# Modélisation et validation des algorithmes non-déterministes de synchronisation des horloges

Pedro Nicolau Faria Da Fonseca

Institut National Polytechnique de Lorraine
École Doctorale IAE+M

Universidade de Aveiro

# Modélisation et validation des algorithmes non-déterministes de synchronisation des horloges

THÈSE

présentée et soutenue publiquement le 21 avril 1999

pour l'obtention du

**Doctorat de l'Institut National Polytechnique de Lorraine**
**(Spécialité Informatique)**

et du

**Doutoramento da Universidade de Aveiro**
**(Especialidade de Engenharia Electrotécnica)**

par

Pedro Nicolau FARIA DA FONSECA

**Composition du Jury :**

| | | |
|---|---|---|
| *Président :* | Francisco VAZ | Prof. à l'univ. d'Aveiro |
| *Rapporteurs :* | Carlos CARDEIRA | Prof. à l'IST, Lisbonne |
| | Francis COTTET | Prof. à l'ENSMA, Poitiers |
| *Examinateurs :* | Nacer BOUDJLIDA | Prof. à l'UHP, Nancy |
| | Carlos COUTO | Prof. à l'univ. du Minho, Braga |
| | José Alberto FONSECA | Prof. à l'univ. d'Aveiro (Codirecteur de thèse) |
| | Zoubir MAMMERI | Prof. à l'univ. Paul Sabatier, Toulouse (Codirecteur de thèse) |
| | Jean-Pierre THOMESSE | Prof. à l'INPL, Nancy |

Institut National Polytechnique de Lorraine
École Doctorale IAE+M

Universidade de Aveiro

# Modelização e validação dos algoritmos não determinísticos de sincronização de relógios

TESE

apresentada publicamente

para a obtenção dos graus de

**Doctorat de l'Institut National Polytechnique de Lorraine**
**(Spécialité Informatique)**

e do

**Doutoramento da Universidade de Aveiro**
**(Especialidade de Engenharia Electrotécnica)**

por

Pedro Nicolau Faria da Fonseca

# Resumé

## Modélisation et validation des algorithmes non-déterministes de synchronisation des horloges

Cette thèse traite le problème d'analyse et de conception des algorithmes de synchronisation non-déterministes dans les systèmes répartis. Les algorithmes non-déterministes constituent une solution intéressante pour le problème de la synchronisation des horloges, ce qui est témoigné par l'intérêt qui leur a été porté pendant les dernières années.

Les algorithmes de synchronisation non-déterministes utilisent des techniques statistiques ou probabilistes et permettent d'obtenir une meilleure précision par rapport aux algorithmes déterministes, le prix à payer étant la probabilité que le système ne réussisse pas à se synchroniser avec la précision souhaitée. Cette probabilité d'échec tend vers zéro avec le nombre de messages échangés et elle peut être réduite à une valeur aussi petite qu'on le souhaite avec un nombre de messages suffisamment grand. Malheureusement, l'évaluation des différentes propositions publiées dans la littérature est difficile, principalement à cause de l'inexistence d'une base commune pour établir les comparaisons.

Nous proposons un modèle analytique pour le fonctionnement des algorithmes de synchronisation non-déterministes. Le but est de trouver une expression que le concepteur d'un système distribué puisse utiliser pour calculer le nombre de messages requis pour un certain algorithme, de façon à fournir la synchronisation avec la précision et la probabilité spécifiées. Ce résultat est la Condition de Garantie de Synchronisation, qui définit une condition suffisante qui garantit le succès de la synchronisation non-déterministe sous les conditions spécifiées. Cette condition est établie à partir de paramètres locaux d'un site et de paramètres du système qui sont facilement calculables, tels que le nombre de sites et les paramètres qui décrivent le délai comme une variable aléatoire.

Les conditions sous-jacentes au modèle proposé sont vérifiées expérimentalement à l'aide d'une plate-forme basée sur le réseau CAN que nous avons développée. Le test des algorithmes sur cette plate-forme a permis de vérifier la validité des hypothèses qui sont associées au modèle.



# Abstract

## Modelling and validation of non-deterministic clock synchronization algorithms

This thesis addresses the problem of analysing and designing non-deterministic clock synchronisation algorithms in distributed systems. Non-deterministic algorithms are a promising solution to the clock synchronisation problem, which can be testified by the attention they have received in recent years.

Non-determinist clock synchronisation algorithms use statistical or probabilistic techniques and they allow a better precision than with deterministic ones; the price to pay is a small probability that the system will fail to synchronise to the desired precision. This probability of failure can be made as small as desired by sending a sufficiently large number of messages. Unfortunately, assessing different algorithms is a difficult task, specially because we lack a common ground to establish comparisons.

We propose an analytical model for the behaviour of non-deterministic clock synchronisation algorithms. The aim is to find an expression that the distributed systems designer can use to estimate the required number of messages, in order to guarantee that a certain algorithm will synchronise to the desired precision and probability of success. This result is a Sufficient Condition for Synchronisation, which states the conditions that must be fulfilled in order to guarantee the desired precision and probability of success. This condition is based on local parameters and on system parameters that are easily computed, such as the number of sites and the description of the delay as a random variable.

The underlying assumptions are experimentally verified. For this purpose, we developed a test-bed based on the Controller Area Network. The results validate the assumptions used for the model.

# Resumo

## Modelização e validação dos algoritmos não determinísticos de sincronização de relógios

Esta tese aborda o problema da análise e da concepção de algoritmos de sincronização de relógios não determinísticos. Os algoritmos não-determinísticos constituem uma solução promissora para o problema da sincronização de relógios, sendo prova disso o interesse que estes têm despertado nos últimos anos.

Os algoritmos de sincronização não-determinísticos utilizam técnicas estatísticas ou probabilísticas para a obtenção do resultado. Obtém-se assim uma melhor precisão do que com os algoritmos determinísticos. O preço a pagar é uma (pequena) probabilidade que o sistema não consiga sincronizar com a precisão desejada. Pode-se tornar esta probabilidade de insucesso tão pequena quanto necessário através da utlização de um número de mensagens suficientemente grande. Infelizmente, a comparação e a avaliação das diferentes soluções propostas é difícil, principalmente devido à inexistência duma base comum para estabelecer essas comparações.

Nesta tese, propomos um modelo analítico para o funcionamento dos algoritmos de sincronização não-determinísticos. O objectivo é obter uma expressão que permita calcular o número de mensagens necessárias para um determinado algoritmo, de modo a que a sincronização ocorra com a precisão e a probabilidade especificadas. Este resultado é a Condição de Garantia de Sincronização, que define uma condição suficiente para garantir o sucesso da sincronização não-determinística sob as condições especificadas de precisão e probabilidade de sucesso. Esta condição é estabelecida a partir de parâmetros locais de um nó e de parâmetros do sistema que são facilmente calculáveis, tais como o número de nós ou os parâmetros que descrevem o atraso de comunicação como uma variável aleatória.

As condições subjacentes ao modelo proposto são verificadas experimentalmente. Para tal, desenvolveu-se uma plataforma baseada na rede CAN (Controller Area Network). As experiências realizadas permitiram verificar a validade das hipóteses associadas ao modelo.

À Margarida
Ao Francisco
por todo o tempo em que eu não estive em casa.

## Remerciements

Je tiens à remercier Monsieur Zoubir MAMMERI, qui a accepté de diriger mon travail de thèse, pour toute sa disponibilité et pour tous ses conseils. L'apport de son esprit méthodique et de ses remarques pertinentes a été inestimable.

Que Monsieur José Alberto FONSECA trouve aussi l'expression des mes remerciements, pour son soutien, son enthousiasme tout au long de ces années, et pas seulement dans le cadre de cette thèse.

Je remercie et j'exprime ma profonde gratitude à Monsieur le Professeur Jean-Pierre THOMESSE, pour m'avoir acueilli dans son équipe, pour son soutien, ses conseils et son encouragement au long de cette thèse. Cette thèse est l'épilogue d'une conversation à la pause café, il y a quelques années...

Je tiens à remercier Messieurs Carlos CARDEIRA, Francis COTTET et Nacer BOUDJLIDA pour avoir accepté la lourde charge de rapporteur de ce mémoire. J'exprime aussi mes plus vifs remerciements à Messieurs Francisco VAZ et Carlos COUTO pour l'honneur qu'ils m'ont fait en participant à mon jury de thèse.

Je voudrais aussi exprimer tous mes remerciements à l'équipe, d'abord InfoIndus du CRIN et maintenant TRIO du LORIA, pour la bonne ambiance qui a toujours régné, pour leur amitié et pour l'entraide qui a permis d'accomplir mes travaux d'une façon à tout moment efficace et agréable. J'addresse un remerciement tout particulier à Luis VEGA, Joël TOUSSAINT et Amor BOUZELAT. L'esprit dans l'équipe a toujours aidé à rendre mes séjours en France plus faciles et agréables. Un mot de remerciement aussi à Norbert GLASER.

E finalmente, *last but not the least*, um agradecimento muito especial a todos aqueles com quem convivo diariamente em Aveiro, em particular à equipa de Sistemas Electrónicos Distribuídos que me acompanhou durante todo este tempo: Alexandre Mota, Luís Almeida, Fernando Santos e Pedro Alvarenga. Quero deixar também aqui uma palavra de agradecimento a todos aqueles com quem tenho trabalhado no DET, docentes, funcionários e alunos.

L'auteur remercie les organismes ci-dessous soit pour avoir financé son séjour en France, soit pour avoir subventionné la participation dans des congrès internationaux:

- Junta Nacional de Investigação Científica e Tecnológica/Fundação para a Ciência e Tecnologia (ref. BD/3729/94)
- Programa PRODEP (Programa 5, Acção 5.2)
- Fundação Calouste Gulbenkian

# Table des matières

# Introduction générale

## Les systèmes temps-réel distribués

Les systèmes temps-réel constituent un ensemble de systèmes informatiques où la validité des résultats produits dépend non seulement de leur valeur numérique ou logique, mais aussi de l'instant auquel ils sont produits. Par exemple, le système d'anti-blocage des freins est un composant de plus en plus commun dans les voitures contemporaines. La commande de ce système doit activer les freins en réponse à la pression sur la pédale de freinage et, en cas de blocage d'une roue, répartir l'action de freinage sur les autres roues de façon à ce que la voiture n'altère pas sa trajectoire. Ceci doit être fait avec des délais de réponse de l'ordre de quelques millisecondes. Le non respect de ces contraintes de temps peut entraîner la perte de contrôle de la voiture et, à la limite, un accident pouvant être grave.

L'exemple précédent met aussi en évidence les besoins de sûreté qui sont imposés à ces systèmes. Les conséquences d'une faute, soit-elle temporelle (le résultat est produit hors de sa fenêtre temporelle de validité) ou logique (la valeur est erronée), peuvent être catastrophiques.

Les systèmes distribués apportent un plus en ce qui concerne la tolérance aux fautes. Dans ces systèmes, plusieurs unités (les *nœuds* ou *sites*) coopèrent pour le même but. Si l'une de ces unités tombe en panne, une autre pourra prendre le relais et l'ensemble du système continuera à fonctionner. Mais ces systèmes posent aussi des problèmes ; en augmentant le nombre d'équipements présents et en utilisant la redondance pour suppléer les nœuds fautifs, on augmente aussi le nombre de fautes potentielles dans le système et sa complexité.

Quand un système est, en même temps, temps-réel et distribué, les nœuds coopèrent entre eux (système distribué), exécutant des actions sous des contraintes temporelles (système temps-réel). Chaque nœud s'apperçoit de l'écoulement du temps à travers son horloge locale. Nous avons donc plusieurs sites qui doivent travailler d'une façon coordonnée, en exécutant des actions à des instants spécifiques, et où chacun a une perception du temps qui lui est propre. Un tel système ne fonctionnera correctement que si les horloges des différents sites sont correctement ajustées.

Considérons un système qui doit exécuter deux actions, $a_1$ et $a_2$. Il y a une contrainte de précédence entre ces deux tâches : $a_1$ doit se terminer avant que $a_2$ démarre. En plus, il y a une contrainte temporelle sur $a_1$ : $a_1$ doit se terminer avant l'instant $t_1$. Si cette contrainte temporelle est respectée, imposer que $a_2$ démarre après l'instant $t_1$ garantit le respect de la contrainte de précédence. Nous traduisons ainsi une contrainte de précédence par une contrainte temporelle. Ceci peut être utilisé, par exemple, pour éviter l'utilisation des protocoles de synchronisation entre tâches et simplifier le fonctionnement du système.

Ces deux actions sont exécutées sur un système distribué, et elles sont placées sur deux nœuds différents : $a_1$ sur $p_1$, $a_2$ sur $p_2$. Maintenant, le respect des contraintes temporelles doit

être fait par rapport à l'horloge locale de chaque site (que nous représentons par $H_1$ et $H_2$), puisqu'il n'existe pas un temps global de référence. Supposons que $T_1$ est la valeur qu'une horloge correcte doit afficher à l'instant réel $t_1$. Alors, $a_1$ doit se terminer avant l'instant où l'horloge de $p_1$ affiche la valeur $T_1$ et $a_2$ pourra démarrer seulement après l'instant où $H_2$ affiche la valeur $T_1$. Mais, si l'horloge de $p_2$ est en avance, $H_1$ affiche la valeur $T_1$ bien après que $H_2$ l'ait fait. Dans ce cas, il est possible que $a_2$ démarre avant que $a_1$ ne soit terminée. Chaque site réussit à faire correctement l'ordonnancement des tâches à exécuter (selon la vue locale dont il dispose), mais le système ne respecte pas les spécifications. La cause de la défaillance est la manque d'accord des horloges de $p_1$ et $p_2$.

Nous avons donc le besoin, dans les systèmes distribués, de garantir l'accord des différentes horloges du système. Malheureusement, ajuster les horloges au moment de la mise en marche du système ne suffit pas. Les horloges des ordinateurs présentent une *dérive* et, au fur et à mesure que le temps passe, les valeurs affichées s'écartent de plus en plus. Pour garantir l'accord des différentes horloges du système à n'importe quel moment, il faut veiller à ce que toutes les horloges du système ne s'écartent les unes des autres plus d'une certaine valeur (*précision*) ou alors qu'elles ne s'écartent plus d'une certaine valeur d'un temps de référence (*exactitude*). La réponse à ce besoin est obtenue par les algorithmes de synchronisation.

## Les algorithmes de synchronisation non-déterministes

Les premiers algorithmes de synchronisation d'horloges par logiciel dans les systèmes répartis étaient du type dit "déterministe". Aujourd'hui, la liste des algorithmes publiés est longue et des travaux de synthèse ont été publiées (*e.g.* [He et al., 1994, Schneider, 1987]), qui présentent un cadre pour l'analyse des algorithmes déterministes et où des pointeurs sur les articles originels peuvent être trouvés.

Dans les dernières années, les algorithmes du type non-déterministe ont fait l'objet d'un intérêt accru. Cet intérêt fait suite à un article, par Lundelius et Lynch, paru en 1984 ([Lundelius et Lynch, 1984]), qui étudiait l'exécution d'un algorithme de synchronisation déterministe sur un système distribué avec $N$ processus communicant par des messages. Le délai de transmission $\Gamma$ pour tout message est borné par $\Gamma_{\min}$ et $\Gamma_{\max}$, tels que $\Gamma_{\min} < \Gamma < \Gamma_{\max}$. Dans ce cas, les auteurs ont prouvé que tout algorithme de synchronisation déterministe ne peut pas garantir une précision inférieure à $\gamma = \epsilon(1 - 1/N)$, où $\epsilon = \Gamma_{\max} - \Gamma_{\min}$.

Ce résultat met en évidence deux caractéristiques, qui sont aussi deux limitations de ce type d'algorithmes. Premièrement, la précision garantie a une borne inférieure non nulle (sauf pour le cas trivial où le délai est constant). Deuxièmement, le délai de transmission d'un message doit être borné pour que la précision soit aussi bornée. Une valeur infinie pour $\Gamma_{\max}$ entraîne une valeur infinie pour $\epsilon$ et donc pour $\gamma$.

Les algorithmes non-déterministes de synchronisation des horloges proposent des solutions en utilisant des techniques statistiques ou probabilistiques pour l'obtention du résultat. Dans ce cas, les limites présentées par Lundelius et Lynch ne s'appliquent plus, mais il y a un prix à payer : le résultat est présenté avec une probabilité associée, qui est inférieure à 1. En principe, c'est possible de rendre cette probabilité aussi proche de 1 qu'on le souhaite à travers l'utilisation d'un nombre de messages suffisamment grand. La performance de tels algorithmes est fonction donc de trois paramètres qui sont interdépendants, et dont la relation doit être maîtrisée par le concepteur du système distribué : la précision de synchronisation, la probabilité de réussite et le nombre de messages échangés. Le concepteur est intéressé par

garantir la précision de synchronisation souhaitée avec une probabilité spécifiée. Ceci doit être fait en utilisant un nombre de messages aussi petit que possible (pour éviter de surcharger le réseau) mais sans jamais mettre en cause la probabilité de réussite spécifiée.

Les algorithmes non-déterministes sont ainsi une solution intéressante pour le problème de la synchronisation des horloges. Le concepteur d'un système distribué qui souhaite les utiliser peut trouver dans la littérature plus d'une dizaine de propositions. Mais le choix de la meilleure solution pour son système est loin d'être un choix facile.

Quand on essaie de comparer les algorithmes non-déterministes de synchronisation des horloges publiés, nous butons souvent sur le problème d'identifier les causes pour la variation de performance entre les différentes propositions. Beaucoup d'algorithmes publiés sont associés à une architecture spécifique ou à un certain modèle. Si un nouvel algorithme présente une performance meilleure que celle de ces prédécesseurs, est-ce que cela est dû aux caractéristiques de l'algorithme? Est-ce qu'un autre algorithme serait aussi performant sur le même système? Si on ne dispose pas d'un cadre commun pour faire les comparaisons, la liste de telles questions peut devenir interminable. Nous avons besoin d'un moyen pour établir les comparaisons entre les différents algorithmes.

Un des objectifs de nos travaux a été de développer un modèle pour l'analyse de la performance des algorithmes de synchronisation non-déterministes. Ce modèle permet de déterminer les conditions qui garantissent que la précision souhaitée est atteinte avec la probabilité spécifiée. Le modèle est simple et général: il couvre l'ensemble des algorithmes qui fonctionnent par des vagues (c'est-à-dire, où l'algorithme s'exécute périodiquement [He, 1993]). L'analyse du comportement des algorithmes de synchronisation est faite indépendemment de la structure ou du type du réseau utilisé. L'impact de l'environnement sur l'exécution de l'algorithme est modélisé à travers un ensemble de paramètres. Ceci permet d'analyser la performance des algorithmes sur une base commune et de vérifier l'effet de la variation d'un paramètre sur la performance de l'algorithme. La simplicité du modèle signifie que des approximations ont été faites, ce qui, forcément, dégrade l'exactitude des résultats. Mais, par contre, la simplicité signifie aussi que le modèle est facile à utiliser et que l'effet de la variation d'un paramètre est plus évidente.

## Organisation de la thèse

Avant de traiter les aspects spécifiques de la synchronisation des horloges, nous faisons dans le chapitre 1 une brève introduction aux systèmes distribués et temps-réel. Le but de ce chapitre est de situer le problème de la synchronisation des horloges dans la problématique des systèmes distribués temps-réel.

Le chapitre 2 rappelle les différents aspects d'une horloge physique. Nous proposons un modèle pour l'horloge physique, et nous le développons avec une définition formelle des propriétés souhaitées pour une horloge, pour établir les conditions qui permettent l'établissement de l'ordre des évènements dans un système distribué à partir des estampilles temporelles de ces évènements. Nos résultats sont conformes aux travaux présentés dans la littérature mais ils sont présentés d'une façon plus générale, où certaines hypothèses sont levées.

Dans le chapitre 3, les solutions pour la synchronisation des horloges sont classifiées en trois groupes: les solutions basées sur la réception d'un signal de référence, les solutions basées sur le matériel et les solutions basées sur l'échange de messages. Le dernier groupe est plus détaillé puisque tout notre travail porte sur ce type de solutions et, en particulier, sur les

algorithmes de synchronisation des horloges non-déterministes. Par ailleurs, nous établissons l'historique des différents algorithmes non-déterministes connus.

Le chapitre 4 présente l'analyse des algorithmes de synchronisation non-déterministes. La mise en œuvre correcte de ces algorithmes demande la maîtrise du rapport entre trois paramètres qui sont interdépendants : la précision souhaitée, la probabilité de réussite et le nombre de messages échangés. Nous proposons un modèle simple pour l'analyse de la performance de ces algorithmes et, basée sur ce modèle, une Condition de Garantie de la Synchronisation, qui établit le rapport entre les paramètres du système (précision, probabilité de réussite, nombre de messages et nombre de sites) qui garantit que la précision souhaitée est atteinte avec une probabilité non inférieure à la probabilité spécifiée. Ce modèle est basé sur un ensemble d'hypothèses, qui sont : la possibilité de calculer le rapport entre la précision souhaitée et l'erreur tolérée dans chaque lecture, l'existence d'un majorant pour la probabilité que l'erreur d'une lecture soit supérieure à un seuil d'erreur spécifié (que nous appelons la fonction d'erreur) et la représentation des erreurs comme des variables aléatoires. Les algorithmes sont classés selon la stratégie de lecture utilisée en trois groupes : régression, moyenne et intervalle. Pour chacun de ces groupes, nous trouvons une expression commune pour la fonction d'erreur et pour le rapport entre l'erreur tolérée et la précision souhaitée.

Le modèle est ensuite validé à travers des expérimentations. Le chapitre 5 présente le système matériel qui servira de support aux expériences. Ce système, totalement développé au cours de ce travail de thèse, est basé sur le réseau CAN et permet, à partir d'un seul ordinateur, le télé-chargement du code pour n'importe quel site du système distribué et le démarrage de l'exécution de ce code. Ceci permet de diminuer le temps nécessaire pour essayer et déboguer une application distribuée.

Les expériences qui permettent de valider le modèle d'analyse de la performance des algorithmes sont décrites dans le chapitre 6. Ce chapitre reprend la discussion de la validité des hypothèses présentées avec le modèle et les vérifie expérimentalement. En particulier, nous cherchons à déterminer :

- la caractérisation du délai comme une variable aléatoire, soit à travers des paramètres (comme la moyenne et la variance), soit à travers une distribution ;
- la validité des hypothèses d'indépendance des délais, notamment l'indépendance temporelle (le délai d'un message est une v.a. indépendante de la v.a. qui représente le délai suivant) et spatiale (le délai d'un message vers un site est une v.a. indépendante de la v.a. qui décrit le délai vers n'importe quel autre site) ;
- la validité du modèle pour fournir des garanties pour la synchronisation.

Les résultats obtenus nous ont permis de trouver la caractérisation du délai comme une variable aléatoire dans différentes situations de charge sur le réseau et confirment les hypothèses d'indépendance. Finalement, les prévisions issues du modèle pour la probabilité de réussite dans la synchronisation sont comparées avec les estimations qui résultent des essais. Nous vérifions que dans tous les cas le modèle fournit une estimation inférieure à la probabilité vérifiée expérimentalement, ce qui confirme sa validité pour établir des conditions qui sont garantes de la synchronisation sous les conditions (précision, nombre de messages et nombre de sites) spécifiées.

Nous terminons cette thèse par une conclusion générale dans laquelle nous faisons le bilan de ce qui a été fait et de ce qui reste à faire.

# Chapitre 1

# Introduction aux systèmes temps-réel et répartis

## 1.1 Le temps dans les systèmes informatiques

Jadis limités à des applications techniques, les systèmes informatiques (SI) sont aujourd'hui omniprésents dans notre vie quotidienne. L'ordinateur personnel est peut-être l'aspect le plus visible de cette influence. Son utilisation est de plus en plus répandue ; on l'utilise au travail et à la maison. Les enfants jouent avec l'ordinateur. L'ordinateur se vend dans les grandes surfaces. Un autre aspect de cette omniprésence des systèmes informatiques, souvent inaperçue aux non-spécialistes, est celui des systèmes embarqués. Ces systèmes utilisent des dispositifs électroniques (des microprocesseurs ou des micro-contrôleurs) pour faire la commande d'une machine ou d'un système ([Khan, 1996, Badami et Chbat, 1998]). En général, tout ce qui relève de l'automatique a aujourd'hui une forte probabilité d'être un système informatique, notamment, un système embarqué. L'utilisation des systèmes qui suivent le paradigme des systèmes embarqués menace même l'industrie de l'ordinateur personnel telle qu'on la connaît aujourd'hui ([Paterson, 1998, in [Schlett, 1998]]). La capacité de calcul de ce type de systèmes devient de plus en plus puissante. Dans la période de novembre 1997 à novembre 98, les fabricants ont lancé sur le marché 20 nouveaux microprocesseurs à 32 et 64 bits pour les systèmes embarqués ([Turley, 1998]). Si, dans beaucoup de cas, ces systèmes sont utilisés dans des applications où leur défaillance ne pose pas de problèmes très graves ([Cardoso, 1990]), dans certains cas la défaillance de ces machines peut entraîner des coûts très importants, voire des pertes de vies humaines. Citons, comme exemple, les systèmes de surveillance des malades dans les hôpitaux ou les systèmes de commande d'avions commerciaux.

Parmi ces SI, quelques uns doivent, non seulement présenter toujours les bonnes valeurs et prendre les bonnes décisions, mais aussi le faire au bon moment. La validité des résultats produits dépend non seulement de leur valeur numérique ou logique, mais aussi de l'instant auquel ils sont présentés. Ces systèmes sont les systèmes *temps-réel* (STR). On peut trouver des dizaines de définitions pour les STR dans la littérature ; une synthèse peut être trouvée dans [Vega, 1996]. Quelques unes de ces définitions sont orientées vers un problème ou un type d'application en particulier ; d'autres sont plus générales, mais sont parfois plus éloignées des problèmes concrets. L'étude des systèmes temps réel est largement au-delà des limites de

cette thèse, et elle constitue un problème qui a attiré l'attention des chercheurs depuis des années et qui est loin d'être épuisé. Parmi les définitions dont nous avons pris connaissance, nous retenons la suivante :

> **Temps réel** : un ensemble de problèmes qui ne sont pas abordés en informatique classique ; ils sont dûs essentiellement à la prise en compte du temps comme contrainte logique de base et non comme facteur de performance. ([Elloy, 1988])

Cette définition met en évidence le rôle du temps dans ce type de systèmes. Ce n'est pas un critère de performance, au sens où on le considère, par exemple, dans certains systèmes transactionnels. Pour un système de vente de billets d'avion, après que le client indique quelle est la destination souhaitée, le système doit répondre avec un délai de quelques secondes. Si ce délai n'est pas respecté, les conséquences ne seront pas catastrophiques. La personne qui est en train de vendre les billets pourra essayer de maintenir la conversation avec son client pendant qu'elle attend l'arrivée de la réponse. Dans cette situation, ayant le choix entre un système qui n'est pas aussi performant qu'on le souhaite, et aucun système du tout, on prend celui qui n'est pas très performant, en attendant qu'une meilleure version du logiciel ou de l'équipement soit disponible.

Pour les systèmes temps-réel, la situation est différente. Le temps ne joue pas un rôle dans l'établissement d'un certain niveau de performances mais de définir une frontière, claire et nette entre ce qui est acceptable et ce qui ne l'est pas. C'est une réponse du type logique : oui ou non. Ayant le choix entre un système qui ne respecte pas les spécifications des fonctions critiques et aucun système du tout, le choix ici serait ni l'un ni l'autre. Le meilleur choix serait de continuer à chercher un autre système, rejetant celui qui n'est pas suffisamment performant.

Les systèmes temps-réel doivent alors exécuter leurs actions sous des contraintes temporelles. Ces contraintes temporelles peuvent prendre des formes très différentes : soit exécuter telle action toutes les $x$ millisecondes, soit exécuter une autre $y$ secondes après un certain événement, soit établir à quel instant un événement (extérieur au SI) a été produit. Nous n'allons pas développer ces concepts ici ; nous prenons comme une évidence l'exécution des actions contraintes par les temps dans un STR.

Exécuter une action contrainte par le temps n'est possible que si le SI a les moyens de perception du temps : une horloge interne. Cette horloge est utilisée par le système informatique pour prendre des décisions qui lui permettent de démarrer et terminer ses tâches et de présenter les bons résultats au bon moment. Dans ce sens, on trouve deux notions du temps. Il y a le temps qui correspond à la grandeur physique, à la quatrième dimension spatio-temporelle. C'est ce temps qui commande l'évolution du procédé physique que le SI commande ou surveille. A côté (on pourrait dire «en même temps»), il y a le temps tel qu'il est perçu par le système informatique. C'est ce temps qui commande l'évolution des actions à l'intérieur du SI et, notamment, les moments auxquels le SI interagit avec l'environnement. Le premier, nous appellerons le temps *physique*[1]. Le deuxième, nous l'appellerons le temps *de l'horloge*. Il y a un seul temps physique (c'est une grandeur universelle) ; il y a une multitude de temps d'horloge (un pour chaque horloge d'un SI). Le lecteur intéressé par les questions

1. Parfois, on trouve aussi la dénomination temps réel pour le temps que nous appelons temps physique, au sens du temps qui existe dans le monde réel. Mais l'expression temps réel a une très forte connotation dans la communauté informatique, d'où notre préférence pour l'expression temps physique.

qui relèvent du temps pourra se référer à des ouvrages qui traitent ce sujet en détail, soit d'un point de vue plutôt philosophique ([Klein, 1995, Piettre, 1994]), soit d'un point de vue scientifique ([Hawking, 1988, Reichenbach, 1958]), orienté dans certains cas vers l'informatique ([Schreiber, 1992]).

| Temps physique | Temps d'horloge |
|---|---|
| Universel | Local au SI |
| Singulier | Multiple |
| Commande l'évolution des procédés (environnement du SI) | Commande l'évolution des actions du SI |

TAB. 1.1: *Les deux notions du temps*

Pour la bonne marche d'un STR, il est essentiel que les deux notions du temps physique et de l'horloge soient cohérentes. La cohérence, dans ce cas, signifie que les observations d'un ensemble d'évènements faites en utilisant le temps d'horloge sont essentiellement identiques à celles faites en utilisant le temps physique. Cette cohérence peut être définie de plusieurs façons : à travers l'ordre des évènements (l'ordre des évènements selon le temps physique doit être l'ordre selon le temps des horloges), à travers la mesure d'un intervalle de temps ou à travers la datation des évènements selon une échelle de temps standard, comme TAI (Temps Atomique International), TMG (Temps Méridien de Greenwich) ou UTC (Temps Universel Coordonné) (voir [Quinn, 1991] pour une présentation des étalons de temps). L'importance de ces définitions dépend de l'application en cause et sera étudiée en détail dans le chapitre 2.

## 1.2 La distribution

Au début de leur utilisation, les systèmes informatiques étaient des équipements chers et rares. L'achat d'un SI était un investissement lourd. À cause de son prix et de sa rareté, il fallait profiter tant que possible du système qu'on avait acheté. Pour la commande d'un procédé dans une usine, on avait un seul ordinateur, auquel tous les capteurs et tous les actionneurs étaient connectés. Cet ordinateur concentrait toutes les fonctions de commande. L'évolution des techniques de fabrication des circuits intégrés a permis la production de circuits à des prix chaque fois plus bas, entraînant la baisse de coût d'un SI. Un coût matériel réduit pour les SI a permis d'envisager d'autres arrangements. Au lieu de concentrer toutes les fonctions de commande dans un seul endroit physique, c'est désormais possible d'avoir plusieurs unités équipées d'un microprocesseur, plus près du procédé commandé (voir, par exemple, [Mota, 1993]). Les raisons pour le faire sont diverses ([Elloy, 1988]) :

- le procédé de contrôle est, par nature, constitué d'équipements multiples ;
- l'intégrité du procédé interdit d'en confier le contrôle à un seul équipement informatique ;
- les contraintes de promptitude de certaines actions gérées par le système de contrôle ne peuvent être techniquement satisfaites que si ces actions sont exécutées en parallèle.

Ces systèmes, où plusieurs unités coopèrent pour le même but, sont les système distribués (SD).

On trouve dans la littérature un grand nombre de définitions pour les systèmes distribués. Les différences dépendent du contexte où l'auteur se place : s'il s'intéresse au matériel ou au logiciel, aux systèmes temps-réel ou pas, aux systèmes embarqués ou aux grands systèmes,... Sa vision particulière attire son attention sur les détails qui sont les plus importants dans son domaine d'intérêt.

Lamport attire l'attention sur le délai des messages entre les sites pour définir un SD : [2]

> A distributed system consists of a collection of distinct processes, which are spatially separated and which communicate with one another by exchanging messages. A system is distributed if the message transmission delay is not negligeable compared to the time between events in a single process. ([Lamport, 1978])

Kopetz et Ochsenreiter proposent une définition proche de la précédente, mais aucune référence n'est faite au délai des messages:

> A distributed real-time system consists of a set of nodes which are interconnected by a local area network and communicate by message passage only. ([Kopetz et Ochsenreiter, 1987])

Pour Le-Lann, la caractéristique distinctive des SD est l'existence de plusieurs localisations de contrôle :

> Distributed system: a computing system whose behaviour is determined by algorithms explicitly designed to work with multiple loci of control and aimed at handling concurrent asynchronous computations ([Le Lann, 1991]).

D'autres auteurs préfèrent mentionner les caractéristiques qui permettent d'identifier un système distribué :

> Five properties for defining a distributed computing system ([Enslow, 1978]):
>
> - multiplicity of general purpose resource components;
> - physical distribution;
> - high level operating system that unifies and integrates control;
> - system transparency;
> - cooperative autonomy.

Il y a même des auteurs qui se refusent à donner une définition claire et nette d'un système distribué, et qui proposent tout simplement les « symptômes » d'un tel système. Si tous ces symptômes se trouvent dans un SI, alors il est fort probable que ce système soit un SD.

> Symptoms of a distributed system ([Mullender, 1989]):
>
> - multiple processing elements;

2. Nous gardons ici la définition en anglais, pour mieux respecter la pensée de l'auteur.

- interconnection hardware ;
- processing elements fail independently ;
- shared state.

Finalement, une autre définition donnée par Lamport focalise l'attention sur le problème de la tolérance aux fautes:

> A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable. (Leslie Lamport, origine inconnue, cité dans plusieurs articles).

Un coût bas du matériel et donc la possibilité d'avoir des systèmes distribués permet d'avoir des systèmes plus tolérants aux fautes. Dans un système distribué, une même fonction peut être attribuée à plus d'un élément du système : si l'un d'entre eux tombe en panne, un autre peut prendre le relais, et le système continue à fonctionner. Mais cet avantage n'est pas immédiat ; comme l'ont mentionné Burns et Wellings, si d'un côté les systèmes distribués apportent les moyens pour faire des systèmes plus sûrs, ils apportent aussi une augmentation du nombre de fautes potentielles dans le système ([Burns et Wellings, 1990]). Les systèmes distribués apportent évidement des avantages, mais le concepteur de tels systèmes doit se prémunir contre des arguments trop simplistes en faveur de la distribution. Selon Peter Neumann :

> The simplicity of those arguments [about the advantages of distributed systems] sweeps under the rug serious problems of security and controllability that arise when the sharing of data and distributed processing are permitted. ([Neumann, 1995])

Cette dichotomie entre les avantages — immédiats et évidents — et les problèmes qu'une telle solution peut apporter — qui peuvent rester cachés pour longtemps — doit être toujours présente dans le développement de tous les systèmes distribués. Le problème de la tolérance aux fautes dans les systèmes distribués continue, depuis des années, à faire l'objet d'attention d'un nombre de chercheurs toujours plus élevé ([Anderson et Lee, 1981, Avizienis, 1997, Boasson, 1995, Burns et Wellings, 1990, Cristian, 1991, Guerraoui et Schiper, 1997, Kopetz, 1997, Kopetz et Ramamritham, 1990, Le Lann, 1991, Neumann, 1995, Simons et Spector, 1989] présentent quelques aspects du problème et des liens vers d'autres ouvrages).

Les motivations pour l'utilisation des SD peuvent être classées en deux groupes: comme un moyen pour atteindre un but, et comme une exigence issue de la situation ([Singhal et Casavant, 1991]). Dans le premier cas, on trouve les systèmes massivement parallèles (« massively parallel systems »), le calcul à haute performance, la tolérance aux fautes et les besoins de réponse temps-réel. Dans le deuxième cas, on trouve les bases de données distribuées, les systèmes de production automatisés et les applications de contrôle et surveillance à distance. Ceux-ci sont, en général, l'application du principe «la forme suit la fonction» ([Kopetz, 1997]).

Dans un système distribué, on trouve plusieurs horloges (en général, une horloge dans chaque site), et donc plusieurs temps d'horloge. Quand les systèmes distribués sont utilisés dans des applications temps-réel, les aspects de cohérence entre le temps physique et les

différents temps d'horloge sont extrêmement importants. Ce sont les temps des horloges qui commandent l'évolution des actions dans chacun des sites ; on a donc différents sites qui essaient de coopérer dans un but commun, où les actions sont commandées par le temps d'horloge local à chacun d'eux. L'ensemble des sites interagit avec un environnement dont les actions sont commandées par l'écoulement du temps physique. Dans cette situation, la localisation du temps d'horloge (le fait qu'il existe seulement à l'intérieur d'un SI) et sa multiplicité (le fait qu'il peut exister plusieurs instances) peuvent poser de sérieux problèmes à la bonne marche d'un système distribué temps-réel. Il est donc indispensable de garantir que les différentes perceptions du temps sont sous le contrôle du système informatique, et que sa localisation et multiplicité ne remettent pas en cause le fonctionnement correct du système. Ceci peut être atteint grâce à la synchronisation des horloges que nous étudierons par la suite.

# Chapitre 2

# Des horloges

Certains systèmes informatiques, en particulier ceux destinés à la commande de procédés physiques, exécutent leurs actions souvent avec des contraintes temporelles, imposées par l'environnement. Ces contraintes peuvent être du type « imposer un intervalle de temps pour l'exécution d'une action », soit du type « déterminer l'instant d'occurrence d'un événement avec une erreur limitée ». L'étude des caractéristiques du rapport entre un SI et son environnement constitue, en soi, une discipline autonome, le temps réel, dont les limites dépassent largement ceux de cette thèse. Nous prendrons comme une évidence l'interaction, contrainte par le temps, des SI avec leur environnement. De cette évidence, résulte le besoin pour les SI d'avoir un moyen de perception du temps. Ce besoin est satisfait par l'utilisation des horloges.

Ce chapitre présente, à travers un exemple pratique, les caractéristiques souhaitables pour les horloges d'un SI. Ces caractéristiques, qui sont issues des spécifications des tâches à accomplir par le SI, définissent un ensemble de propriétés permettant la caractérisation des applications et des horloges sur une base commune : i) les horloges par les propriétés qu'elles possèdent ; ii) les applications par les propriétés qu'elles requièrent. Ceci est fait dans la section 2.1. La structure des horloges telles qu'on les trouve dans les systèmes informatiques les plus communs est décrite dans la section 2.2 ; cette structure constitue l'horloge physique de l'ordinateur. Dans la suite, nous proposons dans la section 2.3, un modèle mathématique qui décrit l'horloge physique. Ce modèle servira pour vérifier quelles sont les propriétés que l'horloge physique possède et, dans la section 2.4, pour établir un ensemble de résultats concernant les conditions qui nous permettent d'établir, avec sûreté, la séquence temporelle dans les systèmes distribués à partir des estampilles locales d'horloge.

## 2.1 Propriétés des horloges

La figure 2.1 représente un système de traitements thermiques qui nous aidera à présenter l'utilisation des horloges dans les systèmes informatiques — dans ce cas, dans un système de commande et supervision. Dans ce système, des palettes contenant des pièces métalliques à traiter dans un four arrivent par un des convoyeurs d'entrée. Pour simplifier, considérons que le traitement consiste tout simplement d'un passage, de durée déterminée, dans le four, à température contrôlée. Les contraintes qui sont imposées pour un bon usage des horloges, permettant le fonctionnement correct du système, serviront à définir les propriétés que ces horloges doivent posséder pour que le système marche selon ses spécifications.

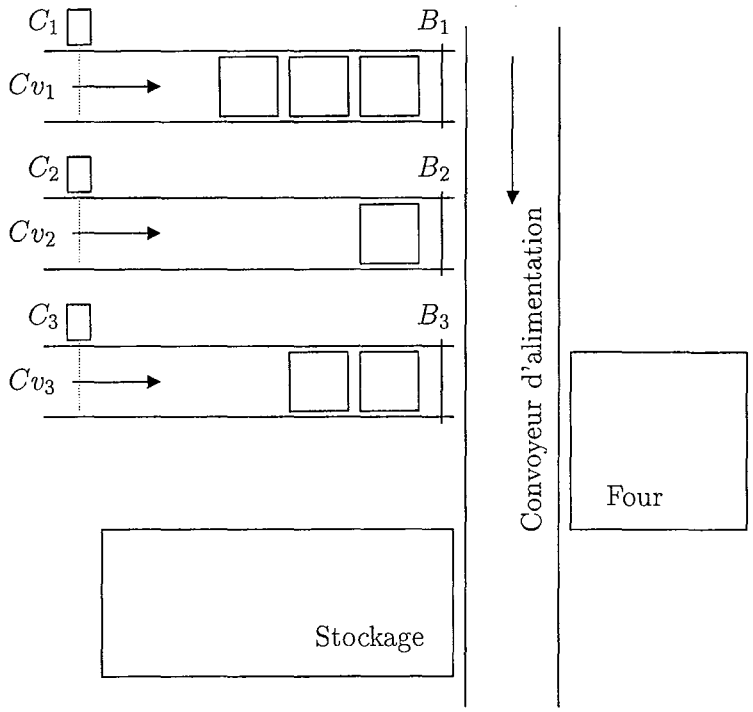



FIG. 2.1: *Système de traitement thermique*

Les palettes portant les pièces à traiter arrivent par un des convoyeurs, $Cv_1$, $Cv_2$ et $Cv_3$. Au moment où le four est libre, une des palettes est placée sur le convoyeur d'alimentation. La palette choisie est celle qui est arrivée en premier à un des convoyeurs d'entrée. L'ensemble des trois convoyeurs d'entrée doit se comporter comme un tampon « premier entré, premier servi », (« first-in, first-out », ou FIFO). La durée du traitement au niveau du four est aussi commandée par le système informatique. Cette durée doit être respectée ; si le traitement est trop court ou trop long, les résultats attendus du traitement thermique ne seront pas atteints, et les pièces seront rejetées. Quand le traitement est terminé, la palette avec les pièces traitées est sortie du four et placée dans la zone de stockage. Une fiche de fin de traitement doit être produite à ce moment, portant l'identification des pièces et l'heure à laquelle le procédé de traitement a été conclu.

