half-open on the left in order to identify $a_i(r)$.

4.2. Bounds $B_{dcr}(i,r)$

Consider every interval $]t_i(d), t_i(d+1)]$, $d \in [1, \nu(i)]$ with $t_i(d+1)$ computed modulo q. We can apply property P_4 to compute the time needed to search such intervals. Indeed, with adversary Z_0, every such interval is full, that is Z_0 generates as many messages as there are indices available on every interval. We need to consider two cases, depending on whether an interval belongs to one tree or «spans» two consecutive trees.

* For $d \in [1, \nu(i)-1]$, the exact search time is $\varphi\{t_i(d), t_i(d+1)\}$ (see annex 1). Exactly $n_i(d+1) = t_i(d+1) - t_i(d)$ messages are sent while searching interval $]t_i(d), t_i(d+1)]$. Hence, $\lambda'(d, d+1)$, the exact time elapsed between end of transmission of message with index t(d) and end of transmission of message with index t(d+1) is :

$$\lambda'(d, d+1) = T_{n_i(d+1)} + s \cdot \varphi\{t_i(d), t_i(d+1)\} \quad ,$$

where notation T_k is as defined section 2.2. As exact message lengths are unknown, bound μ is used, which yields the following bound :

$$\lambda(d, d+1) = \mu \cdot n_i(d+1) + s \cdot \varphi\{t_i(d), t_i(d+1)\} \quad .$$

* For $d = \nu(i)$, not all indices are used, as interval $]t_i(\nu(i)), t_i(1)]$ «bridges» two consecutive trees. Exactly $n_i(1) = Q - t_i(\nu(i)) + t_i(1)$ messages are sent over interval $]t_i(\nu(i)), t_i(1)]$ and the exact search time is $\varphi\{t_i(\nu(i)), t_i(1)+q\}$. Addition is modulo q. Hence :

$$\lambda(\nu(i), 1) = \mu \cdot n_i(1) + s \cdot \varphi\{t_i(\nu(i)), t_i(1)\} \quad .$$

Let Λ_i be the sequence $\lambda(1,2)$, $\lambda(2,3)$,..., $\lambda(\nu(i),1)$ repeated ad infinitum. Sequence $a_i(r)$ is the longest sequence of r consecutive elements that is found over Λ_i. Exactly $\nu(i)$ sequences need be computed to identify $a_i(r)$. Hence :

$$B_{dcr}(i,r) = \max_{d \in [1,\nu(i)]} \left\{ \sum_{k=d}^{d+r-1} \lambda(k, k+1) \right\}$$

Observe that the actual sequence, as generated by Z_0, may contain messages that may be shorter than μ, and therefore could be different from sequence $a_i(r)$. For example, this would happen if small messages only were generated in the first interval of $a_i(r)$, and large messages only in some other interval, making that interval the first one in fact in the «real» $a_i(r)$. Nevertheless, the duration of any sequence of r consecutive elements instantiated by Z_0 is necessarily bounded by the duration of $a_i(r)$.

We will denote $n(r) = \sum_{j=1}^{\nu(i)} n_i(j)$ the exact number of messages transmitted over sequence $a_i(r)$.

Ranking r for a message is determined by the local arrival law and by the scheduling policy used

by its source. Given that messages carry no explicit deadline under problem Ω_1, first-in-first-out seems appropriate. Therefore, a message is ranked r^{th} whenever it enters a waiting queue that contains r-1 messages upon arrival.

4.3. Overhead, channel efficiency and timely influx density

Adversary Z_0 yields bounds $B_{dcr}(i,r)$ as shown above by maintaining the channel constantly busy with sending longest messages and generating repeated global collisions, hence generating repeated tree searches with Q active indices for every tree searched. In doing so, Z_0 maximizes the bounds but also maximizes the efficiency of the tree search algorithm as well as the channel efficiency (see property P_2). Let us examine the overhead incurred and the MAC-layer channel efficiency ratio achieved when using CSMA-DCR against Z_0 over $a_i(r)$. These variables, denoted $\Gamma(i,r)$ and $\Delta(i,r)$, have been introduced section 2.2.

The upper bound of $\Delta(i,r)$ is $\Delta'(i,r) = \mu.n(r)/B_{dcr}(i,r)$. The lower bound of $\Delta(i,r)$, denoted $\Delta''(i,r)$, is obtained by replacing μ with μ_0. This yields $\Delta''(i,r) = \mu_0.n(r)/B_{dcr}(i,r)$. Note however that bounds $B_{dcr}(i,r)$ are also reduced accordingly. That is, μ is replaced with μ_0 in the expressions of the λ's.

Consequently : $\Delta''(i,r) \leq \Delta(i,r) \leq \Delta'(i,r)$,

and $1 - \Delta'(i,r) \leq \Gamma(i,r) \leq 1 - \Delta''(i,r)$.

As illustrated by the numerical example given below, bounds $B_{dcr}(i,r)$ are guaranteed for channels that can be «highly» loaded, corresponding to $\Delta'(i,r)$. Hence, with CSMA-DCR, the adversary is facing the dilemma that it cannot simultaneously maximize bounds B and jeopardize Δ. Shooting for a poor Δ will minimize bounds B accordingly.

The upper bound on source i's timely influx density is $\Phi_{dcr}(i,r) = r/B_{dcr}(i,r)$.

4.4. A numerical example

Consider a channel with a number of sources for which 56 indices are made available. Consider source i, that has been allocated the following 3 indices : $t_i(1) = 18$, $t_i(2) = 41$, $t_i(3) = 50$. The upper bound on message duration is μ. One finds :

$\lambda(1,2) = 23\,\mu + 22$ s, $\lambda(2,3) = 9\,\mu + 9$ s, $\lambda(3,1) = 24\,\mu + 26$ s.
Let us have s = 40 μs.

4.4.1. Bounds for source i with highest channel efficiency

Let us choose $\mu = 300$ μs.

$B_{dcr}(i,1) = \lambda(3,1) = 8.24$ ms, $B_{dcr}(i,2) = \lambda(3,1)+\lambda(1,2) = 16.02$ ms,

$B_{dcr}(i,3) = \lambda(3,1)+\lambda(1,2)+\lambda(2,3) = 19.08$ ms, $B_{dcr}(i,4) = B_{dcr}(i,3) + \lambda(3,1) = 27.32$ ms, etc.

Channel efficiency, overhead and timely influx density

Let us consider r = 3. One finds n(3) = 56. Therefore : $\Delta'(i,3) = 0.88$ (and $\Gamma'(i,3) = 0.12$).

Let us apply these figures to Ethernet technology. We have established that, for the conditions considered, CSMA-DCR can sustain a (stable) MAC-layer throughput as high as 8.8 Mbits/s (i.e. 2,935 messages/second), in which case a bound of 19.08 ms (B(i,3)) is guaranteed for any message that is ranked 3rd in source i's waiting queue, this being obtained for a source i's timely influx density as high as $\Phi_{dcr}(i,3) = 3/19.08\ 10^{-3} = 157$ messages/second.

Let us observe in passing that an additional merit of CSMA-DCR is that it allows the operating of Ethernet technology under load conditions that usually yield complete collapse.

4.4.2. Bounds for source i with lowest channel efficiency

Let us choose $\mu_0 = 60$ µs. One finds :
$B_{dcr}(i,1) = 2.48$ ms, $B_{dcr}(i,2) = 4.74$ ms, $B_{dcr}(i,3) = 5.64$ ms, $B_{dcr}(i,4) = 8.12$ ms, etc.

Channel efficiency, overhead and timely influx density

For r = 3, one finds $\Delta''(i,3) = 0.596$ (and $\Gamma''(i,3) = 0.404$) which corresponds to 9,929 messages/second. Bound 5.64 ms is guaranteed for a source i's timely influx density as high as $\Phi_{dcr}(i,3) = 3/5.64\ 10^{-3} = 531$ messages/second.

5. SOLUTIONS, BOUNDS AND PERFORMANCE FOR PROBLEM Ω_2

In addition to the objectives we set ourselves when considering problem Ω_1, we now seek to demonstrate that adversary arguments are useful also to reason about aperiodic (unknown) message arrivals when messages carry strict relative deadlines that are revealed upon message arrivals only.

There is widespread belief in the off-line camp that such a problem cannot be solved rigorously, i.e., that it is impossible to prove the existence of bounds $B_{alg}(i,r)$. We show that this belief is mistaken.

Messages being tagged with deadlines, we now require that messages in waiting queues be ordered (locally) according to Earliest Deadline First (EDF) policy [LL73].

CSMA-DCR, which has the virtue of being deterministic, can obviously solve problem Ω_2. The bounds $B_{dcr}(i,r)$ established section 4, and guaranteed in the presence of adversary Z_0, remain unchanged when considering problem Ω_2, assuming that FIFO is the local scheduling policy used. This is because CSMA-DCR does not exploit the deadlines associated with messages. Of course, one could envision combining CSMA-DCR with local EDF policy. Corresponding bounds $B_{dcr}(i,r)$ would differ from those given section 4, for the reason that «being ranked r^{th}» for a message would have a different meaning (see further).

We will not explore this combination, for the reason that it is dominated by another algorithmic combination, namely local EDF and DOD/CSMA-CD (see annex 3), which explicitly exploits the deadline information provided on-line in a distributed manner. The major reason why DOD/ CSMA-CD is worth considering is rooted into an optimality result that extends the applicability of (on-line) EDF scheduling to the non-preemptive case. Optimality of non-preemptive EDF in

the absence of overload was first demonstrated for periodic and sporadic tasks in [JSM91]. This result has been recently extended to the case of aperiodic tasks [MR94].

Would DOD/CSMA-CD be a centralized algorithm, then it would follow that DOD/CSMA-CD is optimal, because centralized DOD/CSMA-CD is strictly equivalent to non-preemptive EDF. The binary tree search mechanism used in DOD/CSMA-CD serves the purpose of rendering DOD/CSMA-CD executable in a fully distributed manner. We can only speculate that this particular incarnation of distributed non-preemptive EDF is optimal (no proof developed yet).

To summarize, combining local EDF policy with DOD/CSMA-CD allows for implementing a global EDF policy, which cannot be obtained when combining local EDF policy with CSMA-DCR.

5.1. Strategy of adversary Z_0

Consider some source i, a message m ranked first in waiting queue upon arrival at time τ, with a relative deadline D. Transmission is attempted if the channel is idle. If a collision occurs, or if the channel is busy upon arrival at τ, current m's time index ρ_m is computed (see annex 3). If $\rho_m < F$, then m participates in the current time tree search ; otherwise, source i waits until a reference event occurs to compute a new time index for m.

Consider now that message m is not ranked first in i's waiting queue upon arrival. Rather, over the interval delimited by τ and the time of successful transmission of m, source i will service r-1 messages at least ahead of m. (How to interpret variable r is described section 5.2).

As EDF is the scheduling algorithm used locally, this means that r-1 messages at least are generated by source i over this interval, that have absolute deadlines smaller than or comparable to $E = \tau + D$. Two deadlines are said to be comparable whenever they yield the same time index. The bound on this interval duration is $B_{dod}(i,r)$.

In order to prohibit «early» transmission of m right after τ, the adversary keeps the channel busy, by sending messages back-to-back or by generating collisions resulting into epochs such that m is precluded from participating. Let π be the smallest (earliest) reference time such that m will certainly be associated a time index that will be current (i.e., m will be scheduled).

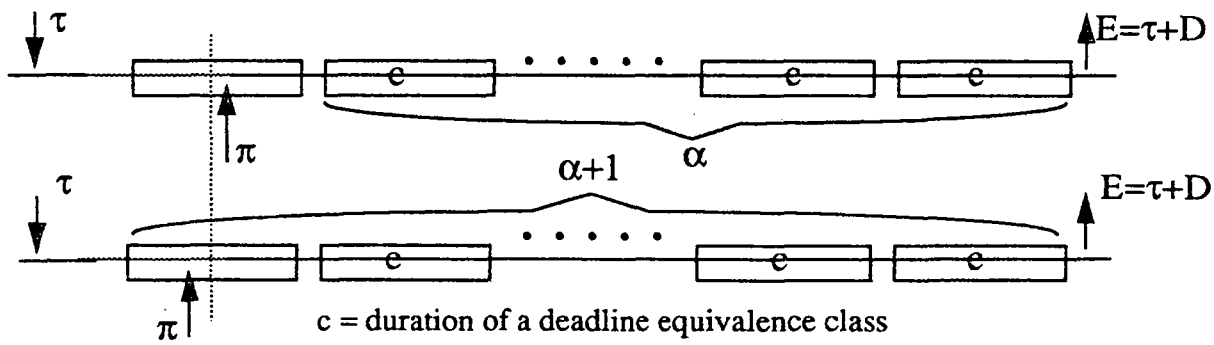


Figure 1 : role of laxity factor α

The rule used to compute time indices is given in annex 3. Given that such a computation involves a laxity variable v replaced with $\lceil v \rfloor$, the integer closest to v, and considering that $\lceil v + 1/2 \rfloor = v$, it is obvious that π verifies $\pi = E - (\alpha + 1/2)c$ (see figure 1). Indeed, for any infinitesimally small δ, at time $t = E - (\alpha + 1/2)c + \delta$, it follows that :

$$\rho_m = \rho_t(i,m) = \max\{0, \lceil \frac{(\alpha + 1/2)c - \delta}{c} \rfloor - \alpha\} + \rho^* + 1 = \max\{0,0\} + \rho^* + 1 = \rho^* + 1,$$

where ρ^* is the value of the time index searched last.

This guarantees that m will be part of the first time tree search initiated after time $E - (\alpha+1/2)c$. It is easy to verify that π cannot be smaller than $E - (\alpha+1/2)c$. The proof is by contradiction. Indeed, for any smaller value of π, one would have $\rho_m > \rho^*+1$. Consequently, the adversary could generate messages having deadlines smaller than E, resulting into time indices ranging between ρ^*+1 and ρ_m-1, thus prohibiting m from being scheduled, thus contradicting the definition of π.

Knowing the value of π, the adversary will delay as much as possible the start of the static tree search that will service message m after time π. The adversary achieves this by triggering a time tree search right before π, noted ts_1, thus excluding m from that search. Obviously, the time index used for that search is ρ^* (see figure 2). Furthermore, the clairvoyant adversary determines the arrival times of $Q - \nu(i)$ messages from knowledge of their relative deadlines, so that the $Q - \nu(i)$ time indices computed from the resulting absolute deadlines are equal to ρ^*+1.

As we seek to establish an upper bound for m's service time, we have to assume that Z_0 is able to involve source i as well in delaying start of service for message m. This translates into having $\nu(i)$ messages processed by source i, with time indices also equal to ρ^*+1. Consequently, static tree search ss_1 that follows ts_1 corresponds to a full static tree.

Under this scenario, termination time of ss_1 is the latest reference time such that ρ_m, as computed by m's source, will be the time index in use with ts_2. This reference time will be referred to as m's latest eligibility time, denoted t_m (see figure 2).

5.2. Bounds $B_{dod}(i,r)$

Let us first observe that being ranked r^{th} at source i for message m means that it could not be transmitted before its latest eligibility time t_m and that between t_m and the time when it will be effectively transmitted, source i will service r-1 other messages ahead of m.

Under problem Ω_2, messages carry explicit deadlines and EDF is the local policy used by every source. Hence, the above is equivalent to saying that over interval $[\pi, \tau + B_{dod}(r-1)]$, exactly $r - 1 + \nu(i)$ messages are serviced by source i ahead of m. By definition, these messages have absolute deadlines smaller than or comparable to $E = \tau + D$. Among these, exactly $\nu(i)$ messages have deadlines smaller than or comparable to E - c. These messages are processed during static tree search ss_1 (see below).

To express $B_{dod}(i,r)$ is equivalent to identifying a sequence of r consecutive intervals similar to sequence $a_i(r)$ as considered for expressing $B_{dcr}(i,r)$. However, trees of height $\log q + \log F$ should

be considered (rather than trees of height log q). We will not bother to do so. As shown below, a good upper bound on the search times involved can be easily found. This bound yields a simpler expression of $B_{dod}(i,r)$[2].

a) Bounding the time tree searches

Let $g' = \lceil r/v(i) \rceil$ be the number of time tree leaves to be searched to service m after time t_m. (Notation $\lceil x \rceil$ stands for ceiling value of x). Given Z_0's strategy, the exact number of time tree searches undertaken after time π is $g = g'+1$.