Les barrières $B_i$ commandent l'accès des palettes au convoyeur d'alimentation du four. Au moment où un des capteurs $C_i$ détecte le passage d'une palette, une estampille de temps est associée à cette palette. Cette estampille servira à choisir quelle palette sera envoyée pour traitement. Ceci pose la première condition à remplir par les horloges du SI : la séquence des valeurs des estampilles doit correspondre à la séquence réelle des événements correspondants (arrivée des palettes à la zone d'attente). Les horloges qui satisfont cette condition possèdent la propriété M, de monotonicité.

**Définition 2.1 (Propriété M (Monotonicité))** *La séquence des valeurs des estampilles doit correspondre à la séquence réelle des événements correspondants.*

Le système de commande du four prend la palette choisie pour le traitement, lit la valeur

de la durée du traitement et place la palette dans le four. Après écoulement du temps de traitement, la palette est retirée du four et envoyée vers la zone de stockage. Pour que le traitement soit correctement exécuté, il faut que la valeur de l'intervalle de temps, telle qu'elle est mesurée par l'horloge du système de commande, corresponde à la vraie valeur de cet intervalle. Cette condition définit une deuxième propriété à imposer aux horloges.

**Définition 2.2 (Propriété D (Dérive bornée))** *La valeur d'un intervalle de temps, telle qu'elle est mesurée par une horloge, doit correspondre à la vraie valeur de cet intervalle.*

Dans certains cas, on pourra envisager que chaque convoyeur soit équipé d'une commande séparée. Ceci peut permettre, par exemple, que chacun soit un module autonome, qui peut être déplacé à n'importe quel instant. Les événements « arrivée d'une palette » sont maintenant estampillés par des horloges différentes. Pour que cet estampillage soit cohérent, c'est-à-dire, pour que la séquence des événements puisse être reconstituée seulement par les estampilles temporelles, il faut que deux horloges affichent la même valeur en même temps.

**Définition 2.3 (Propriété P (Précision))** *La valeur affichée par deux horloges différentes, au même instant, doit être identique.*

Quand les palettes avec les pièces traitées arrivent à la zone de stockage, une fiche doit être produite, portant l'identification du lot des pièces et l'heure à laquelle le lot a été placé en stockage. Les estampilles temporelles doivent correspondre donc à un temps standard, comme TMG ou UTC. L'horloge du système chargé de ces fiches doit afficher, à tout instant, la valeur de ce temps de référence.

**Définition 2.4 (Propriété E (Exactitude))** *La valeur de l'horloge doit correspondre à un temps standard.*

Remarquons que les propriétés énoncées précédemment ne sont pas forcément requises pour tous les types d'applications. Elles ont été présentées en tenant compte de certains besoins. Chacune de ces propriétés signifie que l'horloge est capable de répondre à un certain type de besoins, et que le système qui comporte cette horloge est capable de satisfaire telle ou telle spécification. Pour chaque application (ou même pour chaque tâche que le système doit accomplir), les propriétés que les horloges doivent posséder sont issues des spécifications qui ont été définies.

### 2.1.1 Définitions formelles

Les quatre propriétés (M, D, P, E) peuvent être exprimées d'une façon plus formelle. $H_p(t)$ représente la valeur de l'horloge du site $p$ à l'instant $t$. La propriété M est définie par:

**Définition 2.5 (Propriété M — Monotonicité)** *Une horloge $H_p$ possède la propriété M ssi :*

$$\text{M}: \qquad t_1 > t_2 \Leftrightarrow H_p(t_1) > H_p(t_2) \tag{2.1}$$

La propriété D indique que les intervalles de temps, mesurés par les horloges, ont la valeur correcte, c.-à-d.,

**Définition 2.6 (Propriété D — Dérive bornée)** *Une horloge $H_p$ possède la propriété D ssi :*

$$\mathrm{D}: \qquad \forall_{t_1,t_2}, \quad t_1 - t_2 = H_p(t_1) - H_p(t_2) \tag{2.2}$$

Pour la propriété P, les horloges dans deux sites différents doivent afficher la même valeur en même temps :

**Définition 2.7 (Propriété P — Précision)** *Deux horloges $H_p$ et $H_q$ ont la propriété P ssi :*

$$\mathrm{P}: \qquad \forall_{p\neq q}, \forall_t, \quad H_p(t) = H_q(t) \tag{2.3}$$

Finalement, la propriété E indique que l'horloge doit afficher la valeur correcte selon un certain étalon de temps, $e$:

**Définition 2.8 (Propriété E — Exactitude)** *Une horloges $H_p$ possède la propriété E ssi :*

$$\mathrm{E}: \qquad \forall_t, \quad H_p(t) = H_e(t) \tag{2.4}$$

En général, on considère $H_e(t) = t$.

Les propriétés précédentes correspondent à des situations idéales : des horloges qui affichent exactement la même valeur dans deux sites différents, ... Dans la pratique, nous sommes plutôt intéressés à des définitions qui permettent de considérer des horloges qui, même si elles ne sont pas capables d'atteindre les spécifications strictes, restent quand même assez proches de ces définitions, ce qui permet de les utiliser comme de bonnes approximations. En introduisant des paramètres de tolérance, nous sommes capables de quantifier de combien une horloge s'éloigne de la définition stricte.

### 2.1.2 Paramétrisation des propriétés

L'introduction de paramètres de tolérance dans les définitions présentées précédemment nous permettra de considérer les horloges qui possèdent des caractéristiques qui sont une bonne approximation des propriétés énoncées. Ces paramètres concernent les propriétés qui relèvent d'une mesure, et elles sont: la propriété D (mesure d'un intervalle de temps), la propriété P (mesure de l'instant d'occurrence d'un événement sur des sites différents) et la propriété E (mesure de la valeur du temps physique). On notera la version paramétrée de ces propriétés par un indice avec la valeur du paramètre.

La dérive $\rho$ est une mesure de la qualité de l'approximation des intervalles mesurés par les horloges aux intervalles de temps physique correspondants.

**Définition 2.9 (Propriété $\mathrm{D}_\rho$)**

$$\mathrm{D}_\rho: \qquad \forall_{t_1,t_2}, \quad (1-\rho)(t_2 - t_1) \leq H_p(t_2) - H_p(t_1) \leq (1+\rho)(t_2 - t_1) \tag{2.5}$$

Pour la précision, une définition paramétrée détermine que deux horloges, dans deux sites différents, doivent afficher des valeurs proches l'une de l'autre à tous les instants. La valeur $\delta$ est la borne maximale pour la différence entre les valeurs de deux horloges au même instant:

**Définition 2.10 (Propriété $P_\delta$)**

$$P_\delta : \quad \forall_{p \neq q}, \forall_t \quad |H_p(t) - H_q(t)| \leq \delta \tag{2.6}$$

Une définition alternative est celle qui dit que deux horloges, dans deux sites différents, doivent afficher la même valeur à des instants qui ne soient pas trop éloignés. La valeur $\hat{\delta}$ est la borne maximale pour le délai entre les instants auxquels les horloges affichent la même valeur:

**Définition 2.11 (Propriété $P'_{\hat{\delta}}$)**

$$P'_{\hat{\delta}} : \quad \forall_{p \neq q}, \quad H_p(t_1) = H_q(t_2) \Rightarrow |t_1 - t_2| \leq \hat{\delta} \tag{2.7}$$

La première définition concerne la datation d'événements simultanés[1] par des sites différents. Elle signifie que si deux événements simultanés se produisent sur les sites $p$ et $q$, les estampilles correspondantes ne diffèrent jamais de plus de $\delta$. La deuxième définition concerne l'instant d'occurrence des événements que l'on souhaite voir apparaître simultanément. Si deux actions sont démarrées en même temps (selon les horloges locales) dans deux sites différents, $p$ et $q$, les instants (temps physique) auxquels ces actions démarrent ne diffèrent jamais plus de $\hat{\delta}$. Pour des systèmes où $\rho \ll 1$, $\delta \simeq \hat{\delta}$.

L'exactitude de la valeur de l'horloge (propriété E) est mesurée par le paramètre $\alpha$. La définition paramètrisée de la propriété E est:

**Définition 2.12 (Propriété $E_\alpha$)**

$$E_\alpha : \quad \forall_t, \quad |H_p(t) - t| \leq \alpha \tag{2.8}$$

Les propriétés que nous avons présentées ne sont pas orthogonales ; en particulier, la propriété E implique l'existence des autres trois propriétés. En effet, un système qui possède la propriété $E_\alpha$ possède la propriété $P_\delta$, avec $\delta = 2\alpha$. Si la propriété $E_\alpha$ est garantie pendant un intervalle $\Delta t$, elle implique aussi la propriété $D_\rho$, avec $\rho = 2\alpha/\Delta t$. Finalement, la monotonicité est garantie pour deux lectures qui sont distantes au moins $2\alpha$, c.-à-d., $t_1 > t_2 + 2\alpha \Rightarrow H(t_1) > H(t_2)$.

### 2.1.3 Conclusion

Nous avons donc défini, en partant d'une application, un ensemble de propriétés que les horloges doivent posséder (soit le tout, soit une partie de cet ensemble), de façon à pouvoir répondre aux besoins de l'utilisateur. L'exemple de la section 2.1 a servi pour présenter des cas typiques d'association entre caractéristiques de l'application et propriétés des horloges : l'établissement d'une séquence demande la monotonicité des horloges, la mesure des intervalles de temps demande que la dérive soit bornée, la coordination des actions dans le temps

1. Nous prenons ici la définition la plus commune de la simultanéité, correspondant à un univers Newtonien. Nous ne considérons pas les effets relativistes dans la définition de la simultanéité et nous ne sommes pas occupés — pour le moment— du problème de la vérification de cette simultanéité.

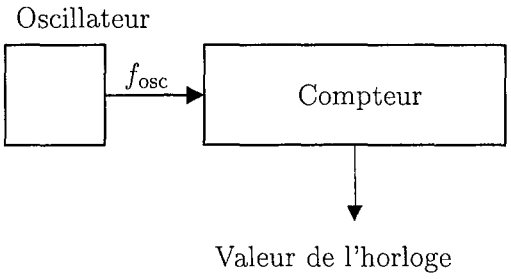



FIG. 2.2: *Structure de l'horloge physique*

sur différents sites demande la précision et la réalisation des actions à des instants définis sur une échelle standard l'exactitude. Maintenant, il faudra vérifier comment ces propriétés sont garanties par les horloges que l'on peut trouver dans les systèmes informatiques. Ceci nous permettra de bien connaître les limites de ces horloges.

## 2.2 L'horloge physique

Les horloges des ordinateurs sont, dans la plupart des cas, basées sur la même structure de base, qu'on appelle l'*horloge physique*. Cette section présente ce dispositif et les propriétés qu'il possède.

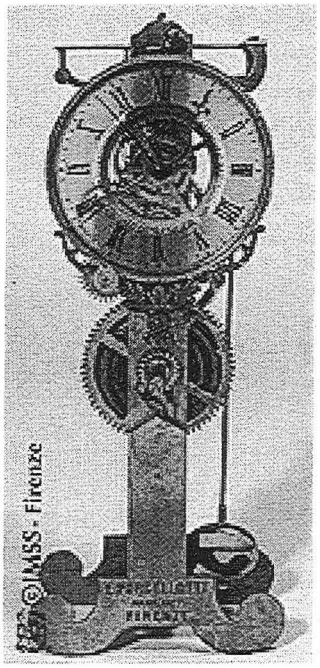
### 2.2.1 Composition de l'horloge physique

La structure d'une horloge physique est décrite par la figure 2.2. Elle est basée sur un oscillateur et un compteur. L'oscillateur est censé produire des événements périodiques, les *tops* de l'horloge, à une certaine fréquence, la fréquence nominale, $f_{\text{nom}}$. À chaque top de l'horloge, le compteur est incrémenté d'une valeur $g_{\text{nom}}$. Pour une horloge qui mesure le temps physique, $g_{\text{nom}} = 1/f_{\text{nom}}$, c'est-à-dire, le compteur est incrémenté d'une valeur identique à la période des tops.

Remarquons au passage que la structure de base d'une horloge n'a pas changé pendant les derniers siècles. Depuis l'invention de l'horloge à pendule, le principe est resté inchangé. Bien sûr, le changement a été complet en ce qui concerne la technologie ; mais on trouve les mêmes éléments fondamentaux dans une ancienne horloge mécanique à pendule comme celle de la figure 2.3 et l'horloge à quartz d'un ordinateur contemporain (tableau 2.1).

| Horloge | Oscillateur | Compteur |
|---|---|---|
| Mécanique | Pendule | Cadrant |
| Quartz | Circuit résonant | Compteur électronique |

TAB. 2.1: *Éléments structuraux d'une horloge*




Horloge mécanique existant au Musée de l'Histoire des Sciences, à Florence, Italie. Cette horloge a été construite par Eustachio Porcellotti, un horloger florentin, en 1877, et c'est un modèle opérationnel du dessin qui apparaît dans une lettre de Galileo Galilei datée de 1637. La précision d'une telle horloge dépend de l'isochronisme des oscillations d'une pendule, ce que Galileo avait démontré dans sa recherche sur la mécanique. (Phot. avec gentillesse du Musée de l'Histoire des Sciences)

FIG. 2.3: *Horloge mécanique à pendule*

### 2.2.2 L'oscillateur

La vraie fréquence de l'oscillateur, $f_{\text{osc}}$ est souvent différente de $f_{\text{nom}}$. Cette déviation de la fréquence de sa valeur nominale est causée par plusieurs facteurs. Par exemple, pour les cristaux à quartz, cette déviation est fonction de la sensibilité à la température, de l'effet des condensateurs de polarisation dans le circuit électrique et du vieillissement. Toutefois, si la valeur de cette déviation n'est pas connue a priori, il est raisonnable de considérer qu'elle est bornée. La dérive $\rho$ d'une horloge est définie comme la borne maximale de la déviation fractionnaire de la fréquence de l'horloge par rapport à sa fréquence nominale :

**Définition 2.13 (Dérive d'une horloge)** *La dérive d'une horloge avec fréquence nominale $f_{\text{nom}}$ et fréquence instantanée $f_{\text{osc}}(t)$ est une valeur $\rho$ telle que :*

$$\forall_t, \qquad \left| \frac{f_{\text{osc}}(t) - f_{\text{nom}}}{f_{\text{nom}}} \right| \leq \rho \tag{2.9}$$

Normalement, $f_{\text{osc}}(t) \simeq f_{\text{nom}}$, ce qui implique $\rho \ll 1$. Ceci permet de simplifier quelques expressions, en négligeant les termes en $\rho^2$ par rapport à 1. Notamment, nous considérerons dans la suite $(1-\rho)^{-1} \simeq 1+\rho$ et $(1+\rho)^{-1} \simeq 1-\rho$.

Pour la plupart des cas, l'oscillateur est basé sur un cristal à quartz. La dérive d'un tel oscillateur se trouve dans la gamme de $10^{-5}$ à $10^{-6}$. Bien que celles-ci soient des valeurs typiques, des imperfections, soit du cristal, soit du circuit résonant, peuvent faire monter cette valeur jusqu'à $10^{-4}$, et des observations de dérives de $2.5 \times 10^{-4}$ ont été déjà référencées ([Mills et Thyagarajan, 1994]).

Les oscillateurs à quartz présentent une bonne performance à un coût, poids et puissance réduits. D'autres solutions existent et ont de meilleures caractéristiques de performance, comme les horloges atomiques à césium ou rubidium. Mais ces solutions sont lourdes (physiquement), plus chères et consomment plus d'énergie, ce qui les rend inadéquates pour une grande partie des applications des SI, comme les systèmes embarqués. Une autre alternative est l'utilisation de mécanismes de compensation des erreurs dans les cristaux, tels que la compensation des erreurs de température. Nous n'étudierons pas ce sujet en détail. Pour plus d'information sur les divers étalons de fréquence se référer à [Mills, 1991b, Sullivan et Levine, 1991, Vig, 1992]. Les caractéristiques des oscillateurs à quartz sont présentés en détail dans [Frerking, 1978, Vig, 1992].

### 2.2.3 Une remarque sur le nom

La dénomination de « horloge physique » pour le dispositif que nous venons juste de présenter mérite quelques remarques de notre part. Cette désignation suppose une opposition entre l'horloge physique (dont la valeur n'est pas ajustée) et l'horloge synchronisée (dont la valeur est ajustée pour posséder certaines propriétés). Mais cette distinction devient ténue quand on considère des horloges synchronisées totalement implémentées sur le matériel. Dans un tel cas, laquelle des horloges est plus « physique » que l'autre?

Le nom de *horloge physique* peut être aussi utilisé par opposition aux horloges *logiques* pour décrire les horloges qui mesurent le temps *physique* (qu'elles soient synchronisées ou pas). Une horloge logique établit des relations de précédence sans mesurer le temps écoulé. L'horloge *physique* serait l'horloge qui mesure le temps *physique*.

Dans la littérature d'expression anglaise, on trouve parfois l'expression « *raw* clock » (à la place de « *hardware* clock ») pour désigner le dispositif qui a une source d'oscillations (c.-à-d. une source d'événements périodiques) et un compteur. Les raisons sont, en bref :

- ce n'est pas forcément une horloge qui est implémentée en matériel. Elle peut même ne pas avoir d'existence *physique* : elle peut être implémentée dans la mémoire d'un processeur, incrémentée par le logiciel.

- le mot *raw* (« brut, cru ») met en évidence le fait que c'est un dispositif sur lequel aucun traitement n'a été fait ; la valeur présentée est dans un état « brut ».

Une dénomination telle que « horloge brute » ou « horloge crue » servirait mieux à mettre en évidence le fait que c'est une valeur telle qu'elle a été produite par un dispositif électronique. Cependant, comme la dénomination d'horloge physique pour le dispositif de la figure 2.2 est d'usage commun, on la gardera dans cette thèse.

## 2.3 Vérification des propriétés de l'horloge physique

En possession d'un modèle pour l'horloge physique, nous allons maintenant vérifier si cette horloge possède les propriétés énoncées dans la section 2.1.

### 2.3.1 Modèle pour l'horloge physique

La valeur d'une horloge qui démarre à l'instant $t = 0$ avec une valeur 0 est représentée par:

$$H(t) = g_{\text{nom}} \left\lfloor \int_0^t f_{\text{osc}}(\tau) d\tau \right\rfloor \tag{2.10}$$

L'intégrale $\int_0^t f_{\text{osc}}(\tau)d\tau$ représente la phase de l'oscillateur à fréquence instantanée $f_{\text{osc}}(t)$ à l'instant $t$. $\lfloor \int_0^t \ldots d\tau \rfloor$ désigne le nombre de tops qui ont été produits par l'oscillateur. La valeur de l'horloge est le nombre de tops multiplié par la granularité $g_{\text{nom}}$.

Cette formulation permet d'obtenir une expression mathématique pour la valeur d'une horloge — qui est discrète et toujours un multiple entier de $g_{\text{nom}}$ — mais qui est basée sur un phénomène physique qui est continu. D'une certaine façon, elle permet de faire le pont entre « le monde des phénomènes continus » (où l'oscillation se déroule à une fréquence $f_{\text{osc}}$) et « le monde des phénomènes discrets » (où se déroule le comptage de l'horloge). Comme nous verrons dans la suite, elle permet aussi d'établir un ensemble de résultats concernant les propriétés des horloges et des conditions qui garantissent l'établissement avec sûreté des relations de précédence dans les systèmes distribués entre événements estampillés sur plusieurs sites. À notre connaissance, cette formulation est originale. Lamport ([Lamport, 1978]) propose une formulation similaire pour $\mathrm{C}_i^t$, une horloge qui est ajustée pour la valeur $\mathrm{C}_i$ à l'instant $t$:

$$\mathrm{C}_i^t(t') = \mathrm{C}_i(t) + \int_t^{t'} [d\mathrm{C}_i(t)/dt]dt \tag{2.11}$$

pour tous les $t' \geq t$. Pour faciliter le traitement mathématique, Lamport considère que les horloges sont des fonctions continues : l'horloge discrète peut être envisagée comme une horloge continue dont l'erreur dans la lecture est, au maximum, $1/2g_{\text{nom}}$. $\mathrm{C}_i(t)$ est une fonction continue et dérivable, sauf sur les points où l'horloge est ajustée. Dans ce cas, $d\mathrm{C}_i(t)/dt$ représente la vitesse à laquelle l'horloge est incrémentée. Dans les travaux postérieurs sur la synchronisation des horloges ([Lamport et Melliar-Smith, 1984, Lamport et Melliar-Smith, 1985]), Lamport ne reprend pas cette formulation.

### 2.3.2 Monotonicité

Les horloges à granularité non nulle ne possèdent plus la propriété M telle qu'elle a été définie dans l'équation 2.1. Entre deux tops, la valeur de l'horloge reste constante. La question qui se pose maintenant est : quel est le plus grand intervalle $\Delta t_0$ pour lequel l'horloge ne change pas sa valeur? La réponse à cette question nous permettra de définir $\Delta t_{\text{min,m}} > \Delta t_0$, le délai minimal à imposer entre les instants d'occurrence de deux événements qui garantit que les estampilles d'horloge qui leur sont associées soient différentes.

**Problème:** Étant donnée une horloge $H_p$, à granularité $g$ et dérive $\rho$ (selon eq. 2.9), quelle est le délai minimal $\Delta t_{\text{min,m}}$ pour lequel $H_p(t + \Delta t_{\text{min,m}}) > H_p(t)$?

**Solution:** Pour trouver $\Delta t_{\text{min,m}}$, nous chercherons d'abord la valeur $\Delta t_0$, la plus grande valeur pour laquelle on a $H_p(t) = H_p(t + \Delta t_0)$. On cherche à trouver

$$\max\{t_2 - t_1\} : H_p(t_2) - H_p(t_1) = 0 \tag{2.12}$$

$$
\begin{aligned}
&H_p(t_2) - H_p(t_1) = \\
&= g\left(\left\lfloor \int_0^{t_2} f_{\text{osc}}(\tau)d\tau \right\rfloor - \left\lfloor \int_0^{t_1} f_{\text{osc}}(\tau)d\tau \right\rfloor\right) \\
&= g\left(\left\lfloor N_1 + \int_{t_{N_1}}^{t_2} f_{\text{osc}}(\tau)d\tau \right\rfloor - N_1\right) \\
&= g\left(\left\lfloor \int_{t_{N_1}}^{t_1} f_{\text{osc}}(\tau)d\tau + \int_{t_1}^{t_2} f_{\text{osc}}(\tau)d\tau \right\rfloor\right) \qquad (2.13)
\end{aligned}
$$

où $gN_1$ est la valeur de l'horloge quand $t = t_1$ et $t_{N_1}$ l'instant auquel la valeur de l'horloge a changé (c.-à-d., $t_{N_1} \triangleq \min\{t : H_p(t) = N_1 g\}$).

De l'équation 2.9 (dérive d'une horloge):

$$(1-\rho)f_{\text{nom}} \leq f_{\text{osc}}(t) \leq (1+\rho)f_{\text{nom}}$$

La valeur $t_2 - t_1$ est maximale dans 2.13 si $t_1 = t_{N_1}$ et $f_{\text{osc}}(t) = (1-\rho)f_{\text{nom}}$. Dans ce cas, la première intégrale est nulle et la deuxième est l'intégrale d'une fonction constante:

$$H(t_2) - H(t_1) = g\lfloor (t_2 - t_1)(1-\rho)f_{\text{nom}} \rfloor$$

La plus grande valeur de $t_2 - t_1$ pour laquelle l'équation précédente est nulle est:

$$(t_2 - t_1)(1-\rho)f_{\text{nom}} = 1^-$$

$$\therefore \Delta t_{\text{min,m}} = \frac{1+\rho}{f_{\text{nom}}} = (1+\rho)g \qquad (2.14)$$

**Fin de solution**

La connaissance de la valeur $\Delta t_{\text{min,m}}$ nous permet de définir la propriété de monotonicité tenant compte du fait que les horloges possèdent une granularité non nulle. Ceci est indiqué par la notation $\text{M}_g$.

**Définition 2.14 (Propriété $\text{M}_g$)**

$$\text{M}_g : \qquad t_2 \geq t_1 + (1+\rho)g \Rightarrow H_p(t_2) > H_p(t_1) \qquad (2.15)$$

Pour le concepteur d'un SI, si $\Delta t_{ee}$ est le plus petit délai entre deux événements successifs, qui doivent être estampillés avec des valeurs d'horloge différentes, la granularité de l'horloge doit respecter l'inégalité :

**Lemme 2.1 (Granularité maximale)**

$$g \leq \Delta t_{ee}(1-\rho) \qquad (2.16)$$

Dans un système où deux événements distincts sont distants d'au moins $\Delta t_{ee}$ dans le temps, la valeur de $g$ définie par le lèmme 2.16 garantit que ces événements seront estampillés avec deux valeurs d'horloge différentes. On pourra alors établir la séquence temporelle correcte à partir des estampilles de l'horloge.

### 2.3.3 Dérive

La définition 2.9 pour la dérive ne prend pas en compte la granularité de l'horloge. Dans cette section, nous analyserons comment une horloge physique, à granularité $g$ et avec une dérive $\rho$ (définie selon l'équation (2.9)), mesure la valeur d'un intervalle de temps de $t_1$ à $t_2$.

**Problème:** L'horloge $H_p$ possède une granularité $g$ et une dérive $\rho$ (selon (2.9)). Quelles sont les limites minimale et maximale pour la mesure d'un intervalle de temps qui va de $t = t_1$ à $t = t_2$?

**Solution:**

$$\begin{aligned} H(t_1) &= g\left\lfloor \int_0^{t_1} f_{\text{osc}}(\tau)d\tau \right\rfloor \\ H(t_2) &= g\left\lfloor \int_0^{t_2} f_{\text{osc}}(\tau)d\tau \right\rfloor \\ H(t_2) - H(t_1) &= g\left(\left\lfloor \int_0^{t_2} f_{\text{osc}}(\tau)d\tau \right\rfloor - \left\lfloor \int_0^{t_1} f_{\text{osc}}(\tau)d\tau \right\rfloor\right) \end{aligned}$$

$$\begin{aligned} &H(t_2) - H(t_1) = \\ &= g\left(\left\lfloor \int_{t_{N_1}}^{t_2} f_{\text{osc}}(\tau)d\tau \right\rfloor\right) \\ &= g\left(\left\lfloor \int_{t_{N_1}}^{t_1} f_{\text{osc}}(\tau)d\tau + \int_{t_1}^{t_2} f_{\text{osc}}(\tau)d\tau \right\rfloor\right) \end{aligned}$$

où $N_1 = H(t_1)/gr$, $t_{N_1} \triangleq \min\{t : H(t) = N_1 g\}$. On sait que :

$$0 \leq \int_{t_{N_1}}^{t_1} f_{\text{osc}}(\tau)d\tau < 1$$

(de la définition de $t_{N_1}$)

$$(1-\rho)(t_2 - t_1)f_{\text{nom}} \leq \int_{t_1}^{t_2} f_{\text{osc}}(\tau)d\tau \leq (1+\rho)(t_2 - t_1)f_{\text{nom}}$$

($H$ a une dérive $\rho$) et

$$x - 1 < \lfloor x \rfloor \leq x$$

(de la définition de $\lfloor x \rfloor$) Alors,

$$g\left((1-\rho)(t_2 - t_1)f_{\text{nom}} - 1\right) \leq H(t_2) - H(t_1) \leq g\left((1+\rho)(t_2 - t_1)f_{\text{nom}} + 1\right)$$

$$\therefore (1-\rho)(t_2 - t_1) - g \leq H(t_2) - H(t_1) \leq (1+\rho)(t_2 - t_1) + g \tag{2.17}$$

**Fin de solution**

Quand $t_2 - t_1 \gg g$, l'équation 2.17 devient identique à l'équation 2.5. Dans ce cas, les définitions 2.9 et 2.13 sont deux définitions équivalentes de la dérive de l'horloge physique.

### 2.3.4 Précision

La définition présentée pour la précision ignore l'effet de la granularité et devient inadéquate quand la granularité n'est pas négligeable par rapport à la précision $\delta$. La solution que nous utiliserons est de définir la précision comme une différence de phase des deux horloges, et non comme une différence de valeurs. Après avoir établi une définition qui prend en compte la granularité de l'horloge, nous vérifierons si les horloges physiques possèdent la propriété de précision.

La phase de l'horloge est définie comme $\Phi(t) = \int_0^t f_{\text{osc}}(\tau)d\tau$.

**Définition 2.15 (Propriété de précision avec granularité)** *Deux horloges, $p$ et $q$, sont synchronisées avec une précision $\delta_\Phi$ ssi :*

$$\text{Pg}_{\delta_\Phi} : \qquad \forall t, \quad |\Phi_p(t) - \Phi_q(t)| \leq \delta_\Phi f_{\text{nom}} \tag{2.18}$$

Dans le cas où la granularité et la fréquence ne sont pas identiques, l'équation 2.18 devient :

$$\forall t, \quad |g_p\Phi_p(t) - g_q\Phi_q(t)| \leq \delta_\Phi \tag{2.19}$$

Nous cherchons à trouver maintenant, pour deux horloges $p$ et $q$ qui possèdent la propriété $\text{Pg}_{\delta_\Phi}$, quelle est la valeur $\delta$ (si elle existe) pour que les horloges possèdent aussi la propriété $\text{P}_\delta$.

**Problème:** Pour deux horloges, $p$ et $q$, synchronisées avec une précision $\delta_\Phi$ (selon la définition de l'équation 2.18), quelle est la différence maximale entre les valeurs $H_p(t)$ et $H_q(t)$?

**Solution:** Si les deux horloges sont synchronisées avec une différence de phase égale ou inférieure à $\delta_\Phi f_{\text{nom}}$, $\Phi_p(t) = \Phi_q(t) + \sigma(t)\delta_\Phi f_{\text{nom}}$, $-1 \leq \sigma(t) \leq 1$. Nous avons donc :

$$\begin{aligned} H_q(t) &= g\left\lfloor \int_0^t f_q(\tau)d\tau \right\rfloor \\ H_p(t) &= g\left\lfloor \int_0^t f_q(\tau)d\tau + \sigma(t)\delta_\Phi f_{\text{nom}} \right\rfloor \end{aligned}$$

$$\begin{aligned} & H_p(t) - H_q(t) \\ &= g\left(\left\lfloor \int_0^t f_q(\tau)d\tau + \sigma(t)\delta_\Phi f_{\text{nom}} \right\rfloor - \left\lfloor \int_0^t f_q(\tau)d\tau \right\rfloor\right) \\ &= g\left(\left\lfloor \int_0^{t_N} f_q(\tau)d\tau + \int_{t_N}^t f_q(\tau)d\tau + \sigma(t)\delta_\Phi f_{\text{nom}} \right\rfloor - \left\lfloor \int_0^{t_N} f_q(\tau)d\tau \right\rfloor\right) \\ &= g\left(\left\lfloor \int_{t_N}^t f_q(\tau)d\tau + \sigma(t)\delta_\Phi f_{\text{nom}} \right\rfloor\right) \end{aligned}$$

$t_N$ est l'instant de la dernière incrémentation de l'horloge $H_q$ avant $t$, et $N_q = H_q(t)/g = H_q(t_N)/g$. De la définition de $t_N$ :

$$0 \leq \int_{t_N}^t f_q(\tau)d\tau < 1$$

$$\begin{aligned} g(\lfloor 0 - \delta_\Phi f_{\text{nom}} \rfloor) &\leq H_p(t) - H_q(t) \leq g(\lfloor 1^- + \delta_\Phi f_{\text{nom}} \rfloor) \\ g\lfloor -\delta_\Phi f_{\text{nom}} \rfloor &\leq H_p(t) - H_q(t) \leq g\lfloor 1^- + \delta_\Phi f_{\text{nom}} \rfloor \\ -g\lceil \delta_\Phi f_{\text{nom}} \rceil &\leq H_p(t) - H_q(t) \leq g\lceil \delta_\Phi f_{\text{nom}} \rceil \end{aligned}$$

$$\therefore |\Phi_p(t) - \Phi_q(t)| \leq \delta_\Phi f_{\text{nom}} \Rightarrow |H_p(t) - H_q(t)| \leq g\lceil \delta_\Phi f_{\text{nom}} \rceil \tag{2.20}$$

**Fin de solution**

L'équation 2.20 nous permet de conclure que deux horloges qui possèdent la propriété $\text{Pg}_{\delta_\Phi}$, possèdent aussi la propriété $\text{P}_\delta$, avec $\delta = g\lceil \delta_\Phi f_{\text{nom}} \rceil$. Il est aussi facile de vérifier que :

$$\lim_{g/\delta_\Phi \to 0} g\lceil \delta_\Phi f_{\text{nom}} \rceil = \delta_\Phi \tag{2.21}$$

Quand $\delta_\Phi \gg g$, l'équation 2.21 signifie que les deux définitions de précision (sans et avec granularité) sont équivalentes, avec $\delta_\Phi = \delta$. Alors, la définition de la propriété $\text{Pg}_{\delta_\Phi}$ selon l'équation 2.18 permet d'avoir une définition de précision qui ne dépend pas de la granularité (la comparaison entre les deux horloges est faite avant arrondi de la valeur au multiple de $g$ inférieur). Elle est donc plus générale et résulte dans une définition identique quand $\delta_\Phi \gg g$ (c'est-à-dire, quand la granularité est négligeable par rapport à la précision).

Cherchons aussi à trouver la valeur $\hat{\delta}$ dans la propriété $\text{P}'$ pour un système d'horloges synchronisées avec une précision $\delta_\Phi$ (selon la définition de l'équation 2.18). Cette valeur est le délai maximal entre deux instants pour lesquels deux horloges affichent la même valeur.

**Problème:** Pour deux horloges, $p$ et $q$, synchronisées avec une précision $\delta_\Phi$ (selon la définition de l'équation 2.18), quelle est le délai maximal entre les instants $t_p$ et $t_q$ tels que $H_p(t_p) = H_q(t_q)$?

**Solution:**

$$\max\{t_p - t_q\} : H_p(t_p) - H_q(t_q) = 0 \tag{2.22}$$

$$\begin{aligned}
H_q(t_q) &= g\left\lfloor \int_0^{t_q} f_q(\tau)d\tau \right\rfloor \\
H_p(t_p) &= g\left\lfloor \int_0^{t_p} f_q(\tau)d\tau + \sigma(t)\delta_\Phi f_{\text{nom}} \right\rfloor \\
H_p(t_p) - H_q(t_q) &= g\left( \left\lfloor \int_0^{t_p} f_q(\tau)d\tau + \sigma(t)\delta_\Phi f_{\text{nom}} \right\rfloor - \left\lfloor \int_0^{t_q} f_q(\tau)d\tau \right\rfloor \right) \\
&= g\left( \left\lfloor \int_{t_{N_q}}^{t_p} f_q(\tau)d\tau + \sigma(t)\delta_\Phi f_{\text{nom}} \right\rfloor \right)
\end{aligned}$$

$N_q = H_q(t_q)/g$ et $t_{N_q} \triangleq \min\{t : H_q(t) = N_q g\}$.

On cherchera la valeur maximale de $(t_p - t_{N_q})$ telle que :

$$\lfloor (t_p - t_{N_q})\bar{f} + \sigma\delta_\Phi f_{\text{nom}} \rfloor = 0$$

où $\bar{f}$ est la valeur moyenne de $f_q(t)$ dans l'intervalle $[t_{N_q}, t_p[$. $(t_p - t_{N_q})$ est maximale quand $\bar{f} = (1-\rho)f_{\text{nom}}$ et $\sigma(t) = -1$. Dans ce cas :

$$\left\lfloor (t_p - t_{N_q})f_{\text{nom}}(1-\rho) - \frac{\delta_\Phi}{g} \right\rfloor = 0$$

$$(t_p - t_{N_q}) f_{\text{nom}}(1-\rho) - \frac{\delta_\Phi}{g} < 1$$

$$(t_p - t_{N_q})(1-\rho) < g + \delta_\Phi$$

$$t_p - t_{N_q} < (g + \delta_\Phi)(1+\rho)$$

Sachant que $t_q \geq t_{N_q}$, $t_p - t_q$ est maximale quand $t_q = t_{N_q}$. On a donc

$$t_p - t_q < (g + \delta_\Phi)(1+\rho) \tag{2.23}$$

La valeur du côté droit de l'équation 2.23 est la plus grande valeur de $t_p - t_q$ pour laquelle l'équation 2.22 est valide.

$$\therefore t_p - t_q \geq (g + \delta_\Phi)(1+\rho) \Rightarrow H_p(t_p) - H_q(t_q) > 0 \tag{2.24}$$

**Fin de solution**

De l'équation 2.23, on conclut que deux horloges $p$ et $q$ qui possèdent la propriété $\text{Pg}_{\delta_\Phi}$, possèdent aussi $\text{P}'_{\hat{\delta}}$, avec $\hat{\delta} = (g + \delta_\Phi)(1+\rho)$.

Après avoir trouvé l'équivalence entre la propriété $\text{Pg}_{\delta_\Phi}$ et les deux autres définitions de la précision, l'étape suivante est la vérification de la propriété $\text{Pg}_{\delta_\Phi}$ pour une horloge physique.

**Problème:** Deux horloges physiques, $p$ et $q$, décrites par l'équation 2.10 et possédant les propriétés $\text{M}_g$ et $\text{D}_\rho$, possèdent-elles la propriété $\text{Pg}_{\delta_\Phi}$?

**Solution:** Par hypothèse, $\Phi_p(0) = \Phi_q(0) = 0$ et $H_p(0) = H_q(0) = 0$.

Remarquons d'abord que:

$$\begin{aligned}\Phi_p(t) - \Phi_q(t) &= \int_0^t f_p(\tau)d\tau - \int_0^t f_q(\tau)d\tau \\ &= \int_0^t f_p(\tau) - f_q(\tau)d\tau \end{aligned} \tag{2.25}$$

Supposons $f_p(t) = (1+\rho)f_{\text{nom}}$ et $f_q(t) = (1-\rho)f_{\text{nom}}$. Dans ces conditions $f_p(t) - f_q(t) = 2\rho f_{\text{nom}}$. Alors,

$$\begin{aligned}|\Phi_p(t) - \Phi_q(t)| &= |\int_0^t f_p(\tau) - f_q(\tau)d\tau| \\ &= |\int_0^t 2\rho f_{\text{nom}} d\tau| \\ &= |2\rho f_{\text{nom}} t| \\ &= 2\rho f_{\text{nom}} t \end{aligned} \tag{2.26}$$

Le résultat précédent dit que, pour deux horloges $p$ et $q$ qui possèdent les propriétés $\text{M}_g$ et $\text{Pg}_{\delta_\Phi}$ :

$$\forall_{\delta_\Phi > 0}, \exists_{t>0} : \quad |\Phi_p(t) - \Phi_q(t)| > \delta_\Phi f_{\text{nom}} \tag{2.27}$$

Ces horloges ne possèdent donc la propriété $\text{Pg}_{\delta_\Phi}$. **Fin de solution**

### 2.3.5 Exactitude

Pour l'exactitude, c'est facile de vérifier qu'une horloge physique ne possède pas la propriété $E_\alpha$, c.-à-d.,

$$\forall_{\alpha>0}, \exists_{t>0} : \quad H_p(t) - t > \alpha \tag{2.28}$$

La démonstration est identique au cas de la précision.

### 2.3.6 Conclusion

Nous avons donc vérifié que les horloges physiques sont des horloges qui possèdent les propriétés $M_g$ et $D_\rho$, et ne possèdent ni la propriété $Pg_{\delta_\Phi}$ ni la propriété $E_\alpha$. Une telle horloge physique peut être caractérisée par deux paramètres: $g$ et $\rho$, la granularité et la dérive. Éventuellement, il faudra y ajouter la fréquence, $f_{\text{nom}}$, au cas où $g \neq 1/f_{\text{nom}}$.

On ne doit pas oublier que les résultats présentés pour la précision et l'exactitude sont valables à n'importe quel instant et pour tout intervalle de temps. Ils signifient qu'une horloge physique, toute seule, n'est pas capable de garantir la précision ou l'exactitude *ad infinitum*. Mais ceci n'est pas vrai pour le cas d'un intervalle de temps limité. Par exemple, pour le cas de la précision, si $\pi < \delta$ est la précision à l'instant $t = t_0$, un système avec la propriété $D_\rho$ sera synchronisé avec précision $\delta$ pour tout instant $t$ tel que $t < t_0 + \frac{\delta - \pi}{2\rho}$. Pour des valeurs très petites de $\rho$, cet intervalle peut être assez long et permet de considérer l'horloge comme possédant la propriété P. C'est cette caractéristique qui permet l'existence de systèmes *plesiochrones* (du grec *plesios*, « presque ») ([Lindsey et al., 1985, Kartaschoff, 1991]).