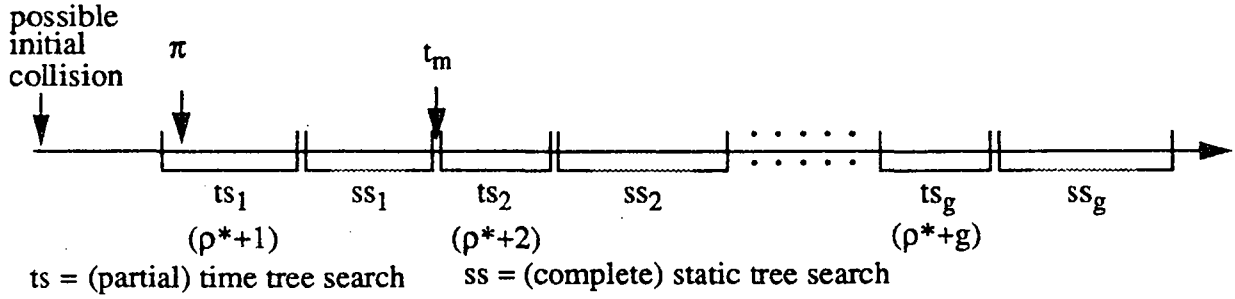


Figure 2 : worst-case scenario after time π

Given the rule used to compute time indices (see annex 3), which yields an increment of 1 exactly for those messages declared schedulable but not serviced yet, it follows that the g time tree leaves searched are adjacent. Hence the total time spent in searching these g consecutive leaves is $\varphi\{X_1,Y_2\}$, as given by property P_4, with $X_1 = \rho^*$ and $Y_2 = \rho^*+ g$, for some ρ^*.

Taking advantage of property P_2, the adversary spreads these g consecutive leaves over the greatest possible number of time trees, all trees being full, to the exception of the first and last ones. The adversary allocates leaves over these two trees according to some worst-case strategy. The number of leaves allocated to each of these two trees is any value comprised between 1 and $F - 1$. Let us write $\gamma = g/F$ and $h = g - \lfloor\gamma\rfloor F$. Depending on the value of h, the adversary has a choice between the following two strategies :

(c_1) construct $\lfloor\gamma\rfloor$ full trees and allocate the remaining h leaves evenly across the first and last trees, with $h < F$,

(c_2) construct $\lfloor\gamma\rfloor - 1$ full trees and allocate the remaining F+h leaves evenly across the first and last trees, with $0 \leq h < F$.

As $]X_1, F - 1]$ (resp. $[0, Y_2]$) is the full interval corresponding to the first (resp., the last) tree, it seems that we should compute a bound for the combined search times over both intervals, for every possible value of the pair (X_1,Y_2). We will not bother to do so. Indeed, for case (c_1), it can be observed that the overall search time is bounded by the time needed to search $\lceil\gamma\rceil$ full time trees. The same observation applies for case (c_2).

Let Ψ_1 be the bound on the overall time tree search needed to process g consecutive time indices.

---

2. We leave it as an exercise for the reader to devise the expression of a better bound

We have shown that this bound is as follows (function θ is defined annex 1, property P_1) :

$$\Psi_1 = \lceil \gamma \rceil \theta_F^F = \lceil \gamma \rceil (F-1) \quad .$$

b) Bounding the static tree searches

With adversary Z_0, all static trees are full, possibly to the exception of the last one. Let x be the index used by source i to service m. We have $x=t_i(v(i)-\omega)$, with $\omega = g'v(i) - r$.

Active indices in the last tree range over complete interval [0,x]. Active indices in other trees range over complete interval [0,Q-1]. Hence, n(r), the corresponding exact number of messages transmitted (or active indices) after time π is as follows (see property P_4) :

$$n(r) = (1 + g' - 1)\,Q + x + 1 = g'\,Q + x + 1.$$

Total static tree search time Ψ_2 is as follows (see property P_4) :

$$\Psi_2 = g' (\varphi^* (Q-1) + \varepsilon) + \varphi^* (x) \quad .$$

c) Overall bound

Let us write $\Psi(i,r) = \Psi_1 + \Psi_2$. Then, the result :

$$B_{dod}(i,r) = \pi - \tau + s \cdot \Psi(i,r) + T_{n(r)}$$

$$\boxed{B_{dod}(i,r) = D - (\alpha + 1/2)\,c + s \cdot \Psi(i,r) + T_{n(r)}}$$

The expression of $B_{dod}(i,r)$ yields an obvious condition on α for m's schedulability. Integer α is the smallest integer verifying the following condition :

$$\alpha \geq [s\Psi(i,r) + T_{n(r)}]/c - 1/2 \quad .$$

Knowledge of feasibility conditions allows for the checking of any given set of messages/ deadlines. Whenever these conditions are satisfied, it is proven that :

(i) the channel cannot enter overload situations,
(ii) bounds $B_{dod}(i,r)$ are guaranteed,
whatever the message arrival times will be. This is how DOD/CSMA-CD can be put to work in practice. This reflects our original motivation, i.e. to take advantage of the demonstrated optimality of centralized non-preemptive EDF in the absence of overload. As indicated before, we consider that DOD/CSMA-CD is an interesting decentralized version of centralized non-preemptive EDF.

5.3. Overhead, channel efficiency and timely influx density

Adversary Z_0 does not necessarily maximize the channel efficiency when maximizing the bounds. Indeed, between time τ and time π, Z_0 is free to keep the channel busy using any strategy ranging between the following two extremes (see property P_2) :

- Q-leaf collisions (static trees) and F-leaf collisions (times trees) only, which yields the highest channel efficiency,
- 2-leaf collisions only, which yields the lowest channel efficiency.

Source i's strategy over interval $[\tau,\pi[$ also influences channel efficiency. Bounds on channel efficiency over interval $[\tau,\pi[$ can be easily computed. Let us focus on those bounds corresponding to interval $[\pi, \tau+B_{dod}(i,r)]$, which is consistent with the analysis given for CSMA-DCR.

The MAC-layer channel efficiency ratio $\Delta(i,r)$ is thus defined as the channel efficiency achieved over interval $[\pi, \tau+B_{dod}(i,r)]$. It follows that :

$$\Delta(i,r) = T_{n(r)}/[\tau + B_{dod}(i,r) - \pi] = T_{n(r)}/[s\,\Psi(i,r) + T_{n(r)}].$$

Variable n(r) is as defined section 5.2. Using the notations introduced section 4.3, lower bound $\Delta''(i,r)$ is obtained considering messages of length μ_0, and upper bound $\Delta'(i,r)$ is obtained considering messages of length μ.

Of course, $\Gamma(i,r) = 1 - \Delta(i,r)$. A detailed analysis of $\Phi_{dod}(i,r)$ falls out of the scope of this paper (such an analysis is related to feasibility conditions, to be addressed in a forthcoming paper). Here, we will simply assume that after time π, messages arrive according to the worst-case scenario as generated by Z_0, which yields $B_{dod}(i,r)$. Consequently, the upper bound on source i's timely influx density is $\Phi_{dod}(i,r) = [r + \nu(i)]/[\tau + B_{dod}(i,r) - \pi]$. Note however that this expression is correct provided that $\nu(i)$ messages arrive «in time» so as to be processed during static tree search ss_1.

5.4. A numerical example

Parameter values used are those given for the CSMA-DCR example (section 4.4). In addition, we select the following values : F = 8, c = 17 ms, and D = 60 ms for all messages. Consider further that feasibility conditions computed for other sources have led to choose $\alpha = 3$. The possibility of computing $B_{dod}(i,r)$ results in determining the highest value of r for which the constraint D = 60 ms is always satisfied by source i.

5.4.1. Bounds for source i with highest channel efficiency

Recall that we have $\mu = 300$ µs. Variable $d = D - (\alpha + 1/2)c = 0.5$ ms.

$B_{dod}(i,1) = 0.5 + 0.04\,[64 + \varphi^*(18)] + 75 \times 0.3 = 26.44$ ms,

$B_{dod}(i,2) = 0.5 + 0.04\,[64 + \varphi^*(41)] + 98 \times 0.3 = 34.22$ ms,

$B_{dod}(i,3) = 0.5 + 0.04\,[64 + \varphi^*(50)] + 107 \times 0.3 = 37.28$ ms,

$B_{dod}(i,4) = 0.5 + 0.04\ [121 + \varphi^*(18)] + 131 \times 0.3 = 45.52$ ms,

$B_{dod}(i,5) = 0.5 + 0.04\ [121 + \varphi^*(41)] + 154 \times 0.3 = 53.3$ ms,

$B_{dod}(i,6) = 0.5 + 0.04\ [121 + \varphi^*(50)] + 163 \times 0.3 = 56.36$ ms,

$B_{dod}(i,7) = 0.5 + 0.04\ [200 + \varphi^*(18)] + 187 \times 0.3 = 65.48$ ms.

The highest acceptable value for r is 6. Relative deadline D = 60 ms is always satisfied for every possible arrival scenario that does not lead source i to process more than 9 messages between time τ + 0.5 ms and time τ + 56.36 ms. This yields maximum source i's timely influx density $\Phi_{dod}(i, 6) = 9/55.86\ 10^{-3} = 161$ messages/second.

Channel efficiency

Let us choose r = 3 again. Then, n(3) = 107. This yields :

$$\Delta'(i,3) = 107 \times 0.3 / [107 \times 0.3 + 0.04 \times 117] = 0.872.$$

Bound $B_{dod}(i,3)$ given above (37.28 ms) is obtained along with a MAC-layer channel efficiency ratio as high as 0.872 (i.e. for a global arrival rate equal to $107/36.78\ 10^{-3} = 2{,}909$ messages/ second).

5.4.2. Bounds for source i with lowest channel efficiency

Bounds are established as those computed above, by replacing $\mu = 300$ µs with $\mu_0 = 60$ µs in the expression of $T_{n(r)}$. Therefore :

$B_{dod}(i,1) = 8.44$ ms, $B_{dod}(i,2) = 10.7$ ms, $B_{dod}(i,3) = 11.6$ ms,

$B_{dod}(i,4) = 14.08$ ms, $B_{dod}(i,5) = 16.34$ ms, etc.,

$B_{dod}(i,28) = 59.48$ ms, $B_{dod}(i,29) = 61.74$ ms.

The highest acceptable value for r is 28. Relative deadline D = 60 ms is always satisfied for every possible arrival scenario that does not lead source i to process more than 31 messages between time τ + 0.5 ms and time τ + 59.48 ms. This yields This yields maximum source i's timely influx density $\Phi_{dod}(i, 28) = 31/58.98\ 10^{-3} = 525$ messages/second (from source i).

Channel efficiency

Again, consider r = 3 and n(3) = 107. This yields :

$$\Delta''(i,3) = 107 \times 0.06 / [107 \times 0.06 + 0.04 \times 117] = 0.578.$$

Bound $B_{dod}(i,3)$ given above (11.6 ms) is obtained along with a MAC-layer channel efficiency ratio not lower than 0.578 (i.e. for a global arrival rate equal to $107/11.1\ 10^{-3} = 9{,}639$ messages/ second.

Applying these results to Ethernet technology, we have demonstrated that, for the conditions considered, with DOD/CSMA-CD, the bounds and the timely influx densities given are guaranteed for a channel sustaining a (stable) MAC-layer throughput as high as 8.72 Mbits/s.

Again, for regular Ethernets, such speeds usually yield total collapse.

6. CONCLUSION

We have explored two scheduling problems that arise in the context of «hard real-time» communications taking place over a distributed broadcast channel. We have described two on-line algorithms that are solutions to these problems.

CSMA-DCR (CSMA-CD with Deterministic Collision Resolution) is an algorithm that is appropriate for handling real-time messages that have implicit deadlines, as often considered by designers of token passing/polling protocols or time-slotted synchronous networks.

DOD/CSMA-CD (Deadline Oriented Deterministic CSMA-CD) is an algorithm that is appropriate for handling real-time messages tagged with arbitrary and strict deadlines.

Bounds on message transmission delays and on collision resolution latency are given for both algorithms.

Such bounds are interesting for the reason that they have been established without making any assumption w.r.t. message arrival laws and timings. They have been established using adversary arguments, which seems to be a fruitful technique when trying to solve a «hard real-time» problem for aperiodic arrivals and arbitrary deadlines. We believe that designs similar to ours, conducted under minimal advance knowledge, are the only correct designs in the case of critical hard real-time systems. Proven bounds are always valid, contrary to those established under conventional off-line approaches, that rest on too many clairvoyance assumptions, that can be easily violated at run-time.

Too many flawed arguments have been and are being used by backers of off-line approaches. We view this paper as a contribution to a much needed clarification of the off-line versus on-line debate.

Regarding the applicability of our results, it is important to understand that CSMA-DCR and DOD/CSMA-CD can be implemented «on top of» or «within» conventional Ethernet or ISO/OSI 8802/3 chips. Recall that CSMA-DCR and DOD/CSMA-CD are «plug-to-plug» compatible with standard CSMA-CD technology. This technology being so much widespread, the cost involved in bringing slight modifications to current VLSI chip designs should not be an issue, compared to the size of the markets waiting for real proven solutions to the hard real-time communication problem. Although not shown here, CSMA-DCR and DOD/CSMA-CD algorithms can be used over arbitrarily «fast» channels (e.g., 100 Mbits/s Ethernets that implement a ternary channel). Obviously, these algorithms can be efficiently implemented over «slow» or «small» networks, such as those used in transportation systems (e.g. cars, trains, aircrafts, spaceships) or in ground-based critical systems (e.g., nuclear plants, chemical plants).

Acknowledgments

We would like to thank the REFLECS group at INRIA for putting up with early presentations of this work. Special thanks go to Philippe Jacquet and Paul Mühlethaler for valuable discussions on function ξ_k^t.

References

[BFR87] J. Boudenant, B. Feydel, P. Rolin, «An IEEE 802.3 compatible deterministic protocol», IEEE INFOCOM'87, San Francisco, April 1987, 573-579.

[GR93] B. di Gennaro, N. Rivierre, «Schedulability analysis of multiaccess protocols for distributed real-time systems», 4th IEEE Workshop on Future Trends in Distributed Computing Systems, Lisbon, Portugal, Sept. 1993, 332-338.

[JSM91] K. Jeffay, D.F. Stanat, C.U. Martel, «On non-preemptive scheduling of periodic and sporadic tasks», IEEE Real-Time Systems Symposium, San Antonio, Dec. 1991, 129-139.

[K et al.89] H. Kopetz, A. Damm, Ch. Koza, M. Mulazzani, W. Schwabl, Ch. Senft, R. Zainlinger, «Distributed fault-tolerant real-time systems : The MARS approach», IEEE Micro, Feb. 1989, 25-40.

[KG85] J. Komlós, A.G. Greenberg, «An asymptotically nonadaptive algorithm for conflict resolution in multiple-access channels», IEEE Trans. on Information Theory, Vol. IT-31, 2, March 1985, 302-306.

[L78] L. Lamport, «Time, clocks and the ordering of events in a distributed system», Communications ACM, Vol. 21, 7, July 1978, 558-565.

[LL73] C.L. Liu, J.W. Layland, «Scheduling algorithms for multiprogramming in a hard real-time environment», Journal ACM, Vol. 20, 1, Jan. 1973, 46-61.

[LL77] G. Le Lann, «Distributed systems - Towards a formal approach», IFIP Congress, Toronto, Canada, 1977, 155-160.

[LL87] G. Le Lann, «The 802.3 D protocol : a variation on the IEEE 802.3 standard for real-time LANs», INRIA Technical Report, 1987, 12 p.

[LR84] G. Le Lann, P. Rolin, «Procédé et dispositif pour la transmission de messages entre différentes stations à travers un réseau local à diffusion», French patent n°8416957, Nov. 1984.

[LS86] J.P. Lehoczky, L. Sha, «Performance of real-time bus scheduling algorithms», ACM Performance Evaluation Review, 14, 1, May 1986, 44-53.

[MR94] P. Muhlethaler, N. Rivierre, «Optimality and non-preemptive scheduling revisited», INRIA Research Report, to appear in 1994.

[PS91] K. Prodromides, W.H. Sanders, «Performability evaluation of CSMA/CD and CSMA/DCR protocols under transient fault conditions», IEEE Trans. on Reliability, Vol. 42, 1, March 1993, 116-127.

[R93] J. Rushby, «Formal methods and the certification of critical systems», SRI Technical Report, to appear in 1993.

[SM89] J.K. Strosnider, T.E. Marchok, «Responsive deterministic IEEE 802.5 token ring scheduling», Journal of Real-Time Systems, Vol. 1, Sept. 1989, 133-158.