## 2.4 Établissement de l'ordre temporel dans un système d'horloges

Dans cette section, nous utiliserons le modèle présenté pour l'horloge physique pour analyser le fonctionnement d'un système d'horloges synchronisées en ce qui concerne la datation d'événements et l'établissement de la relation de précédence, c'est-à-dire, d'identifier, à partir des estampilles associées à des événements qui se sont produits dans deux sites différentes, leur ordre d'occurrence. Ce système est composé d'un ensemble d'horloges qui possèdent les propriétés $M_g$ et $D_\rho$, et sur lesquelles on fait l'hypothèse de synchronisation, c.-à-d., on suppose que les horloges possèdent la propriété $Pg_{\delta_\Phi}$. La façon dont cette dernière propriété est garantie ne nous intéresse pas pour le moment.

Pour le cas d'un système avec une seule horloge, la propriété $M_g$ nous garantit que deux estampilles de temps différentes correspondent à un seule ordre possible pour les instants associés. Nous avons aussi calculé quel est le délai le plus court entre deux événements pour que les estampilles de l'horloge permettent d'établir leur ordre sans ambiguïté. Nous allons dans la suite essayer de trouver les valeurs correspondantes pour les systèmes distribués. Dans un système distribué, la situation est plus complexe, puisqu'on accepte que les valeurs de deux horloges soient différentes pour le même instant de temps physique. Quelles sont les conditions pour que l'ordre des événements puisse être établie à partir de la datation de ces événements dans deux sites différents? Quel est l'intervalle le plus petit entre deux événements qui garantit que l'ordre peut être établi par l'estampillage sur deux sites différents?

**Problème:** Soit un système de deux horloges, $H_p$ et $H_q$, synchronisées avec une précision $\delta_\Phi$

(équation 2.18). Les deux horloges ont une granularité $g$. Quelle est la différence minimale entre les valeurs $H_p(t_p)$ et $H_q(t_q)$, $\Delta H_{\min,d}$, qui garantit que l'ordre des valeurs des horloges correspond à l'ordre des instants de temps physique?

**Solution:** De la propriété $\mathrm{Pg}_{\delta_\Phi}$ et de l'équation (2.20) on sait que:

$$|H_p(t) - H_q(t)| \leq g\lceil \delta_\Phi f_{\text{nom}} \rceil$$

Alors,

$$H_p(t_p) - H_q(t_q) \geq g\left(\lceil \delta_\Phi f_{\text{nom}} \rceil + 1\right) \Rightarrow t_p > t_q \tag{2.29}$$

La démonstration peut être faite par l'absurde. Supposons que $H_p(t_p) - H_q(t_q) \geq g\left(\lceil \delta_\Phi f_{\text{nom}} \rceil + 1\right)$ et $t_p < t_q$. Si les horloges sont monotones, $H_p(t_q)$ ne peut être inférieure à $H_p(t_p)$, puisque $t_q > t_p$ (par hypothèse). Dans ce cas, $H_p(t_q) - H_q(t_q) \geq g\left(\lceil \delta_\Phi f_{\text{nom}} \rceil + 1\right) > g\lceil \delta_\Phi f_{\text{nom}} \rceil$, ce qui est impossible.

$$\therefore \Delta H_{\text{min,d}} = g\left(\lceil \delta_\Phi f_{\text{nom}} \rceil + 1\right) \tag{2.30}$$

**Fin de solution**

**Problème:**

Soit un système de deux horloges, $H_p$ et $H_q$, synchronisées avec une précision $\delta_\Phi$ (équation 2.18). Les deux horloges ont une granularité $g$ et possèdent les propriétés $\mathrm{M}_g$ et $\mathrm{D}_\rho$. Quelle est l'intervalle minimal (temps physique) entre deux événements sur deux sites différents, $\Delta t_{\text{min,d}}$ qui garantit que son ordonnancement peut être établi à partir des estampilles de temps associées?

**Solution:** Pour répondre à cette question, nous cherchons le plus grand intervalle pour lequel on ne peut pas être sûr de retrouver l'ordre des événements. C'est-à-dire, nous cherchons:

$$\max\{t_p - t_q\} : H_p(t_p) - H_q(t_q) = g\left\lceil \frac{\delta_\Phi}{g} \right\rceil \tag{2.31}$$

$$\begin{aligned}
H_q(t_q) &= g\left\lfloor \int_0^{t_q} f_q(\tau)d\tau \right\rfloor \\
H_p(t_p) &= g\left\lfloor \int_0^{t_p} f_q(\tau)d\tau + \sigma(t)\delta_\Phi f_{\text{nom}} \right\rfloor \\
H_p(t_p) - H_q(t_q) &= g\left(\left\lfloor \int_0^{t_p} f_q(\tau)d\tau + \sigma(t)\delta_\Phi f_{\text{nom}} \right\rfloor - \left\lfloor \int_0^{t_q} f_q(\tau)d\tau \right\rfloor\right) \\
&= g\left(\left\lfloor \int_{t_{N_q}}^{t_p} f_q(\tau)d\tau + \sigma(t)\delta_\Phi f_{\text{nom}} \right\rfloor\right)
\end{aligned}$$

$N_q = H_q(t_q)/g$ et $t_{N_q} \triangleq \min\{t : H_q(t) = N_q g\}$.

On cherche la valeur maximale de $(t_p - t_{N_q})$ telle que:

$$\left\lfloor (t_p - t_{N_q})\bar{f} + \sigma\delta_\Phi f_{\text{nom}} \right\rfloor = \left\lceil \frac{\delta_\Phi}{g} \right\rceil$$

où $\bar{f}$ est la valeur moyenne de $f_q(t)$ dans l'intervalle $[t_{N_q}, t_p[$. $(t_p - t_{N_q})$ est maximale quand $\bar{f} = (1-\rho)f_{\text{nom}}$ et $\sigma(t) = -1$. Dans ce cas :

$$\left\lfloor (t_p - t_{N_q}) f_{\text{nom}}(1-\rho) - \frac{\delta_\Phi}{g} \right\rfloor = \left\lceil \frac{\delta_\Phi}{g} \right\rceil$$

$$(t_p - t_{N_q}) f_{\text{nom}}(1-\rho) - \frac{\delta_\Phi}{g} \leq \left\lceil \frac{\delta_\Phi}{g} \right\rceil + 1$$

$$(t_p - t_{N_q})(1-\rho) \leq g\left(\left\lceil \frac{\delta_\Phi}{g} \right\rceil + 1\right) + \delta_\Phi$$

$$t_p - t_{N_q} \leq \left[ g\left(\left\lceil \frac{\delta_\Phi}{g} \right\rceil + 1\right) + \delta_\Phi \right](1+\rho)$$

Sachant que $t_q \geq t_{N_q}$, $t_p - t_q$ est maximal quand $t_q = t_{N_q}$. On a donc

$$t_p - t_q \leq \left[ g\left(\left\lceil \frac{\delta_\Phi}{g} \right\rceil + 1\right) + \delta_\Phi \right](1+\rho) \tag{2.32}$$

La valeur du côté droit de l'équation 2.32 est la plus grande valeur de $t_p - t_q$ pour laquelle l'équation 2.31 est valide.

$$\therefore t_p - t_q > \left[ g\left(\left\lceil \frac{\delta_\Phi}{g} \right\rceil + 1\right) + \delta_\Phi \right](1+\rho) \Rightarrow H_p(t_p - H_q(t_q) > g\left(\lceil \delta_\Phi f_{\text{nom}} \rceil + 1\right) \tag{2.33}$$

et

$$\Delta t_{\text{min,d}} = \left[ g\left(\left\lceil \frac{\delta_\Phi}{g} \right\rceil + 1\right) + \delta_\Phi \right](1+\rho) \tag{2.34}$$

**Fin de solution**

L'équation (2.30) garantit que, si la différence entre les valeurs des estampilles est égale ou supérieur à $\Delta H_{\text{min,d}}$, alors l'ordre des événements correspond à l'ordre des valeurs des estampilles associées. L'équation (2.33) garantit que, si les événements sont distants d'au moins $\Delta t_{\text{min,d}}$, la différence entre les estampilles ne sera jamais inférieure à $\Delta H_{\text{min,d}}$ et, donc, l'ordre des événements est établi sans ambiguïté par le système.

Considérons la situation où deux événements se produisent sur deux sites, $p$ et $q$, à des instants $t_p$ et $t_q$, respectivement. Chacun des sites attribue une estampille temporelle locale à l'événement qu'il observe, $H_p(t_p)$ et $H_q(t_q)$. $\Delta t = t_p - t_q$ est le délai temps physique et $\Delta H = H_p(t_p) - H_q(t_q)$ la différence entre les valeurs de l'horloge associées. La figure 2.4 présente les rapports entre les valeurs des délais de temps physique et les différences des valeurs des horloges. Les flèches indiquent la plus grande valeur de $\Delta H$ (resp. $\Delta t$) qui peut correspondre à une certaine valeur de $\Delta t$ (resp. $\Delta H$). Les courbes sont symétriques au tour de l'origine des axes ; on représentera seulement la moitié droite (des valeurs positives).

Les résultats nous permettent d'identifier trois régions sur l'axe du délai temps physique, qui représente le délai entre les événements :

- $\Delta t \in ]0, (g + \delta_\Phi)(1+\rho)]$ : dans ce cas, il n'y a aucune garantie sur la cohérence entre l'ordre des événements et l'ordre des estampilles des horloges. Pour ces valeurs de $\Delta t$, $\Delta H$ peut être négative ou positive.

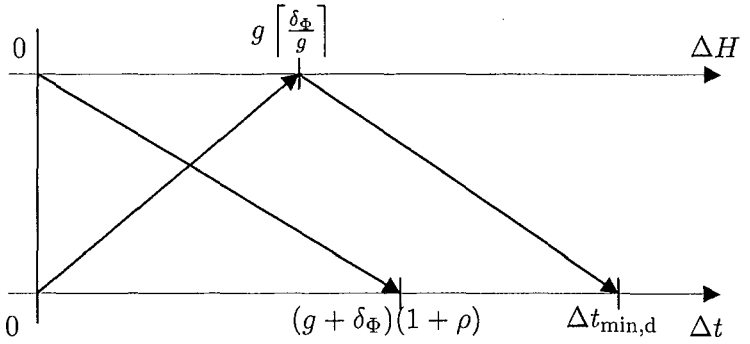



FIG. 2.4: *Rapport entre les valeurs de temps et horloges*

| Système | Paramètre | Valeur | Propriétés |
|---|---|---|---|
| Mono-site | $\Delta H_{\min}$ | $g$ | $M_g$ |
| | $\Delta t_{\min}$ | $g(1+\rho)$ | $M_g, D_\rho$ |
| Distribué | $\Delta H_{\min}$ | $g\left(\left\lceil\frac{\delta_\Phi}{g}\right\rceil+1\right)$ | $M_g, Pg_{\delta_\Phi}$ |
| | $\Delta t_{\min}$ | $\left[g\left(\left\lceil\frac{\delta_\Phi}{g}\right\rceil+1\right)+\delta_\Phi\right](1+\rho)$ | $M_g, D_\rho, Pg_{\delta_\Phi}$ |

TAB. 2.2: *Conditions pour l'établissement de l'ordre des événements*

- $\Delta t \in ](g+\delta_\Phi)(1+\rho), \Delta t_{\min,d}]$ : $\Delta t$ et $\Delta H$ sont cohérents: l'ordre établi à partir des différences des valeurs de l'horloge correspond à l'ordre des instants temps physique ; mais, pour un observateur placé à l'intérieur du système informatique, qui ne possède que les valeurs des horloges, rien ne peut être conclu sur $\Delta t$. Les valeurs de l'horloge qu'il connaît peuvent correspondre à $t_p > t_q$ ou à $t_q > t_p$.

- $\Delta t \in ]t_{\min,d}, +\infty]$ : $\Delta t$ et $\Delta H$ sont cohérents, et un observateur qui ne possède que les valeurs des horloges peut connaître, avec certitude, l'ordre des événements.

Il y a donc un segment sur la droite des $\Delta t$ où l'ordre des estampilles des horloges correspond à l'ordre des instants d'occurrence des événements mais qui n'est pas utile pour un observateur qui ne connaît que les valeurs des horloges. Pour les $\Delta t$ qui appartiennent à ce segment, il n'est pas garanti que les $\Delta H$ correspondants permettent à l'observateur de prendre une décision. Une telle garantie n'existe que pour les valeurs de $\Delta t$ supérieures à $\Delta t_{\min,d}$.

Le tableau 2.2 présente les résultats obtenus en ce qui concerne l'établissement de l'ordre des événements à partir des estampilles. Il y a des conditions qui sont imposées soit à la différence minimale entre les valeurs des horloges ($\Delta H_{\min}$), soit au délai minimal entre événements ($\Delta t_{\min}$), pour un système avec une seule horloge (mono-site) et pour un système distribué (avec plusieurs horloges). Ce tableau indique aussi les propriétés qui sont requises pour chacun des cas. La propriété de monotonicité est omniprésente: elle est requise pour toutes les situations. La propriété de la dérive bornée est requise pour la validité du $\Delta t_{\min}$ pour tous les systèmes (mono-site ou distribués). Remarquons que les systèmes distribués requièrent toujours la propriété de précision, ce qui conduit à la conséquence suivante: sans imposer cette propriété, aucune conclusion ne peut être faite sur l'ordre des événements qui

apparaissent sur plusieurs sites. Les résultats pour les systèmes mono-site sont un cas particulier des résultats pour les systèmes distribués, correspondant à $\delta_\Phi = 0$.

### 2.4.1 Formulation générale

Les résultats précédents nous ont permis d'établir des bornes pour les différences entre les valeurs des horloges en fonction du délai respectif, et vice-versa, pour des cas particuliers. Dans cette section, nous présenterons la généralisation de ces résultats. Nous cherchons à répondre aux deux questions suivantes :

**Problème:** Pour un système distribué, où les propriétés $Pg_{\delta_\Phi}$ et $D_\rho$ sont garanties,

- Quelles sont les valeurs minimale et maximale pour la différence entre horloges, $\Delta H$, qui correspondent à un délai de temps physique $\Delta t$?
- Quelles sont les valeurs minimale et maximale pour le délai de temps physique $\Delta t$ qui correspondent à une différence entre horloges $\Delta H$?

**Solution:**

$$\begin{aligned} \Delta t &= t_p - t_q \\ \Delta H &= H_p(t_p) - H_q(t_q) \\ H_p(t_p) &= g\left\lfloor \int_0^{t_p} f_q(\tau)d\tau + \sigma(t)\delta_\Phi f_{\text{nom}} \right\rfloor \\ H_q(t_q) &= g\left\lfloor \int_0^{t_q} f_q(\tau)d\tau \right\rfloor \end{aligned}$$

Après quelques manipulations, on arrive à

$$\Delta H = g\left\lfloor \int_{t_{N_q}}^{t_p} f_q(\tau)d\tau + \sigma(t)\delta_\Phi f_{\text{nom}} \right\rfloor$$

$\bar{f}_q$ représente la valeur moyenne de $f_q(t)$ dans l'intervalle $[t_{N_q}, t_p]$. De la propriété $D_\rho$, nous savons que $(1-\rho)f_{\text{nom}} \leq \bar{f}_q \leq (1+\rho)f_{\text{nom}}$. L'équation précédente peut être réécrite :

$$\Delta H = g\left\lfloor \bar{f}_q(t_p - t_{N_q}) + \sigma(t)\delta_\Phi f_{\text{nom}} \right\rfloor \tag{2.35}$$

Dans cette section, les paramètres qui varient sont $\bar{f}_q$, $(t_p - t_{N_q})$ et $\sigma(t)$. Les résultats seront trouvés en remplaçant ces quantités par leurs valeurs extrêmes qui maximisent ou minimisent la quantité dont on cherche à trouver les bornes.

**a) Calcul des bornes de $\Delta H$**

| Par. | $\max \Delta H$ | $\min \Delta H$ |
|---|---|---|
| $\bar{f}_q$ | $(1+\rho)f_{\text{nom}}$ | $(1-\rho)f_{\text{nom}}$ |
| $\sigma$ | 1 | $-1$ |
| $t_p - t_{N_q}$ | $\Delta t + (1-\rho)g$ | $\Delta t$ |

TAB. 2.3: *Calcul des bornes pour $\Delta H$.*

En remplaçant les valeurs du tableau 2.3 dans (2.35), on obtient :

$$g\lfloor(1-\rho)f_{\text{nom}}\Delta t - \delta_\Phi f_{\text{nom}}\rfloor \leq \Delta H \leq g\lfloor(1+\rho)f_{\text{nom}}\Delta t + \delta_\Phi f_{\text{nom}} + 1\rfloor \tag{2.36}$$

$$(1-\rho)\Delta t - \delta_\Phi - g \leq \Delta H \leq (1+\rho)\Delta t + \delta_\Phi + g \tag{2.37}$$

La première équation est plus restrictive que la deuxième, mais celle-ci est par contre plus facile à traiter analytiquement.

**b) Calcul des bornes de $\Delta t$**

| Par. | $\max \Delta t$ | $\min \Delta t$ |
|---|---|---|
| $f_q$ | $(1-\rho)f_{\text{nom}}$ | $(1+\rho)f_{\text{nom}}$ |
| $\sigma$ | $-1$ | $1$ |
| $t_p - t_{N_q}$ | $\Delta t$ | $\Delta t + (1-\rho)g$ |

TAB. 2.4: *Calcul des bornes pour $\Delta t$.*

En remplaçant les valeurs du tableau 2.4 dans (2.35), on obtient :

$$(1-\rho)(\Delta H - \delta_\Phi - g) \leq \Delta t \leq (1+\rho)(\Delta H + \delta_\Phi + g) \tag{2.38}$$

**Fin de solution**

En partant de l'hypothèse qu'un système distribué possède les propriétés $Pg_{\delta_\Phi}$ et $D_\rho$, les résultats obtenus nous permettent donc d'établir, pour un intervalle de temps $\Delta t$, les bornes minimale et maximale pour la différence des valeurs des horloges $\Delta H$ qui lui correspondent (équations 2.36 et 2.37) et, réciproquement, les $\Delta t$ qui correspondent à une certaine valeur de $\Delta H$ (2.38).

### 2.4.2 Comparaison avec des résultats existants

Le problème d'établissement des relations de précédence entre événements à partir des estampilles associées a déjà été abordé. Cette problématique dans le domaine des systèmes distribués a été étudiée par Kopetz ([Kopetz, 1997]) et Veríssimo([Veríssimo, 1994]). Kopetz établit la « condition de la valeur raisonnable » (*reasonableness condition* [Kopetz, 1997] ou *granularity condition* [Kopetz, 1992]) : $\delta_\Phi < g$. Cette condition garantit que les estampilles associées au même événement produites dans plusieurs sites ne diffèrent jamais de plus de $g$. En partant de l'équation (2.36), avec $\Delta t = 0$, il est facile de vérifier que :

$$\delta_\Phi < g \Rightarrow |\Delta H| \leq g$$

Dans la situation limite, $\delta_\Phi = g^-$, on obtient une valeur de $\Delta H_{\text{min,d}} = 2g$, ce qui est conforme aux résultats présentés dans [Kopetz, 1997]. La mesure d'un intervalle de temps ([Kopetz, 1997, Sect. 3.2.2]) est conforme à eq. 2.38, sauf pour l'effet de la dérive, que Kopetz ne considère pas (le système considéré par Kopetz a la propriété $D_0$). Sous la condition de

valeur raisonnable, $\Delta t_{\min,d} = (1+\rho)3g$ ; ceci correspond à un système à précédence du type $0/3g$ (« $0/3g$-precedent » selon la dénomination de Kopetz, [Kopetz, 1997, Sect. 3.2.3]).

Dans le cas des travaux de Veríssimo ([Veríssimo, 1994]), nous remarquons que les résultats sur l'établissement de l'ordre des événements à partir des estampilles ([Veríssimo, 1994, Figure 6]) sont des cas particuliers de l'équation 2.38 pour un système qui possède la propriété $D_0$. Nos résultats sont aussi conformes à ceux présentés par Veríssimo.

## 2.5 Conclusions

Ce chapitre a étudié le problème de l'usage des horloges dans les systèmes informatiques en général, et dans un système distribué en particulier. Les spécifications possibles pour une application définissent quatre propriétés, qui permettent de vérifier si une horloge est adéquate à l'application considérée. Ces propriétés, la monotonicité, la dérive, la précision et l'exactitude, sont paramétrables, ce qui permet de quantifier la qualité d'une horloge.

Un modèle pour l'horloge physique a permis de vérifier qu'une telle horloge possède seulement les propriétés de monotonicité et dérive ; elle peut être caractérisée par sa granularité et par sa dérive. Si les propriétés de précision et d'exactitude sont requises, elles doivent être garanties par des moyens supplémentaires.

Le problème d'établissement de l'ordre des événements à partir des estampilles a été étudié. Même si l'hypothèse de synchronisation est faite, des conditions additionnelles doivent être imposées pour que l'ordre soit correctement établi. Notamment, nous avons présenté des valeurs minimales pour l'intervalle entre événements et pour la différence entre les estampilles associées. Les résultats sont conformes aux travaux présentés dans la littérature, mais ils sont présentés d'une façon plus générale, où aucune hypothèse a-priori n'est faite sur le rapport entre les différents paramètres. En particulier, nos résultats permettent de considérer l'effet de la dérive des horloges et de rassembler des cas différentes sous une même formule mathématique et avec une approche unifiée.

# Chapitre 3

# Solutions pour la synchronisation des horloges

Dans les chapitres précédents, nous avons mis en évidence le besoin des horloges pour les systèmes informatiques, et en particulier pour les systèmes distribués. Quatre propriétés ont été identifiées : la monotonicité, la dérive bornée, la précision et l'exactitude. Ces quatre propriétés permettent de vérifier si une horloge ou un ensemble d'horloges sont adéquats pour une application donnée. Le dispositif sur lequel se basent les horloges des ordinateurs contemporains, l'horloge physique, n'est capable de garantir que deux de ces propriétés : M et D. Des moyens supplémentaires sont nécessaires pour les applications qui demandent soit la propriété P, soit la propriété E. Dans ce chapitre, nous introduisons les méthodes à la disposition du concepteur des systèmes distribués pour garantir que les horloges de son système possèdent les propriétés P ou E.

Dans notre cas, les problèmes qui se posent au concepteur d'un SD sont : « Comment garantir que les horloges d'un système distribué affichent toutes la même valeur au même instant? » (garantir la précision) et « Comment garantir que les horloges d'un système distribué affichent une valeur proche d'un temps standard? » (garantir l'exactitude). Nous remarquons ici une certaine similitude entre les deux problèmes. Dans un cas général, il s'agit de mettre l'horloge locale de chaque site en accord avec une référence donnée, normalement extérieure au site. Dans le cas de la précision, cette référence doit être identique pour tous les sites, mais sans être forcément en accord avec un temps standard. Dans le cas de l'exactitude, cette référence doit être en accord avec un temps standard [1].

1. *Un* temps standard et non *le* temps standard, car il y en a plusieurs. Même des auteurs renommés dans la synchronisation des horloges divergent sur le standard à choisir ; Mills, par exemple, préfère UTC ([Mills, 1991a]) alors que Kopetz défend l'usage de TAI ([Kopetz et Ochsenreiter, 1987]). Ce choix est aussi fonction du domaine de travail et de l'application envisagée. L'algorithme de Mills synchronise les horloges du réseau Internet. L'intérêt c'est d'avoir une valeur pour l'horloge de l'ordinateur identique à celle de la montre que l'utilisateur porte. Le temps affiché par cette horloge doit prendre en compte des évènements comme les secondes ajoutées ou retirées à la valeur du temps UTC pour le garder proche de UT0. À cause de ces corrections, UTC n'est pas chronoscopique. Kopetz, qui travaille surtout dans le domaine du temps réel, préfère un temps chronoscopique, comme TAI. Un exemple tiré de [Neumann, 1995] peut aider à comprendre le choix de Kopetz. Dans une fonderie en Allemagne, le système de commande utilisait pour ajuster son horloge l'heure diffusée par l'observatoire de Braunschweig. À la fin avril 1993, on est passé à l'horaire d'été. Cette nuit, l'horloge du système a avancé de 1h59mn à 3h00mn en une seule minute. Quand le système de commande a pensé que le temps de refroidissement du métal fondant était écoulé (une heure en avance), le résultat a été le versement de métal encore trop chaud, avec des dommages considérables pour l'usine.

Un ensemble d'horloges qui possèdent la propriété $E_\alpha$ possède aussi la propriété $P_{2\alpha}$. Ceci dit, il serait apparemment suffisant de garantir E, puisque P n'en serait qu'un cas particulier. Cette conclusion n'est pas vraie ; deux observations permettent de le constater:

- il est possible d'avoir des applications qui demandent les propriétés $E_\alpha$ et $P_\delta$, où $\alpha$ a une valeur supérieure à $2\delta$. L'accord avec le temps standard peut être fait avec une erreur d'une seconde, par exemple, mais on peut avoir besoin d'établir la séquence des actions avec une granularité beaucoup plus petite. Dans ce cas, la précision, telle qu'elle est garantie par l'exactitude des horloges, n'est pas suffisante ; il faudra des moyens supplémentaires pour garantir la précision telle qu'elle a été spécifiée.

- l'accès à un étalon de temps peut ne pas être possible pour certaines applications. Dans ce cas, on peut avoir les moyens pour garantir la précision, mais pas l'exactitude.

Alors, si d'une part, on trouve des similitudes entre le problème de garantir la précision et celui de garantir l'exactitude, il y a d'autre part aussi des raisons pour les traiter comme des problèmes séparés.

## 3.1 Une structure commune

Du point de vue abstrait, la synchronisation des horloges dans un système distribué consiste à associer, sur chaque site, une valeur d'horloge à un instant de temps physique. C'est-à-dire, à accrocher le temps d'horloge au temps physique.

Le concept d'associer une valeur d'horloge à un instant temps physique a été formalisé par Schneider ([Schneider, 1986, Schneider, 1987]) à travers la notion d'une « source de temps sûre » (*Reliable Time Source*, ou RTS). Cette source de temps est censée: i) générer un événement qui force le site à resynchroniser son horloge ; et ii) produire une valeur pour que le site ajuste son horloge. Les algorithmes de synchronisation diffèrent par la façon dont ils construisent la source de temps et, en particulier, par les choix faits concernant trois aspects:

- la détection de l'événement ;

- la lecture des horloges distantes ;

- l'usage des valeurs reçues pour le calcul de la nouvelle valeur.

Les deux premiers aspects déterminent comment le site construit un ensemble de données de temps d'horloges et instants de temps physique. Ces données seront ensuite utilisées pour calculer la nouvelle valeur pour l'horloge et l'instant de resynchronisation, ce qui est l'objet du troisième aspect. Dans la suite de ce chapitre, nous présenterons les différentes méthodes qui sont disponibles pour accomplir ces deux objectifs: détecter un événement de synchronisation et lire une horloge distante.

## 3.2 Solutions pour la synchronisation

On trouve aujourd'hui un grand nombre de méthodes pour la synchronisation des horloges dans les systèmes distribués. Celles-ci vont des solutions qui demandent des infrastructures

matérielles spécifiques, jusqu'à celles qui sont basées seulement sur l'échange de messages sur le réseau. On peut regrouper les solutions existantes en trois classes ([Fonseca et al., 1998c]) :

- solutions basées sur la réception d'un signal de référence ;
- solutions basées sur le matériel ;
- solutions basées sur l'échange de messages sur le réseau.

Cette classification est identique à celle proposée par Olson et Shin ([Olson et Shin, 1994]). Nous la considérons comme une classification des *solutions* pour la synchronisation des horloges, pas seulement des algorithmes (le concept d'algorithme est presque inexistant dans la première classe, par exemple). Remarquons que seulement le deux dernières sont des solutions orientées vers les systèmes distribués. Dans la première, le fait que les sites à synchroniser soient intégrés dans un SD ou pas est indifférent. Chaque site se comporte comme une entité autonome par rapport au site de référence (émetteur du signal).

Nous ne nous proposons pas à faire une classification extensive des algorithmes de synchronisation ; pour les aspects de classification, le lecteur intéressé peut s'adresser à des travaux comme [Ellingson et Kulpinski, 1973, He, 1993, Kartaschoff, 1991, Lindsey et al., 1985, Ramanathan et al., 1990b, Simons et al., 1989, Suri et al., 1994]. Notre objectif est de mettre en évidence l'importance des solutions algorithmiques basées sur l'échange de messages dans le réseau, surtout en comparaison avec celles basées sur la réception du signal de référence.

Le prix de certaines solutions — comme celles basées sur le système GPS — a baissé considérablement durant les dernières années. C'est donc inévitable, à propos des efforts pour améliorer la synchronisation par des méthodes algorithmiques, de tenir compte de la question : « Pourquoi tant d'effort à developper les algorithmes si on peut faire la synchronisation par GPS? » Nous sommes d'avis que les systèmes comme GPS peuvent apporter un plus au problème de la synchronisation des horloges mais ne sont pas (au moins, pour le moment) une réponse universelle aux problèmes qui sont posés. Cette section aborde cette problématique, en identifiant les avantages et inconvénients de chacune des solutions.

### 3.2.1 Solutions basées sur la réception du signal de référence

Les solutions de cette classe sont basées sur la réception d'une « marque de temps », qui est émise à partir de sites dotés d'une horloge de très haute qualité. Ces sites peuvent être accédés par appel téléphonique, par réception de certaines stations radio ou par la réception des satellites du système GPS. Normalement, l'horloge de ces sites est une horloge atomique.

Dans l'accès par appel téléphonique, l'heure affichée par une horloge de très haute qualité est codifiée dans un message et envoyée à travers la ligne jusqu'au récepteur. La transition correspondant à un des bits du message marque l'instant associé à la valeur transmise. Des services de ce type existent, a notre connaissance, en Europe (Suède, Italie, Autriche, Royaume-Uni et Pays Bas), Amérique (États Unis et Canada) et Océanie (Nouvelle Zélande et Australie). Ce sont des services à usage local: l'heure fournie est l'heure locale, sans indication du fuseau horaire. Le format du message envoyé change selon le fournisseur de service. Cette solution demande l'installation d'un modem et accès au réseau téléphonique. Le serveur doit être accessible à un tarif raisonnable.

Certaines stations radio offrent un service d'accès à une horloge atomique ou à une autre horloge de très haute qualité. Des exemples sont les stations WWV, WWVH et WWVB

aux États-Unis (gérés par le NIST), DFC77 en Allemagne et France-Inter, en France. L'information sur l'heure est incluse dans le codage des signaux émis ; des récepteurs radio permettent de capter les signaux de ces stations et de produire une valeur pour l'horloge et l'instant associé. Une synthèse des techniques utilisées pour le faire peut être trouvée dans [Lichtenecker, 1997].

L'utilisation des signaux radio permet d'éliminer le câblage. Mais la mise en œuvre d'une telle solution a d'autres aspects qui doivent être pris en compte. Son implantation demande la maîtrise des aspects de réception et traitement des signaux radio, ce qui est normalement au-delà des compétences du concepteur de systèmes distribués. Celui-ci doit donc utiliser des solutions matérielles (et parfois, logicielles) développées par quelqu'un d'autre, et qui sont insérées dans le système comme une boîte noire. Cette approche contraint son choix aux solutions qui lui sont proposées et limite le développement de solutions spécifiques, adaptées au système considéré. Cet aspect va influencer tout l'ensemble d'un projet de synchronisation basé sur cette technique.

Il faut placer les sites à synchroniser sur des endroits où la réception des signaux radio soit possible avec la qualité requise. Les endroits où la réception est trop faible, ou où le niveau des interférences électromagnétiques est trop fort, sont donc éliminés. Ceci peut interdire son application dans certains bâtiments (à cause de l'effet de blindage magnétique de la structure) et dans les environnements industriels (où le niveau d'émissions électromagnétiques peut être trop élevé).

L'interface du récepteur doit être compatible avec le matériel et le logiciel existant dans le site. Cet aspect est spécialement important quand on utilise des solutions fournies « prêtes à utiliser ». Les dimensions physiques du récepteur et de l'antenne doivent être prises en compte. La consommation d'énergie par le récepteur peut être importante, surtout dans le cas d'équipements autonomes. Finalement, le prix de cette solution est aussi un aspect à considérer. Surtout pour les systèmes où la réduction des coûts est un facteur critique (typiquement, le cas de beaucoup de systèmes distribués embarqués), l'installation d'un récepteur dans chaque site peut être impossible. Le tableau 3.1 présente une résumé des aspects à prendre en compte dans cette solution, et son impact sur le système.

| Aspect | Impact |
|---|---|
| Produits développés par une tierce partie | Limite à la liberté de choix |
| Réception | Localisation des sites |
| Interface | Compatibilité avec le matériel et logiciel existant |
| Prix | Coût du système (multiplié par le nombre de sites) |
| Dimensions | Encombrement de l'ensemble |
| Consommation | Autonomie |

TAB. 3.1: *Aspects à considérer dans la technique d'émission*

Le système GPS (*Global Positioning System*) peut être aussi utilisé comme source de temps. GPS est basé sur une constellation de 24 satellites (y compris 3 suppléants), dont 4 doivent être visibles en ligne de vue par le récepteur pour que le système atteigne la qualité maximale. Ces quatre satellites sont nécessaires pour que le système soit capable d'estimer

la position du récepteur GPS et l'heure actuelle (UTC).

Ce système permet une estimation du temps et de la position avec une très grande qualité. Cette qualité dépend du type de service utilisé. Dans le service standard, disponible pour les utilisateurs civils, les signaux émis par les satellites sont corrompus intentionnellement, pour dégrader la qualité de service. Le service standard (*Standard Positioning System*, SPS) permet une synchronisation avec une erreur de 340ns avec une probabilité de 95%, et une estimation de position (absolue) avec une erreur de 100m (95%) dans l'horizontale et de 156m (95%) dans la verticale. L'intention de cette dégradation est mise en clair par son nom : *Selective Availability*, SA, ou disponibilité sélective. Pour les utilisateurs militaires, il y a un service de précision, (*Precise Positioning Service*, PPS), sans dégradation de signal, qui permet une synchronisation avec une erreur de 200ns (95%) et une erreur de position de 22m (95%, horizontale) et 27,7m (95%, verticale). L'usage de PPS est limité aux utilisateurs militaires autorisés par le Département de Défense des États-Unis. Le lecteur intéressé par ce sujet pourra trouver plus de détails dans [Dana, 1997, Kaplan, 1996]; des exemples d'application de GPS à la synchronisation des horloges sont présentés dans [Sterzbach, 1997, Veríssimo et al., 1997, Wannenmacher et Halang, 1994].

Les remarques faites à propos de l'utilisation de l'émission radio basée sur terre comme source de temps s'appliquent aussi à GPS. L'utilisation de GPS comme source de temps présente en plus deux particularités. La première est le temps de stabilisation du système. Après sa mise en marche, un récepteur peut prendre jusqu'à 12 minutes pour que la qualité totale de service soit atteinte (ceci en supposant que 4 satellites sont en ligne de vue). Il dépend de l'application pour laquelle on doit savoir si un tel délai est acceptable ou pas. Un autre problème potentiel est le fait que GPS est un système militaire. Il a été conçu et développé et il est maintenu par le Département de la Défense des États Unis. Les décisions concernant le fonctionnement du système GPS sont d'abord guidées par les intérêts stratégiques des États-Unis, même si la vaste majorité des utilisateurs sont des civils. Pour l'utilisateur, ceci peut entraîner la dégradation — ou même l'interruption — du service en cas de conflit armé, en permanence ou temporairement, selon la région du planète. Curieusement, pendant des opérations militaires des États-Unis (à Haiti et dans la Guerre du Golfe) l'effet a été l'inverse : la qualité du service SPS a été temporairement équivalent à celle de PPS ([Dana, 1997]).

| Aspect | Impact |
|---|---|
| (Tous les aspects des techniques d'émission) | |
| Temps de stabilisation | Délai de service (jusqu'à 12 minutes) |
| Système militaire | Possible dégradation ou interruption de service |

TAB. 3.2: *Aspects de la synchronisation par GPS*

### 3.2.2 Solutions basées sur le matériel

Dans ce type de solutions, la synchronisation des horloges est garantie par un ensemble de dispositifs qui commandent l'horloge locale. Les signaux des horloges physiques d'un ou

de plusieurs sites sont distribués à tous les sites du système. Chaque site doit produire, en fonction des signaux qui lui arrivent, un signal de référence qui est utilisé pour commander l'horloge locale.

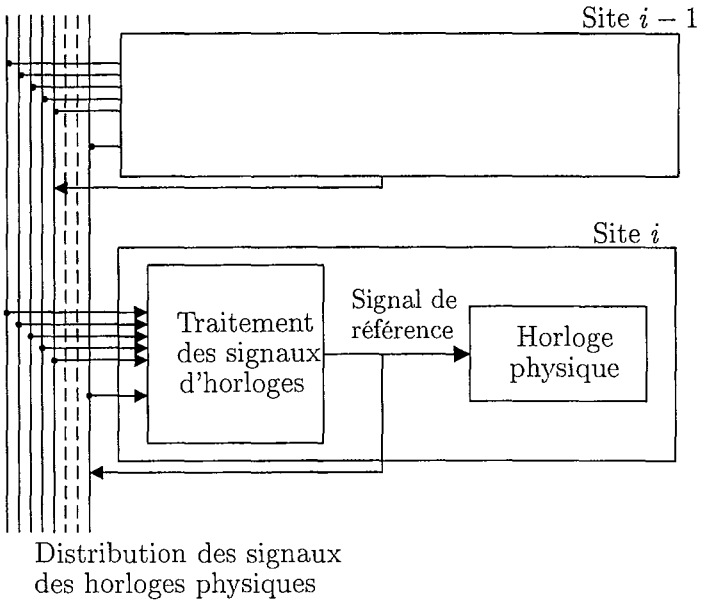



FIG. 3.1: *Synchronisation basée sur le matériel*

Le principe est celui d'une boucle de capture de phase (*phase-lock loop*). La fréquence de l'horloge est générée par l'oscillateur contrôlée par tension (*VCO*) de la boucle de phase. L'entrée de cet oscillateur contrôlé par tension est la sortie d'un détecteur de phase, qui compare la phase de l'horloge locale avec celles des signaux qui arrivent. Ce signal fait augmenter ou diminuer $f_{\text{osc}}$, ce qui fait avancer ou reculer l'horloge locale par rapport aux autres horloges. Chaque site observe les horloges des autres sites et calcule la différence de phase. Ces différences de phase sont ordonnées par valeur croissante ; la médiane est choisie, et la boucle de phase se synchronise avec le signal correspondant à cette médiane ([Hopkins et al., 1978]). Parfois, des fonctions supplémentaires sont ajoutées, comme l'élimination de l'effet du délai de transmission ([Lindsey et al., 1985, Shin et Ramanathan, 1988]).

Cette solution apporte une synchronisation avec une erreur très petite, mais demande l'installation de câblage dédié au transport des signaux d'horloges et du matériel spécifique pour la génération du signal de référence. Les lignes dédiées au transport des signaux des horloges distantes peuvent limiter la distance entre les sites.

Comme l'horloge physique est commandée par un autre dispositif (le sous-système chargé de recevoir les signaux des autres sites et de produire la référence), cette solution doit être considérée dès le début du projet du système distribué. Ce n'est pas une solution facile à intégrer dans un système déjà existant et qui n'était pas conçu à l'origine pour cette méthode de synchronisation. Une synthèse de l'application de cette méthode est présentée dans [Krishna, 1990] et des exemples particuliers dans [Baek et al., 1994, Choi et al., 1990, Davies et Wakerly, 1978, Hopkins et al., 1978, Kessels, 1984, Krishna et al., 1985,

Ramanathan et al., 1990b, Ramanathan et al., 1990a, Shin et Ramanathan, 1987, Shin et Ramanathan, 1988, Vasanthada et al., 1986, Vasanthada et Marinos, 1988]. Il faut remarquer aussi que cette solution concerne la *syntonisation* des horloges (garantir la même fréquence), et pas la synchronisation (garantir la même valeur). Pour synchroniser les horloges, il faudra un moyen supplémentaire de les ajuster avec la même valeur.

### 3.2.3 Solutions basées sur l'échange de messages d'horloges

Une troisième option est celle d'utiliser le réseau qui relie les différents sites du système pour construire une source de temps. Dans ce cas, les sites envoient des messages par le réseau ; le contenu de ces messages et l'instant où ils sont envoyés sont définis par l'algorithme de synchronisation. Ces algorithmes fonctionnent par *vagues* [He, 1993]. Une vague commence au début d'une exécution de l'algorithme de synchronisation et se termine à la prochaine exécution de l'algorithme. La durée d'une vague, appelée *intervalle de resynchronisation*, est notée $R$. Au début de chaque vague, les sites s'échangent des messages, calculent la valeur de la correction à imposer à l'horloge locale et font son ajustement. La durée effective de l'exécution de l'algorithme est appelée *durée de synchronisation* et est notée $S$.

Un des avantages de cette solution est l'utilisation d'une infrastructure déjà existante (le réseau), sans avoir besoin d'équipement spécifique. Se dispenser d'une pièce d'équipement supplémentaire va éliminer (ou, du moins, réduire) les problèmes posés par la consommation, l'encombrement, le prix et l'interface qui s'étaient posés pour les solutions précédentes. L'utilisation du réseau permet de résoudre les problèmes causés par la visibilité des stations émettrices[2] et ne demande aucun câblage additionnel. Finalement, c'est une solution qui apporte, en général, la plus grande liberté de choix au concepteur du système distribué, puisqu'il peut maîtriser les concepts et la technologie utilisés (ce sont les mêmes qu'il utilise pour mettre le système en marche).