[SSL89] B. Sprunt, L. Sha, J.P. Lehoczky, «Aperiodic task scheduling for hard real-time systems», Journal of Real-Time Systems, Vol. 1, 1, June 1989, 27-60.

Glossary

CSMA-CD	Carrier Sense Multiple Access - Collision Detection
CSMA-DCR	CSMA - Deterministic Collision Resolution
DOD/CSMA-CD	Deadline Oriented Deterministic/CSMA-CD
EDF	Earliest Deadline First
Z_0	Unrestricted adversary
T_k	Physical transmission time for k messages
s	Channel slot time duration
$B_{alg}(i,r)$	Upper bound on message service time where alg is the multiaccess/ scheduling algorithm used, i is the message source considered and r is the rank in source i's waiting queue of the message considered
$\Delta(i,r)$	MAC-layer channel efficiency, where i and r are as introduced with $B_{alg}(i,r)$
$\Phi(i,r)$	Upper bound on source i's timely influx density (messages from source i per time unit)
$\Gamma(i,r)$	MAC-layer overhead, where i and r are as introduced with $B_{alg}(i,r)$
μ	Upper bound on message length
μ_0	Lower bound on message length
Q	Number of static indices used (Q-1 is the highest index)
q	Smallest power of 2 greater than or equal to Q
$t_i(j)$	j^{th} static index allocated to source i, indices being ranked by increasing

	values
$\nu(i)$	Number of static indices allocated to source i
c	Duration of a deadline equivalence class
F	Time tree size, in number of leaves (time indices), a power of 2
ξ_k^t	Upper bound on search time for isolating k leaves in a t-leaf balanced binary tree
θ_k^t	Tight upper bound on search time for isolating k leaves in a t-leaf balanced binary tree ; $\theta_k^t = \min(\xi_k^t, t-1)$
u_k	Tree search efficiency
α	Laxity factor
ρ	Time index
D	Relative deadline of a message
E	Absolute deadline ($E = \tau + D$) of a message arrived at time τ
$\varphi\{X,Y\}$	Exact search time over full interval]X,Y]
$\varphi^*\{Y\}$	Exact search time over full interval]0,Y]
ε	Exact search time to «leave a tree» after full interval [0,z] has been searched, with $z < Q$

Annex 1 : essential properties

The balanced binary tree properties useful in establishing bounds $B_{alg}(i,r)$ are given below. Search times are expressed in tree nodes visited, for t-leaf trees. A tree node corresponds to a collision or to an empty channel slot (search of an empty sub-tree). Variable t is equal to F when considering time trees, and t is equal to q when considering static trees. Notation log stands for log base 2.

In this paper, we adopt the convention that the «winning» set is the subset formed with «small» indices (e.g., subset [0, t/2[for the first dichotomy). Also, tree searches start from the left (that is from index/leaf # 0 for the first dichotomy).

Property P_1

A good upper bound ξ_k^t on search time for isolating k leaves in a t-leaf balanced binary tree was given by different authors (see [KG85] for a summary). This bound is as follows, for $1 < k \leq t$ (integer part of x denoted $\lfloor x \rfloor$) :

$$\xi_k^t = k + \lfloor k \log (t/k) \rfloor \cdot$$

It can be observed that for certain values of k, this bound is not tight. This can be established by considering the first derivative of continuous function $C\xi_k^t = k\,[1 + \log (t/k)] - 1$ defined over the continuous interval [2,t]. This derivative is $1 + [ln (t/k) - 1] / ln2$ (where ln stands for neperian log), which is positive for $k \leq 2t/e$ and negative for $2t/e < k \leq t$. Therefore, $C\xi_k^t$ is a strictly increasing function of k for $k \leq 2t/e$ and a strictly decreasing function of k for $k > 2t/e$. Observing that $0 < \xi_k^t - C\xi_k^t \leq 1$, that $\xi_2^t = 2 \log t$ and that $\xi_t^t = t$, it follows that $\xi_k^t \geq t - 1$ for $k_0 \leq k \leq t$, for some k_0 comprised between 2 and 2t/e.

As it is known that the search time cannot be greater than t-1, we will use the following bound :

P_1 : $$\theta_k^t = \min \{t - 1, \xi_k^t\} \cdot$$

Consequently, if T_k is the time needed to physically transmit (without collisions) the k messages (i.e. leaves) searched, then the time needed to fully resolve a k-message collision is bounded by $T_k + s\theta_k^t$.

Property P_2

P_2 : Tree search efficiency has a lower bound that is an increasing positive function of k, the number of indices searched (i.e., messages involved in a collision).

The lower bound of tree search efficiency is $u_k = k / (k + \theta_k^t)$.

Proof

Given that u_k is a discontinuous function, we proceed by proving that u_{k+1} is always greater than u_k. The sign of $u_{k+1} - u_k$ is the sign of $G = (k+1)\,\theta_k^t - k\;\theta_{k+1}^t$.

Let us first consider function $D = (k+1)\,\xi_k^t - k\xi_{k+1}^t = (k+1)\lfloor X_k \rfloor - k\lfloor X_{k+1} \rfloor$, where $X_k = k\;\log(t/k)$. We seek to establish that $D > 0$, that is :

$$\lfloor X_k \rfloor + k\{\lfloor X_k \rfloor - \lfloor X_{k+1} \rfloor\} > 0 .$$

Knowing that $\lfloor U \rfloor - \lfloor V \rfloor = \lfloor U - V \rfloor$ or $\lfloor U - V \rfloor + 1$, and selecting the most constraining equality, we have to establish :

$$(\omega) \qquad \lfloor X_k \rfloor + k\lfloor \log(k+1) - \log t + k\;\log(1 + 1/k) \rfloor > 0 \quad .$$

For $k > 1$, it is known that $\log(1 + 1/k) > 1/k$. Therefore, (ω) is demonstrated if the following holds true : $\lfloor X_k \rfloor + k - k\lfloor \log(t/k) + \log[k/(k+1)] \rfloor > 0$.

Knowing that $\lfloor kV \rfloor \geq k\lfloor V \rfloor$, it suffices to show that :

$$k\lfloor \log(t/k) \rfloor + k - k\lfloor \log(t/k) + \log[k/(k+1)] \rfloor > 0 \quad ,$$

which is trivially true. Hence, $D > 0$ for $1 < k \leq t$, which implies $u_{k+1} > u_k$ when considering function ξ.

For $k \geq k_0$, $\theta_k^t = t - 1$, whereas $\xi_k^t \geq t - 1$. In order to complete the demonstration that $G > 0$, two cases need be considered only, as ξ_k^t is an increasing function of k (see property P_1).

* **Case** $\theta_k^t = \xi_k^t < t - 1$ **and** $\theta_{k+1}^t = t - 1 < \xi_{k+1}^t$

In this case, we have $\xi_{k+1}^t > t - 1$. The proof given above applies a fortiori when replacing ξ_{k+1}^t with t - 1 in the expression of D, which yields :

$$G = (k+1)\,\xi_k^t - k(t-1) > D > 0 \quad .$$

Hence, $u_{k+1} > u_k$.

* **Case** $\theta_k^t = \theta_{k+1}^t = t - 1$

The sign of $u_{k+1} - u_k$ is the sign of : $G = (k+1)\,\theta_k^t - k\;\theta_{k+1}^t = t - 1$.

Hence, $u_{k+1} > u_k$.

End of proof.

It follows that Δ_k, the lower bound of channel efficiency $\left[\Delta_k = T_k/(T_k + s\theta_k^t)\right]$ also is an

increasing positive function of k. This is easily established by developing $\Delta_{k+1} - \Delta_k$. The sign of $\Delta_{k+1} - \Delta_k$ is the sign of $m(k+1)\theta_k^t - mk\theta_{k+1}^t$ (where $m = T_k/k$), that is the sign of G.

Property P_3

P_3 : ε, the exact search time to «leave a tree», index z being the last one searched and interval [0,z] being full, is the exact search time over (empty) interval [z+1, t-1], that is :

$$\varepsilon = \sigma(W) - 1,$$

where W = t - z and $\sigma(W)$ is the number of powers of 2 in W.

Proof

W = t - z is polynomial in powers of 2. Each power of 2 represents a binary sub-tree. Every such sub-tree being empty, search time is 1 (the root) per sub-tree. The only sub-tree being non empty is the one that contains index # z. That sub-tree search (which is 1) should not be accounted for. Hence the result.
End of proof.

Property P_4

P_4 : The exact search time over full interval]X,Y], denoted $\varphi\{X,Y\}$ is as follows :

$$\varphi\{X, Y\} = Y - X + \sigma(X) - \sigma(Y) + \delta[z + \varepsilon + \log t - \sigma(z)],$$

with $\delta = 0$ if X and Y belong to the same tree, $\delta = 1$ if X and Y belong to two adjacent trees. Variables $\sigma(z)$ and ε are as defined in property P_3. Index z is the last index searched in the first tree, when considering interval]X,Y] spanning two adjacent trees.

Let us first establish the following **property P'_4** :

P'_4 : The exact search time over full interval [0,V], denoted $\varphi^*(V)$, is as follows :

$$\varphi^*(V) = \log t + V - \sigma(V) \quad .$$

Proof for P'_4 :

Any integer V can be developed as follows :

$$V = 2^{a_1} + 2^{a_2} + \ldots + 2^{a_{\sigma(V)}} \quad .$$

Each power of 2 corresponds to a full binary sub-tree (having all its leaves searched) and to the first leaf of the subsequent sub-tree. Consider a tree T_h of height h = log t (see figure A1). Consider in T_h the leftmost tree T_{a_1} of height a_1 and consider interval $]0, 2^{a_1}]$, decomposed into interval $]0, 2^{a_1}[$ and leaf # 2^{a_1}. Let $\varphi^*(2^{a_1})$ be the time needed to search interval $]0, 2^{a_1}]$, starting from leaf # 0. This time is equal to the time needed to search (full) T_{a_1} (think of leaf # 2^{a_1} replaced with leaf # 0), starting from root a_1. That is : $\varphi^*(2^{a_1}) = 2^{a_1} - 1$. This reasoning can be iterated over V, which yields :

$$\varphi^*(2^{a_1}) = 2^{a_1} - 1$$

$$\varphi^*(2^{a_2}) = 2^{a_2} - 1$$

. . . .

$$\varphi^*(2^{a_{\sigma(V)}}) = 2^{a_{\sigma(V)}} - 1 \quad .$$

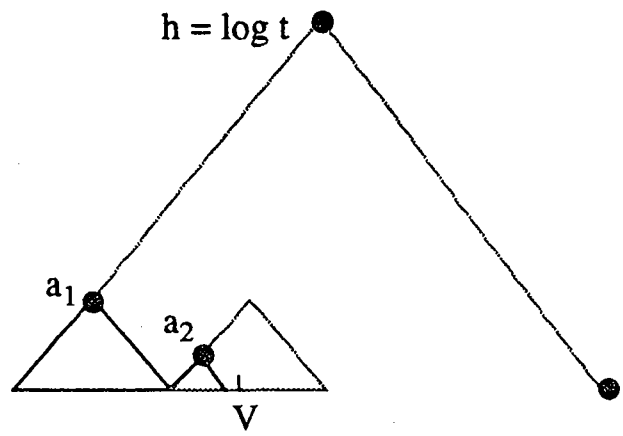


Figure A1 : a binary tree representation of an integer

The time needed to reach leaf # 0, starting from the root of T_h, has to be accounted for. By summing up, we obtain the desired result :

$$\varphi^*(V) = \log t + V - \sigma(V)$$

In the particular case when V = t - 1 (full tree), it is known that σ(V) = log t. Therefore :

$$\varphi^*(t-1) = t-1.$$

End of proof.

Proof for P_4

Consider first X and Y belonging to the same tree. Obviously, we can write $\varphi\{X, Y\} = \varphi^*(Y) - \varphi^*(X)$.

It follows that : $\varphi\{X, Y\} = Y - X + \sigma(X) - \sigma(Y)$.

Consider now X and Y belonging to two adjacent trees. We have $\varphi\{X, Y\} = \varphi\{X, z\} + \varepsilon + \varphi^*(Y)$. Using the results established previously, the general expression of φ{X,Y} is easily found.

End of proof.

Annex 2 : the CSMA-DCR algorithm

Principles

CSMA-DCR is a deterministic variation of the Ethernet standard or of the ISO/OSI 8802/3 standard. CSMA-DCR stands for Carrier Sense Multiple Access with Deterministic Collision Resolution (or 802.3D for Deterministic 802.3). This algorithm was first described in [LR84]. It resembles other deterministic tree protocols. Presentation of CSMA-DCR salient features can be found in [BFR87] and [L87]. An analysis of CSMA-DCR performance in faulty channels can be found in [PS91]. Various options of this algorithm have been implemented in a number of products [e.g., the FACTOR factory LAN from CAP-APTOR (France), the Récital/Ethernet D-103 VME board from Dassault Electronique (France), the 82596 Ethernet chip from Intel (USA)].

CSMA-DCR is fully compatible with the ISO/OSI 8802/3-Ethernet standards. In other words, conventional CSMA-CD chips and CSMA-DCR chips can co-exist and exchange messages over the same shared medium. CSMA-DCR is compliant with standard interfaces defined for OSI layers 1 and 2^+ (physical and logical link layers). The reason is that the variation simply consists in replacing the probabilistic binary exponential back-off algorithm with a deterministic binary tree search algorithm. Clearly, this modification has no impact on the other elements of the CSMA-CD protocol or chips.

Binary tree protocols belong to the well explored class of tree protocols, that have been shown to exhibit very interesting properties (stability and efficiency in particular).

While no collision occurs, CSMA-DCR behaves exactly like CSMA-CD. Upon collision occurrence, CSMA-DCR iteratively dichotomizes the set of message sources. Only an upper bound on the number of sources need be known at configuration time.

Description

Every message source uses the following run-time variables :

* channel status, a ternary variable (the only global one), representing the following channel states :
  - channel busy, normal transmission underway,
  - channel busy, collision,
  - channel idle,

* a set unique indices (possibly only 1).

Of course, a source that is allocated $t > 1$ indices does not «collide with itself» when it has several messages pending. If a source experiences a collision while it is transmitting one of its messages, and the channel was initially idle, then, it will transmit up to t pending messages during the resulting epoch. To others, this will look like competing with up to t other sources.

CSMA-DCR comes under different options, selected at configuration time, represented by the

following variables :

* a boolean, indicating whether the tree mode selected is «general» (full tree search starting from the root) or «leaf» (tree search starting from the rightmost or leftmost «leaf», i.e. the smallest or the highest index)

* a boolean, indicating whether the epoch entry mode selected is «blocked» (only those messages involved in the original collision can be transmitted within the corresponding epoch) or «open» (messages transmitted within the epoch are those involved in the original collision plus those that arrive later but in time, i.e. before their index has been searched).

Under the leaf mode option, CSMA-DCR performs at least as well as the SS-TDMA or polling protocols. Property P_2 demonstrates that higher efficiency is obtained when minimizing the number of tree searches needed to transmit a given number of messages. Hence, the open entry mode is more efficient than the blocked mode. In the sequel, we consider the general mode and the open epoch entry mode exclusively.

Every source follows the CSMA-DCR algorithm, even if inactive (no message pending).

Let Q be the total number of consecutive indices allocated to the message sources. An index is the (static) name of a balanced binary tree leaf. Let v_i be the number of indices allocated to source i. Let q be the smallest power of 2 greater than or equal to Q. Consider a collision occurs. Let k be the number of messages that need be isolated among Q (potential messages) during the resulting epoch, i.e. the resulting binary search. Each source knows which indices it owns and is able to locally sense the ternary CSMA-CD channel (channel idle, channel jammed (collision), regular (clear) transmission). Upon collision occurrence, every source dichotomizes the (virtual) binary tree which has Q leaves potentially active among a total of q. Only one of the two resulting subsets (the «winning» set) is allowed to remain active. The «winning» sources are allowed to try again as soon as the channel gets back to idle (i.e. in s time units). The iteration is reversed whenever the winning set either is empty or contains exactly 1 active source (that transmits its message collision-free). Obviously, the complete resolution of a collision is conducted in bounded time (see property P_1, annex 1).

In this paper, we adopt the convention that the «winning» set is the subset formed with «small» indices (e.g., subset [0, q/2[for the first dichotomy). Also, tree searches start from the left (that is from index/leaf # 0 for the first dichotomy).

Properties of search times in balanced binary trees are used to develop the expressions of the bounds (see annex 1).