La précision et l'exactitude de la synchronisation par échange de messages sont normalement de qualité moindre par rapport aux méthodes précédentes. Cette solution présente aussi l'inconvénient d'augmenter la charge des processeurs qui doivent exécuter l'algorithme de synchronisation en plus des autres activités. La charge sur le réseau est aussi augmentée. Cette augmentation — de la charge des processeurs et du réseau — peut conduire à une dégradation de la qualité du service. Dans un processeur ou un réseau très chargé, on peut s'attendre à une plus grande incertitude dans les délais de réponse ou de transmission d'un message que pour des processeurs ou des réseaux identiques qui ne sont pas trop chargés. Quand une très grande précision ou exactitude sont requises, l'utilisation de matériel spécialisé pour le traitement des messages peut apporter de bons résultats ([Hank, 1997, Kopetz et Ochsenreiter, 1987, Kopetz, 1989, Schossmaier et al., 1997]).

2. Les problèmes de visibilité dont nous parlons ici sont ceux de visibilité de la station, ou des stations, émettrices du signal qui véhicule l'heure, soit une station radio ou soit un satellite GPS. Si on considère un système distribué avec un réseau qui utilise des connexions radio comme couche physique, on s'expose bien sûr à tous les problèmes typiques de la transmission radio, y compris ceux de visibilité et interférence électromagnétique. Mais, dans ce cas, ces problèmes sont intrinsèques au système, et pas spécifiques à la solution de synchronisation des horloges. Si la source de temps était rendue inutilisable à cause de ces problèmes, alors ce serait tout le système qui ne marcherait plus. Pour le moment, nous faisons l'hypothèse que le reste du système marche bien, et nous nous concentrons sur les problèmes spécifiques à la synchronisation d'horloges.

### 3.2.4 Conclusion

Les méthodes d'implantation d'une source de temps qui ont été présentées ne sont pas exclusives. On peut avoir des solutions où plus d'une méthode est appliquée ; des exemples se trouvent dans [Schmid, 1995, Veríssimo et al., 1997]. Dans cette thèse, nous traitons le cas de la synchronisation par échange de messages. L'intérêt pour cette solution est dû principalement à deux raisons : i) c'est une solution qui ne demande pas de l'équipement spécifique ; ii) elle demande des compétences qui sont celles du concepteur des systèmes distribués. Elle est donc facilement intégrée dans un système distribué.

## 3.3 La synchronisation par échange de messages

La synchronisation des horloges dans les systèmes répartis par échange de messages sur le réseau a attiré l'attention des chercheurs, et depuis des années un grand nombre de solutions a été proposé. Peut-être les premières références à ce problème sont celles de Wensley *et al* ([Wensley et al., 1978]) et de Hopkins *et al* ([Hopkins et al., 1978]). Les motivations pour la proposition de nouveaux algorithmes ou méthodes d'analyse a été, entre autres, la tolérance aux fautes ([Cristian et al., 1986, Lamport et Melliar-Smith, 1985]) ou des implémentations plus efficaces ([Clegg et Marzullo, 1996, Drummond et Babaoğlu, 1993, Halpern et al., 1983, Olson et Shin, 1994, Srikanth et Toueg, 1987]). Nous trouvons aussi des solutions spécifiques pour une certaine classe de systèmes : CAN (Controller Area Network) ([Gergeleit et Streich, 1994]), FIP (Factory Information Protocol ou Flux Information Processus) ([He et al., 1990, He et Mammeri, 1995]), FMS/MAP (Flexible Manufacturing Systems, Manufacturing Automation Protocol) ([Gora et al., 1988]), UNIX ([Gusella et Zatti, 1989]) ou l'Internet ([Mills, 1991a]). D'autres algorithmes peuvent être aussi trouvés dans [Kopetz et Ochsenreiter, 1987, Lundelius-Welch et Lynch, 1988, Marzullo et Owicki, 1983, Verissimo et Rodrigues, 1992]. Des synthèses sur ce sujet se trouvent dans [He et al., 1994, Mammeri, 1997, Mammeri et He, 1996a, Mammeri et He, 1996b, Simons et al., 1989, Suri et al., 1994] et une liste de la bibliographie sur le temps et les états globaux dans les systèmes répartis (bien que datée de 1993) se trouve dans [Yang et Marsland, 1993]. [Attiya et al., 1996, Dolev et al., 1986, Fischer et al., 1990, Lundelius et Lynch, 1984, Patt-Shamir et Rajsbaum, 1994] présentent des résultats fondamentaux sur la synchronisation d'horloges par échange de messages.

Un des résultats de première importance est celui présenté par Lundelius et Lynch ([Lundelius et Lynch, 1984]). Dans leur article, les auteurs étudient l'exécution d'un algorithme de synchronisation par échange de messages sur un système réparti, avec $n$ sites. Les horloges avancent avec une dérive nulle (le système est du type $D_0$) et il n'y a pas de pannes. Les valeurs des horloges ont, au départ, un écart non borné et il y a une incertitude de $\epsilon$ dans le délai des messages (c.-à-d., le délai d'un message, $\Gamma$, est tel que $\Gamma_{\min} \leq \Gamma \leq \Gamma_{\max}$ et $\Gamma_{\max} - \Gamma_{\min} = \epsilon$). Dans un tel système, la précision d'accord garantie par tout algorithme de synchronisation ne pourra jamais être inférieure à $\gamma = \epsilon(1 - 1/n)$. Nous pouvons interpréter ce résultat qualitativement comme suit : l'horloge d'un site peut avancer (ou retarder) d'une valeur $\Delta \leq \gamma$ sans que les autres sites ne s'en aperçoivent ; la variation de la valeur de l'horloge peut être masquée par la variation du délai des messages.

Ce résultat met en évidence deux caractéristiques des algorithmes de synchronisation. Premièrement, la précision garantie a une borne inférieure non-nulle (sauf pour le cas trivial

où le délai est constant). Deuxièmement, le délai de transmission d'un message doit être borné pour que la précision soit aussi bornée. Une valeur infinie pour $\Gamma_{max}$ entraîne une valeur infinie pour $\epsilon$ et donc pour $\gamma$. Une remarque doit être faite à propos de la limite de la précision d'accord. Le travail de Lundelius et Lynch ne dit pas que les algorithmes de synchronisation sont *incapables* d'atteindre une précision d'accord inférieure à $\gamma$. Elles disent plutôt qu'ils sont incapables de *garantir* une précision inférieure à cette valeur. Quand on prend le résultat de Lundelius et Lynch, on ne doit jamais oublier qu'il s'agit d'une condition qui s'applique dans le pire des cas.

Le fait que la précision d'accord a une borne minimale et que le délai de transmission doit être borné nous laisse avec deux questions :

- « Est-il possible de garantir une précision d'accord avec une valeur inférieure à $\epsilon(1 - 1/n)$? »
- « Est-il possible de faire fonctionner des algorithmes de synchronisation sur des systèmes où le délai n'est pas borné? »

Une des stratégies proposées pour essayer de répondre (affirmativement...) à ces deux questions est celle des algorithmes dits non-déterministes.

## 3.4 Les algorithmes non-déterministes

Le résultat de Lundelius et Lynch a mis en évidence deux limitations des algorithmes de synchronisation dits déterministes : la précision garantie a une borne inférieure non-nulle qui dépend de l'incertitude du délai de communication et ce délai doit être borné pour que l'erreur soit aussi bornée. Les algorithmes de synchronisation non-déterministes (ASND) essayent de proposer des solutions auxquelles ces limites ne s'appliquent pas. Ceci est fait à travers l'utilisation de techniques statistiques ou probabilistiques pour l'obtention du résultat. On obtient donc une meilleure précision ; le prix à payer pour cette précision est une (petite) probabilité que le système ne réussisse pas à se synchroniser avec la précision souhaitée. Cette probabilité d'échec tend vers zéro avec le nombre de messages échangés et elle peut être réduite à une valeur aussi petite qu'on le souhaite avec un nombre de messages suffisamment grand (pourvu qu'on dispose du temps, de la largeur de bande et de la capacité de calcul nécessaires).

### 3.4.1 L'application des ASND aux systèmes temps-réel

Le fait que les ASND présentent des résultats avec une probabilité associée peut causer une certaine réserve en ce qui concerne leur application à des systèmes tels que les STR. Peut-on accepter, dans un système qu'on cherche à développer de façon à garantir un fonctionnement sûr, des algorithmes qui présentent un certaine probabilité d'échec? Pourquoi ne pas utiliser les algorithmes déterministes? Dans ces algorithmes, nous avons toujours la garantie de la présentation du bon résultat.

La garantie présentée par les algorithmes déterministes est basée sur certaines hypothèses, notamment l'existence d'une borne maximale pour le délai. Les hypothèses sous-jacentes aux ASND sont beaucoup plus étendues. Qu'est-ce qui se passe si les hypothèses requises par les algorithmes déterministes ne se vérifient pas? Aucune garantie n'est donnée par l'algorithme dans une telle situation.

Alors, et pour répondre à la question « Les ASND sont-ils adéquats pour les STR? », nous pouvons répondre qu'ils sont adéquats pour tous les systèmes qui ne garantissent pas une borne maximale pour le délai de transmission ou pour les systèmes où cette borne est trop élevée pour la précision souhaitée.

### 3.4.2 Description des algorithmes non-déterministes

La première référence à un algorithme de synchronisation non-déterministe est l'algorithme de Duda *et al* ([Duda et al., 1987]), qui fait l'hypothèse que la valeur de l'horloge de chaque site $p$ peut être décrite comme une fonction linéaire du temps, $H_p(t) = a_p t + b_p$. La relation entre la valeur de l'horloge et le temps physique ne peut pas être déterminée directement, mais l'hypothèse permet de représenter l'horloge d'un site $p$ comme une fonction linéaire de l'horloge d'un autre site $q$: $H_p(t) = aH_q(t) + b$. L'algorithme calcule les paramètres (pente et ordonnée à l'origine) qui décrivent cette relation. Un site envoie un message avec la valeur de son horloge à un autre qui enregistre l'instant d'arrivée du message et son contenu. Le récepteur répond avec un message qui contient la valeur de l'instant d'arrivée du premier message et la valeur de l'horloge locale à l'instant d'envoi de la réponse. Après plusieurs échanges, les paires de valeurs (instant de départ, instant d'arrivée) permettent d'établir le rapport entre les deux horloges.

Une approche différente est prise dans l'algorithme de Cristian ([Cristian, 1989b]). Un site envoie un message de requête au site distant. Celui-ci doit répondre sans délai avec la valeur de son horloge. La stratégie de requête-réponse permet au site initiateur (celui qui fait la requête) d'estimer une borne maximale pour l'erreur en fonction de la durée de l'aller-retour. La lecture de l'horloge distante réussit et la valeur reçue est acceptée si l'erreur maximale garantie (fonction du délai d'aller et retour) est inférieure à l'erreur souhaitée. Dans le cas où la lecture échoue, la requête est répétée, jusqu'au premier succès ou jusqu'à un nombre maximal d'essais.

Dans l'algorithme d'Arvind ([Arvind, 1989]), le site distant envoie une séquence de messages. Chaque message contient la valeur de l'horloge de l'émetteur à l'instant d'émission. Le récepteur enregistre la valeur de son horloge à l'instant d'arrivée. La connaissance du délai moyen pour la transmission d'un message permet d'estimer l'écart entre les deux horloges à partir de la valeur moyenne des instants de départ et d'arrivée, mesurées par les horloges de l'émetteur et du récepteur, respectivement.

L'algorithme de Le-Lann ([Le Lann, 1990]) s'adresse aux systèmes qui ont des moyens de diffuser des messages et aux structures distribuées. Chaque site envoie plusieurs messages sur un réseau à diffusion. Les messages ne contiennent pas la valeur de l'horloge de l'émetteur ; seulement son identification et un numéro de séquence. Dans une première phase, chaque site émet un certain nombre de messages et tous les sites enregistrent les instants d'arrivée (selon leurs horloges). À la fin de la première phase, chaque site $p$ calcule la valeur moyenne pour les instants d'arrivée, $M_p$. Cette valeur, différente pour chacun des sites, correspond, a peu près, au même instant temps physique. Dans la deuxième phase, les sites échangent les valeur calculés localement. Ceci permet d'obtenir une valeur globale pour la moyenne des instants d'arrivée $M$, qui est connue de tous les sites. À ce moment, tous les sites ont une référence commune pour la valeur de l'horloge et pour un instant temps physique, ce qui permet de les synchroniser.

Ces quatre algorithmes définissent un ensemble de stratégies qui sont communes à tous les

algorithmes non déterministes publiés. Comme nous le verrons dans la suite, les algorithmes proposés utilisent un des principes déjà mentionnés ; leur contribution est, en général, leur adaptation à un scénario différent (par exemple, un environnement plus complexe, un degré de connectivité différent ou l'addition d'hypothèses qui permettent d'obtenir une erreur plus petite).

Les deux algorithmes proposés par Olson et Shin ([Olson et Shin, 1991]), l'algorithme des intervalles et l'algorithme des moyennes, reprennent les algorithmes de Cristian et d'Arvind et les adaptent pour un réseau où il n'y a pas de connexion directe entre tous les sites. Les hypothèses concernant le réseau sont : i) l'existence d'un chemin qui part d'un site, traverse tous les autres sites et se termine au site de départ (un cycle ou une boucle hamiltonienne) ; et ii) la connaissance de la valeur minimale (pour l'algorithme des intervalles) ou de la valeur moyenne (pour l'algorithme des moyennes) du délai de transmission entre deux sites voisins sur ce chemin. L'algorithme des intervalles est basé sur l'algorithme de Cristian ; un message est envoyé selon le cycle à tous les sites. Chaque site ajoute au message la valeur de son horloge locale à l'instant de passage du message et le renvoie vers le site suivant dans le cycle. Quand le message complète un tour, le site détermine les intervalles probables pour la valeur de chacune des horloges distantes. La méthode d'estimation de l'horloge distante est identique à celle de Cristian ; la valeur minimale pour le délai d'aller est remplacée par la somme de tous les délais minimaux dans le chemin d'aller, et de même pour le délai de retour. L'estimation retenue est l'intersection de tous les intervalles. Dans le cas de l'algorithme des moyennes, le principe est identique : un message est envoyé à travers une boucle qui passe par tous les sites à synchroniser. Quand le message complète le tour, le site qui reçoit le message estime la valeur de toutes les autres horloges par la méthode décrite par Arvind, où le délai de transmission est estimé par la somme de tous les délais moyens des liaisons traversées.

Rangarjan et Tripathi ([Rangarajan et Tripathi, 1991]) proposent l'adaptation de l'algorithme d'Arvind à un réseau non totalement connecté. Les sites sont connectés de telle façon que chaque site communique directement avec $2k(n^{1/k} - 1)$ sites et est capable d'accéder à tout autre site en traversant un nombre de liaisons égale ou inférieur à $k$, où $k$ est une valeur choisie selon le degré de connectivité (en effet, $k$ est le diamètre du graphe de connectivité). Les sites envoient des messages à tous les sites voisins, contenant la valeur de leurs horloges et les valeurs reçues des autres sites. À partir d'un certain moment, les sites s'arrêtent d'envoyer la valeur de leur horloge et ne font que le relais des messages reçus, pour garantir que tout message arrive à son destinataire. À la fin, la valeur d'une horloge distante est calculée de façon identique à celle de l'algorithme des moyennes d'Olson et Shin.

L'algorithme de Couvet *et al* ([Couvet et al., 1991]) est basé sur des hypothèses identiques à celles de Duda *et al*, notamment en ce qui concerne la représentation de l'horloge comme une fonction linéaire du temps et l'utilisation de la régression pour établir le rapport entre deux horloges. L'amélioration proposée par Couvet *et al* est le calcul de l'ordonnée à l'origine par l'algorithme de Cristian, et non par des techniques de régression.

Alari et Ciuffoletti ([Alari et Ciuffoleti, 1992, Alari et Ciuffoletti, 1997]) proposent une amélioration à l'algorithme de Cristian basée sur des hypothèses plus restrictives. Notamment, les auteurs démontrent que, si une certaine précision est garantie avant l'exécution de l'algorithme, le diamètre de l'intervalle d'incertitude associée à la valeur lue peut être réduit.

L'algorithme de Duda *et al* est repris par Maillet et Tron ([Maillet et Tron, 1995]) qui présentent une implémentation et quelques heuristiques pour l'améliorer.

Cristian et Fetzer ([Cristian et Fetzer, 1994]) proposent une évolution de l'algorithme ori-

ginal de Cristian. La contribution est l'adaptation de l'algorithme à une structure distribuée et le découplement des messages de requête et réponse. Ce découplement permet à un site de répondre, avec un seul message diffusé sur le réseau à toutes les requêtes reçues antérieurement.

La figure 3.2 présente une « généalogie » des algorithmes de synchronisation non-déterministes recensés dans la littérature.

## 3.5 Une remarque sur le vocabulaire utilisé

La dénomination de « non-déterministe » pour les algorithmes présentés est retenue dans cette thèse pour une question de convenance, car elle est devenue d'usage courant. Mais elle peut être aussi un peu inadaptée.

Le concept de « déterminisme » est assez bien établi dans le domaine de la mathématique. Selon Sedgewick ([Sedgewick, 1988, page 364]), un algorithme est déterministe si, pour n'importe quelle action que l'algorithme est en train d'exécuter, il y a une seule autre action qu'il peut prendre dans la suite.

Présenté avec les mêmes valeurs, un algorithme déterministe produit toujours les mêmes résultats et prend toujours les mêmes décisions. Un algorithme non-déterministe est défini comme celui qui n'est pas déterministe. Un tel algorithme peut exécuter des actions et prendre des décisions différentes, présenté deux fois dans la même situation et avec les mêmes données. Ces algorithmes sont normalement implémentés par recours à des variables aléatoires : les décisions à prendre sont définies en fonction : a) des données ; b) d'une valeur aléatoire générée au niveau interne. Cette caractéristique est primordiale pour distinguer les algorithmes non-déterministes : la différence des actions internes entre deux exécutions réalisées sous les mêmes conditions. Comme exemple d'application, dans [Ben-Or, 1983, Rabin, 1983, Perry, 1985] les algorithmes non-déterministes sont utilisés dans le problème du consensus distribué.

En regardant les algorithmes proposés dans la littérature comme « non-déterministes », nous trouvons qu'ils sont tous des algorithmes déterministes, selon la définition présentée précédemment. Ses actions sont bien identifiées et déterminées ; à aucun moment l'algorithme ne fait un choix (non-déterministe) parmi un ensemble d'actions possibles. Mais, en même temps, ces algorithmes présentent des caractéristiques qui permettent de les distinguer, même s'ils ne sont pas inclus dans la définition présentée de non-déterminisme.

La caractéristique distinctive de ces algorithmes peut être la suivante : bien que leur comportement soit déterministe, ils sont conçus pour s'exécuter en des situations où les données changent d'une exécution à la suivante. Ces données sont produites, par exemple, par un processus stochastique. L'indéterminisme dans l'exécution de l'algorithme provient, non de l'algorithme, mais plutôt du phénomène observé ou mesuré. Alors, et dans le but de trouver les caractéristiques qui nous permettent de classer un algorithme de synchronisation comme étant non-déterministe (au sens qu'on trouve dans la littérature de la synchronisation des horloges), nous proposons la définition suivante :

**Définition 3.1 (Algorithme de synchronisation non-déterministe)** *Les algorithmes de synchronisation non-déterministes sont ceux qui exploitent le caractère non-déterministe des phénomènes sous-jacents, notamment le fait que la durée de transmission d'un message n'est pas constante mais qu'elle a une valeur aléatoire.*

## 3.6 Conclusion

Ce chapitre a examiné les solutions disponibles pour le concepteur d'un système distribué pour synchroniser les horloges de son système, de façon à garantir les propriétés de précision ou d'exactitude. Les solutions proposées sont très diverses et incluent les solutions basées sur la réception d'un signal de référence, sur le matériel et sur l'échange de messages. Les dernières, synchronisation par l'échange de messages, présentent l'avantage de ne pas demander d'équipement additionnel et ce sont des solutions qui apportent une très grande liberté de choix au concepteur du système distribué, puisqu'il peut maîtriser les concepts et la technologie utilisés. Un des inconvénients est la limitation de la précision que ces méthodes peuvent atteindre. Une des réponses à ce problème est celle des algorithmes non-déterministes, dont nous avons présenté l'historique et le principe de fonctionnement des plus significatifs. Dans la suite, nous allons analyser en détail ces algorithmes.

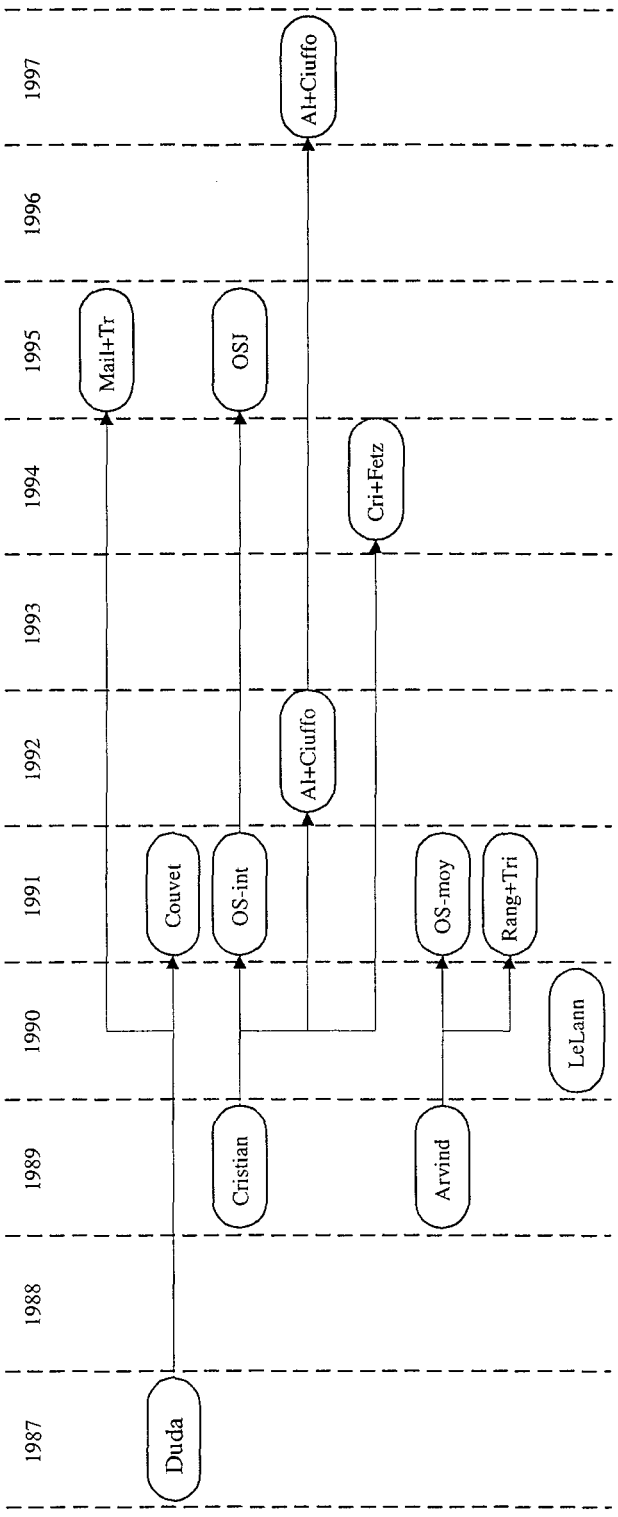



FIG. 3.2: *Généalogie des algorithmes non-déterministes*

# Chapitre 4

# Analyse des algorithmes non-déterministes

## 4.1 Introduction

De par leur nature, les algorithmes de synchronisation non-déterministes (ASND) offrent une très petite précision de synchronisation en échange d'une probabilité de réussite inférieure à 1. Dans le cas général, cette probabilité peut être aussi proche de 1 qu'on le souhaite ; il faut envoyer un nombre suffisamment grand de messages (en supposant que le système est capable de transmettre et traiter tous les messages de synchronisation). Il y a donc trois paramètres qui sont de première importance et dont la relation doit être maîtrisée par le concepteur du système distribué : la précision de synchronisation, la probabilité de synchronisation et le nombre de messages échangés.

Ce chapitre essaie de présenter un cadre analytique qui décrit le rapport entre ces trois paramètres. Le but de cette analyse est de trouver une expression que le concepteur d'un système distribué puisse utiliser pour calculer le nombre de messages requis pour un certain algorithme, de façon à fournir la synchronisation avec la précision et la probabilité spécifiées. Cette valeur doit être présentée en fonction d'une série de paramètres qui soient facilement calculables.

Pour cela, les ASND sont classifiés en grands groupes. Cette classification permet de trouver les caractéristiques qui sont communes à chacun de ces groupes, notamment en identifiant une stratégie spécifique. L'analyse de l'interdépendance entre l'erreur et le nombre de messages échangés est donc réduite aux stratégies identifiées.

Dans la section 4.2, nous présentons la classification des algorithmes et nous détaillons les principes de fonctionnement de chaque classe. La section 4.3 traite le problème de calculer le rapport entre le nombre de messages échangés dans l'exécution de l'algorithme et la probabilité que l'erreur dans une lecture soit supérieure à un seuil spécifié. Dans cette section, la méthodologie pour calculer cette erreur pour chacune des classes est detaillée. Finalement, les sections 4.4 et 4.5 traitent de l'extension des résultats des sections précédentes à l'ensemble d'un système distribué et présentent quelques indications qui permettent de comparer la performance des algorithmes non-déterminites de synchronisation des horloges. Une version antérieure de ce travail a été présenté dans [Fonseca et Mammeri, 1996].

## 4.2 Classification des algorithmes

Dans le chapitre précédent, nous avons présenté les principaux algorithmes non-déterministes de synchronisation d'horloges. Cette première analyse nous a permis de déterminer quelques algorithmes comme étant de base dans le domaine. Dans cette section, nous essayerons d'identifier les caractéristiques communes à ces algorithmes, afin de pouvoir les comparer.

En analysant les stratégies adoptées par chaque algorithme, nous distinguons trois types de stratégies :

**Régression.** La première stratégie est celle de Duda *et al.* Elle est basée sur la représentation d'une horloge comme une fonction linéaire d'une autre horloge, et l'utilisation de techniques de régression linéaire pour déterminer les paramètres de cette fonction : la pente et l'ordonnée à l'origine.

**Intervalle.** Une autre stratégie est celle qui consiste à déterminer des intervalles où se trouve la valeur de l'horloge distante. C'est la stratégie adoptée dans les algorithmes de Cristian et d'Olson et Shin.

**Moyenne.** La troisième stratégie consiste à enregistrer une suite d'observations (de temps de départ, d'arrivée) des messages et de déterminer une approximation pour l'horloge distante basée sur la moyenne des valeurs de ces suites. Cette stratégie est utilisé par Arvind, Rangarajan et Tripathi, Olson et Shin et Le Lann.

### Algorithmes utilisant la régression

Ces algorithmes sont basés sur la représentation de chaque horloge comme une fonction linéaire du temps. Ceci permet de représenter la valeur d'une horloge comme une fonction linéaire de la valeur d'une autre. Pour déterminer le rapport entre les deux horloges, les algorithmes utilisent des techniques de régression linéaire. Pour représenter $H_p$ comme une fonction linéaire de $H_q$, $H_p = aH_q + b$, $p$ et $q$ échangent une série de messages. Pour chaque message $(i)$, $p$ et $q$ enregistrent la valeur de leur horloge locale à l'instant de départ ou d'arrivée, selon qu'ils soient émetteur ou récepteur (Fig. 4.1). Dans la figure, $T_p^{(i)} = H_p(t_m^{(i)})$ où les $t_m^{(i)}$ sont les instants d'émission ou de réception du message $i$, selon la direction de ce message. Chaque message transporte la valeur de l'horloge de l'émetteur à l'instant d'émission et l'instant de réception du message précédent. Chaque site obtient un ensemble de paires de valeurs $(T_q^{(i)}, T_p^{(i)})$. Dans un plan $T_q \times T_p$, ces points définissent deux droites parallèles ; chacune de ces droites correspond à un des sens pour les messages (aller ou retour) (Fig. 4.2). La droite calculée en fonction de tous les points de l'ensemble donne la représentation de $H_q$ comme une fonction linéaire de $H_p$. Il n'est pas nécessaire de connaître la valeur du délai d'aller ou retour. Pour que l'estimation ne soit pas biaisée, l'espérance mathématique des délais d'aller et retour doit être identique et le nombre de points utilisés doit être le même pour les deux sens. Pour un traitement plus approfondi du sujet de l'estimation des paramètres de la regression, voir par exemple [Kempthorne, 1952, pg. 54].

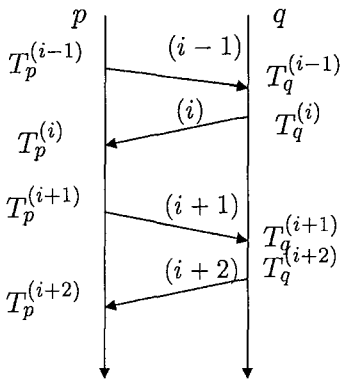



FIG. 4.1: *Algorithmes de régression : principe de datation des événements et d'echange de messages.*

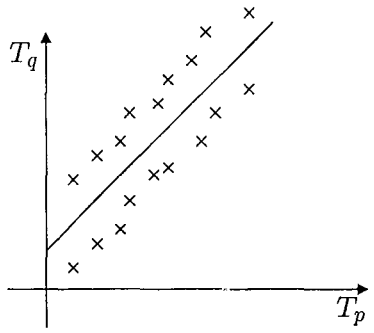



FIG. 4.2: *Représentation graphique des points* $(T_p^{(i)}, T_q^{(i)})$

### Algorithmes utilisant les intervalles

Les algorithmes de ce type déterminent un intervalle de valeurs où se trouve la valeur de l'horloge distante (fig. 4.3). Les bornes de cet intervalle sont déterminées par le délai d'aller et retour d'un message. La connaissance d'une borne minimale pour le délai de transmission, $\Gamma_{\min}$, permet de réduire la distance entre les bornes de l'intervalle, et donc l'erreur dans la détermination de la valeur de l'horloge distante. La valeur de l'horloge distante est placée au centre de l'intervalle de probabilité. La raison est la suivante: si la valeur de l'horloge distante est comprise dans l'intervalle calculé, l'erreur est inférieure ou égale à la distance entre l'estimation et l'extrémité de cet intervalle la plus éloignée. Cette valeur est minimisée en plaçant l'estimation au centre de l'intervalle.

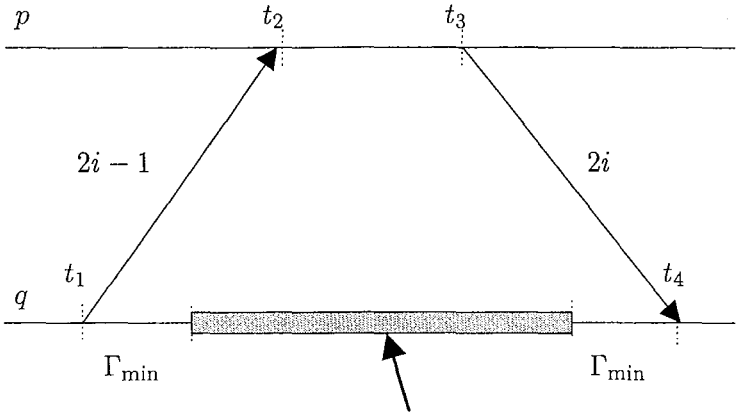



FIG. 4.3: *Algorithmes des intervalles: calcul de l'intervalle.*

Ces algorithmes présentent la caractéristique d'être capables de garantir un seuil maximal pour l'erreur dans la lecture d'une horloge distante, car la valeur de l'horloge distante est forcément dans l'intervalle calculé. Cette capacité peut être employée de différentes façons. Cristian prend le premier intervalle avec une valeur de l'erreur égale ou inférieure au seuil souhaité, Olson et Shin prennent l'intersection des intervalles. L'aspect commun est le fait que l'erreur de l'estimation retenue est garantie avec une probabilité égale à 1 et la valeur de l'horloge distante est dans l'intervalle calculé.

### Algorithmes utilisant la moyenne

Ces algorithmes sont basés sur l'obtention d'une suite de valeurs (normalement, des instants d'émission ou de réception de messages). La moyenne des valeurs de cette suite fournit une estimation qui est utilisée pour le calcul de l'horloge distante ou pour la détermination d'un instant de temps physique. Ces algorithmes utilisent le théorème de la limite centrale. Le concept sous-jacent est que la moyenne d'une suite d'observations d'une valeur est une

estimation fiable. L'algorithme de Arvind est l'archétype des algorithmes de moyenne. Normalement, ces algorithmes nécessitent la connaissance du délai moyen de transmission d'un message.

## 4.3 L'erreur dans les méthodes de lecture

Cette section traite le problème de calcul du rapport entre le nombre de messages échangés dans l'exécution de l'algorithme et la probabilité d'erreur pour chacune des stratégies présentées. Nous introduisons le concept d'une *étape* d'un algorithme de synchronisation non-déterministe. L'algorithme est exécuté en étapes, et une étape consiste à envoyer $r$ messages. À la fin de l'étape, une valeur est calculée. $\epsilon(r)$ est l'erreur de la valeur calculée après l'envoi de $r$ messages. Nous allons calculer la probabilité $\mathbf{P}\left[|\epsilon(r)| \leq \epsilon^{\max}\right]$ en fonction de la caractérisation du délai comme une variable aléatoire.

Le délai du message $i$ est représenté par $\Gamma^{(i)}$. Pour une suite de messages, nous faisons l'hypothèse que les $\Gamma^{(i)}$ sont des variables aléatoires (v.a.) indépendantes et identiquement distribuées (i.i.d.), représentées par la v.a. $\Gamma$. Ceci signifie, entre autres, que l'intervalle entre l'envoi de deux messages est suffisamment long pour que les causes d'un possible retard ou avance du message $i$ n'affectent pas la transmission du message $i+1$. Par exemple, si le premier message a été retardé à cause d'un encombrement du réseau, le deuxième ne doit pas être retardé par la même cause. Sinon, les délais $\Gamma^{(i)}$ et $\Gamma^{(i+1)}$ auraient une corrélation positive et ils ne seraient plus indépendants. $F_\Gamma$ est la fonction de distribution de $\Gamma$ et $f_\Gamma$ la fonction densité de probabilité (fdp). $\mathrm{E}[\Gamma] = \hat{\Gamma}$ est l'espérance de $\Gamma$ et $\mathrm{Var}[\Gamma] = \sigma_\Gamma^2$ la variance.

### 4.3.1 Analyse de l'envoi d'un message

L'envoi d'un message d'un site $p$ à un autre site $q$ a été décrit, en ce qui concerne sa durée, par l'identification de deux instants de temps physique, $t_e$ et $t_r$, les instants d'envoi et de réception, respectivement. En regardant ce processus en plus de détail, nous trouvons des aspects qu'il faut spécifier davantage: à quel instant doit-on associer la valeur $t_r$? L'instant où le message arrive au site $q$? L'instant où $q$ reçoit une indication de l'arrivée du message et commence son traitement? L'instant où l'horloge $H_q$ est lue pour estampiller le message qui arrive?

Le processus d'envoi d'un message $i$ est décrit par la figure 4.4. À l'instant $t_1$, l'horloge du site $p$ est lue. En fonction de l'implantation spécifique, ceci peut impliquer la lecture d'un dispositif périphérique et le transfert d'une valeur vers la mémoire du processeur. Cette valeur doit être incorporée dans le message, ce qui signifie créer toute la structure du message (en-tête, ... ), éventuellement en ajoutant d'autres données (identification de l'émetteur, numéro de série, ... ). Ce message est placé dans la file pour transmission ($t = t_2$). À un certain moment, le message est envoyé à travers le réseau ($t = t_3$) et arrive à la destination ($t = t_4$). L'arrivée du message au site $q$ est signalé au processeur local, qui traite le message. Ce traitement inclut l'estampillage du message avec la valeur de l'horloge locale ($t = t_5$).

Nous cherchons à trouver les valeurs qui seront utilisées pour calculer une estimation pour la différence des valeurs des deux horloges, $H_p$ et $H_q$. Ceci sera fait forcément avec la perception que chacun de ces sites aura de l'envoi du message. Pour $p$, le moment où le message est envoyé est $t = t_1$ ; la valeur de l'horloge associée à l'instant d'envoi du message

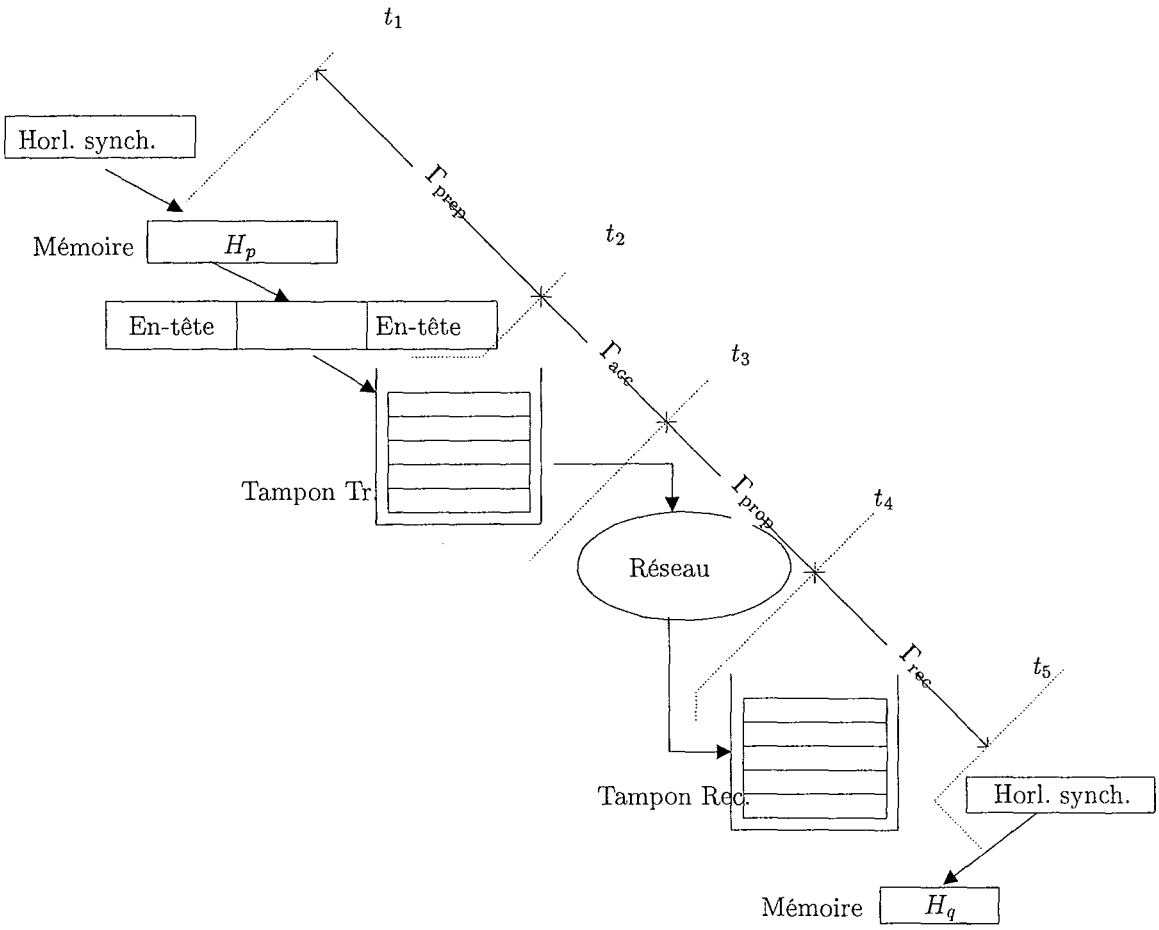



FIG. 4.4: *Envoi d'un message*

est $H_p(t_1)$. Pour $q$, tout se passe comme si le message était arrivé à l'instant $t = t_5$, car la valeur de son horloge associée à l'arrivée du message est $H_q(t_5)$. Tous les autres instants ($t_2$, $t_3$ et $t_4$) sont invisibles ou pas mesurables par les processeurs de $p$ et $q$. Alors, l'instant d'envoi est $t_e = t_1$ et l'instant de réception est $t_r = t_5$.

Le délai de transmission d'un message $i$, $\Gamma^{(i)}$, a été décomposé en 4 composants: $\Gamma^{(i)} = \Gamma_{\text{prep}} + \Gamma_{\text{acc}} + \Gamma_{\text{prop}} + \Gamma_{\text{rec}}$. Ce modèle a été proposé par Kopetz (par exemple, dans [Kopetz et Ochsenreiter, 1987]). $\Gamma_{\text{prep}} = t_2 - t_1$ est le délai de préparation du message, de la lecture de l'horloge locale au placement du message dans la file de transmission. Le message attend dans cette file pour un temps $\Gamma_{\text{acc}} = t_3 - t_2$, le temps pour accéder au canal de transmission. $\Gamma_{\text{prop}} = t_4 - t_3$ correspond au délai de propagation du message, de l'instant où le message quitte le tampon de l'émetteur à l'instant où il arrive à la file de réception. Finalement, $\Gamma_{\text{rec}} = t_5 - t_4$ correspond au délai chez le récepteur.

L'introduction de ce modèle a deux objectifs. Premièrement, il sert à attirer l'attention sur le fait que l'envoi d'un message, normalement décrit comme une durée, à laquelle on associe un instant de début et un instant de fin, est en fait plus compliqué que ne le laisse paraître cette analyse, et que notamment tout un nombre de phénomènes contribuent à l'incertitude de la valeur de ce délai. Deuxièmement, cette décomposition en un ensemble de phénomènes plus simples rend plus facile la représentation du délai comme une variable aléatoire. En décrivant le délai comme une somme de quatre variables aléatoires différentes, ceci pourra faciliter la comparaison de différentes techniques, ou à trouver plus facilement la cause d'une différence de performance entre deux systèmes différents.