Example

An example for a 16-index, 1 index/source configuration is shown figure A2. If we assume that messages have an equal duration of 6s (s is channel slot time) and that s = 40 μs, it follows that the 6-message collision is fully resolved in exactly 1.8 ms after entering the epoch (i.e. start of channel slot #1).

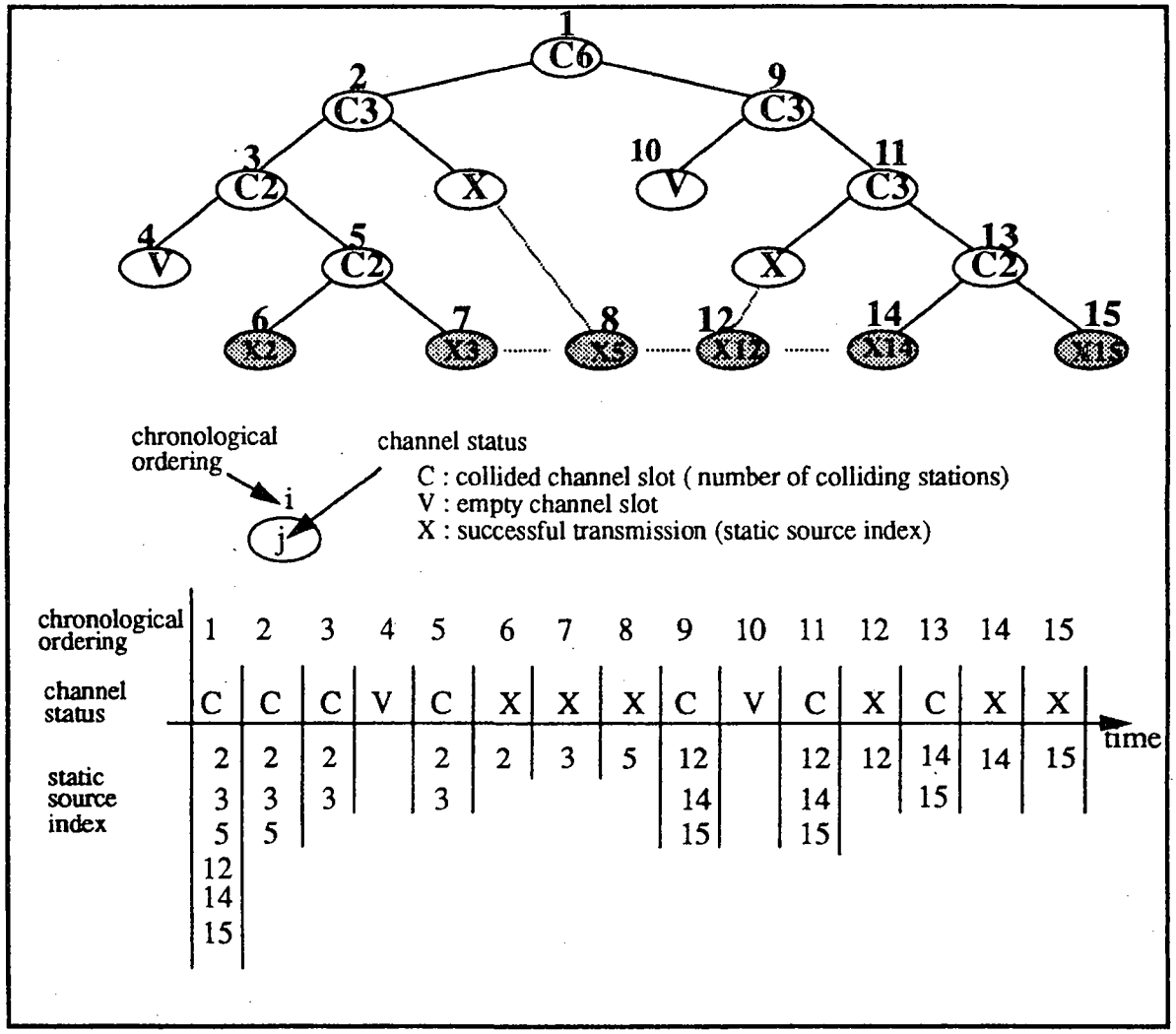


Figure A2 : a CSMA-DCR epoch

Annex 3 : the DOD/CSMA-CD algorithm

Principles

DOD/CSMA-CD is similar to CSMA-DCR. The major difference stems from using indices that are computed on-line, rather than resting solely on indices that are pre-allocated to sources.

Indices computed dynamically are called time indices. They represent deadline equivalence classes. All messages that are (dynamically) assigned the same time index have «comparable» deadlines (see further). A virtual time tree is defined, of size F. Tree searches being more efficient with full trees (see property P_2), it is wise to pick up F as a power of 2 (our assumption).

Consider a source x that attempts to transmit a message m at time τ (first attempt). Message m has a relative deadline D. Let $E = \tau + D$ be its absolute deadline (see figure 1). We need to consider two cases.

Case «channel idle»
At time τ, sensing the channel idle, source x starts transmitting m. If no collision occurs, m is done. If a collision occurs, x proceeds as described below.

Case «channel busy»
«Channel busy» means that a collision has just occurred or has occurred in the past, not fully resolved yet (i.e., the corresponding time tree search still is in progress).

In order to determine whether or not it is allowed to try transmitting m, source x must compute ρ_m, the time index associated with m. Obviously, a (dynamic) reference time that is common to all sources need be defined.

A reference time is the time of occurrence of a reference event. There are two types of reference events :

- initial collision, i.e. the opening of a time tree search,
- termination of a static tree search (à la CSMA-DCR).

The current reference time is the reference time of the most recent reference event. A static tree search is initiated when a collision occurs while a time tree leaf is being searched. This is an indication that several sources are trying to transmit a message belonging to that deadline equivalence class. Time tree searches and static tree searches are conducted using CSMA-DCR.

Description

A scheduling horizon H is implicitly defined. This is needed to approximate perfect (centralized) EDF. Recall that in any real-time system, processors (and therefore message sources) maintain a common global time reference, via synchronized clocks (this need is not specifically related to the utilization of DOD/CSMA-CD). In the sequel, we ignore issues related to the impossibility of

having perfectly synchronized clocks. In general, with broadcast channels, the precision (denoted η) maintained by a set of synchronized clocks is «good», i.e. in the order of a few channel slot times, even with deterministic synchronization algorithms. The fact that distributed time precision η cannot be zero does not jeopardize the behavior of DOD/CSMA-CD. At worst, this can only yield deadline assignment inversions limited to an amplitude of η. It might be worth noticing that reference times used with DOD/CSMA-CD are not obtained by reading a clock. Such times are defined after channel state transitions.

A source that has several messages pending ranks them by increasing deadlines (local EDF (Earliest Deadline First [LL73])).

The size of a deadline equivalence class, which is denoted c, is first determined. This defines H=cF. However, DOD/CSMA-CD uses F rather than H to approximate perfect EDF. There is a trade-off decision involved in choosing c. Too small a value for c, for a given F, would reduce H, this possibly resulting into deadline misses for feasible scenarios. Augmenting F would result into increased overhead for searching the time tree. Too large a value for c would jeopardize the objective of approximating perfect EDF. In the sequel, we do not consider any particular choice for c. We simply want to ensure that all time tree leaves can be searched.

Consider an initial collision occurring at reference time t_0. All sources involved compute their current time indices. For a message with absolute deadline E, time index ρ_0 is as follows :

$$\rho_0 = \lceil (E - t_0) / c \rfloor - \alpha ,$$

where $\lceil k \rfloor$ stands for integer closest to k. Constant α (a positive integer) is a laxity factor, known to all sources, whose purpose is to permit transmission attempts that are not necessarily constrained to take place within the last (before deadline) deadline equivalence class.