### 4.3.2 Régression

Dans cette stratégie, le site $q$ utilise des techniques de régression pour établir la relation linéaire entre $H_q$ et $H_p$. $p$ et $q$ échangent un total de $r$ messages ; un nombre identique de messages ($r/2$) est envoyé de $q$ à $p$ et de $p$ à $q$. Un couple de valeurs $(T_q^{(i)}, T_p^{(i)})$ est associé à chaque message $i$, où $T_q^{(i)}$ (resp. $T_p^{(i)}$) est la valeur de l'horloge $H_q$ (resp. $H_p$) à l'instant d'envoi ou de réception du message, selon que $q$ (resp. $p$) est l'émetteur ou le récepteur. La représentation des points $(T_q^{(i)}, T_p^{(i)})$ dans un graphique définit une droite et, donc, la relation entre les deux horloges. Pour simplifier la notation, les $T_q^{(i)}$ seront représentés par $x^{(i)}$ et les $T_p^{(i)}$ par $y^{(i)}$. Nous avons donc un ensemble de points $\{(x^{(i)}, y^{(i)})\}$ et nous cherchons à trouver la valeur des coefficients qui expriment la dépendance entre les $x$ et les $y$ : $y = ax+b$. L'ensemble des points $\{(x^{(i)}, y^{(i)})\}$ se divise en deux sous-ensembles, qui correspondent aux messages aller et aux messages retour. Nous définissons le sens «aller» comme de $q$ à $p$. Pour les messages aller, $x^{(i)}$ est l'instant de départ et $y^{(i)}$ l'instant d'arrivée. La valeur $y^{(i)}$ correspond à la valeur de l'horloge de $p$ à l'instant d'envoi du message $(ax^{(i)} + b)$ plus le délai du message, $\Gamma^{(i)}$ : $y^{(i)} = ax^{(i)} + b + \Gamma^{(i)}$. Pour les messages de retour, $y^{(i)} + \Gamma^{(i)} = ax^{(i)} + b$. Considérant les messages correspondant au sens aller comme ayant un numéro pair, et ceux de retour avec un numéro impair, nous avons donc les ordonnées des points définies par

$$y^{(i)} = ax^{(i)} + b - (-1)^i \Gamma^{(i)} \tag{4.1}$$

Pour que les méthodes de régression puissent s'appliquer, il faut que l'erreur dans la position des points $\{(x^{(i)}, y^{(i)})\}$ soit une v.a. avec espérance nulle. Comme cette erreur correspond à l'éloignement des points de la droite $y = ax + b$, cette condition revient à imposer que le délai soit symétrique.

L'erreur $\epsilon(r)$ de l'estimation $y_0$ de l'ordonnée du point sur la droite avec abscisse $x_0$ est une v.a. normale, de moyenne nulle, définie par ([Ross, 1987]):

$$\epsilon(r) \sim \mathcal{N}\left(0, \sigma^2\left[\frac{1}{r} + \frac{(x_0 - \bar{x})^2}{\sum_{i=1}^{r}(x^{(i)} - \bar{x})^2}\right]\right) \tag{4.2}$$

La variance $\sigma^2$ n'est pas connue, mais elle peut être estimée par :

$$\sigma^2 = \frac{1}{r-2}\sum_i (y^{(i)} - ax^{(i)} - b)^2 \tag{4.3}$$

En conséquence de l'utilisation de cette approximation, l'erreur devient une v.a. avec distribution $t$ avec $r-2$ degrés de liberté:

$$\frac{\epsilon(r)}{\frac{1}{r-2}\sum_i (y^{(i)} - ax^{(i)} - b)^2 \left[\frac{1}{r} + \frac{(x_0-\bar{x})^2}{\sum_{i=1}^{r}(x^{(i)}-\bar{x})^2}\right]} \sim t_{r-2} \tag{4.4}$$

La variance de l'erreur dépend de deux termes de valeur encore inconnue : le terme $\sum_i (y^{(i)} - ax^{(i)} - b)^2$ (la somme des carrés des résidus) et le terme $(x_0 - \bar{x})^2 / \sum (x^{(i)} - \bar{x})^2$.

Pour le premier terme, nous avons (4.1) :

$$\begin{aligned}\sum_i (y^{(i)} - ax^{(i)} - b)^2 &= \sum_i (ax^{(i)} + b - (-1)^i \Gamma^{(i)} - ax^{(i)} - b)^2 \\ &= \sum_i \left(-(-1)^i \Gamma^{(i)}\right)^2 \\ &= rM_2 \end{aligned} \tag{4.5}$$

où $M_2 = \mathrm{E}\left[\Gamma^2\right] = \hat{\Gamma}^2 + \sigma_\Gamma^2$.

Le calcul du terme $(x_0 - \bar{x})^2 / \sum (x^{(i)} - \bar{x})^2$ demande que quelques hypothèses soient faites. La méthode de régression offre une très grande liberté de choix (du nombre de messages $r$, de l'intervalle entre deux messages consécutifs, de la distribution des instants d'envoi des messages durant l'étape, ... ). Tous ces choix influencent le résultat, ce qui rend difficile de trouver des résultats valides pour toutes les situations. Il faut donc nous restreindre à des cas particuliers, qu'on espère être représentatifs de beaucoup d'applications. Nos hypothèses sont les suivantes:

- les messages sont envoyés durant un intervalle $[x_a, x_b]$ ;
- les instants d'envoi des messages sont identiquement distribués le long de cet intervalle ;
- l'estimation de la valeur de l'horloge distante est faite à la fin de l'envoi des messages, c-à-d, $x_0 = x_b$.

Ces hypothéses correspondent à un système où la synchronisation est faite par des vagues. L'exécution de l'algorithme, qui correspond à la durée de synchronisation, se divise en deux phases. Dans la première, les sites envoient des messages et, dans la deuxième, ils calculent la nouvelle valeur de l'horloge. La première phase se déroule pendant l'intervalle $[x_a, x_b]$. Normalement, il y a un délai constant entre l'envoi de deux messages consécutifs. Les instants

d'envoi des messages sont équi-distants ; la distribution des instants est uniforme dans l'intervalle $[x_a, x_b]$. Après l'envoi des messages, les valeur des horloges distantes sont calculées. En faisant $x_0 = x_b$, nous faisons l'hypothèse que le temps nécessaire pour les calculs est négligeable par rapport à $x_b - x_a$, la durée de la phase d'envoi des messages.

Avec ces hypothèses, nous avons donc :

$$\begin{aligned} \bar{x} &= \frac{x_a + x_b}{2} \\ x_0 - \bar{x} &= \frac{x_b - x_a}{2} \\ \sum_i (x^{(i)} - \bar{x})^2 &= (r-1)\frac{(x_b - x_a)^2}{12} \end{aligned}$$

et

$$\begin{aligned} \frac{(x_0 - \bar{x})^2}{\sum_i (x^{(i)} - \bar{x})^2} &= \frac{\left(\frac{x_b - x_a}{2}\right)^2}{(r-1)\frac{(x_b - x_a)^2}{12}} \\ &= \frac{3}{r-1} \end{aligned} \tag{4.6}$$

En utilisant (4.5) et (4.6), l'expression pour la variance de l'erreur dans (4.4) peut être simplifiée :

$$\begin{aligned} \mathrm{Var}\left[\epsilon(r)\right] &= \left[\frac{1}{r} + \frac{3}{r-1}\right]\frac{r}{r-2}M_2 \\ &= \frac{4r-1}{(r-1)(r-2)}M_2 \\ &\approx \frac{4M_2}{r-2}, \qquad r \gg 1 \end{aligned} \tag{4.7}$$

Remplaçant la valeur obtenue dans (4.7) dans (4.4) :

$$\frac{\epsilon(r)}{\sqrt{\frac{4M_2}{r-2}}} \sim t_{r-2}$$

$$\sqrt{\frac{r-2}{M_2}}\frac{\epsilon(r)}{2} \sim t_{r-2} \tag{4.8}$$

La probabilité que l'erreur soit inférieure à $\epsilon_{\max}$ est donc :

$$\begin{aligned} \mathbf{P}\left[|\epsilon(r)| \leq \epsilon_{\max}\right] &= \mathbf{P}\left[-\epsilon_{\max} \leq \epsilon(r) \leq \epsilon_{\max}\right] \\ &= \mathbf{P}\left[-\sqrt{\frac{r-2}{M_2}}\frac{\epsilon_{\max}}{2} \leq \epsilon(r) \leq \sqrt{\frac{r-2}{M_2}}\frac{\epsilon_{\max}}{2}\right] \\ &= T_{r-2}\left(\sqrt{\frac{r-2}{M_2}}\frac{\epsilon_{\max}}{2}\right) - T_{r-2}\left(-\sqrt{\frac{r-2}{M_2}}\frac{\epsilon_{\max}}{2}\right) \end{aligned} \tag{4.9}$$

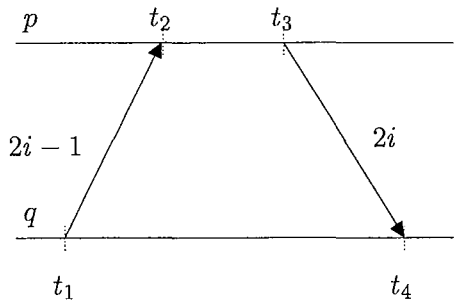



FIG. 4.5: *Requête de l'horloge distante.*

où $T_{r-2}$ représente la fonction de distribution d'une v.a. $t$ avec $r-2$ degrés de liberté. Pour une valeur de $r$ suffisamment grande (typiquement, $r \geq 30$), l'erreur peut être décrite par une v.a. normale et :

$$\mathbf{P}\left[|\epsilon(r)| \leq \epsilon_{\max}\right] = \Phi\left(\sqrt{\frac{r-2}{M_2}}\frac{\epsilon_{\max}}{2}\right) - \Phi\left(-\sqrt{\frac{r-2}{M_2}}\frac{\epsilon_{\max}}{2}\right) \tag{4.10}$$

où $\Phi(x)$ est la fonction de distribution d'une v.a. normale de moyenne nulle et variance unitaire.

### 4.3.3 Intervalle

La stratégie des intervalles est basée sur la requête de la valeur de l'horloge distante. Le processus d'échange de messages est décrit par la figure 4.5 : le site $q$ envoie une requête au site $p$ à l'instant $t = t_1$ et reçoit une réponse à l'instant $t = t_4$. L'objectif est de calculer $H_p^{(q)}(t_4)$, l'estimation par le site $q$ de $H_p(t_4)$, la valeur de l'horloge distante à l'instant de réception du message de réponse ($t = t_4$). Dans cette stratégie nous trouvons deux variantes pour calculer l'intervalle des valeur possibles à travers l'envoi de $r$ requêtes :

- la répétition des requêtes jusqu'à ce que l'erreur garantie pour la paire requête-réponse soit inférieure à un seuil spécifié au préalable ou jusqu'à ce que $r$ messages soient envoyés ;

- la répétition des requêtes avec intersection de tous les intervalles associés aux couples requête-réponse.

Nous traiterons d'abord le cas d'un seul couple requête-réponse, comment estimer la valeur de l'horloge distante et quelle est l'erreur maximale associée. Ensuite, nous calculerons la probabilité que l'erreur après $r$ messages soit inférieure à un seuil $\epsilon^{\max}$ pour chacune des variantes présentées.

**Analyse d'un couple requête-réponse**

Définissons les quantités $\Delta H_q$, $\Delta H_p$, $\Delta_p$ et $\Delta_q$ comme suit:

$$\begin{aligned} \Delta H_q &= H_q(t_4) - H_q(t_1) \\ \Delta_q &= t_4 - t_1 \\ \Delta H_p &= H_p(t_3) - H_p(t_2) \\ \Delta_p &= t_3 - t_2 \end{aligned}$$

Le système a la propriété $D_\rho$ avec $\rho \ll 1$. Nous avons donc:

$$(1-\rho)\Delta H_q \leq \Delta_q \leq (1+\rho)\Delta H_q \tag{4.11}$$
$$(1-\rho)\Delta H_p \leq \Delta_p \leq (1+\rho)\Delta H_p \tag{4.12}$$
$$(1-\rho)\Delta_q \leq \Delta H_q \leq (1+\rho)\Delta_q \tag{4.13}$$
$$(1-\rho)\Delta_p \leq \Delta H_p \leq (1+\rho)\Delta_p \tag{4.14}$$

La valeur $H_p(t_4)$ dépend de $H_p(t_3)$ et du délai $t_4 - t_3 = \Gamma^{(2i)}$. Le site $q$ connaît $H_p(t_3)$ mais pas $\Gamma^{(2i)}$. Il peut estimer un intervalle $I^{(i)}$où se trouve $H_p(t_4)$ par:

$$I^{(i)} = \left[H_p(t_3) + (1-\rho)\mathrm{Min}\,\Gamma^{(2i)}, H_p(t_3) + (1+\rho)\mathrm{Max}\,\Gamma^{(2i)}\right] \tag{4.15}$$

où $\mathrm{Min}\,\Gamma^{(2i)}$ (resp. $\mathrm{Max}\,\Gamma^{(2i)}$) représente la plus petite valeur (resp. la plus grande valeur) possible pour $\Gamma^{(2i)}$. En outre, $q$ connaît des bornes pour les valeurs possibles pour $\Gamma^{(2i)}$, (4.11) et (4.12):

$$\begin{aligned} \Gamma_{\min} \leq \Gamma^{(2i)} &\leq \mathrm{Max}\Delta_q - \mathrm{Min}\Delta_p - \Gamma_{\min} \\ \Gamma_{\min} \leq \Gamma^{(2i)} &\leq (1+\rho)\Delta H_q - (1-\rho)\Delta H_p - \Gamma_{\min} \end{aligned} \tag{4.16}$$

De (4.15) et (4.16) et en négligeant les termes en $\rho^2$ par rapport à 1, le site $q$ peut calculer $I^{(i)}$ par:

$$I^{(i)} = [H_p(t_3) + (1-\rho)\Gamma_{\min}, H_p(t_3) + \Delta H_q - \Delta H_p + 2\rho\Delta H_q - (1+\rho)\Gamma_{\min}] \tag{4.17}$$

Pour minimiser l'erreur, $H_p^{(q)}(t_4)$ sera placé au centre de l'intervalle des valeurs possibles, pour garantir une borne maximale de l'erreur ($BME$) égale à la demi-largeur de cet intervalle:

$$H_p^{(q)}(t_4) = H_p(t_3) + \frac{\Delta H_q - \Delta H_p}{2} + \rho\left(\Delta H_q - \Gamma_{\min}\right) \tag{4.18}$$

$$BME = \frac{\Delta H_q - \Delta H_p}{2} + \rho\Delta H_q - \Gamma_{\min} \tag{4.19}$$

L'équation 4.19 définit la borne maximale pour l'erreur associée à un seul essai. Les deux variantes de cette stratégie diffèrent dans la façon dont la BME est calculée pour une suite d'essais.

### Essai jusqu'au premier succès

Cette stratégie est utilisée, par exemple, dans l'algorithme de Cristian ([Cristian, 1989b]). Dans le $i$-ème essai, le site qui fait la requête envoie le message numéro $2i-1$ et reçoit le message numéro $2i$, avec les délais $\Gamma^{(2i-1)}$ et $\Gamma^{(2i)}$, respectivement. Le site $q$ réussit à lire l'horloge du site $p$ si au moins un des essais a une erreur associée inférieure à $\epsilon^{max}$. Dans cette section, nous allons calculer la borne maximale de l'erreur, BME, associée à une suite de $r$ messages. Cette borne est la plus petite des bornes associées à chacun des essais.

Pour calculer la probabilité de réussite après $r$ essais il faut introduire l'hypothèse de l'indépendance des essais :

**Hypothèse 4.1 (Indépendance des essais)** *Chaque essai de lecture de l'horloge distante est un événement aléatoire, dont le résultat est indépendant du résultat des tous les autres essais.*

Il faut calculer la valeur *BME* dans eq. (4.19) en fonction des délais des messages. Remarquons que:

$$\Delta_q = \Gamma^{(2i-1)} + \Delta_p + \Gamma^{(2i)} \tag{4.20}$$

De (4.20) avec (4.13) et (4.14) on a :

$$\begin{aligned} \Delta H_q - \Delta H_p &\leq \Delta_q - \Delta_p + \rho(\Delta_q + \Delta_p) \\ &\leq \Gamma^{(2i-1)} + \Gamma^{(2i)} + \rho\left(\Gamma^{(2i-1)} + \Gamma^{(2i)} + 2\Delta_p\right) \end{aligned} \tag{4.21}$$

Il est maintenant possible de calculer la borne maximale pour l'erreur après une suite de $r$ essais en fonction des délais des messages :

$$BME = \min_i \left\{ \frac{\Gamma^{(2i-1)} + \Gamma^{(2i)}}{2} - \Gamma_{\min} + \rho\left[\frac{3}{2}(\Gamma^{(2i-1)} + \Gamma^{(2i)}) + 2\Delta_p\right] \right\} \tag{4.22}$$

Cette expression est difficile à traiter en termes de la distribution des délais à cause des termes en $\rho$. Pour rendre ce traitement possible, nous faisons l'hypothèse que ces termes peuvent être négligés.

**Hypothèse 4.2** *Les termes en $\rho$ de l'équation (4.22) peuvent être négligés.*

Avant de continuer, il faut établir les conditions qui nous permettent de considérer l'hypothèse 4.2 comme valable. Cette hypothèse est équivalente à dire que :

$$(\Gamma^{(2i-1)} - \Gamma_{\min}) + (\Gamma^{(2i)} - \Gamma_{\min}) \gg 3\rho(\Gamma^{(2i-1)} + \Gamma^{(2i)}) + 4\rho\Delta_p \tag{4.23}$$

Pour que l'équation précédente soit valide, il faut que :

$$\frac{(\Gamma^{(2i-1)} - \Gamma_{\min}) + (\Gamma^{(2i)} - \Gamma_{\min})}{\Gamma^{(2i-1)} + \Gamma^{(2i)}} \gg 3\rho \tag{4.24}$$

$$\frac{(\Gamma^{(2i-1)} - \Gamma_{\min}) + (\Gamma^{(2i)} - \Gamma_{\min})}{\Delta p} \gg 4\rho \tag{4.25}$$

Le terme $(\Gamma^{(2i-1)} - \Gamma_{\min}) + (\Gamma^{(2i)} - \Gamma_{\min})$ représente la partie variable des délais. Alors, l'hypothèse que les termes en $\rho$ peuvent être négligés est valable si les deux conditions suivantes sont vraies:

$$\frac{\text{Partie var. du délai}}{\text{Délai}} \gg \rho \tag{4.26}$$

$$\frac{\text{Partie var. du délai}}{\text{Atteinte au site } p} \gg \rho \tag{4.27}$$

L'expression pour la borne maximale de l'erreur est réduite à:

$$BME = \frac{\min_i \left\{\Gamma^{(2i-1)} + \Gamma^{(2i)}\right\}}{2} - \Gamma_{\min} \tag{4.28}$$

et la probabilité que l'erreur après $r$ essais soit inférieure ou égale à $\epsilon^{\max}$ est la probabilité que BME soit inférieure à $\epsilon^{\max}$:

$$\begin{aligned}
\mathbf{P}\left[\epsilon(r) \leq \epsilon^{\max}\right] &= \mathbf{P}\left[\frac{\min_i \left\{\Gamma^{(2i-1)} + \Gamma^{(2i)}\right\}}{2} - \Gamma_{\min} \leq \epsilon^{\max}\right] \\
&= \mathbf{P}\left[\min_i \left\{\Gamma^{(2i-1)} + \Gamma^{(2i)}\right\} \leq 2(\epsilon^{\max} + \Gamma_{\min})\right] \\
&= \mathrm{F}_A(2(\epsilon^{\max} - \Gamma_{\min}))
\end{aligned} \tag{4.29}$$

où $A \triangleq \min_i \left\{\Gamma^{(2i-1)} + \Gamma^{(2i)}\right\}$ ($\triangleq$ représente « égale par définition »). Dans le cas où on connaît la distribution du délai, $\mathrm{F}_\Gamma$:

$$\mathrm{F}_A(x) = 1 - \left[1 - \mathrm{F}_{\Gamma_\Sigma^{(2)}}(x))\right]^r \tag{4.30}$$

où $\Gamma_\Sigma^{(2)}$ est définie par $\Gamma_\Sigma^{(j)} \triangleq \sum_{i=k}^{k+j-1} \Gamma^{(i)}$ ($\Gamma_\Sigma^{(j)}$ est l'addition de $j$ v.a. $\Gamma$).

### Intersection des intervalles

Olson et Shin proposent dans [Olson et Shin, 1991] que l'intervalle associé à une suite d'essais soit l'intersection des intervalles obtenus dans chaque essai. Les intervalles qui sont considérés pour l'intersection sont les intervalles qui contiennent la valeur de l'écart entre les horloges de $p$ et de $q$, et non les intervalles qui contiennent l'estimation de l'horloge distante.

Avant d'analyser la stratégie d'intersection des intervalles, nous présentons un exemple qui aidera à mieux comprendre les mécanismes de son fonctionnement et, en particulier, ceux qui affectent l'erreur.

La figure 4.6 représente l'échange de messages entre les sites $p$ et $q$. Les horloges diffèrent de 3 unités, et celle de $p$ est en avance ($\forall_t, H_p(t) = H_q(t) + 3$). $\Gamma_{\min} = 1$ et le système est du type $D_0$ ($\rho = 0$). Le tableau 4.1 contient les valeurs de $H_q(t_3^{(i)})$ (la valeur de l'horloge de $q$ à l'instant de réception de la réponse), $H_p(t_2^{(i)})$ (la valeur de l'horloge envoyée par $p$), $\Delta q^{(i)}$ (le délai de l'aller-retour mesuré par $q$), $\mathrm{Int}^{(i)}$ (l'intervalle des valeur possibles pour $H_p^{(q)}(t_3^{(i)})$, l'estimation de l'horloge de $p$ par $q$ à l'instant de réception de la réponse) et $\Delta^{(i)}$ (l'intervalle des valeurs possibles pour la différence entres les horloges: $\Delta^{(i)} = \mathrm{Int}^{(i)} - H_q(t_3^{(i)})$).

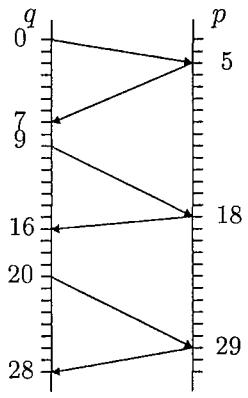



FIG. 4.6: *Stratégie d'intersection des intervalles*

| $H_q(t_3^{(i)})$ | $H_p(t_2^{(i)})$ | $\Delta q^{(i)}$ | $\text{Int}^{(i)}$ | $\Delta^{(i)}$ |
|---|---|---|---|---|
| 7 | 5 | 7 | [6, 11] | [-1, +4] |
| 16 | 18 | 7 | [19, 24] | [+3, +8] |
| 28 | 29 | 8 | [30, 36] | [+2, +8] |

TAB. 4.1: *Stratégie d'intersection des intervalles*

Pour l'exemple présenté, l'intersection de tous les intervalles $\Delta = \cap_i \Delta^{(i)} = [+3, +4]$. En regardant cet exemple avec un peu d'attention, notamment en analysant l'évolution de l'intersection des intervalles calculés par $q$, on pourra vérifier que la distance de l'extrémité inférieure (resp. supérieure) de $\Delta$ à la valeur correcte de $H_p - H_q$ dépend du plus court délai dans la transmission de la réponse (resp. de la requête).

Après cet exemple, reprenons l'analyse détaillée de cette méthode. Définissons $\Delta_{pq}(t) = H_p(t) - H_q(t)$ comme l'écart entre les horloges de $p$ et de $q$ et $\Delta^{(i)}$ comme l'intervalle qui contient les valeurs possibles pour $\Delta_{pq}$ obtenu dans l'essai $i$. Définissons aussi l'opération algébrique d'addition d'une valeur $x$ à un intervalle $I = [a, b]$ $(x, a, b \in \mathbb{R})$ comme: $I + x = [a+x, b+x]$. L'opération de soustraction est définie de façon identique. L'intervalle des écarts est défini comme:

$$\begin{aligned} \Delta^{(i)} &= I^{(i)} - H_q(t_4) \\ &= [H_p(t_3) - H_q(t_4) + (1-\rho)\Gamma_{\min}, \\ &\quad H_p(t_2) - H_q(t_1) + 2\rho\Delta H_q - (1+\rho)\Gamma_{\min}] \end{aligned} \tag{4.31}$$

**Hypothèse 4.3** *L'écart entre les horloges peut être considéré constant durant la durée de synchronisation.*

L'hypothèse 4.3 est vraie si $\Delta_{pq} \gg 2\rho S$ (où $S$ est la durée de synchronization, définie au chapitre 3) et permet de considérer $\Delta_{pq}(t_j) = \Delta_{pq}$, $j = 1..4$. Remarquons au travers des deux lemmes suivants :

**Lemme 4.1**

$$\Delta_{pq} + (1-\rho)\Gamma^{(2i-1)} \leq H_p(t_2) - H_q(t_1) \leq \Delta_{pq} + (1+\rho)\Gamma^{(2i-1)}$$

**Lemme 4.2**

$$\Delta_{pq} - (1+\rho)\Gamma^{(2i)} \leq H_p(t_3) - H_q(t_4) \leq \Delta_{pq} - (1-\rho)\Gamma^{(2i)}$$

L'intervalle $\Delta^{(i)}$ correspond aux valeurs calculées par le site $q$, mais il n'est pas exprimé en fonction des délais. Nous allons maintenant calculer un intervalle $\Delta'^{(i)} \supseteq \Delta^{(i)}$ qui est défini en fonction de l'écart $\Delta_{pq}$ et des délais. L'erreur associée à $\Delta'^{(i)}$ est supérieure à l'erreur associée à $\Delta^{(i)}$. Dans ce cas, si les spécifications pour l'erreur sont respectées avec l'intervalle $\Delta'^{(i)}$, elles le seront aussi avec l'intervalle $\Delta^{(i)}$.

$$\begin{aligned} \Delta'^{(i)} = \Big[&\Delta_{pq} - (1+\rho)\Gamma^{(2i)} + (1-\rho)\Gamma_{\min}, \\ &\Delta_{pq} + (1+\rho)\Gamma^{(2i-1)} + 2\rho(1+\rho)\Delta_q - (1+\rho)\Gamma_{\min}\Big] \end{aligned} \tag{4.32}$$

$\Delta$ est l'intervalle qui servira à estimer l'écart entre les horloges :

$$\begin{aligned}\Delta &= \bigcap_i \Delta'^{(i)} \\ &= \Big[\Delta_{pq} - \min_i\left\{(1+\rho)\Gamma^{(2i)}\right\} + (1-\rho)\Gamma_{\min}, \\ &\qquad \Delta_{pq} + \min_i\left\{(1+\rho)\Gamma^{(2i-1)} + 2\rho\Delta_q\right\} - (1+\rho)\Gamma_{\min}\Big] \\ &\approx \Big[\Delta_{pq} - \min_i\left\{\Gamma^{(2i)}\right\} + \Gamma_{\min} - \rho(\min_i\left\{\Gamma^{(2i)}\right\} + \Gamma_{\min}), \\ &\qquad \Delta_{pq} + \min_i\left\{\Gamma^{(2i-1)}\right\} - \Gamma_{\min} + \rho(\min_i\left\{\Gamma^{(2i-1)}\right\} + \Gamma_{\min})\Big] \qquad (4.33)\end{aligned}$$

où le dernier passage est justifié par l'hypothèse :

**Hypothèse 4.4**

$$(1+\rho)\Gamma^{(2i-1)} \gg 2\rho\Delta_q$$

Remarquons au passage à propos de l'équation (4.33) qu'elle confirme l'observation que l'erreur dans la borne maximale dépend du délai aller et celle de la borne minimale du délai de retour, ce qui avait été observé d'une façon informelle dans l'exemple présenté.

La borne maximale de l'erreur associée à cet intervalle est :

$$\begin{aligned}BME = &\frac{1}{2}\Big[\min_i\left\{\Gamma^{(2i-1)}\right\} + \min_i\left\{\Gamma^{(2i)}\right\} - 2\Gamma_{\min} \\ &+\rho\left(\min_i\left\{\Gamma^{(2i-1)}\right\} + \min_i\left\{\Gamma^{(2i)}\right\} + 2\Gamma_{\min}\right)\Big] \qquad (4.34)\end{aligned}$$

Comme pour le cas précédent, nous faisons l'hypothèse :

**Hypothèse 4.5** *Les termes en $\rho$ de l'équation (4.34) peuvent être négligés.*

La condition qui valide cette hypothèse est :

$$\begin{aligned}&\min_i\left\{\Gamma^{(2i-1)}\right\} + \min_i\left\{\Gamma^{(2i)}\right\} - 2\Gamma_{\min} \gg \\ &\qquad \rho\left(\min_i\left\{\Gamma^{(2i-1)}\right\} + \min_i\left\{\Gamma^{(2i)}\right\} + 2\Gamma_{\min}\right) \qquad (4.35)\end{aligned}$$

Pour que l'équation (4.35) soit valide, il faut que les deux expressions suivantes soient vraies :

$$\frac{\left(\min_i\left\{\Gamma^{(2i-1)}\right\} - \Gamma_{\min}\right) + \left(\min_i\left\{\Gamma^{(2i-1)}\right\} - \Gamma_{\min}\right)}{\min_i\left\{\Gamma^{(2i-1)}\right\} + \min_i\left\{\Gamma^{(2i)}\right\}} \gg \rho \qquad (4.36)$$

$$\frac{\left(\min_i\left\{\Gamma^{(2i-1)}\right\} - \Gamma_{\min}\right) + \left(\min_i\left\{\Gamma^{(2i-1)}\right\} - \Gamma_{\min}\right)}{\Gamma_{\min}} \gg \rho \qquad (4.37)$$

En termes informels, les équations précédentes signifient que les minima de la partie variable des délais doivent être beaucoup plus grands que $\rho$ fois les délais minimaux vérifiés (en effet, la première condition implique la deuxième).

L'expression pour la borne maximale de l'erreur dans le cas de l'intersection des intervalles devient :

$$BME = \frac{\min_i \left\{\Gamma^{(2i-1)}\right\} + \min_i \left\{\Gamma^{(2i)}\right\}}{2} - \Gamma_{\min} \tag{4.38}$$

En comparant (4.38) avec (4.28) il est possible d'identifier l'avantage apporté par l'intersection des intervalles. La stratégie d'essai jusqu'au premier succès utilise la valeur minimale de la somme de deux délais (aller et retour). Dans la stratégie de l'intersection des intervalles, la valeur minimale retenue à la fin des $r$ essais est, non le minimum d'une somme de deux valeurs, mais la somme de deux minima indépendants. En conséquence, la *BME*, pour le cas de l'intersection des intervalles, est toujours égale ou inférieure au cas précédent, puisque pour deux suites de valeurs réelles $A^{(i)}$ et $B^{(i)}$, $\min\{A^{(i)} + B^{(i)}\} \geq \min\{A^{(i)}\} + \min\{B^{(i)}\}$. En tout cas, l'équation (4.38) ne prend pas en compte le fait que l'information sur la différence $\Delta_{pq}$ devient obsolète avec l'écoulement du temps. Pour des grandes valeurs de la durée de resynchronisation $S$ ou pour des valeurs très petites pour $\Delta_{pq}$, l'hypothèse 4.3 ($\Delta_{pq} \gg 2\rho S$) peut ne plus être valable.

La probabilité que l'erreur après $r$ essais soit inférieure à $\epsilon^{\max}$ est :

$$\begin{aligned}
\mathbf{P}\left[|\epsilon(r)| \leq \epsilon^{\max}\right] &= \mathbf{P}\left[\frac{\left(\min_i \left\{\Gamma^{(2i-1)}\right\} + \min_i \left\{\Gamma^{(2i)}\right\}\right)}{2} - \Gamma_{\min} \leq \epsilon^{\max}\right] \\
&= \mathbf{P}\left[\min_i \left\{\Gamma^{(2i-1)}\right\} + \min_i \left\{\Gamma^{(2i)}\right\} \leq 2(\epsilon^{\max} - \Gamma_{\min})\right] \\
&= \mathrm{F}_A(2(\epsilon^{\max} - \Gamma_{\min}))
\end{aligned} \tag{4.39}$$

où $A \triangleq \min_i \left\{\Gamma^{(2i-1)}\right\} + \min_i \left\{\Gamma^{(2i)}\right\}$. Pour calculer $F_A$, décomposons $A$ en $A = X + Y$, où $X \triangleq \min_i \left\{\Gamma^{(2i-1)}\right\}$ et $Y \triangleq \min_i \left\{\Gamma^{(2i)}\right\}$. Ces deux distributions sont identiques sous l'hypothèse que les délais soient symétriques. Dans le cas où on connaît la distribution du délai, $\mathrm{F}_\Gamma$ :

$$\begin{aligned}
\mathrm{F}_X(x) &= 1 - (1 - \mathrm{F}_\Gamma(x))^r \\
\mathrm{f}_Y(x) &= r\,(1 - \mathrm{F}_\Gamma(x))^r\,\mathrm{f}_\Gamma(x) \\
\mathrm{F}_A(x) &= \int_{-\infty}^{+\infty} \mathrm{F}_X(x - \xi)\mathrm{f}_Y(\xi)d\xi
\end{aligned} \tag{4.40}$$

Les équations (4.39) et (4.40) permettent d'obtenir une expression pour la probabilité que l'erreur soit inférieure à un seuil $\epsilon^{\max}$ en termes de la caractérisation du délai $\Gamma$ comme une variable aléatoire.

### 4.3.4 Moyenne

On trouve deux variantes des algorithmes qui utilisent la moyenne : l'envoi de l'horloge distante et la perception d'un événement commun. La première variante est utilisée, par exemple, dans l'algorithme de Arvind ([Arvind, 1994]) et la seconde dans l'algorithme de Le Lann ([Le Lann, 1990]).

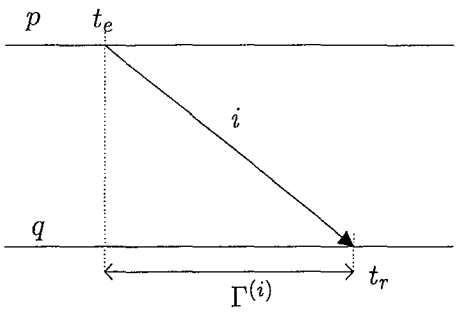



FIG. 4.7: *Envoi de la valeur de l'horloge*

### Envoi de l'horloge distante

Le site $p$ envoie un message au site $q$ qui contient la valeur de l'horloge de l'émetteur à l'instant où le message est envoyé (Fig. 4.7). $p$ envoie le message à l'instant (temps physique) $t_e$ ; $t_r$ est l'instant de réception du message par $q$. $\Gamma^{(i)} = t_r - t_e$ est le délai de transmission du message $i$. À l'instant de la réception, le site $q$ peut estimer la valeur de l'horloge de $p$ par :

$$H_p^{(q)}(t_r) = H_p(t_e) + \hat{\Gamma} \tag{4.41}$$

où $\hat{\Gamma}$ est une estimation du délai de transmission.

L'erreur associée au message $i$ est :

$$\begin{aligned} \epsilon^{(i)} &= H_p(t_r) - H_p^{(q)}(t_r) \\ &= H_p(t_r) - H_p(t_e) - \hat{\Gamma} \end{aligned} \tag{4.42}$$

De (4.42) et sachant que l'horloge de $p$ a une dérive $\rho$ on a :

$$\Gamma^{(i)} - \hat{\Gamma} - \rho\Gamma^{(i)} \leq \epsilon^{(i)} \leq \Gamma^{(i)} - \hat{\Gamma} + \rho\Gamma^{(i)} \tag{4.43}$$

L'équation (4.43) met en évidence deux termes qui contribuent à l'erreur de cette méthode :

- la différence entre la vraie valeur du délai et l'estimation, $\Gamma^{(i)} - \hat{\Gamma}$ ;
- l'effet de la dérive de l'horloge du site $p$ pendant la durée de la transmission du message, $\rho\Gamma^{(i)}$.

Nous ferons dans la suite l'hypothèse :

**Hypothèse 4.6** *Les termes en $\rho$ de l'équation (4.43) peuvent être négligés.*

Cette hypothèse permet l'approximation $\epsilon^{(i)} = \Gamma^{(i)} - \hat{\Gamma}$ et est valide ssi :

$$\left|\Gamma^{(i)} - \hat{\Gamma}\right| \gg \rho\Gamma^{(i)}$$

Cette hypothèse est-elle acceptable? Elle signifie que la principale contribution pour l'erreur est celle de la différence entre le vrai délai et l'estimation de ce délai. Car $\rho \ll 1$ ($\rho \approx 10^{-6}$ pour une horloge à quartz), l'hypothèse 4.6 est encore valide pour des valeur de $\Gamma^{(i)}$ qui différent de $\hat{\Gamma}$ par une part sur mille: pour un délai de l'ordre de quelques millisecondes, une différence de micro-seconde fait que la contribution de cette différence pour l'erreur soit à peu près 1000 fois supérieure à celle de la dérive. L'hypothèse ne sera plus valable si la différence relative entre $\Gamma^{(i)}$ et $\hat{\Gamma}$ est de l'ordre de $\rho$. La probabilité que l'hypothèse 4.6 soit invalide peut être quantifiée comme suit. Dire que $\left|\Gamma^{(i)} - \hat{\Gamma}\right| \gg \rho\Gamma^{(i)}$ est équivalent à dire que $\left|\Gamma^{(i)} - \hat{\Gamma}\right| > k\rho\Gamma^{(i)}$ pour $k \gg 1$. La valeur $k$ exprime le rapport minimal qu'on accepte entre la contribution de la différence des délais et celle de la dérive pour que cette dernière puisse être négligée. Pour le cas où $k\rho \ll 1$, cette probabilité est approximativement $\mathrm{F}_\Gamma((1+k\rho)\hat{\Gamma}) - \mathrm{F}_\Gamma((1-k\rho)\hat{\Gamma}) = \mathbf{P}\left[(1-k\rho)\hat{\Gamma} \leq \Gamma^{(i)} \leq (1+k\rho)\hat{\Gamma}\right]$, où $\mathrm{F}_\Gamma$ est la fonction de distribution du délai $\Gamma$.

Pour une série de $r$ essais, l'erreur à la fin de la série est la moyenne des erreurs:

$$\begin{aligned}
\epsilon(r) &= \frac{1}{r}\sum_{i=1}^{r} \epsilon^{(i)} \\
&= \frac{1}{r}\sum_{i=1}^{r} \left(\Gamma^{(i)} - \hat{\Gamma}\right) \\
&= \frac{1}{r}\sum_{i=1}^{r} \Gamma^{(i)} - \hat{\Gamma} \\
&= \overline{\Gamma}^{(r)} - \hat{\Gamma}, \quad \overline{\Gamma}^{(r)} \triangleq \frac{1}{r}\sum_{i=1}^{r} \Gamma^{(i)}
\end{aligned} \tag{4.44}$$

$$\begin{aligned}
\mathbf{P}\left[|\epsilon(r)| \leq \epsilon^{\max}\right] &= \mathbf{P}\left[-\epsilon^{\max} \leq \epsilon(r) \leq \epsilon^{\max}\right] \\
&= \mathbf{P}\left[-\epsilon^{\max} \leq \overline{\Gamma}^{(r)} - \hat{\Gamma} \leq \epsilon^{\max}\right] \\
&= \mathrm{F}_{\overline{\Gamma}^{(r)}}(\hat{\Gamma} + \epsilon^{\max}) - \mathrm{F}_{\overline{\Gamma}^{(r)}}(\hat{\Gamma} - \epsilon^{\max})
\end{aligned} \tag{4.45}$$

où $\mathrm{F}_{\overline{\Gamma}^{(r)}}(x)$ est la fonction de distribution de $\overline{\Gamma}^{(r)}$. Cette probabilité peut être calculée de deux façons: soit à travers le calcul de $\mathrm{F}_{\overline{\Gamma}^{(r)}}(x)$ en termes de $\mathrm{F}_\Gamma$, soit à travers le Théorème de la Limite Centrale (TLC).

Pour le premier cas, définissons $\Gamma_\Sigma^{(j)} \triangleq \sum_{i=1}^{j} \Gamma^{(i)}$. La fonction de densité de probabilité (fdp) de $\Gamma_{\Sigma^{(j)}}$, $\mathrm{f}_{\Gamma_{\Sigma^{(j)}}}$ est la convolution des $j$ fdp des $\Gamma^{(i)}$:

$$\mathrm{f}_{\Gamma_\Sigma^{(2)}} = \mathrm{f}_\Gamma * \mathrm{f}_\Gamma \tag{4.46}$$

$$\mathrm{f}_{\Gamma_\Sigma^{(j)}} = \mathrm{f}_{\Gamma_\Sigma^{(j-1)}} * \mathrm{f}_\Gamma \tag{4.47}$$

$\mathrm{F}_{\overline{\Gamma}^{(r)}}(x)$ peut être calculée en utilisant la propriété qui dit que, pour deux v.a. $X$ et $Y$ telles que $Y = kX$ ($k \in \mathbb{R}$), $\mathrm{F}_Y(x) = \mathrm{F}_X(x/k)$:

$$\mathrm{F}_{\overline{\Gamma}^{(r)}}(x) = \mathrm{F}_{\Gamma_\Sigma^{(r)}}(rx) \tag{4.48}$$

Ce procédé peut être laborieux et compliqué ; calculer $\mathrm{F}_{\overline{\Gamma}}$ peut demander du temps et des moyens de calcul considérables. Cependant, pour quelques cas particuliers, il peut être relativement facile de calculer les fonctions de distribution et de densité de probabilité de $\overline{\Gamma}^{(r)}$. Un de ces cas est quand les $\Gamma^{(i)}$ sont des v.a. avec distribution Gamma. La fdp d'une v.a. Gamma avec paramètres $\alpha$ et $\lambda$, $\mathrm{f}_{\Gamma(\alpha,\lambda)}$, est:

$$\mathrm{f}_{\Gamma(\alpha,\lambda)}(x) = \begin{cases} \frac{\lambda(\lambda x)^{(\alpha-1)}e^{-\lambda x}}{\Gamma(\alpha)} & , x \geq 0 \\ 0 & , x < 0 \end{cases} \tag{4.49}$$

Si $X$ est une v.a. avec distribution Gamma avec paramètres $\alpha$ et $\lambda$, ces deux paramètres se rapportent avec l'espérance et la variance par les équations (4.50) et (4.51).

$$\alpha = \frac{(\mathrm{E}[X])^2}{\mathrm{Var}[X]} \tag{4.50}$$

$$\lambda = \frac{\mathrm{E}[X]}{\mathrm{Var}[X]} \tag{4.51}$$

Le graphe de cette fonction avec paramètres $\alpha = 2$ et $\lambda = 1$ est présenté dans la figure 4.8.

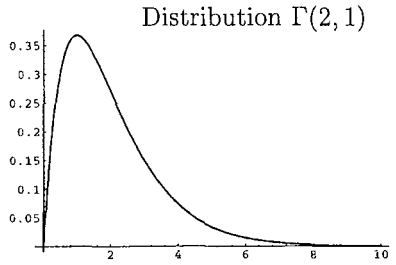



FIG. 4.8: *Fdp d'une v.a.* $\Gamma(2,1)$.

Cette fonction nous intéresse pour deux raisons. D'abord, elle ressemble à la distribution présentée par Cristian dans [Cristian, 1989b, Cristian, 1989a] (qui, par ailleurs, a été prise comme sacro-sainte pour les années qui ont suivi). En principe, elle sera adéquate, car elle représente une v.a. continue qui prend seulement des valeurs positives et qui peuvent être arbitrairement grandes (avec une probabilité d'occurrence qui décroît à mesure que la valeur augmente).

Du point de vue du traitement mathématique, elle est intéressante parce que l'addition de $r$ v.a. Gamma avec des paramètres $\alpha$ et $\lambda$ est elle aussi une v.a. Gamma avec paramètres $r\alpha$ et $\lambda$. Sous l'hypothèse $\Gamma \sim \Gamma(\alpha,\lambda)$, $\Gamma_\Sigma \sim \Gamma(r\alpha,\lambda)$ et $\mathrm{F}_{\overline{\Gamma}^{(r)}}(x) = F_{\Gamma_\Sigma}(rx) = \mathrm{F}_{\Gamma(r\alpha,\lambda)}(rx)$. Dans ce cas particulier, eq. (4.45) devient :

$$\mathbf{P}\left[|\epsilon(r)| \leq \epsilon^{\max}\right] = \mathrm{F}_{\Gamma(r\alpha,\lambda)}(r(\hat{\Gamma} + \epsilon^{\max})) - \mathrm{F}_{\Gamma(r\alpha,\lambda)}(r(\hat{\Gamma} - \epsilon^{\max})) \tag{4.52}$$

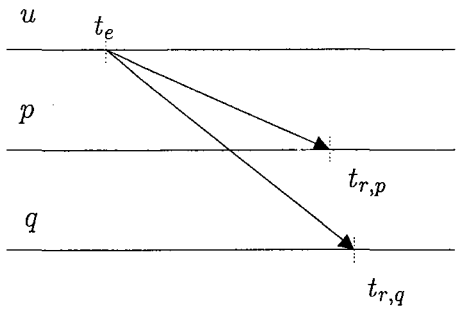



FIG. 4.9: *Perception d'une événement commun*

Pour les cas où il est impossible ou trop difficile d'obtenir une expression pour $F_{\overline{\Gamma}}$, on peut utiliser le Théorème de la Limite Centrale. Si $E[\Gamma] = \overline{\Gamma}$ et $Var[\Gamma] = \sigma_\Gamma^2$ sont connus, il est possible de calculer $E[\overline{\Gamma}] = \hat{\Gamma}$ et $Var[\overline{\Gamma}] = \sigma_\Gamma^2/r$. Pour une valeur de $r$ suffisamment grande, $\overline{\Gamma}$ tend vers une distribution normale (gaussienne) et

$$\mathbf{P}\left[|\epsilon(r)| \leq \epsilon^{\max}\right] = \Phi\left(\frac{\sqrt{r}\epsilon^{\max}}{\sigma_\Gamma}\right) - \Phi\left(-\frac{\sqrt{r}\epsilon^{\max}}{\sigma_\Gamma}\right) \tag{4.53}$$

où $\Phi(x)$ est la fonction de distribution d'une v.a. normale de moyenne nulle et variance unitaire.

### Perception d'une événement commun

Cette méthode essaye de créer un événement commun, qui est perçu par l'ensemble des sites. Un site $u$ envoie un message, qui est reçu par $p$ et $q$ et estampillé à l'instant d'arrivée (Fig. 4.9). $t_{r,p}$ et $t_{r,q}$ sont les instants de réception du message par $p$ et $q$, respectivement ; $\Gamma_p$ et $\Gamma_q$ sont les délais de transmission de $u$ à $p$ et $q$.

Le site $q$ fait l'hypothèse que les deux événements de réception sont simultanés, $t_{r,p} = t_{r,q}$. L'erreur associée à cette méthode est la dispersion des perceptions locales de l'événement global, c.-à-d, les différences des instants d'arrivée des messages sur plusieurs sites :

$$\epsilon^{(i)} = \Gamma_p^{(i)} - \Gamma_q^{(i)} \tag{4.54}$$

L'erreur associée à une suite de $r$ essais est la moyenne des erreurs :

$$\begin{aligned}
\epsilon(r) &= \frac{1}{r}\sum_{i=1}^{r} \epsilon^{(i)} \\
&= \frac{1}{r}\sum (\Gamma_p^{(i)} - \Gamma_q^{(i)}) \\
&= \frac{1}{r}\sum \Delta\Gamma^{(i)} \qquad \Delta\Gamma^{(i)} \triangleq \Gamma_p^{(i)} - \Gamma_q^{(i)} \\
&= \Delta\overline{\Gamma}
\end{aligned} \tag{4.55}$$

Les procédés de calcul de cette probabilité sont identiques au cas de l'envoi de l'horloge distante. Dans le cas de la perception d'un événement commun, l'erreur dépend de la v.a. définie par $\Delta\overline{\Gamma}$. La valeur $\Delta\Gamma^{(i)} = \Gamma_p^{(i)} - \Gamma_q^{(i)}$ est une fonction de deux variables aléatoires qui ne sont pas normalement indépendantes : elles correspondent aux délais d'un message de diffusion. Reprenons le modèle pour le délai présenté précédemment, c'est facile de constater que, pour un réseau à diffusion, les composants $\Gamma_{\text{prep}}$ et $\Gamma_{\text{acc}}$ sont identiques pour les deux délais. La différence entre $\Gamma_p^{(i)}$ et $\Gamma_q^{(i)}$ est dûe aux composants $\Gamma_{\text{prop}}$ et $\Gamma_{\text{rec}}$, qui ne sont plus forcément identiques pour deux sites différents et pour le même message. Ceci permet d'exprimer $\Delta\Gamma^{(i)}$ comme une fonction de variables aléatoires indépendantes :

$$\begin{aligned}\Delta\Gamma^{(i)} &= {\Gamma_p^{(i)}}_{\text{prop}} + {\Gamma_p^{(i)}}_{\text{rec}} - {\Gamma_q^{(i)}}_{\text{prop}} - {\Gamma_q^{(i)}}_{\text{rec}} \\ &= {\Delta\Gamma^{(i)}}_{\text{prop}} + {\Delta\Gamma^{(i)}}_{\text{rec}} \quad ({\Delta\Gamma^{(i)}}_{\text{prep}} = {\Delta\Gamma^{(i)}}_{\text{acc}} = 0)\end{aligned} \tag{4.56}$$

En faisant l'hypothèse que les délais vers $p$ et $q$ sont des v.a. avec la même espérance et indépendants :

$$\mathrm{E}[\Delta\Gamma_{\text{prop}}] = \mathrm{E}[\Delta\Gamma_{\text{rec}}] \quad \Rightarrow \quad \mathrm{E}[\Delta\Gamma] = 0 \tag{4.57}$$

$$\begin{aligned}\mathrm{Var}[\Delta\Gamma] &= \mathrm{Var}[\Gamma_{p\text{prop}}] + \mathrm{Var}[\Gamma_{p\text{rec}}] + \mathrm{Var}[\Gamma_{q\text{prop}}] + \mathrm{Var}[\Gamma_{q\text{rec}}] \\ &= 2\left(\mathrm{Var}[\Gamma_{\text{prop}}] + \mathrm{Var}[\Gamma_{\text{rec}}]\right) \\ &\triangleq \sigma^2_{\text{diff}}\end{aligned} \tag{4.58}$$

L'espérance et la variance de $\Delta\overline{\Gamma}$ sont donc :

$$\mathrm{E}\left[\Delta\overline{\Gamma}\right] = \mathrm{E}[\Delta\Gamma] = 0 \tag{4.59}$$

$$\mathrm{Var}\left[\Delta\overline{\Gamma}\right] = \sigma^2_{\Delta\overline{\Gamma}} = \frac{1}{r}\sigma^2_{\text{diff}} \tag{4.60}$$

La connaissance de l'espérance et de la variance de la variable $\Delta\overline{\Gamma}$ dont l'erreur dépend permet, pour une valeur de $r$ suffisamment grand (typiquement, $r \geq 30$) de calculer, en utilisant le Théorème de la Limite Centrale, la probabilité

$$\begin{aligned}\mathbf{P}\left[|\epsilon(r)| \leq \epsilon^{\max}\right] &= \mathbf{P}\left[-\epsilon^{\max} \leq \epsilon(r) \leq \epsilon^{\max}\right] \\ &= \mathbf{P}\left[\epsilon^{\max} \leq \Delta\overline{\Gamma} \leq \epsilon^{\max}\right] \\ &= \Phi\left(\frac{\epsilon^{\max}}{\sigma_{\Delta\overline{\Gamma}}}\right) - \Phi\left(-\frac{\epsilon^{\max}}{\sigma_{\Delta\overline{\Gamma}}}\right) \\ &= \Phi\left(\frac{\sqrt{r}\epsilon^{\max}}{\sigma_{\text{diff}}}\right) - \Phi\left(-\frac{\sqrt{r}\epsilon^{\max}}{\sigma_{\text{diff}}}\right)\end{aligned} \tag{4.61}$$

### 4.3.5 Analyse des résultats

Pour chacune des stratégies identifiées dans la section 4.2 nous pouvons associer une formulation pour la probabilité d'erreur qui lui est propre (tableau 4.2). La probabilité de l'erreur pour les algorithmes de régression suit une distribution $t$ de Student avec $r-2$ degrés de liberté ; pour les algorithmes d'intervalle, c'est la probabilité que la borne maximale de l'erreur soit inférieure au seuil maximal de l'erreur. Pour les algorithmes de moyenne, la

| Stratégie | | $\mathbf{P}\left[\lvert\epsilon(r)\rvert \leq \epsilon^{\max}\right]$ |
|---|---|---|
| Régression | | $T_{r-2}\left(\frac{\sqrt{r-2}}{2}\frac{\epsilon^{\max}}{\sqrt{M_2}}\right) - T_{r-2}\left(-\frac{\sqrt{r-2}}{2}\frac{\epsilon^{\max}}{\sqrt{M_2}}\right)$ |
| Intervalle | Ess. 1 suc | $\mathrm{F}_{BME}(\epsilon^{\max})$ |
| | Int. Int. | $\mathrm{F}_{BME}(\epsilon^{\max})$ |
| Moyenne | EHD | $\mathrm{F}_A(\epsilon^{\max}) - \mathrm{F}_A(-\epsilon^{\max})$ |
| | PEC | $\mathrm{F}_A(\epsilon^{\max}) - \mathrm{F}_A(-\epsilon^{\max})$ |

TAB. 4.2: *Probabilité d'erreur*

| Stratégie | | V.a. |
|---|---|---|
| Régression | | $M_2 = \hat{\Gamma}^2 + \sigma_\Gamma^2$ |
| Intervalle | Ess. 1 suc | $BME = \frac{\min_i\{\Gamma^{(2i-1)}+\Gamma^{(2i)}\}}{2} - \Gamma_{\min}$ |
| | Int. Int. | $BME = \frac{\min_i\{\Gamma^{(2i-1)}\}+\min_i\{\Gamma^{(2i)}\}}{2} - \Gamma_{\min}$ |
| Moyenne | EHD | $A = \overline{\Gamma}^{(r)} - \hat{\Gamma}$ |
| | PEC | $A = \overline{\Gamma}_p^{(r)} - \overline{\Gamma}_q^{(r)}$ |

TAB. 4.3: *Variables aléatoires qui affectent l'erreur*

probabilité d'erreur est la probabilité qu'une variable aléatoire qui dépend de la différence entre deux délais soit inférieure, en valeur absolue, à l'erreur maximale.

Les résultats présentés ont été calculés en utilisant certaines hypothèses communes à l'ensemble des algorithmes non-déterministes et qui définissent le type de système pour lequel les résultats sont valables. Ces hypothèses peuvent se résumer en deux grandes classes:

- l'écart entre deux horloges peut être considéré constant pendant l'exécution de l'algorithme. Nous considérons seulement l'effet accumulé de la dérive et négligeons sa contribution à la variation de l'écart pendant la durée d'exécution de l'algorithme. En termes mathématiques, ceci correspond à $\Delta_{pq} \gg \rho S$.

- le rapport entre l'incertitude du délai et les délais dans l'exécution de l'algorithme (les délais de transmission d'un message et les délais en attendant son traitement) est beaucoup plus grand que $\rho$. Ceci permet de négliger tous les termes en $\rho$ par rapport à l'incertitude du délai et considérer celle-ci comme la principale source d'erreur.

La variable qui affecte l'erreur pour chacune des méthodes (tableau 4.3) représente l'aspect du système qui doit être contrôlé pour contrôler l'erreur. Dans les algorithmes de régression, cette variable est le moment de deuxième ordre du délai $\Gamma$, $\mathrm{E}\left[\Gamma^2\right] = \hat{\Gamma}^2 + \sigma_\Gamma^2$. Cette caractéristique est unique entre les stratégies mentionnées et pénalise l'utilisation des réseaux où le délai a une très petite variance (comportement proche d'un réseau déterministe) mais où le délai moyen a une valeur considérable par rapport à $\epsilon^{\max}$. Dans ce cas, garantir la probabilité de succès avec de petites valeur pour $\epsilon^{\max}$ peut demander un nombre de messages élevé.

L'erreur dans les algorithmes qui utilisent la stratégie des intervalles dépend de la partie variable du délai et ils utilisent la valeur minimale trouvée dans la suite des $r$ essais. Dans ce cas, nous sommes intéressés à utiliser ce type d'algorithmes dans les réseaux où la fonction de densité de probabilité des délais est asymétrique et biaisée vers la gauche (c'est-à-dire, les valeurs les plus probables se trouvent proches de la valeur minimale). Ceci garantit une proba-

| Stratégie | Situation favorable | Paramètres |
|---|---|---|
| Régression | Délai moyen et variance avec valeurs petites | Moy : petite<br>Min : —<br>Var : petite |
| Intervalles | Forte probabilité qu'une valeur du délai soit proche de la valeur minimale | Moy : proche de Min<br>Min : —<br>Var : — |
| Moyenne | | |
| - EHD | Petite variance du délai; rapport entre Min et Moy n'influence pas directement | Moy : —<br>Min : —<br>Var : petite |
| - PEC | Petite dispersion des instants d'arrivée aux différents sites | Moy : —<br>Min : —<br>Var : petite |

TAB. 4.4: *Situations favorables aux différentes stratégies*

bilité raisonnable pour l'occurrence d'une lecture avec une petite incertitude (qui correspond à des délais dont la valeur de la partie variable est très petite).

Pour le cas des algorithmes de la moyenne, l'erreur dépend de la différence entre deux estimations du délai. Dans le cas de l'envoi de l'horloge distante, c'est la différence entre le délai moyen durant les $r$ essais et l'estimation obtenue a priori. Pour le cas de la perception d'un événement commun, c'est la différence entre les délais moyens durant les $r$ essais du site émetteur aux sites récepteurs. Ce type d'algorithmes s'adaptent bien à des réseaux où le délai a une petite variance, même si sa valeur moyenne est significativement plus grande que la valeur minimale. Pour la stratégie de la perception d'un événement commun, il est fondamental que la dispersion dans le temps des instants d'arrivée du même message aux différents sites soit aussi petite que possible.

Le tableau 4.4 présente, d'une façon qualitative, quelques indications, pour le concepteur d'un système distribué, sur les caractéristiques des réseaux (en ce qui concerne le délai de tranmission) qui sont les plus favorables aux différentes stratégies. La colonne «Paramètres» exprime comment ces caractéristiques se traduisent dans la description statistique des délais, à travers la moyenne, la valeur minimale et la variance. Le tiret (—) signifie que l'influence du paramètre respective a une importance mineure par rapport aux autres paramètres.

Les expressions analytiques présentées dans le tableau 4.5 servent à mieux comprendre le rapport entre les grandeurs dont la probabilité de l'erreur dépend. Pour les algorithmes de régression et de moyenne, l'expression est identique, ce qui est issu du fait que l'erreur est considéré comme une v.a. avec distribution normale. Dans ces cas, la probabilité que l'erreur soit inférieure au seuil maximal défini croît avec $r$ et $\epsilon^{\max}$ et diminue avec la variance du délai (pour les algorithmes de moyenne) ou avec le moment de deuxième ordre du délai (pour les algorithmes de régression). Pour les algorithmes d'intervalle, on trouve dans les deux expressions des termes du type $F_\Gamma(2(\epsilon^{\max} + \Gamma_{\min}))$, la probabilité que le délai soit inférieur à $2(\epsilon^{\max} + \Gamma_{\min})$. Les expressions dépendent de la distribution des délais. La probabilité pour

| Stratégie | $\mathbf{P}\left[\lvert\epsilon(r)\rvert \leq \epsilon^{\max}\right]$ |
|---|---|
| Régression | $\Phi\left(\frac{\sqrt{r-2}}{2}\frac{\epsilon^{\max}}{\sqrt{M_2}}\right) - \Phi\left(-\frac{\sqrt{r-2}}{2}\frac{\epsilon^{\max}}{\sqrt{M_2}}\right)$ |
| Intervalle : | |
| - Ess. 1 suc. | $1-(1-p)^r, \quad p = \mathbf{P}\left[\Gamma^{(2i)} + \Gamma^{(2i+1)} \leq 2(\epsilon^{\max} + \Gamma_{\min})\right]$ |
| - Int. Int. | $\int_{-\infty}^{+\infty} [1-(1-\mathrm{F}_\Gamma(2(\epsilon^{\max}+\Gamma_{\min})-\xi))^r]\, r\,(1-\mathrm{F}_\Gamma(\xi))^r\, \mathrm{f}_\Gamma(\xi) d\xi$ |
| Moyenne : | |
| - EHD | $\Phi\left(\frac{\sqrt{r}\epsilon^{\max}}{\sigma_\Gamma}\right) - \Phi\left(-\frac{\sqrt{r}\epsilon^{\max}}{\sigma_\Gamma}\right)$ |
| - PEC | $\Phi\left(\frac{\sqrt{r}\epsilon^{\max}}{\sigma_{\text{diff}}}\right) - \Phi\left(-\frac{\sqrt{r}\epsilon^{\max}}{\sigma_{\text{diff}}}\right)$ |

TAB. 4.5: *Probabilité d'erreur (expression analytique)*

le cas de l'essai jusqu'au premier succès peut être présentée dans une forme simplifiée, en fonction de la probabilité que l'addition de deux délais soit inférieure à une valeur donnée. Pour le cas de l'intersection des intervalles, l'erreur dépend de l'addition de deux valeurs minimales pour deux suites d'essais ; ce découplement rend l'expression plus compliquée, et la simplification n'est plus possible.

## 4.4 Analyse de l'exécution d'un algorithme

Dans la section précédente, nous avons établi le rapport entre les paramètres les plus importants pour la performance des algorithmes non-déterministes de synchronisation des horloges : l'erreur maximale tolérée, la probabilité que l'erreur n'excède pas le seuil spécifié et le nombre de messages nécessaires pour garantir cette probabilité. L'utilité de ces résultats est encore réduite pour le concepteur d'un système distribué car les résultats présentés concernent un seul site.

Dans la suite de cette section, nous essayerons de systématiser l'exécution d'un ASND dans un système distribué. Le but est de trouver le rapport entre les conditions locales à un site et les conditions globales pour le système (la probabilité que le système se synchronise avec une précision spécifiée). La motivation est le fait que les conditions locales, intrinsèques à un site, sont plus faciles à vérifier que des conditions globales.

Nous nous sommes intéressés à analyser le rapport entre trois paramètres : la probabilité de synchronisation, la précision et le nombre de messages. Nous ne nous sommes pas intéressés, pour le moment, aux aspects algorithmiques au sens strict ; ceux-ci nous intéressent parce qu'ils établissent le rapport entre les grandeurs mentionnées.

L'exécution d'un algorithme de synchronisation non-déterministe dans un système distribué peut être considérée comme la production d'une ou plusieurs suites de messages par un certain nombre de sites. Il peut se passer qu'un seul site sur tout le système produise des messages (comme par exemple dans une structure centralisée) ou il est possible que tous les sites soient producteurs de messages de synchronisation (dans une structure distribuée). Pour chaque message produit, il y a un ou plusieurs sites qui en sont les consommateurs. Chaque site producteur envoie ses messages en *étapes*. Une étape correspond à la plus petite unité d'exécution de l'ASND et elle consiste à envoyer une suite de $r$ messages de synchronisation. Tous les messages dans une étape ont le même destinataire. Le résultat de chaque étape est le calcul d'une valeur pour chaque site consommateur, à laquelle une erreur $\epsilon^{\text{val}}$ est associée.

Le processus d'exécution est cyclique, avec une période $R$.

Nous considérons que le système à analyser se compose de $n$ sites. Définissons $n_p$ comme le nombre de sites producteurs ($1 \leq n_p \leq n$) et $n_c$ comme le nombre de consommateurs pour chaque message ($1 \leq n_c \leq n$). Par exemple, pour un système où chaque site produit des messages pour tous les autres, le nombre total de sites consommateurs est $n$ mais chaque message s'adresse à $n-1$ sites : $n_c = n-1$. Définissons aussi $n_e$ comme le nombre d'étapes dans chaque site. Nous avons donc un total de $n_p n_e$ étapes dans chaque exécution de l'algorithme, et donc $n_p n_e n_c$ couples production/consommation de suites de $r$ messages.

Pour mieux comprendre le concept de l'étape et la signification des paramètres $n_p$, $n_e$ et $n_c$ nous présentons quelques exemples:

**Algorithme distribué dans un réseau à diffusion** [1] Le rôle de tous les sites est identique, donc tous sont producteurs de messages : $n_p = n$. Chaque site doit envoyer $r$ messages à tous les autres sites. La diffusion permet de le faire d'un seul coup : une étape d'envoi de $r$ messages est suffisante pour que tous les autres sites reçoivent les messages. Dans ce cas, $n_e = 1$. Chaque message est adressé à $n-1$ sites : $n_c = n-1$.

**Algorithme distribué dans un réseau point-à-point** Comme dans le cas précédent, $n_p = n$ (algorithme distribué). Pour que tous les autres sites reçoivent $r$ messages produits par un site $p$, celui doit répéter l'envoi $n-1$ fois (une fois pour chacun des sites distants). Il faut donc $n_e = n-1$ étapes. Dans chaque étape, il y a qu'un seul consommateur pour les messages : $n_c = 1$.

**Maitre-Esclave avec maître producteur** Le maître envoie la valeur de son horloge aux sites esclaves ; il est le seul producteur ($n_p = 1$). Si le réseau est à diffusion, une seule étape est nécessaire ($n_e = 1$ et $n_c = n-1$). Pour un réseau point-à-point, il faut une étape pour chaque site esclave ($n_e = n-1$ et $n_c = 1$).

**Maître-esclave avec maître consommateur** Chaque site esclave envoie $r$ messages au maître. Il y a $n_p = n-1$ site producteurs (les esclaves). Chaque producteur a besoin d'une seule étape où chaque message a un seul destinataire, indépendamment du type de réseau (diffusion ou point-à-point) : $n_e = n_c = 1$.

Le tableau 4.6 résume les valeurs attribuées aux paramètres $n_p$, $n_e$ et $n_c$ pour des différents types de structure (distribuée et maître-esclave) et de réseau (diffusion ou point-à-point). La dernière colonne présente le nombre de consommateurs associés à chaque producteur.

| Struct. | Réseau | $n_p$ | $n_e$ | $n_c$ |
|---|---|---|---|---|
| Dist. | Diff. | $n$ | 1 | $n-1$ |
| Dist. | P-à-p | $n$ | $n-1$ | 1 |
| M-E (M. prod.) | Diff. | 1 | 1 | $n-1$ |
| M-E (M. prod.) | P-à-p | 1 | $n-1$ | 1 |
| M-E (M. cons.) | Diff./P-à-p | $n-1$ | 1 | 1 |

TAB. 4.6: *Paramètres $n_p$, $n_e$ et $n_c$.*

1. Nous supposons que le réseau est à diffusion *fiable*

### 4.4.1 Garantie de la précision de synchronisation

Comme nous l'avons dit précédemment, l'algorithme se déroule en étapes ; à chaque site producteur correspondent $n_e$ étapes, donc il y a un total de $n_p n_e$ étapes. À chaque étape correspondent $n_c$ valeurs qui sont calculées par l'algorithme (un pour chaque couple producteur/consommateur). Nous supposons que les erreurs dans le calcul des valeurs associées aux consommateurs sont des v.a. i.i.d. :

**Hypothèse 4.7 (Indépendance des erreurs des valeurs)** *Les erreurs associées aux sites consommateurs d'une suite de messages d'une étape, $\epsilon^{\text{val}}$, sont des variables aléatoires indépendantes et identiquement distribuées. Elles sont représentées par la v.a. $\epsilon(r)$.*

Une des conséquences de l'hypothèse 4.7 est :

$$\mathbf{P}\left[\epsilon^{\text{etap}} < \epsilon^{\text{tol}}\right] = \left(\mathbf{P}\left[\epsilon(r) < \epsilon^{\text{tol}}\right]\right)^{n_c} \tag{4.62}$$

L'erreur d'une étape est inférieure à un seuil $\epsilon^{\text{tol}}$ ssi toutes les valeurs calculées dans cette étape ont une erreur inférieure à $\epsilon^{\text{tol}}$.

Sans le démontrer pour le moment, faisons l'hypothèse qu'il existe une valeur $\epsilon^{\text{tol}}$ telle que, si toutes les erreurs des étapes sont égales ou inférieures à $\epsilon^{\text{tol}}$, la précision $\delta$ est égale ou inférieure à $\delta^{\text{spec}}$. Supposons aussi que les erreurs associées aux étapes sont des v.a. indépendantes et identiquement distribuées.

**Hypothèse 4.8 (Existence de la valeur $\epsilon^{\text{tol}}$)**

$$\exists_{\epsilon^{\text{tol}}>0} \quad : \quad \forall_{p=1..n_p}, \forall_{i=1..n_e}, \quad \epsilon^{\text{etap}}_{p,i} \leq \epsilon^{\text{tol}} \Rightarrow \delta \leq \delta^{\text{spec}} \tag{4.63}$$

**Hypothèse 4.9 (Indépendance des erreurs des étapes)** *Les erreurs associées aux étapes sont des variables aléatoires indépendantes et identiquement distribuées. Elles sont représentées par la v.a. $\epsilon^{\text{etap}}$.*

La probabilité $\mathbf{P}\left[\epsilon(r) \leq \epsilon^{\text{tol}}\right]$ peut être difficile à calculer, mais il est souvent possible d'en calculer un majorant. Faisons l'hypothèse que ce majorant, qu'on appellera $f^{\text{err}}$, existe :

**Hypothèse 4.10**

$$\exists_{f^{\text{err}}(\epsilon^{\text{tol}},r)} \quad : \quad \mathbf{P}\left[\epsilon(r) \leq \epsilon^{\text{tol}}\right] \geq 1 - f^{\text{err}}(\epsilon^{\text{tol}}, r) \tag{4.64}$$

Les conditions qui ont été établies nous permettent maintenant d'établir la condition qui garantit que le système se synchronise avec une précision $\delta^{\text{spec}}$ et une probabilité $\mathrm{P}_{\text{spec}}$, basée seulement sur des observations locales à un site :

**Théorème 4.1 (Condition de garantie de la synchronisation)**

$$f^{\text{err}}(\epsilon^{\text{tol}}, r) \leq \frac{1 - \mathrm{P}_{\text{spec}}}{n_p n_e n_c} \Rightarrow \mathbf{P}\left[\delta \leq \delta^{\text{spec}}\right] \geq \mathrm{P}_{\text{spec}} \tag{4.65}$$

**Preuve** Pour prouver le théorème 4.1 nous introduisons le lemme:

**Lemme 4.3**

$$\forall_{x\in[0,1[}, \forall_{n\in\mathbb{N}}, \quad (1-x)^n \geq 1 - nx$$

$$\begin{aligned}
\mathbf{P}\left[\delta \leq \delta^{\text{spec}}\right] &\geq \mathbf{P}\left[\epsilon^{\text{etap}} \leq \epsilon^{\text{tol}} \text{ pour toutes les étapes}\right] \quad \text{(Hyp. 4.8)} \\
&= \left(\mathbf{P}\left[\epsilon^{\text{etap}} \leq \epsilon^{\text{tol}}\right]\right)^{n_p n_e} \quad \text{(Hyp. 4.9)} \\
&= \left(\mathbf{P}\left[\epsilon(r) \leq \epsilon^{\text{tol}}\right]\right)^{n_p n_e n_c} \quad \text{(Hyp. 4.7)} \\
&\geq 1 - n_p n_e n_c \mathbf{P}\left[\epsilon(r) \leq \epsilon^{\text{tol}}\right] \quad \text{(Lemme 4.3)} \\
&\geq 1 - n_p n_e n_c f^{\text{err}}(\epsilon^{\text{tol}}, r) \quad \text{(Hyp. 4.10)} \qquad (4.66)
\end{aligned}$$

De l'équation (4.66), on obtient

$$1 - n_p n_e n_c f^{\text{err}}(\epsilon^{\text{tol}}, r) \geq \mathrm{P}_{\text{spec}} \Rightarrow \mathbf{P}\left[\delta \leq \delta^{\text{spec}}\right] \geq \mathrm{P}_{\text{spec}}$$

et le théorème est prouvé. □

Le théorème 4.1 nous permet de trouver les conditions locales qui sont suffisantes pour garantir une caractéristique globale du système. Ceci nous permet de nous concentrer sur les conditions locales qui sont plus faciles a calculer et à maîtriser. Le résultat est basé sur les hypothèses suivantes :

1. l'indépendance des erreurs
2. l'existence de la borne $\epsilon^{\text{tol}}$ qui garantit la précision $\delta^{\text{spec}}$
3. l'existence de la fonction $f^{\text{err}}$, le majorant pour la probabilité que l'erreur dans une étape soit supérieure à $\epsilon^{\text{tol}}$.

La validité du résultat dépend de la validité de ces hypothèses. Nous étudierons ce sujet dans la section suivante.

### 4.4.2 Analyse de la validité des hypothèses

Dans cette section, nous traitons la question de la validité des hypothèses qui sont à la base du théorème 4.1. Ceci permettra de mieux spécifier les conditions sous lesquelles ce résultat est valide.

#### Indépendance des erreurs

L'indépendance des erreurs de chaque site est mentionnée dans [Rangarajan et Tripathi, 1991] comme le pire des cas. Cette affirmation mérite un peu de réflexion, car il faut bien vérifier les conditions qui la rendent valide.

Prenons l'exemple d'une valeur qui est calculée avec le résultat de deux lectures. L'erreur associée aux lectures est $\epsilon_1$ et $\epsilon_2$ ; l'erreur associée à la valeur calculée est $\epsilon = k(\epsilon_1 + \epsilon_2)$. Pour simplifier, considérons $k = 1$. La probabilité que l'erreur excède un certain seuil est fonction de sa variance, $\mathrm{Var}[\epsilon]$.

$$\begin{aligned}
\mathrm{Var}[\epsilon] &= \mathrm{Var}[\epsilon_1 + \epsilon_2] \\
&= \mathrm{Var}[\epsilon_1] + \mathrm{Var}[\epsilon_2] + \mathrm{Cov}[\epsilon_1, \epsilon_2] \qquad (4.67)
\end{aligned}$$

Si $\epsilon_1$ et $\epsilon_2$ sont indépendantes, $\mathrm{Cov}[\epsilon_1, \epsilon_2] = 0$ et l'expression précédente est réduite à :

$$\mathrm{Var}[\epsilon] = \mathrm{Var}[\epsilon_1] + \mathrm{Var}[\epsilon_2] \tag{4.68}$$

Dire que l'indépendance des erreurs est le pire des cas est équivalent à affirmer que la variance de l'erreur quand les lectures sont indépendantes a une valeur supérieure que quand les lectures ne le sont pas :

$$\begin{aligned} \mathrm{Var}[\epsilon|\epsilon_1, \epsilon_2 \text{ indep.}] &\geq \mathrm{Var}[\epsilon|\epsilon_1, \epsilon_2 \text{ dep.}] \\ \mathrm{Var}[\epsilon_1] + \mathrm{Var}[\epsilon_2] &\geq \mathrm{Var}[\epsilon_1] + \mathrm{Var}[\epsilon_2] + \mathrm{Cov}[\epsilon_1, \epsilon_2] \\ \mathrm{Cov}[\epsilon_1, \epsilon_2] &\leq 0 \end{aligned} \tag{4.69}$$

L'hypothèse que l'indépendance des erreurs est le pire des cas est équivalent à supposer que les erreurs sont indépendantes ou alors que la relation qui existe est telle que la covariance des erreurs est négative. Sachant que $\mathrm{E}[\epsilon_1] = \mathrm{E}[\epsilon_2] = 0$, $\mathrm{Cov}[\epsilon_1, \epsilon_2] = \mathrm{E}[\epsilon_1 \times \epsilon_2]$. En supposant que la distribution des erreurs est symétrique (ce qui est vrai par exemple pour les algorithmes de moyenne), $\mathrm{Cov}[\epsilon_1, \epsilon_2] \leq 0 \Rightarrow \mathbf{P}[\epsilon_1\epsilon_2 < 0] > \mathbf{P}[\epsilon_1\epsilon_2 > 0]$, c'est-à-dire, il est plus probable que les erreurs soient de signes contraires que du même signe.

L'hypothèse de l'indépendance comme le pire des cas signifie que les erreurs sont indépendantes ou alors que les causes pour l'augmentation de l'erreur dans le calcul d'une valeur provoquent la diminution de l'erreur dans une autre valeur. Le résultat net est que la variance de la somme d'une suite de valeurs qu'on considère comme v.a. n'excède pas la somme des variances : il y a des phénomènes qui se contrebalancent et rendent le résultat plus stable. Dans un réseau qui est encombré de messages issus des applications qui s'exécutent dans le système et où les messages de synchronisation sont retardés à cause de cet encombrement, la covariance des erreurs sera probablement positive : si l'erreur d'une lecture augmente à cause de l'encombrement, il est probable que les erreurs dans les autres sites augmentent aussi.

Pour les algorithmes d'intervalle, cette hypothèse dépend de la distribution des délais.

En conclusion, l'hypothèse de l'indépendance des erreurs est valable dans certains cas particuliers. Avant d'utiliser les résultats issus de cette hypothèse, il faudra bien vérifier si, dans le système considéré, les conditions qui la valident sont présentes.

### Existence d'une borne $\epsilon^{\mathrm{tol}}$

Nous considérons deux cas : une structure Maître-Esclave et une structure distribuée. Dans la structure Maître-Esclave chaque esclave estime l'horloge du Maître et utilise cette estimation pour ajuster son horloge. Pour que les horloges restent synchronisées avec une précision $\delta^{\mathrm{spec}}$, la précision immédiatement après l'exécution de l'algorithme, $\pi_{\mathrm{MS}}$, doit être telle que : $\pi_{\mathrm{MS}} + 2\rho R \leq \delta^{\mathrm{spec}}$ ce que conduit à $\pi_{\mathrm{MS}} \leq \delta^{\mathrm{spec}} - 2\rho R$. Si chaque site commet une erreur $\epsilon^{\mathrm{tol}}$ dans la valeur estimée, $\pi_{\mathrm{MS}} \leq 2\epsilon^{\mathrm{tol}}$. Alors, $\epsilon^{\mathrm{tol}} \leq \delta^{\mathrm{spec}}/2 - \rho R$ garantit que la précision est égale ou inférieure à $\delta^{\mathrm{spec}}$.

Pour les structures distribuées, chaque site estime l'horloge de tous les autres sites et calcule une nouvelle correction pour son horloge. Ceci est fait à travers la Fonction de Convergence (FdC). Pour les FdC moyenne, médiane et point-centrale (« mid-point »), la précision après synchronisation se présente sous la forme $\pi_{\mathrm{dist}} = \varepsilon + f_{\mathrm{tf}}(k)$ ([Schneider, 1987]) où $k$ est le degré de tolérance aux fautes (le nombre de fautes tolérées), $f_{\mathrm{tf}}$ un terme qui exprime comment la précision se dégrade avec la tolérance aux fautes et $\varepsilon$ est la plus grande différence entre deux valeurs correspondantes en deux sites différents. Dans notre cas, $\varepsilon = 2\epsilon^{\mathrm{tol}}$.

Si on veut garantir une précision $\delta^{\text{spec}}$ a tout l'instant, il faut que $\pi_{\text{dist}} + 2\rho R \leq \delta^{\text{spec}}$ ou $\epsilon^{\text{tol}} \leq \delta^{\text{spec}}/2 - \rho R - f_{\text{tf}}(k)/2$.

### Existence de $f^{\text{err}}$

Dans la section 4.3 nous avons déjà calculé la probabilité $\mathbf{P}\left[\epsilon(r) \leq \epsilon^{\max}\right]$ pour l'ensemble des algorithmes trouvés dans la littérature (tableau 4.5).

Les algorithmes de moyenne utilisent tous le même principe général. Du point de vue statistique, la valeur calculée, qu'on appellera $W$, est la moyenne d'une suite de $s$ observations de la v.a. $X$ : $W = 1/s\Sigma_{i=1}^{s} X^{(i)}$. Cette variable $X$ est toujours associée au délai du message. La taille de l'échantillon dépend du nombre de messages envoyés dans chaque étape : $s = Cr$, où $C$ est le *coefficient de coopération* qui dépend de l'algorithme. Dans les algorithmes recensés dans la littérature, $C = n$ pour l'algorithme de Le Lann et $C = 1$ pour tous les autres[2]. L'erreur est l'écart entre la valeur calculée pour $W$ et son espérance mathématique, $\mathrm{E}[W]$ (le délai moyen pour EHD, 0 pour PEC), et la probabilité que l'erreur dépasse une certaine valeur est fonction de la variance de l'erreur : $\mathrm{Var}[\epsilon] = \mathrm{Var}[W] = \mathrm{Var}[X]/Cr$. La variance de $X$ est la variance du délai associé à l'algorithme. Pour les algorithmes qui utilisent EHD, le délai est le délai de bout en bout, $\Gamma_{\text{bb}} = \Gamma_{\text{prep}} + \Gamma_{\text{acc}} + \Gamma_{\text{prop}} + \Gamma_{\text{rec}}$, selon la décomposition du délai présenté dans la section 4.3.1. Chacun de ces termes peut être considéré comme une variable aléatoire indépendante. La variance de $\Gamma_{\text{bb}}$ est alors la somme de la variance de tous les termes qui la composent. Si $\Gamma_{\text{bb}}$ est le délai de traversée d'une liaison et $D$ la borne maximale pour le nombre de liaisons traversées, alors $\mathrm{Var}[X] \leq D\mathrm{Var}[\Gamma_{\text{bb}}] \triangleq \sigma_{\text{bb}}^2$.

L'erreur dans les algorithmes du type PEC dépend de la différence entre les instants d'arrivée dans un réseau à diffusion. Dans ce cas, la variance de $X$ dépend seulement des termes $\Gamma_{\text{prop}}$ et $\Gamma_{\text{rec}}$ : $\mathrm{Var}[X] = \mathrm{Var}[\Gamma_{\text{prop}}] + \mathrm{Var}[\Gamma_{\text{rec}}] \triangleq \sigma_{\text{diff}}^2$.