The time tree search is conducted à la CSMA-DCR by all sources which have computed $\rho_0 < F$. That source with the smallest time index, say ρ^*, transmits first, unless at least two sources own that same time index. In that case, a static tree search is initiated, also conducted à la CSMA-DCR. When this tree search is over, say at reference time t_1, a new time index ρ_1 is computed by those sources that have messages pending. In order to search the time tree consistently, it has to be that ρ_1 is greater than ρ^*. Furthermore, we do not want to make any assumption on the policy applied w.r.t. messages that may have missed their deadlines (they may be filtered out or not). If such «late» messages are not detected and suppressed, they could be assigned negative time indices. This would yield backward time tree searches, which is unacceptable. In order to account for cases where ρ_1 computed at time t_1, as shown above, would yield $\rho_1 \leq \rho^*$, time index ρ_1 must be computed according to the following rule :

$$\rho_1 = \max \{0, \lceil (E - t_1) / c \rfloor - \alpha\} + \rho^* + 1 \quad .$$

Observe now that this formula may yield $\rho_1 > \rho_0$. This would be the case, in particular, whenever static tree searches last less than c. Consequently, a message that was initially considered as being schedulable ($\rho_0 < F$) may become non schedulable (its transmission may be relegated to some subsequent time tree search). This is quite normal, as the opposite may also occur, i.e. this

message may become schedulable again in the same time tree. This would be case for example in the presence of a sufficiently long static tree search. Note, however, that once time E - (α + 1/2)c has been reached (time π in section 5.1), a schedulable message remains schedulable.

Let ρ^* be the time index corresponding to the time tree node whose search has resulted into the most recent collision. Let t be the termination time of the corresponding static tree search.

Using t as a reference time, source i computes ρ_m as follows :

$$\rho_m = \rho_t(i, m) = \max \{0, \lceil (E - t) / c \rfloor - \alpha\} + \rho^* + 1$$

Obviously, ρ^*+1 is the value of the time index to be searched next (i.e., during channel slot]t, t+s]).

If ρ_m is the smallest time index computed, then m is transmitted right away, if m also is first in i's waiting queue and in the case no collision occurs. It takes exactly $\lceil r/v(i) \rceil$ time tree/static tree searches to process m successfully, v(i) being the number of static indices allocated to source i, and r being the rank of message m (local EDF ranking). If $\rho_m \geq F$, source i does not participate in the current time tree search (i.e., source i remains silent).

A time tree search might terminate searching empty leaves/inactive indices (all time indices computed are greater than F). When a time tree search is completed, the channel is considered to be back to idle.

Every source records the value of the current reference time. Consider a source z that becomes active (a message n arrives in its empty waiting queue). Time index ρ_n is computed as shown above. Assume that ρ_n is current (i.e., is the time index being searched) and that a collision is being resolved (a static tree search is in progress). Assume further that a static index (allocated to z) is still available for n and not searched yet. Then, n is processed during that search. This is how the open entry mode option (see annex 2) operates under DOD/CSMA-CD.

Example

An extension of the example used in annex 2 is shown figure A3. Messages have an identical transmission duration of 6s (s is the channel slot time). Relative deadlines upon occurrence of the initial collision (time t_0) are given in table 1, as multiples of s.

Additional parameters are F = 4, c = 35s and α = 0. The first reference time is t_0 (initial collision is detected). That is, t_0 occurs one channel slot after the epoch is entered. Sources compute their time indices (shown table 1) and start searching the time tree. A static tree search is necessary to service those sources that have static indices 5 and 15. Message # 5 is done at time t_0 + 8s. Message # 15 is done at time t_0 + 14s.

The second reference time is t_1 (end of the first static tree search), that is $t_1 = t_0 + 14s$. Sources re-compute their time indices (shown table 2), with $\rho^* = 0$. Message # 3 is done at time t_0 + 20s. A static tree search is necessary to service those sources that have static indices 12 and 14. Message # 12 is done at time t_0 + 32s. Message # 14 is done at time t_0 + 38s. Notice that source # 2 stops

participating in the epoch (its new time index = F).

The third reference time is $t_2 = t_0 + 38s$. Source # 2 re-computes its time index, with $\rho^* = 2$ (see table 3). Given that $\rho > F-1$ (5 > 3), message # 2 still is not schedulable. It takes 1 channel slot to complete the current time tree search. Message # 2, which is transmitted within a 1 message time tree (not shown on figure A3), is done at time $t_0 + 45$ s (no time index computed as there is no collision).

Hence, assuming that s = 40 μs, it follows that transmission of the 6 messages considered is completed in 1.8 ms after detection of the initial collision, that is 1.84 ms after entering the epoch.

Note that this scenario would not have been feasible with CSMA-DCR for the example considered. Indeed, message # 5 would be done at time t + 23s and message # 15 would be done at time t + 45s (with $t = t_0 - s$). Both messages would miss their deadlines. Hence, in this particular case, DOD/CSMA-CD is able to solve a problem that cannot be solved with CSMA-DCR, this being achieved at an additional cost of 1 channel slot.

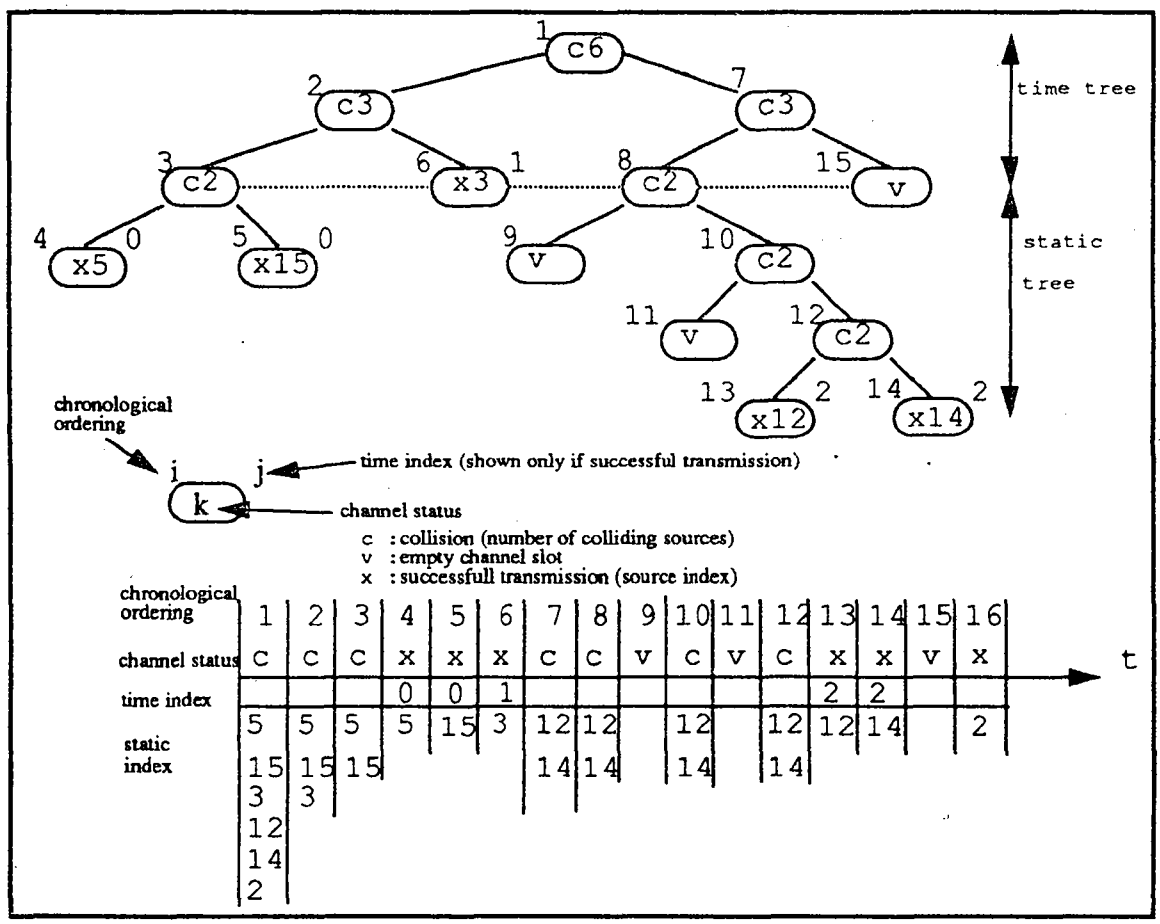


Figure A3 : a DOD/CSMA-CD epoch

source static index	5	15	3	12	14	2
relative deadline (initial collision)	10s	17s	29s	65s	56s	104s
computed time index ρ	0	0	1	2	2	3

Table 1: parameters at reference time t_0

source static index	3	12	14	2
relative deadline (at t_1)	15s	51s	42s	90s
computed time index ρ	1	2	2	4

Table 2: parameters at reference time t_1

source static index	2
relative deadline (at t_2)	66s
computed time index ρ	5

Table 3: parameters at reference time t_2

Unité de Recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

Unité de Recherche INRIA Lorraine Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)
Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Sophia Antipolis 2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399