L'erreur du résultat dépend de la variance de $W$, $\mathrm{Var}[W] = \sigma_W^2 = \mathrm{Var}[X]/Cr \leq D\sigma_{\text{alg}}^2/Cr$, où $\sigma_{\text{alg}}$ est la variance du délai associé à la méthode ($\sigma_{\text{bb}}$ ou $\sigma_{\text{diff}}$). Nous avons alors :

$$\begin{aligned}\mathbf{P}\left[|W - \mathrm{E}[W]| > \epsilon^{\text{tol}}\right] &= 1 - \left[\Phi\left(\frac{\epsilon^{\text{tol}}}{\sigma_W}\right) - \Phi\left(-\frac{\epsilon^{\text{tol}}}{\sigma_W}\right)\right] \\ &\leq 1 - \left[\Phi\left(\sqrt{\frac{rC}{D}}\frac{\epsilon^{\text{tol}}}{\sigma_{\text{alg}}}\right) - \Phi\left(-\sqrt{\frac{rC}{D}}\frac{\epsilon^{\text{tol}}}{\sigma_{\text{alg}}}\right)\right] \qquad (4.70)\end{aligned}$$

Pour les algorithmes de régression et de moyenne, l'expression de l'erreur est en termes de $\Phi(x)$, la fonction de distribution d'une v.a. gaussienne normalisée (moyenne nulle et variance unitaire). Cette expression est de traitement analytique difficile ; en particulier, aucune méthode pour l'inversion de cette fonction n'est connue, ce qui rend impossible, par exemple, d'obtenir une expression pour $r$ en fonction des autres paramètres. Quand une telle approche est nécessaire, et car nous sommes intéressés à un majorant pour la probabilité d'erreur, nous pouvons utiliser le lemme suivant ([Parzen, 1960]):

2. Le fait que $C = 1$ pour tous les autres algorithmes n'est pas une contrainte fondamentale. Ceci est vrai pour les algorithmes tels qu'ils ont été publiés. Par exemple, pour l'algorithme d'Arvind, c'est facile de l'adapter pour $C = n - 1$ : il suffit que la valeur calculée soit la moyenne de toutes les valeurs reçues de tous les autres sites.

**Lemme 4.4**

$$1 - [\Phi(x) - \Phi(-x)] \leq \sqrt{\frac{2}{\pi}} \frac{1}{x} \exp\left(-\frac{x^2}{2}\right)$$

Le lemme 4.4 permet, si on le désire, de traiter analytiquement la probabilité d'erreur. Remarquons que l'inversion de cette expression pour obtenir une expression en $r$ est encore difficile car on obtient pour la probabilité de l'erreur des expressions du type $A\sqrt{r}\exp(B/r)$.

Pour les algorithmes d'intervalle, il faudra utiliser l'expression du tableau 4.5, sauf pour des cas particuliers où une expression pour la probabilité d'erreur plus facile à traiter existe.

## 4.5 Analyse de la performance

La mise en œuvre d'un ASND impose un compromis entre des critères opposés. On cherche à avoir une erreur très petite avec une certaine probabilité de succès (en général très proche de 1). Ceci peut être garanti avec un nombre suffisant de messages. Mais, d'autre part, on ne peut pas accepter que le nombre de messages augmente sans limite, car on risquerait d'encombrer le réseau seulement avec les messages de synchronisation.

Le résultat du théorème 4.1 nous permet d'établir la relation asymptotique entre le nombre de messages $r$ et les différents paramètres dont il dépend. Nous présentons cette analyse pour deux cas particuliers : quand l'erreur tolérée, $\epsilon^{\text{tol}}$, tend vers zéro, et quand le nombre de sites $n$ augmente.

### 4.5.1 Variation du nombre de messages avec l'erreur tolérée

La réduction de la valeur maximale pour l'erreur, $\epsilon^{\text{tol}}$, est un facteur qui provoque l'augmentation du nombre de messages, $r$. Nous allons ensuite étudier la variation de $r$ quand $\epsilon^{\text{tol}}$ est réduite et la probabilité d'erreur reste constante : $r(\epsilon^{\text{tol}})$ (Tableau 4.7).

| Stratégie | $r(\epsilon^{\text{tol}})$ |
|---|---|
| Régression | $O\left(\frac{1}{\epsilon^{\text{tol}^2}}\right)$ |
| Intervalle | $O\left(\frac{1}{F_{\Gamma_\Sigma(2)}(2(\epsilon^{\text{tol}}+\Gamma_{\min}))}\right)$ |
| Moyenne | $O\left(\frac{1}{\epsilon^{\text{tol}^2}}\right)$ |

TAB. 4.7: *Variation de $r$ avec $\epsilon^{\text{tol}}$.*

Pour tous les algorithmes qui utilisent des méthodes statistiques (régression et moyenne), le nombre de messages croit avec $\frac{1}{\epsilon^{\text{tol}^2}}$ quand $\epsilon^{\text{tol}} \to 0$.

Pour les algorithmes d'intervalle de probabilité, la variation de $r$ avec $\epsilon^{\text{tol}}$ dépend de la distribution des délais. Si la fonction de distribution des délais augmente très lentement dans le voisinage de $\Gamma_{\min}$, la probabilité $F_{\Gamma_\Sigma(2)}(2(\epsilon^{\text{tol}} + \Gamma_{\min}))$ reste petite et $r$ atteint des valeur très élevées. Si cette probabilité croît rapidement, la valeur de $r$ pour des petites valeurs de $\epsilon^{\text{tol}}$ diminue.

### 4.5.2 Variation du nombre de messages avec le nombre de sites

Le résultat présenté pour la condition de garantie de la synchronisation nous permet aussi d'obtenir des estimations pour la valeur de $r$, le nombre de messages envoyés dans chaque étape, qui est suffisant pour garantir que le système se synchronise avec la précision et la probabilité spécifiées. Dans cette section, nous présenterons des exemples pour les algorithmes de moyenne et régression. La méthode de calcul est identique pour les autres algorithmes ; l'exemple est présenté avec les algorithmes de moyenne et régression car ceux-ci admettent une fonction d'erreur analytique commune.

Le lemme 4.4 fournit une expression analytique pour la fonction d'erreur des algorithmes dont l'erreur peut être considérée comme une v.a. normale :

$$f^{\text{err}}(\epsilon^{\text{tol}}, r) = \sqrt{\frac{2D}{\pi r C}} \frac{\epsilon^{\text{tol}}}{\sigma_{\text{alg}}} \exp\left(-\frac{rC}{2D} \frac{\epsilon^{\text{tol}^2}}{\sigma_{\text{alg}}^2}\right) \tag{4.71}$$

Une condition suffisante pour la garantie de la synchronisation selon le théorème 4.1 et basée sur (4.71) est :

$$r \geq \frac{2D}{C} \frac{\sigma_{\text{alg}}^2}{\epsilon^{\text{tol}^2}} \ln\left(\sqrt{\frac{2D}{\pi C}} \frac{\epsilon^{\text{tol}}}{\sigma_{\text{alg}}} \frac{n_p n_e n_c}{1 - \text{P}_{\text{spec}}}\right) \tag{4.72}$$

L'équation (4.72) permet, par exemple, de trouver l'ordre de la variation de $r$ avec $n$ quand $n$ augmente de façon que la probabilité de réussite de la synchronisation ne diminue pas, $r(n)$. Ceci est présenté dans le tableau 4.8 ; dans ce tableau, nous présentons aussi les valeurs de $C$ et $D$ pour chacun des algorithmes.

| Algorithme | $C$ | $D$ | $r(n)$ |
|---|---|---|---|
| Duda | 1 | 1 | $O(\ln(n))$ |
| Arvind | 1 | 1 | $O(\ln(n))$ |
| OS(int) | 1 | $n$ | $O(n \ln(n))$ |
| Rang+Tr | 1 | $\ln(n)$ | $O(\ln^2(n))$ |
| Le Lann | $n$ | 1 | $O(\ln(n)/n)$ |

TAB. 4.8: *Variation de $r$ avec $n$.*

Les formules présentées peuvent être aussi utilisées pour calculer la valeur de $r$ nécessaire pour garantir la synchronisation avec une précision $\delta^{\text{spec}}$ et une probabilité de succès $\text{P}_{\text{spec}}$ dans un système avec $n$ sites.

Les graphiques des figures 4.10, 4.11 et 4.12 présentent la dépendance du nombre de messages $r$ avec la probabilité d'échec $P_{\text{fail}}$, l'erreur commise par chaque site $e$ et le nombre de sites $n$, respectivement. L'erreur est présentée comme l'erreur normalisée, définie comme le rapport entre l'erreur $\epsilon^{\text{tol}}$ et la variance du délai Var [Γ]. Cette comparaison est faite pour les algorithmes d'Arvind, de Cristian, de Le Lann, des moyennes d'Olson et Shin (OSmoy) et de Ragarajan et Tripathi (RT). Le système type, qui définit les paramètres qui ne sont pas spécifiés dans chaque cas, correspond aux valeurs $n = 8$, $e = .5$ et $P_{\text{fail}} = .005$.

Nous pouvons vérifier que les algorithmes de base (Cristian, Arvind et Le-Lann) présentent un meilleur comportement par rapport au nombre de messages $r$ que les algorithmes dérivés (des moyennes d'Olson et Shin et de Rangarajan et Tripathi). C'est patent que le paramètre

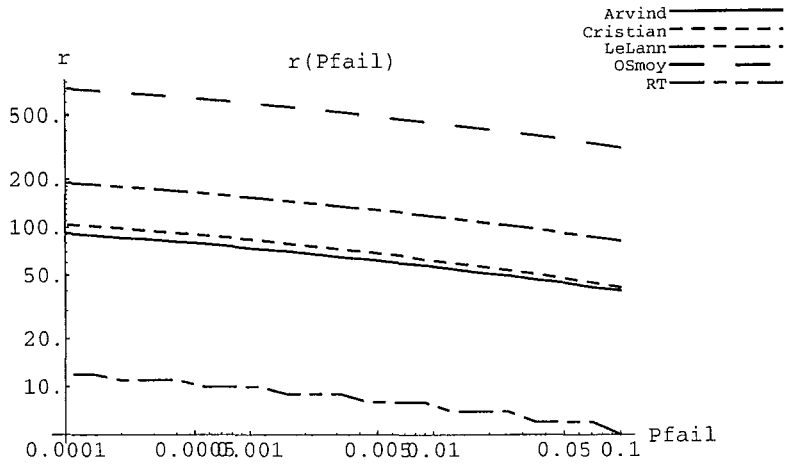



FIG. 4.10: *Dépendance de r avec* $P_{\text{fail}}$.

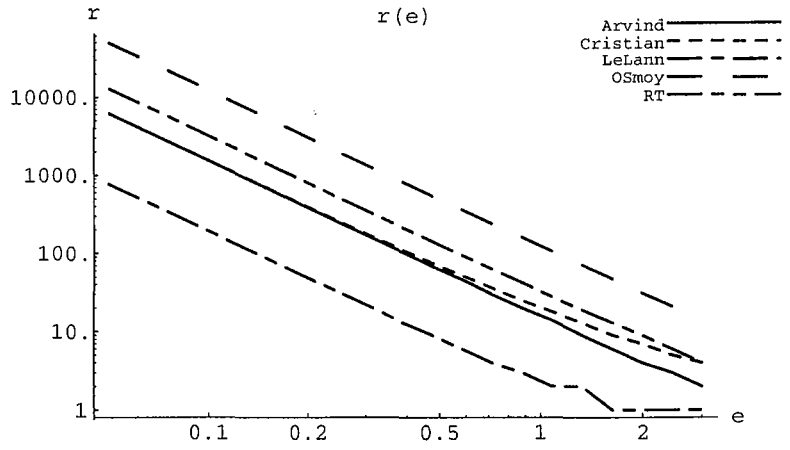



FIG. 4.11: *Dépendance de r avec e.*

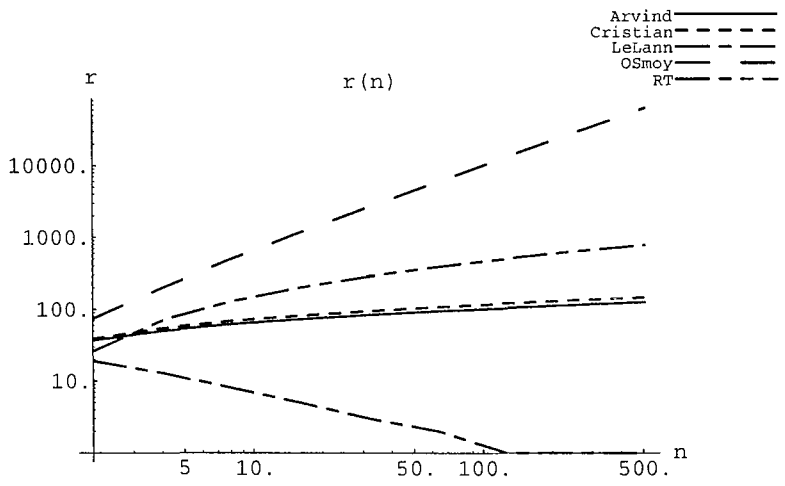



FIG. 4.12: *Dépendance de r avec n.*

qui influence de plus le nombre de messages est l'erreur normalisée $e$. Dans des cas extrêmes, $r$ peut atteindre des valeurs qui sont manifestement irréalisables (par exemple, de l'ordre des millier ou même des dizaines de millier pour $e < .1$).

Le mécanisme de diffusion dans l'algorithme de Le Lann lui apporte un avantage, et cet avantage résulte dans un résultat qui est un peu surprenant : le nombre de messages envoyés par un site peut diminuer quand le nombre de sites augmente. Nous pouvons, qualitativement, trouver des arguments pour valider ce résultat. L'écart entre les horloges synchronisées dépend de la variance du délai, $\sigma_\Gamma$. L'algorithme détermine, dans chaque site, la moyenne d'un échantillon de taille $r*n$. La variance de cet échantillon est $\sigma_\Gamma^2/(rn)$. Alors, augmenter le nombre de processeurs est équivalent à augmenter le nombre d'échantillons dans chaque site et à diminuer la valeur de sa variance. Pour un $r$ constant, l'erreur diminue. Il faut cependant ajouter que le résultat est basé sur l'hypothèse que la variance du délai, $\sigma_\Gamma^2$, ne change pas avec l'introduction de nouveaux sites dans le système.

Dans l'algorithme des moyennes d'Olson et Shin, proposé pour diminuer le nombre de messages, on remarque que la croissance de $r$ avec $n$ est supérieure à celle de l'algorithme d'Arvind original. Chaque message doit faire le tour de tous les sites : l'incertitude associée aux valeurs transportées augmente. Pour compenser cet effet, on doit augmenter le nombre de messages envoyés par un site, $r$, donc on perd l'avantage initial. La comparaison des résultats pour les algorithmes de Olson et Shin (moyennes) et de Rangarajan et Tripathi rend cet effet évident. Chaque message doit traverser $n$ liaisons (dans l'algorithme d'Olson et Shin) et, au maximum, $\ln(n)$ liaisons (dans l'algorithme de Rangarajan et Tripathi). Ceci correspond à l'excès de messages dans chacun de ces algorithmes, par rapport à celui d'Arvind. La conclusion à tirer, pour les cas de l'algorithme des moyennes d'Olson et Shin, est que celui-ci n'apporte des avantages que pour le cas où la boucle hamiltonienne est le seul arrangement pour les connexions entre les sites. Dans un cas plus général, on peut observer, pour l'algorithme d'Arvind et ses variantes (l'algorithme des intervalles d'Olson et Shin et l'algorithme de Rangarajan et Tripathi), que les améliorations proposées sont valables pour une situation très spécifique. Avant d'utiliser une telle solution, il faut toujours bien vérifier

si, pour la situation propre à son application, elle porte effectivement un avantage. Et il est même possible de voir les proposants d'un algorithme admettre, quelques années après, que leur proposition serait peut-être moins efficace qu'on le souhaiterait (*e.g.* [Olson et al., 1995] à propos de [Olson et Shin, 1991]).

## 4.6 Conclusion

Les algorithmes de synchronisation des horloges non-déterministes permettent de contourner la limitation fondamentale que subissent les algorithmes déterministes (dont la précision a une borne qui est fonction de l'incertitude du délai), et offrent la possitiblité d'atteindre une meilleure précision de synchronisation. Le prix à payer est une probabilité non-nulle que le système ne réussisse pas à se synchroniser. En principe, cette probabilité d'échec peut être aussi petite qu'on le souhaite en utilisant un nombre suffisamment grand de messages. Il y a donc trois paramètres qui déterminent la performance des algorithmes non-déterministes : la précision souhaitée, la probabilité de réussite et le nombre de messages. L'utilisation correcte de ce type d'algorithmes demande la maîtrise du rapport entre ces paramètres.

Dans ce chapitre, nous avons présenté un modèle pour l'analyse de la performance des algorithmes de synchronisation non-déterministes. Ce modèle s'applique à l'ensemble des algorithmes non-déterministes de synchronisation basés sur le concept de vague de synchronisation et établit les conditions qui garantissent que la précision souhaitée sera atteinte avec une probabilité non inférieure à la probabilité spécifiée en utilisant un certain nombre de messages. Le modèle est fondé sur trois hypothèses :

1. l'indépendance des erreurs
2. l'existence de la borne $\epsilon^{\text{tol}}$ qui garantit la précision $\delta^{\text{spec}}$
3. l'existence de la fonction $f^{\text{err}}$, le majorant pour la probabilité que l'erreur dans une étape soit supérieure à $\epsilon^{\text{tol}}$.

Pour la détermination de la fonction $f^{\text{err}}$, nous avons classifié les algorithmes parus dans la littérature selon trois stratégies fondamentales : la régression, les intervalles et la moyenne. Cette classification a permis d'établir, pour l'ensemble des algorithmes de chaque classe, le rapport entre l'erreur, le nombre de messages et la probabilité de réussite pour la communication entre un couple de sites. Une fois ces expressions déduites pour un des algorithmes, leur extension aux autres algorithmes de la même classe peut être faite par mimétisme.

Les résultats ont permis de mettre en évidence les avantages et les incovénients de chacune des stratégies identifiées et son rapport avec la caractérisation du délai comme une variable aléatoire. Cette connaissance peut être utilisée pour identifier quels sont les types d'algorithmes qui s'adaptent mieux à un certain type de réseau, en fonction de la description statistique du délai de transmission.

Les résultats ne sont utiles pour le concepteur d'un système distribué que s'ils concernent l'exécution d'un algorithme sur l'ensemble d'un système. Ceci est l'objet de la Condition de Garantie de la Synchronisation. Cette condition établit les conditions locales à un site (c'est-à-dire, les conditions observables et vérifiables sur un seul site) qui garantissent que le système se synchronise avec la précision et probabilité spécifiées.

L'utilisation de méthodes numériques a permis l'obtention de valeurs numériques pour les paramètres dont dépend la performance du système, notamment pour le nombre de messages

$r$. Les valeurs obtenues ne sont pas optimales, mais en tout cas elles indiquent que, pour des situations extrêmes, le nombre de messages nécessaires pour garantir la synchronisation peut être trop élevé pour une réalisation pratique.

Des trois hypothèses sur lesquelles le modèle se base, il ne reste qu'une qui n'a pas été, pour le moment, validée : celle de l'indépendance des erreurs. Encore, il faut vérifier si les expressions trouvées pour l'erreur tolérée et la fonction d'erreur $f^{\text{err}}$sont correctes. La vérification de la validité de ces hypothèses sera le sujet des chapitres suivants.

# Chapitre 5

# Plate-forme expérimentale

Ce chapitre décrit le système développé pour l'essai des algorithmes de synchronisation d'horloges. Ce système étant basé sur le protocole CAN, nous ferons d'abord une présentation de ce protocole et de ses caractéristiques (section 5.1). Pour réaliser un système reparti autour de ce réseau, nous pouvons disposer de nombreux circuits intégrés et de différentes façons de placer le logiciel sur chacun des sites. C'est pourquoi nous présentons dans les sections 5.2 et 5.3 les différentes solutions qui s'offraient à nous avant de détailler, dans la section 5.4 l'architecture des sites de notre plate-forme d'expérimentation conçue avec les cartes CANivete.

## 5.1 Le protocole CAN

Cette section présente le protocole CAN : son origine, ses principes de fonctionnement et ses caractéristiques principales. Ceci est fait dans le but de présenter les fondements pour mieux comprendre le fonctionnement de la plate-forme développée ; le lecteur plus intéressé par les détails de CAN peut se référer à des ouvrages comme [Bosch, 1991, Lawrenz, 1997, Paret, 1996].

Le protocole CAN trouve son origine dans l'industrie automobile. Les besoins de confort et de sécurité dans les voitures ont poussé, surtout dans les dernières années, à l'utilisation d'un nombre toujours croissant de dispositifs électriques et électroniques. Parmi ces dispositifs, nous pouvons mentionner les systèmes de climatisation, la commande électrique des vitres, des sièges, des rétroviseurs — en ce qui concerne le confort —, le système ABS et l'airbag — en ce qui concerne la sécurité — l'injection électronique et la boîte à vitesses automatique. La prise en compte de l'impact de l'utilisation de ces voitures sur l'environnement a conduit à l'usage de systèmes de contrôle de plus en plus sophistiqués — voire électroniques — de façon à réduire autant que possible (ou autant que le demandent les normes gouvernementales) l'émission de gaz et fumées polluants.

L'utilisation de ces dispositifs pose forcément des problèmes d'interconnexion. Par exemple, le système anti-dérapage pour l'accélération (ASC, *acceleration skid control*) demande la conjugaison des valeurs de la rotation du moteur et de la position du carburateur de façon à éviter que les pneus glissent au départ d'une voiture. Rassembler l'information provenant de tous les capteurs et boîtiers de commande en utilisant une paire de fils par signal est devenu impraticable. Pour résoudre ces problèmes, une solution basée sur un réseau à dif-

fusion a été proposée par Bosch ([Bosch, 1991]) : le Controller Area Network, ou CAN. Cette solution a été adoptée par des constructeurs comme Mercedes, BMW, Volkswagen, Opel et Volvo ([Tindell, 1998]) pour leurs voitures de haut de gamme. Renault, qui avait d'abord choisi VAN (un autre protocole pour l'industrie automobile) a décidé d'utiliser CAN. Nous avons trouvé aussi des références d'utilisation de CAN par Maserati et Ssangyong. Vite, CAN est devenu une norme ISO en deux variantes : la norme ISO-11519, dite de « basse vitesse », avec un débit de 125 kb/s ([ISO, 1994]) et la norme ISO-11898, de « haute vitesse », avec un débit de 1 Mb/s ([ISO, 1993]).

Les caractéristiques de CAN comme un réseau d'utilisation générique ont rapidement étendu son utilisation à d'autres domaines que les applications automobiles. Ces applications vont de la commande de procédés industriels aux équipements militaires ([Purdy, 1998]) et aux sous-marins autonomes pour la recherche ([Bradley et al., 1998]). Selon le consortium CiA (CAN in Automation), pour des applications automobiles et d'autres telles que les appareils électroménagers et la commande de procédés industriels, un total de 15 millions de circuits ont été vendus jusqu'au printemps de 1997. L'application de CAN au domaine industriel a été accompagnée par le développement d'un ensemble de spécifications pour la normalisation des couches hautes du modèle OSI, notamment pour la couche application ; voir par exemple les propositions du consortium CiA ([CAN in Automation, 1996]), de Allen Bradley pour DeviceNet ([Bradley, 1997]) et de Honeywell pour SDS (Smart Distributed System, [Honeywell, 1996]), dont une comparaison peut être trouvée dans [Etschberger, 1997]. D'autres travaux sur la comparaison des protocoles de haut niveau sont parus dans [Lennartsson, 1998, Schulze, 1998].

### 5.1.1 Description générale

Le protocole CAN est établi pour des réseaux à diffusion. La structure des messages échangés est basée sur deux champs : un champ d'identification (ID) et un champ de données (DATA) (fig. 5.1). Le champ ID identifie le *contenu* du message ; il n'y a pas d'identification


FIG. 5.1: *Structure de base d'une trame CAN*

du destinataire d'un message. Un message envoyé sur le réseau peut être reçu par n'importe quelle station : chaque station réceptrice décide si elle garde le message ou le rejette, en fonction de la valeur du champ ID. CAN suit donc un modèle Producteur/Consommateur. L'identification du message doit être unique : deux stations différentes ne peuvent pas produire des messages avec le même identificateur.

### 5.1.2 Contrôle d'accès au médium

Une des caractéristiques les plus distinctives de CAN est la gestion du droit de parole des stations. Le contrôle d'accès au médium (MAC, *medium access control*) se fait bit à bit et est basé sur le principe des bits *dominants* et *récessifs*. Dans CAN, le bit dominant a la valeur 0 et le bit récessif la valeur 1. Quand deux stations émettent en même temps deux valeurs différentes, la valeur lue sur la ligne est la valeur correspondant au bit dominant.

Ce principe signifie donc que :

- en cas de collision la valeur présente sur la ligne de transmission reste une valeur valide (elle correspond à une valeur de bit 1 ou 0) ;
- dans une collision entre bits dominants et récessifs, la ligne garde la valeur du bit dominant, quel que soit le nombre de stations qui émettent un bit récessif.

La résolution des conflits d'accès au médium est faite comme suit :

- chaque station prend l'initiative de transmettre un message si et seulement si la ligne de transmission est au repos (silence).
- les stations vérifient l'état de la ligne après l'émission de chaque bit. Si le bit lu est identique au bit envoyé, la transmission continue pour le prochain bit ; sinon, la station reconnaît qu'elle est en compétition avec une autre et renonce à la transmission.

Le champ ID est le premier champ à être transmis, précédé d'un bit de SOF (*start of frame*, « début de trame »), qui est identique pour toutes les trames. Comme l'identification des messages est unique, à la fin de la transmission du champ ID une et une seule station continue à émettre : la priorité d'un message est donc définie par la valeur du champ ID.

Regardons l'exemple de la figure 5.2. Les stations A, B et C décident d'émettre simultanément une trame. Les identificateurs des messages sont, respectivement, $100_h$ (=0010-0000000$_b$), $600_h$ (=11000000000$_b$) et $1B8_h$ (=00110111000$_b$). Chaque station envoie d'abord le bit de SOF (début de trame, dominant), suivi de l'identification respective. Les points indiquent l'instant où chaque station a perdu l'arbitrage. Ceci correspond à l'instant où la station a lu un bit dominant (0) sur la ligne alors qu'elle avait envoyé un bit récessif (1). La plus haute priorité correspond donc à la plus petite valeur de l'identificateur.

Les réseaux CAN possèdent des caractéristiques qui les rendent très intéressants pour les applications temps-réel. D'abord, comme nous l'avons vu, le message de plus haute priorité parmi celles qui essayent d'accéder au bus est envoyé sans délai additionnel. Pour les autres messages, des études sur les temps de latence des messages peuvent être trouvés dans [Rufino et Veríssimo, 1995, Tindell, 1995, Tindell et Burns, 1994a, Tindell et Burns, 1994b]). Plusieurs études ont été publiés sur le sujet l'application de CAN aux systèmes temps réel (voir par exemple [Berger et al., 1998, Bohannon et Heffernan, 1998, Gifford et al., 1998, Henderson et al., 1998, Navet et Song, 1997, Navet et Song, 1998, Song et al., 1997, Rodriguez et Campelo, 1998, Rüdiger, 1998]).

### 5.1.3 Format de trame

Le document issu de Bosch qui définit le protocole CAN ([Bosch, 1991]) décrit deux variantes pour le format de la trame. Ces deux variantes (CAN 2.0A et 2.0B) diffèrent dans la

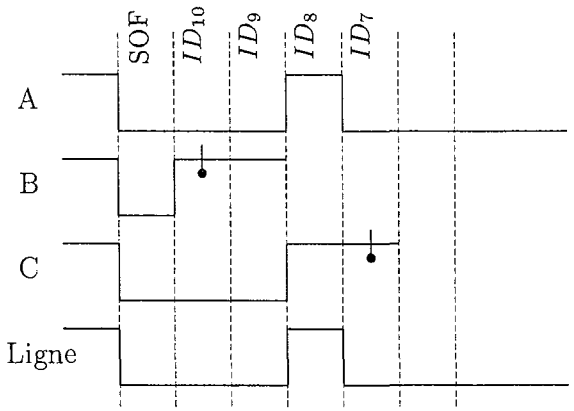



FIG. 5.2: *Arbitrage en CAN*

taille de l'identificateur : 11 bits pour CAN 2.0A (format standard) et 29 bits pour CAN 2.0B (format étendu). Le format d'une trame pour la version 2.0A est présenté dans la figure 5.3. Une trame CAN se compose des champs SOF (début de trame), ARBITRATION (arbi-

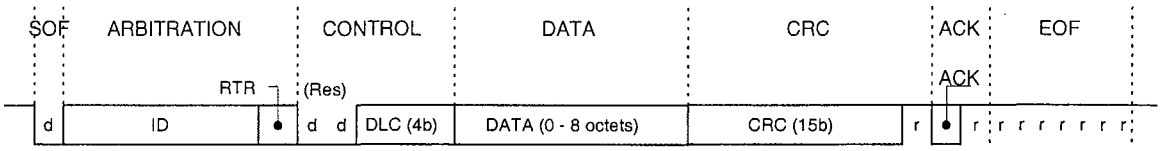



FIG. 5.3: *Format de trame CAN 2.0A*

trage), CONTROL (contrôle), DATA (données), CRC (code de redondance cyclique), ACK (acquittement) et EOF (fin de trame). Le bit SOF est toujours dominant. Le champ ARBITRATION est composé de l'ID du message (11 bits, le plus significatif est envoyé en premier) et du bit RTR (Remote Transmission Request). À la fin de la transmission de ce champ, une seule station reste en possession du droit d'accès au bus. Le champ CONTROL contient deux bits réservés (qui sont dominants pour CAN 2.0A) et le DLC (Data Length Code). Le DLC indique la taille du champ de données (le nombre d'octets envoyés dans le champ DATA) ; cette taille est comprise entre 0 et 8. Les bits réservés servent à garantir la compatibilité avec versions futures (par exemple, le premier bit réservé passe à récessif pour indiquer une trame du format étendu). Le champ DATA contient les données à transmettre (de 0 à 8 octets). Le code à redondance cyclique (CRC) est calculé à partir des valeurs des champs ARBITRATION, CONTROL et DATA, et suivi du bit CRC-delimiter, toujours récessif. Le champ ACK se compose des bits ACK slot et ACK delimiter. Toutes les stations qui ont

vérifié avec succès le CRC du message, écrivent la valeur dominante dans le bit ACK slot, ce qui va écraser la valeur récessive écrite par l'émetteur. Finalement, la trame se termine par une suite de 7 bits récessifs.

Le format des trames est choisi de façon à que les trames de la version 2.0A aient une priorité plus élevée que celles de la version 2.0B. La norme impose qu'un contrôleur CAN 2.0B puisse émettre et recevoir des trames du format standard et du format étendu. Un contrôleur de la version 2.0A émet et reçoit seulement des messages du format standard, et rejette les trames du format étendu. Le rejet est signalé en émettant une trame d'erreur qui détruit la trame originale. La transmission de trames du format étendu est alors impossible avec des contrôleurs qui implémentent la version 2.0A. Pour permettre la coexistence de stations qui travaillent en format standard avec d'autres qui travaillent avec le format étendu, certains contrôleurs implémentent le format standard et acceptent le format étendu, sans pour autant avoir la capacité de le traiter (ils acceptent tout simplement les trames du format étendu sans générer des erreurs). Ces contrôleurs sont normalement appelés contrôleurs CAN 2.0B passif.

### 5.1.4 Longueur de la ligne de transmission

La résolution des conflits d'accès au médium dans CAN demande que la même valeur du bit soit présente en même temps dans toutes les stations. Prenons le cas où une station à une extrémité du segment de la ligne de transmission vérifie que la ligne est en silence et commence à émettre une trame. Le signal émis se propage à travers la ligne (à une vitesse d'environ $2 \times 10^8$m/s pour un câble électrique). À l'autre extrémité de la ligne, une autre station, un « pico-instant » avant l'arrivée du signal de la première station, voit la ligne en silence et décide d'émettre aussi. Le signal émis par la deuxième station entre en collision avec le premier signal et le signal résultant se propage maintenant vers la première station. Le résultat de la collision (qui détermine la station qui aura droit à continuer à émettre) doit arriver à la première station avant la fin de la durée du bit. La longueur maximale pour un segment de ligne dans un réseau CAN est alors inférieure à la moitié de la longueur d'un bit (la « longueur d'un bit » est $\tau \times v$, où $\tau$ est la durée du bit et $v$ la vélocité de propagation). Il faut aussi considérer l'effet des délais dans le traitement des signaux au niveau des récepteurs et des émetteurs, ce qui réduit encore la longueur maximale. Le tableau 5.1 présente quelques valeurs pour la longueur maximale d'un réseau (dans le cas d'un contrôleur 80C200, de Philips).

## 5.2 Réalisation d'un réseau CAN

Le concepteur d'un système basé sur un réseau CAN trouve à sa disposition quatre types de circuits intégrés :

- les gestionnaires de protocole ;
- les contrôleurs à gestionnaire intégré ;
- les interfaces de ligne (les *drivers*) ;
- les unités simples d'entrée-sortie (SLIO, ou *serial link input-output*).

| Débit | Longueur max. |
|---|---|
| 1,6 Mb/s | 10 m |
| 1 Mb/s | 40 m |
| 500 kb/s | 130 m |
| 250 kb/s | 270 m |
| 125 kb/s | 530 m |
| 100 kb/s | 620 m |
| 50 kb/s | 1,3 km |
| 20 kb/s | 3,3 km |
| 10 kb/s | 6,7 km |
| 5 kb/s | 10 km |

TAB. 5.1: *Longueur maximale d'un réseau CAN ([Paret, 1996])*

### 5.2.1 Les gestionnaires de protocole

Les gestionnaires de protocole sont le « cœur » d'une réalisation d'un réseau CAN. Leur usage est présenté dans la figure 5.4.

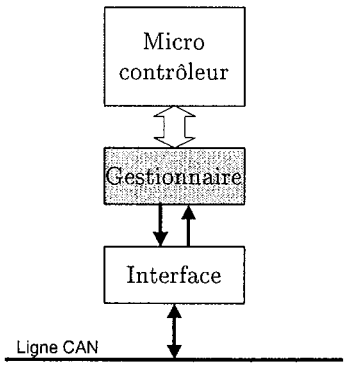



FIG. 5.4: *Gestionnaire de protocole CAN.*

Le gestionnaire fait l'interface entre le micro-contrôleur ou microprocesseur et le réseau. Il est normalement installé comme un périphérique de l'unité centrale et il est chargé de réaliser toutes les actions spécifiées dans la norme CAN. Les trames CAN qui arrivent ou partent sont codées comme une suite de bits dont les niveaux de tension sont ceux de la famille du circuit (typiquement, des valeurs de tension de 0V pour le niveau 0 et 3 à 5V pour le niveau 1). Le gestionnaire reçoit les trames, vérifie leur validité, génère les acquittements et décode la trame, qui est stockée dans une mémoire accédée par l'unité centrale. Inversement pour l'envoi, le gestionnaire prend un message stocké en mémoire, crée la trame respective et procède à son envoi. Du point de vue du modèle OSI, il est responsable de la couche 2 (liaison de données) et de la sous-couche PLS (Physical Signaling) de la couche 1.

Les gestionnaires CAN proposent un système de filtrage des messages. Ce système permet de soulager l'unité centrale, puisqu'elle n'a pas besoin de traiter tous les messages qui circulent

dans le réseau, mais seulement un sous-ensemble défini au préalable. Quelques gestionnaires offrent des capacités sophistiquées pour le filtrage des messages qui arrivent, notamment un stockage basé sur objets-message ("*message objects*", [Intel Co., 1995]). D'autres n'offrent qu'un système très simple, basé sur des masques: si les 8 bits les plus significatifs sont conformes aux bits définis dans le masque, le message reçu est accepté, sinon le message est perdu. Ces derniers gestionnaires sont souvent appelés *BasicCAN* et les premiers *FullCAN*. Il faut remarquer trois aspects à propos de cette nomenclature. D'abord elle n'est définie nulle part ; elle fait partie du jargon de la communauté CAN et elle est d'une certaine façon utile pour faire une classification des gestionnaires disponibles. Deuxièmement, un gestionnaire BasicCAN réalise le protocole CAN dans sa totalité ; ses limitations sont au niveau du traitement des messages « du côté du contrôleur ». Finalement, la classification BasicCAN et FullCAN ne présente que deux cas typiques, et toute variante est possible entre les deux. Les contrôleurs BasicCAN présentent le problème d'une charge plus élevée pour l'unité centrale, par contre leur coût est plus faible que celui des gestionnaires FullCAN qui demandent une surface de silicium plus grande. Le circuit 80C200 de Philips, est un exemple d'un gestionnaire BasicCAN ([Philips, 1997]). Comme exemple de gestionnaires proposés par d'autres fabricants, nous pouvons citer le circuit 82527 de Intel ([Intel Co., 1995]), ou le 81C92 de Siemens.

### 5.2.2 Les contrôleurs à gestionnaire intégré

Dans ce cas, l'unité centrale et le gestionnaire CAN sont rassemblés dans un seul circuit intégré. Les avantages de cette solution sont :

- la réduction de la surface des circuits intégrés et du nombre de ces circuits en chaque site ;

- le transfert plus rapide de données entre l'unité centrale et le gestionnaire.

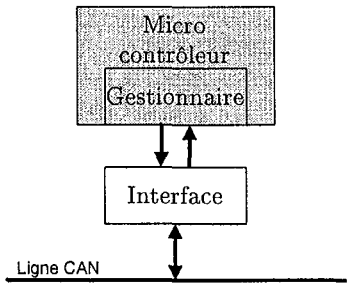



FIG. 5.5: *Contrôleur à gestionnaire intégré*

Le contrôleur 80592, de Philips, est une variante du micro-contrôleur 80C51 qui réunit le contrôleur et le gestionnaire 80C200 dans un même circuit intégré. D'autres contrôleurs sont le 87C196 de Intel, le COP684/884 de National Semiconductors et le C505 de Siemens.

### 5.2.3 Les interfaces de ligne

Ces circuits (connus aussi sous le terme anglais *line-drivers*) sont responsables pour la conversion entre les niveaux de tension utilisés par le gestionnaire (en général, 0 et 5V pour les bits 0 et 1) et les signaux qui circulent sur le réseau (tensions, lumière, ...).

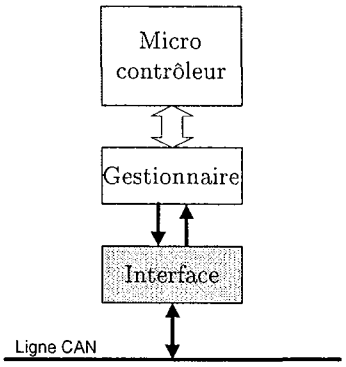



FIG. 5.6: *Circuit d'interface de ligne*

La figure 5.7 présente un des schémas les plus simples pour faire l'interface entre un gestionnaire 80C200 (ou un contrôleur 80C592) et la ligne CAN. Ce type d'interface a l'avantage d'être simple, mais il convient seulement pour des applications qui ne sont pas trop exigeantes en termes de sensibilité aux interférences électromagnétiques.

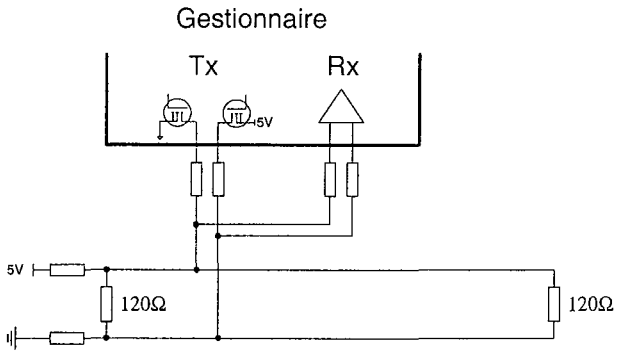



FIG. 5.7: *Interface de ligne simple*

Pour des applications plus exigeantes et plus performantes, et pour garantir le respect des normes de la couche physique, notamment en ce qui concerne les niveaux de tension, on peut utiliser des circuits spécifiques, tels que le circuit PCA82C250 de Philips.

### 5.2.4 Les unités simples d'entrée-sortie

Dans un système distribué, on trouve parfois des noeuds où les exigences de calcul sont inexistantes ou très réduites, et où la seule fonction est l'interface avec le monde extérieur : la lecture de valeurs (de capteurs, ...) ou l'envoi d'une commande (allumer ou éteindre une lampe, régler la vitesse d'un moteur, ...). Équiper ces noeuds d'un micro-contrôleur pour faire l'interface avec le reste du système est une solution qui gaspille des ressources, notamment la capacité de traitement de l'unité centrale, qui n'est utilisée que pour faire la commande du gestionnaire. Quelques fabricants proposent des circuits à utiliser dans de telles situations : ce

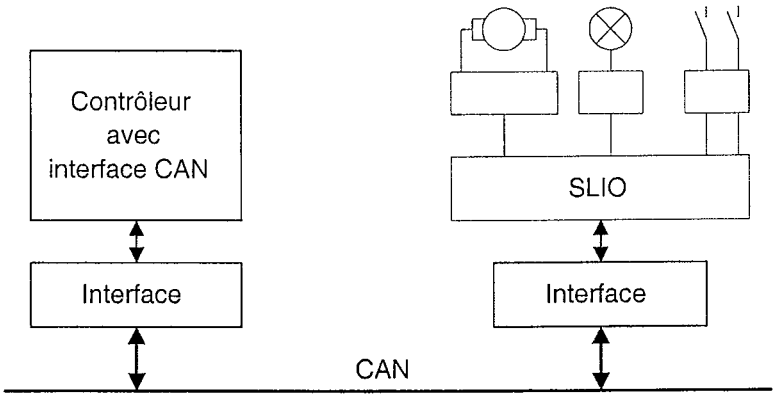



FIG. 5.8: *Exemple d'application avec un circuit SLIO*

sont les unités simples d'entrée-sortie ou SLIO (*serial link input-output*, fig. 5.8). Ces unités disposent d'un ensemble d'entrées et sorties (analogiques et numériques) et chaque SLIO a une adresse de base. Cette adresse (programmable) définit le champ ID des messages que l'unité va traiter. Le nombre maximal de circuits SLIO présents dans un réseau CAN est limité par le nombre d'adresses disponibles.

Les sorties sont programmées par un message de données et les sorties sont lues avec un message de requête de données. Comme les SLIO n'ont pas de capacité de traitement, une des stations du réseau CAN est chargée de leur programmation. Le circuit P82C150 de Philips est un exemple de SLIO.

## 5.3 Développement de systèmes embarqués répartis

Le développement de systèmes embarqués présente quelques caractéristiques distinctives par rapport au développement des systèmes informatiques plus traditionnels. Les systèmes embarqués disposent de moyens limités d'interface avec l'utilisateur : les claviers sont rares et les consoles, quand elles existent, sont, pour la plupart des cas, capables de présenter seulement quelques lignes de caractères ASCII. Développer un programme en utilisant toujours le système embarqué pour faire l'édition des fichiers source et leur compilation serait une tâche vraiment pénible. C'est donc naturel d'utiliser une autre machine, plus puissante du point de vue d'interface avec l'utilisateur, pour faire le développement du logiciel et sa compilation. Normalement, on utilise un ordinateur personnel du type PC ou une station de travail pour

éditer les fichiers source (en langage C, par exemple) et pour les compiler pour obtenir un fichier exécutable par l'unité centrale du système embarqué.

### 5.3.1 Solutions pour la programmation de la cible

L'étape qui suit la compilation d'un programme est son transfert de l'ordinateur où il a été généré (le « hôte ») vers celui qui va l'exécuter (la « cible »). Il y a plusieurs solutions pour le faire ; nous ferons rapidement une revue des options principales. Celles-ci sont ([Fonseca et al., 1998b]) :

- la programmation d'EPROMs ;
- l'émulation de l'EPROM ;
- l'émulation du contrôleur (ICE, ou *in-circuit emulator*)
- les moniteurs ;
- le télé-chargement de code.

#### La programmation d'EPROMs

C'est l'option la plus immédiate et la plus proche de la façon dont le système va fonctionner ; elle demande l'existence de l'équipement nécessaire pour programmer la mémoire. Son utilisation est possible avec n'importe quelle unité centrale qui utilise une mémoire de programme extérieure ; des unités centrales avec mémoire de programme intégrée pourront demander l'acquisition de l'équipement de programmation spécifique. Elle peut devenir fastidieuse et monotone car toute modification du programme demande le remplacement d'un circuit intégré, son effacement et sa reprogrammation. Les facilités pour tester le circuit sont minimales ; l'utilisation d'un analysateur logique est possible mais pas trop intéressante du point de vue de la rentabilité.

#### L'émulation de l'EPROM

L'émulation de l'EPROM consiste à remplacer le circuit intégré de la mémoire de programme par un circuit électronique connecté au système hôte. Le code du programme à exécuter est télé-chargé dans le simulateur, qui se comporte — pour le système cible — de la même façon que la mémoire EPROM avec le programme.

C'est une solution destinée aux unités centrales qui lisent le programme d'une mémoire externe. Pour des systèmes très rapides et performants, elle peut présenter des problèmes de temps d'accès (délai dans la présentation des valeurs pendant l'accès à la mémoire simulée). Elle demande un circuit simulateur pour chaque système cible. Les facilités de test sont identiques à celles de la programmation de l'EPROM, mais les modifications dans le programme sont beaucoup plus faciles.

#### L'émulation du contrôleur

Dans ce cas, l'unité centrale du système cible est remplacée par un circuit connecté au système hôte, le simulateur du contrôleur ou ICE (*in-circuit emulator*). Ce simulateur reproduit, en temps réel, les actions qui seraient prises par l'unité centrale si elle était branchée

sur le circuit. Les capacités de test peuvent être très puissantes, puisque l'hôte peut avoir accès à toutes les actions de l'unité centrale simulée. Les simulateurs sont normalement des équipements chers, surtout quand ils sont capables de simuler plusieurs types d'unité centrale.

#### Les moniteurs

Un moniteur est un petit programme qui permet au système hôte de communiquer avec le système cible. À travers cette communication, l'utilisateur est capable de télé-charger des programmes, de vérifier et déboguer le code, de démarrer et d'arrêter son exécution et même d'avoir accès au contenu de la mémoire et des registres internes de l'unité centrale. Le système cible doit être équipé d'une ligne pour communiquer avec le système hôte. Cette solution peut fournir un très bon niveau d'inspection de l'unité centrale à un prix raisonnable. La portabilité du programme moniteur est très réduite, car il demande l'accès aux registres internes de l'unité centrale (pour démarrer et arrêter l'exécution, pour l'exécution pas-à-pas).

#### Le télé-chargement de code

De même que pour les moniteurs, le télé-chargement de code est basé sur un petit programme qui réside dans la cible, dont la fonction principale est de recevoir le code envoyé par l'hôte, le stocker en mémoire et démarrer son exécution.

Contrairement au moniteur, le télé-chargeur ne présente pas de fonctionalités comme la consultation du contenu des registres internes et de la mémoire ou l'éxécution du programme en pas-à-pas. Parfois, le télé-chargeur peut même être retiré de la mémoire aprés le chargement ; dans ce cas, le flux d'exécution ne retourne jamais au télé-chargeur. Au contraire, les moniteurs sont toujours présents en mémoire et l'éxécution retourne toujours au moniteur après que le programme de l'utilisateur se termine.

### 5.3.2 Une solution pour les systèmes répartis

Pour un système embarqué qui est aussi un système distribué, les solutions qui sont normalement utilisées pour les systèmes mono-site ne sont pas toujours adéquates. Les principales raisons sont :

- la multiplicité des sites équipés d'un micro-contrôleur ou microprocesseurs ;
- leur distribution physique.

Le premier aspect concerne le nombre de sites où il faut installer le programme à exécuter (soit en installant une ROM avec le programme, soit en connectant l'hôte aux cibles). La plupart des méthodes présentées sont inadéquates dans cette situation. La gestion des EPROMs avec les nouvelles versions d'un logiciel en phase de développement, qui changent continuellement, est déjà difficile pour un seul site ; pour une multiplicité de sites, les difficultés augmentent. Les simulateurs (de mémoire ROM et d'unité centrale) deviennent des solutions très chères : il faut en installer un dans chaque site du système distribué. Il reste encore le problème de connecter plusieurs simulateurs à un seul hôte (tous les simulateurs ne le permettent pas).

La distribution physique des sites peut rendre les choses encore plus difficiles. Il faut avoir accès en permanence aux sites du système distribué pour y installer les EPROMs, si on utilise

la première méthode présentée. Pour les simulateurs il faut avoir un moyen de communication (un câble, un faisceau hertzien, ...) entre l'hôte et les cibles.

Il faut donc trouver une solution adéquate aux situations de travail avec un système distribué. Les caractéristiques que nous cherchons pour cette solution sont :

1. le transfert du code d'un seul hôte vers de multiples systèmes cibles ;

2. la possibilité d'avoir des systèmes cibles dispersés (et non concentrés près de l'hôte).

D'autres caractéristiques sont aussi souhaitables : le test des systèmes cibles (surtout pour des applications avec une grande dispersion physique ou où les cibles sont inaccessibles), le démarrage et l'arrêt des programmes contrôlés par l'hôte, ...

Puisque le système distribué est bâti sur un réseau, une solution possible est l'utilisation de ce réseau pour faire la communication entre l'hôte et les systèmes cible. Les avantages sont les suivantes:

- c'est une solution qui ne demande pas d'équipement additionnel à brancher sur le système ;

- la communication entre l'hôte et les sites distants est faite par une infrastructure (le réseau) déjà existante.

Ceci nous conduit à des solutions basées sur le télé-chargement du code à exécuter : les moniteurs et les chargeurs de code. Nous avons développé un système basé sur ce type de solution, fonctionnant sur un réseau CAN ([Fonseca et al., 1998d, Fonseca et al., 1998a]) et utilisant le principe du télé-chargeur de code. Les avantages prévues dans cette solution ont été :

- la possibilité d'avoir un système qui exécute du code identique à celui qui serait programmé dans une mémoire (le code pour les systèmes à base d'un moniteur doit être toujours décalé dans la mémoire pour laisser de la place au programme moniteur qui reste toujours présent).

- la simplicité du développement (les moniteurs sont des programmes plus complexes que les télé-chargeurs de code).

Avoir un système qui exécute du code identique à celui que serait enregistré dans la mémoire ROM d'un système autonome (les anglophones utilisent l'expression *prommable code*) permet de réduire le temps et l'effort nécéssaires pour passer d'un système de test (tel que le système basé sur le télé-chargeur de code) à un système autonome. Si le code peut être enregistré dans une mémoire ROM directement, il suffit de remplacer la mémoire non-volatile (ROM, PROM ou EPROM) qui contient le programme télé-chargeur par une autre, avec le programme de l'utilisateur. Dans le cas des systèmes à base d'un moniteur, celui est toujours présent et le programme de l'utilisateur s'exécute au dessus du programme moniteur. Le même programme de l'utilisateur doit être adapté pour s'exécuter à partir d'adresses différentes — selon il sera exécuté dans un système autonome ou dans un système à base d'un moniteur. Cette adaptation peut ne pas être toujours facile. Encore, après qu'un programme a été testé avec un moniteur, il faut aussi vérifier que continue à s'exécuter correctement, et que son exécution correcte ne dépend pas de resources qui sont fournies par le moniteur.

Le deuxième avantage prévue est la simplicité de développement. Les moniteurs sont des programmes plus complexes que les télé-chargeurs de code. Étant limités par le temps, il a fallu choisir la solution qui garantissait un système en état de marche, vérifié et débogué dans le délai le plus court possible.

## 5.4 La carte CANivete

La carte CANivete[1] (fig. 5.9) est le noyau d'un système destiné a l'aide au développement de systèmes distribués embarqués. Chaque carte CANivete est un noeud dans un réseau CAN qui est capable de recevoir le programme à exécuter à travers la connexion au réseau CAN et démarrer et arrêter son exécution. Ceci permet de connecter plusieurs systèmes distants (29 dans la version courante) à un seul hôte. La carte a deux modes : « télé-chargement » (*download*) et « exécution » (*run*). Après mise en marche, la carte est en mode télé-chargement. Dans ce mode, les sites attendent le programme à exécuter, qui sera envoyé à travers le réseau CAN. Après réception et stockage du programme dans la mémoire locale, le système se prépare pour entrer en mode exécution, ce qui aura lieu après la prochaine RAZ. Dans la

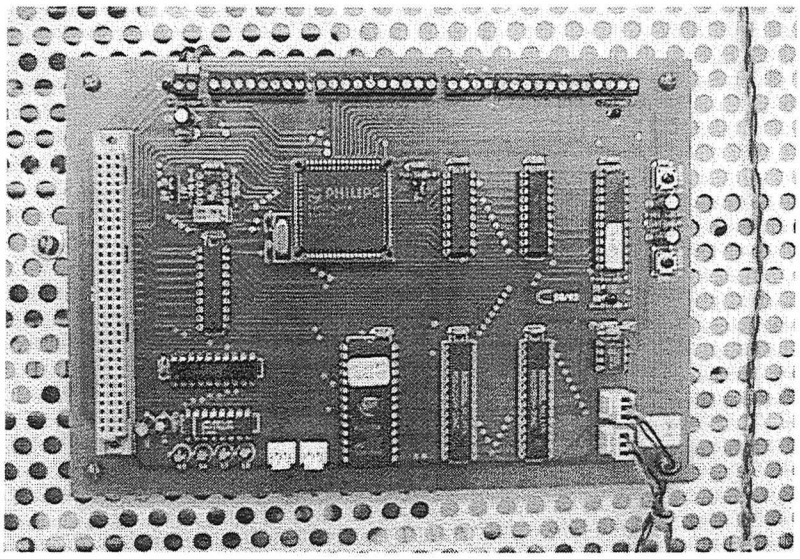



FIG. 5.9: *Carte CANivete*

suite, nous ferons une brève présentation de la carte CANivete. Une description plus détaillée peut être trouvé dans [Fonseca et al., 1998e].

### Principe de fonctionnement

Le principe de fonctionnement de la carte est présenté dans la figure 5.10. La mémoire ROM contient le programme du télé-chargeur de code, RAM P sert à stocker le programme à exécuter et RAM D est la mémoire de données pour le programme de l'utilisateur.

Quand la carte est en mode « télé-chargement », l'unité centrale (CPU) voit la ROM comme la mémoire de programme et RAM P comme mémoire de données. Après la RAZ,

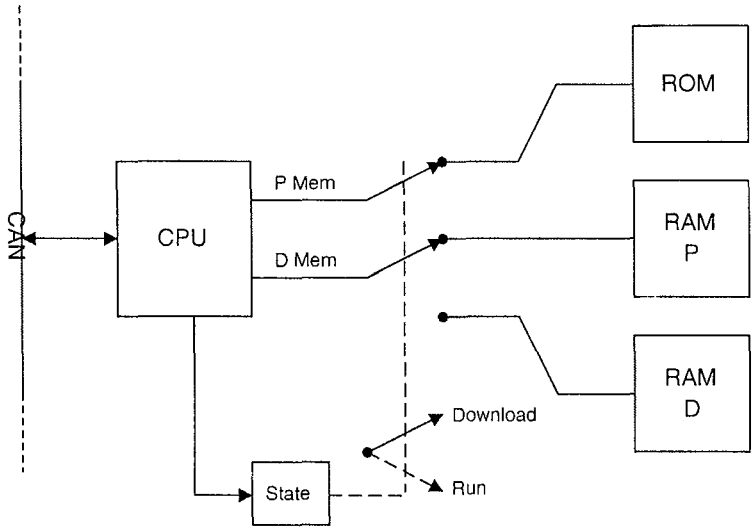



FIG. 5.10: *CANivete : principe de fonctionnement.*

l'exécution du programme stocké en ROM démarre. Ce programme consiste à surveiller le réseau CAN, en attendant l'envoi du programme. Le programme reçu est stocké dans la mémoire de données (RAM P dans ce mode). Après la réception du programme, l'unité centrale commute vers le mode « exécution ». La mémoire RAM P lui apparaît maintenant comme la mémoire de programme et l'unité centrale démarre l'exécution du programme stocké dans RAM P.

Une possible configuration pour le système est présentée dans la figure 5.11. Le PC contient le logiciel de dialogue avec l'utilisateur. Normalement, la compilation et l'assemblage du code à exécuter sont faits dans le même PC. Chaque carte porte un numéro d'identifica-

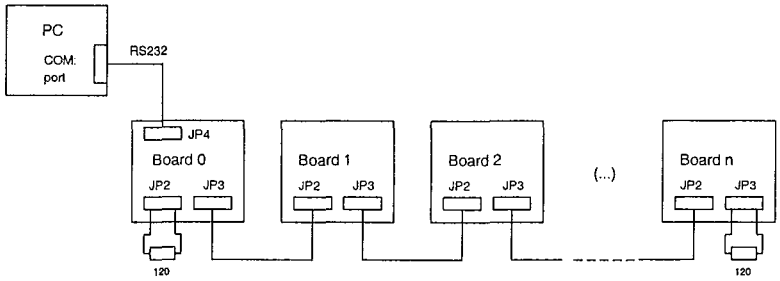



FIG. 5.11: *Configuration du système*

tion, qui doit être unique pour tout le système. La carte numéro 0 (zéro) est la carte qui est connectée à l'hôte et est en charge de la communication avec les autres cartes. Les numéros d'identification des cartes ne sont pas forcément consécutifs. Il est seulement nécessaire que la carte 0 soit branchée au PC et que tous les numéros d'identification soient uniques.

Le système utilise un sous-ensemble des identificateurs pour le télé-chargement des programmes (de $704_h$ à $7EF_h$) ; la transmission est faite à une vitesse de 128 kb/s. Le nombre maximal de stations est limité à 29. Cette limite est imposée par le format des identificateurs.

Le système minimal est composé du PC, la carte 0 et une carte distante. Un système hyper-minimal serait composé du PC et de la carte 0, mais aucune usage ne serait fait du réseau CAN.

### 5.4.1 Le matériel

La carte CANivete est un système autonome, basé sur le micro-contrôleur 80592 de Philips.

#### Gestion de mémoire

Du point de vue de l'utilisateur, le système dispose de 32k octets de mémoire de programme, et 32k octets de mémoire de données. 32k adresses dans la moitié supérieure de l'espace d'adressage sont réservées pour des dispositifs périphériques. Cet espace pour périphériques consiste en 128 blocs identiques de 256 adresses (on peut adresser au maximum 256 dispositifs périphériques différents).

| Adresses | Code | Données |
|---|---|---|
| 0000h-7FFFh | Programme (32k) | Données (32k) |
| 8000h-FFFFn | (Indisponible) | Peripheriques (256) |

TAB. 5.2: *Gestion de mémoire*

#### Interface

La carte CANivete offre un ensemble d'entrées et sorties, numériques et analogiques. Elles sont à la disposition de l'utilisateur sur plusieurs connecteurs (figure 5.12). La description détaillée de cette interface peut être trouvée dans [Fonseca et al., 1998d]. L'interface disponible dans la carte CANivete est : un port (8 entrées/sorties) numérique, des entrées/sorties à fonctions spéciales (lignes d'interruption et contrôle des compteurs), entrées/sorties analogiques et un connecteur d'expansion.

Le port P4 du 80592 est disponible dans le connecteur JP6. C'est un port bidirectionnel (entrée/sortie). Le bloc d'entrées/sorties spéciales consiste en 4 lignes d'interruption (INT2, INT3, INT4 et INT5) et deux signaux pour la commande du compteur TIMER2 (T2 et RT2), disponibles sur le connecteur JP7. Pour l'interface analogique, l'utilisateur dispose de 2 signaux PWM (*Pulse Width Modulation*) et 8 entrées de signaux analogiques. Tous ces signaux d'interface, à l'exception des entrées de tension analogiques, sont aussi présents sur le connecteur d'expansion.

#### Communication

La carte CANivete contient l'interface CAN et deux connexions RS-232. L'interface avec la ligne de transmission CAN est fait par un circuit intégré 82C250, de Philips, qui est conforme aux niveaux de tension décrits dans la norme ISO 11898 ([ISO, 1993]).

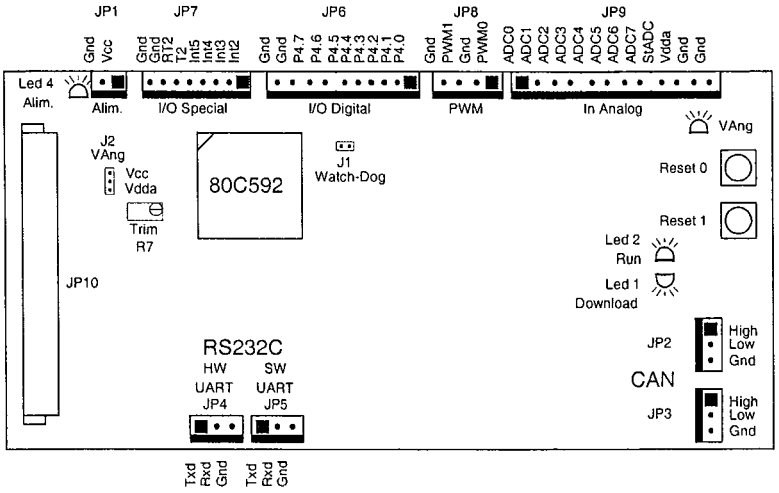



FIG. 5.12: *Interface de la carte CANivete*

### 5.4.2 Le logiciel

Le programme `PCLoad` est l'interface de l'utilisateur dans l'hôte. Dans la version actuelle, ce programme permet le télé-chargement du code vers le site distant. Les commandes pour démarrer l'exécution du code dans le site distant ne sont pas nécessaires dans cette version du système, car les noeuds se reconfigurent automatiquement pour exécuter le programme à la fin de la réception. `PCLoad` utilise un des ports de communication série du PC, et la syntaxe est la suivante:

```
PCLoad progfile Station_ID [COM_Port]
```

`progfile` est le fichier avec le code du programme à exécuter dans le site distant, dans le format Intel HEX 1. `Station_ID` est le numéro d'identification de la station réceptrice et `COM_Port` est un argument optionnel qui définit le port du PC à utiliser pour la communication.

## 5.5 Conclusion

Ce chapitre a traité le problème de la mise en œuvre de solutions logicielles dans les systèmes distribués et, en particulier, le cas des systèmes embarqués. Le protocole CAN a été présenté comme un exemple d'un protocole sur lequel on peut baser un tel système. Les problèmes spécifiques au développement des systèmes distribués embarqués ont été identifiés. Nous avons présenté une solution pour l'aide au concepteur d'un tel système. Par rapport aux solutions les plus communes, qui sont orientées vers des systèmes mono-site, notre proposition offre une interaction plus facile avec l'ensemble du système distribué.

# Chapitre 6

# Vérification expérimentale

Dans le chapitre 4, nous avons présenté un modèle pour l'estimation du nombre de messages requis pour garantir la synchronisation des horloges pendant l'exécution d'un algorithme non-déterministe, avec une précision et une probabilité données et en utilisant un certain nombre de messages.

La question de la validité des hypothèses sous-jacentes était discutée dans la suite de la présentation du modèle, mais quelques questions ont été laissées ouvertes. Dans ce chapitre, nous reprenons cette discussion, et nous essayons de vérifier la validité de ces hypothèses et du modèle. Ceci sera fait en utilisant la plate-forme basée sur le réseau CAN décrite dans le chapitre précédent.

Le modèle présente des estimations pour le nombre de messages en fonction de la variance du délai, la probabilité d'échec et du nombre de sites. En particulier, la Condition de Garantie de la Synchronisation, qui est issue de ce modèle, établit une condition suffisante qui garantit que la synchronisation réussit avec une probabilité non inférieure à une valeur spécifiée.

La section 6.1 présente les aspects d'implémentation du système qui servira de support aux travaux expérimentaux et qui est basé sur la plate-forme CAN ; en particulier, nous décrirons en détail les aspects qui concernent l'implantation d'une horloge interne et l'organisation du système pour l'essai des algorithmes de synchronisation. Les résultats expérimentaux sont décrits dans dans les sections 6.2 et 6.3. La première section s'occupe de la dépendance de la description statistique des délais par rapport à la valeur des identifieurs des messages de synchronisation et la deuxième présente les expériences qui testent la validité du modèle présenté. Nous présentons les conclusions dans la section 6.4.

Pour les essais qui seront décrits au cours de ce chapitre, il ne sera pas possible de vérifier les résultats en ce qui concerne la variation du nombre de messages avec le nombre de sites. Le nombre de sites du réseau était limité par le matériel disponible pour les essais.

## 6.1 Mise en œuvre du système

La mise en œuvre du système pour l'essai des algorithmes de synchronisation demande un ensemble de décisions dont dépend sa performance. Si, d'un côté, ces décisions ne sont pas directement liées à l'objet scientifique du travail que nous nous proposons de développer, la qualité des résultats dépendra forcement de la correction de ces choix. Dans le cas extrême, une mauvaise implantation peut engendrer des résultats qui suggèrent l'impossibilité d'une

solution qui, autrement, serait faisable.

### 6.1.1 L'horloge interne

Dans cette section, nous décrirons les choix concernant la construction de l'horloge interne de chaque nœud du système.

Le principe de fonctionnement de l'horloge synchronisée est décrit dans la figure 6.1. L'horloge synchronisée est construite à travers l'horloge physique et une correction, qui est ajoutée à la valeur de l'horloge physique pour établir la valeur de l'horloge synchronisée. Le système n'a pas de moyens d'agir sur l'horloge physique. Cette horloge est un compteur qui est incrémenté avec une fréquence $\phi$ (ou plutôt, avec une fréquence $\phi(t)$, car cette fréquence n'est pas forcément constante). La valeur de la correction à imposer à l'horloge physique est calculée par l'algorithme de synchronisation.

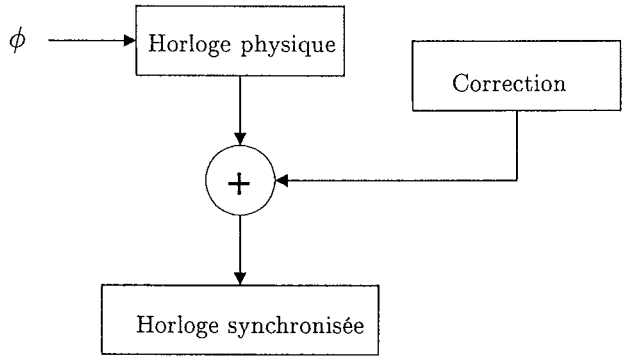



FIG. 6.1: *Principe de fonctionnement de l'horloge synchronisée*

La construction de l'horloge synchronisée commence alors par l'implantation de l'horloge physique. Pour cela, nous avons choisi d'utiliser un des compteurs internes du micro-contrôleur. Les avantages de cette solution sont:

- l'horloge physique est implantée sur le matériel (et non par le logiciel); ceci permet un vrai parallélisme entre l'actualisation de la valeur de l'horloge physique et les tâches du système;
- l'utilisation d'un compteur interne dispense l'utilisation de matériel supplémentaire; et
- le fait que le compteur est interne permet une lecture plus rapide que dans le cas de l'utilisation d'un circuit externe au micro-contrôleur.

Un autre avantage apporté par cette solution est le fait qu'elle peut être implantée pratiquement sur tous les micro-contrôleurs du marché.

Il faut ensuite choisir la fréquence nominale de l'horloge physique, $\phi_{\text{nom}}$, d'où dépend la granularité nominale $g_{\text{nom}}$. En principe, on serait intéressé par une granularité aussi petite que possible: ceci nous permettrait, par exemple, de distinguer les instants d'occurrence de deux événements distincts, qui autrement seraient estampillés avec la même valeur de l'horloge. Mais le fait que l'horloge physique a une taille (en nombre de bits) finie impose la prise en compte d'autres aspects. La taille finie du compteur de l'horloge physique fait que ce

compteur déborde après un intervalle de temps égale à $2^{n_b}g$, où $n_b$ est le nombre de bits du compteur. Le système doit être alors capable de traiter correctement les valeurs d'une horloge qui déborde, ce qui signifie une surcharge du point de vue logiciel (où alors l'introduction de matériel spécifique). Par exemple, après le débordement d'un compteur de 16 bits, la valeur "1" pour l'horloge correspond à un instant postérieur à celui de la valeur "65535", même si 65535 est supérieur à 1. Dans les horloges du "monde réel", la propriété de la monotonicité (telle qu'elle a été définie dans le chapitre 2) n'est pas toujours respectée. Plus petite est la granularité, plus fréquents sont les débordements, ce qui peut signifier un délai accru pour calculer la valeur de l'horloge et, par conséquent, des erreurs dans la valeur de l'horloge et une dégradation de la qualité du service.

Dans notre cas, nous avons décidé d'utiliser pour l'horloge physique le compteur Timer2, un compteur de 16 bits. La fréquence de comptage est définie par rapport à la fréquence de l'horloge du système, $\phi_{\text{sys}}$. Au niveau interne, cette fréquence est divisée par 12 et après par un facteur $k$, que l'utilisateur peut choisir entre 1, 2, 4 et 8. Nous avons utilisé le facteur $k = 8$, ce qui correspond à une fréquence de l'horloge $\phi_{\text{nom}} = 115200\text{Hz}$ et à une granularité $g_{\text{nom}} = 8.68\mu\text{s}$.

$$\begin{aligned} \phi_{\text{nom}} &= \phi_{\text{sys}}/(12 * k) \\ &= 11.0592 \times 10^6/(12 * 8) \\ &= 115200 \quad [\text{Hz}] \end{aligned} \tag{6.1}$$

À la fréquence nominale, le compteur Timer2 déborde tous les 568 ms, ce qui est un intervalle trop court. Un compteur auxiliaire de 16 bits, qui correspond à la variable `ClockAux`, est incrémentée à tout débordement du compteur Timer2. Ce compteur permet d'avoir une horloge physique de 32 bits, avec une période de débordement de 37 283 s = 10h 21m 23s. Une telle période sera peut-être trop courte pour une application industrielle, mais elle est suffisante pour les essais que nous souhaitons faire. La structure de l'horloge physique de chaque nœud est alors présentée dans la figure 6.2.

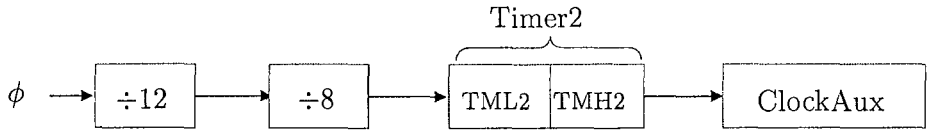



FIG. 6.2: *Structure de l'horloge physique.*

Un autre choix qui concernait aussi l'horloge est de la considérer comme un compteur de granules tout court, et de ne pas essayer de la convertir en des unités plus proches de l'usage commun (heures, minutes, secondes, dixièmes de seconde, ...). Ceci a été fait dans une première phase du travail, mais a été postérieurement abandonné. D'un côté, une telle structure n'apportait rien aux résultats obtenus (elle ne sert qu'à rendre les valeurs de l'horloge plus faciles à lire par l'Homme), toute conclusion valable pour une représentation est aussi valable pour l'autre. En outre, la gestion de cette structure représente une surcharge pour le nœud (soit pour l'affichage de la valeur, soit — et surtout — pour le calcul de différences entre deux valeurs des horloges). Par exemple, le temps pris seulement pour mettre à jour les registres d'une telle horloge correspondait à 5.6% de la charge du processeur!

### 6.1.2 Codage en langage C

Les algorithmes de synchronisation, comme tout le système, sont réalisés par des procédures écrites dans un langage de programmation, C dans notre cas. La représentation de la structure de l'horloge dans ce langage doit permettre l'exécution des tâches qui se rapportent à la gestion du temps et de l'horloge synchronisée dans un délai le plus court possible et avec la plus petite surcharge pour le processeur. Ce facteur devient critique dans le cas de la plate-forme utilisée, car celle ci est basée sur un micro-contrôleur à 8 bits, dont la puissance de calcul est assez réduite.

L'ensemble d'outils pour la programmation en C utilisé a été le ICC8051 de IAR Systems. Les structures de données de ce compilateur sont décrites dans le tableau 6.1. La taille de

| Type | Taille | $T_{\text{inc}}$ |
|---|---|---|
| `char` | 8 bits | $20\mu s$ |
| `int` | 16 bits | $28\mu s$ |
| `long` | 32 bits | $250\mu s$ |

TAB. 6.1: *Structure de données dans le compilateur C.*

l'horloge correspond alors à une valeur du type long (32 bits). Mais l'utilisation de variables de ce type est à éviter, car le temps d'exécution des opérations avec des valeurs du type long est très grand, même pour des opérations simples. La colonne $T_{\text{inc}}$ représente le temps pris pour faire l'incrémentation d'une variable du type char, int et long. La différence de performance du micro-contrôleur dans les variables du type long par rapport aux variables du type int ou char est évidente. Il faudra alors chercher à utiliser des stratégies qui permettent d'éviter l'usage de variables de 32 bits en continuant à utiliser une horloge de 32 bits.

Le constructeur `struct` de C permet de résoudre ce problème. L'horloge du système est décrite par la structure `Clock` (fig. 6.3).

```
typedef union {
  unsigned long Time;
  struct{
    int High;
    unsigned int Low;
  } Counter;
  unsigned char Register[4];
} Clock;
```

FIG. 6.3: *Structure de données* `Clock`

Cette structure permet de choisir, pour chaque opération, la méthode plus efficace pour accéder à la valeur de l'horloge. Regardons par exemple la fonction `time()`, qui lit l'horloge physique (fig. 6.4). Chaque partie de la valeur de l'horloge physique est lue soit comme un octet (TML2 et TMH2), soit comme un entier (ClockAux).

```
/*

  time()

  Lecture de l'horloge physique.

*/
monitor long time(void)
{
  Clock TimeNow;

  P1.0=1;                /* stockage de la valeur actuelle
                            de TIMER2 dans les registres de
                            capture CTH0 et CTL0 */
  TimeNow.Counter.High = ClockAux;
  TimeNow.Register[TIMER2HIGH] = CTH0;
  TimeNow.Register[TIMER2LOW] = CTL0;
  P1.0=0;

  return TimeNow.Time;
}
```

FIG. 6.4: *Fonction* `time()`

### 6.1.3 Messagerie

Les messages échangés entre les sites du système distribué ont la structure définie par la struct `TMesg` (figure 6.5). Un message transporte la valeur de l'horloge et un champ (de la taille d'un octet) appelé DATA. Ce champ est utilisé pour transporter, par exemple, le nombre de séquence du message.

```
typedef struct {
  unsigned char Data;
  Clock TimeSent;
} TimeMess;
```

FIG. 6.5: *Structure de données* `TimeMess`

### 6.1.4 Système pour l'analyse des délais

Pour l'analyse du comportement du système par rapport aux délais de transmission, nous avons monté la plate-forme expérimentale comme décrit dans la suite (fig. 6.6):

- trois stations (un maître et deux esclaves) exécutent l'algorithme de synchronisation ("Algorithme" dans la figure);
- deux autres stations sont responsables de charger le réseau CAN ("Charge"). Cette charge consiste en un ensemble de messages périodiques, de taille et période différentes, qui sont produits par ces deux stations.

L'ensemble de messages est basé sur la messagerie de l'application PSA ([Navet et Song, 1998], tableau 6.2). Dans sa présentation originale, cet ensemble occupe à peu près 40% de la capacité nominale du réseau CAN à 125 kbits/s. En faisant varier la période des messages, on peut faire varier la charge sur le réseau. La messagerie PSA a été choisie pour représenter un ensemble de messages "typique" dans une application basée sur le réseau CAN. Deux des équipements sont décrits par $x$ et $y$ pour des raisons de confidentialité.

## 6.2 Expérience 1 : Effet du champ ID

Cette expérience a pour objectif de vérifier l'effet de la valeur du champ ID des messages de synchronisation dans le comportement (du point de vue statistique) du délai de transmission dans un réseau chargé. Pour cela, nous avons créé une charge sur le réseau, représentant l'application qui s'exécute sur le système distribué, et nous avons mis l'algorithme de synchronisation en marche. Dans le premier cas (identifié par "lo"), les valeurs du champ ID des messages de synchronisation étaient inférieures à celles des messages de l'application (la charge). La priorité des messages de synchronisation est supérieure à la priorité des messages de l'application. Dans le deuxième cas (identifié par "hi"), les identificateurs des messages de synchronisation étaient supérieurs aux identificateurs de l'application. Les messages de synchronisation ont, dans ce cas, une priorité inférieure aux messages de l'application.

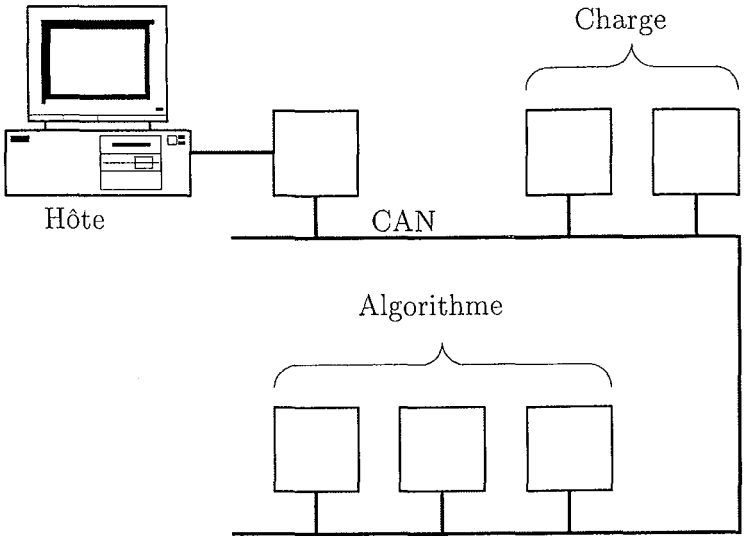



FIG. 6.6: *Système utilisé pour les expériences*

L'algorithme exécuté est l'algorithme de Cristian et le délai mesuré est le délai d'aller et retour. Cet algorithme est décrit en pseudo-code dans la figure 6.7.

Les paramètres qui déterminent la séquence temporelle des actions du maître sont: $InterRoundTime$ (l'intervalle entre deux essais consécutifs de lecture du site distant), $DtoNextAttempt$ (l'intervalle entre un essai échoué et l'essai suivant) et $InterSlavePhase$ (l'intervalle entre l'instant où démarrent les essais de lecture d'un site distant et l'instant où démarrent les essais pour le site suivant). Les valeurs utilisées dans cet expérience sont présentées dans le tableau 6.3. La valeur $DMax$ est suffisamment grande pour que l'essai réussisse toujours.

Remarquons que, dans les deux cas, la charge imposée par les messages de synchronisation est négligeable (6 messages de 87 bits tous les 5s, ce qui correspond à une charge de 104 bit/s, c.-à-d., inférieure à 0.1% de la capacité du réseau à 125 kbit/s).

Dans toutes les situations, la charge est définie en pourcentage de la charge effective totale (c'est-à-dire, y compris tous les bits "non utiles" de la trame CAN — SOF, RTR, ACK, ...). En faisant varier la valeur de la charge imposée sur le réseau, nous avons mesuré l'effet sur quatre paramètres: les valeurs moyenne ("Moy"), minimale ("Min"), maximal ("Max") et l'écart-type ("Ec-t") du délai. Les résultats sont présentés dans les tableaux 6.4 et 6.5 et figures 6.8, 6.9, 6.10 et 6.11.

On vérifie que le comportement des délais ne change pas d'une façon significative pour une charge sur le réseau inférieure à 60% et les courbes des deux cas, "hi" et "lo", sont similaires. À partir d'une charge de 60%, les courbes de valeurs "lo" (pour des IDs inférieurs aux IDs de la charge) et les courbes des valeurs "hi" s'écartent visiblement. L'effet est surtout évident dans la valeur du délai maximal, qui multiplie sa valeur par 6 quand la charge passe de 85 à 95% (de 993 à 5796 et de 992 à 5956). Pour les délais du groupe "lo", les valeurs maximale et minimale restent presque constantes pour toutes les valeurs de la charge, pendant que la

| Priorité | Site émetteur | DLC | Période |
|---|---|---|---|
| 1 | contrôle moteur | 8 | 10 ms |
| 2 | capteur angle volant | 3 | 14 ms |
| 3 | contrôle moteur | 3 | 20 ms |
| 4 | boite vitesse | 2 | 15 ms |
| 5 | équipement $x$ | 5 | 20 ms |
| 6 | équipement $x$ | 5 | 40 ms |
| 7 | équipement $x$ | 4 | 15 ms |
| 8 | calcul. carrosserie | 5 | 50 ms |
| 9 | équipement $y$ | 4 | 20 ms |
| 10 | contrôle moteur | 7 | 100 ms |
| 11 | boite vitesse | 5 | 50 ms |
| 12 | équipement $x$ | 1 | 100 ms |

TAB. 6.2: *Messagerie de l'application PSA (de [Navet et Song, 1998])*

```
every(InterRoundTime)
do
    now = time();
    for i : = 0 to NSlaves − 1 do
        at(now + i ∗ InterSlavePhase)
        do
            Success[i] : = False;
            NTries : = 0;
            while (¬Success[i] ∧ NTries < NMaxTries) do
                    NTries + +;
                    TimeSent : = time();
                    SendTimeRequest(i);
                    ReceiveTime(i);
                    D = time() − TimeSent;
                    Success[i] : = (D < Dmax);
                    if (¬Success[i])
                      wait(DtoNextAttempt); fi
            od
        od
    od
od
```

FIG. 6.7: *Algorithme de Cristian*

| Paramètre | Valeur |
|---|---|
| $InterRoundTime$ | 5s |
| $DtoNextAttempt$ | 0.1s |
| $InterSlavePhase$ | 1s |

TAB. 6.3: *Paramètres de l'algorithme de Cristian*

| | Esclave 0 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Hi | | | | Lo | | | |
| Charge | Moy | Ec-t | Min | Max | Moy | Ec-t | Min | Max |
| 10% | 344.54 | 23.65 | 328 | 491 | 344.76 | 20.37 | 328 | 510 |
| 20% | 353.01 | 29.57 | 328 | 476 | 355.02 | 27.79 | 328 | 474 |
| 30% | 362.10 | 32.75 | 327 | 467 | 356.41 | 25.60 | 330 | 465 |
| 40% | 370.35 | 38.13 | 327 | 512 | 373.51 | 38.98 | 330 | 504 |
| 50% | 408.07 | 104.89 | 326 | 959 | 378.82 | 38.51 | 329 | 502 |
| 60% | 370.38 | 36.57 | 327 | 503 | 390.18 | 39.20 | 330 | 504 |
| 70% | 507.58 | 170.19 | 328 | 998 | 394.74 | 42.63 | 333 | 512 |
| 80% | 567.35 | 200.88 | 330 | 992 | 403.25 | 37.63 | 328 | 512 |
| 85% | 600.62 | 191.08 | 334 | 993 | 402.49 | 34.65 | 338 | 501 |
| 90% | 1003.10 | 457.85 | 331 | 1967 | 416.19 | 38.43 | 335 | 513 |
| 95% | 2724.50 | 1442.00 | 427 | 5796 | 411.35 | 36.95 | 338 | 512 |

TAB. 6.4: *Statistiques des délais (Esclave 0)*

| | Esclave 1 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Hi | | | | Lo | | | |
| Charge | Moy | Ec-t | Min | Max | Moy | Ec-t | Min | Max |
| 10% | 343.56 | 23.27 | 328 | 478 | 343.85 | 19.29 | 328 | 421 |
| 20% | 352.17 | 26.94 | 328 | 462 | 352.80 | 27.94 | 329 | 506 |
| 30% | 360.54 | 31.25 | 327 | 445 | 365.95 | 36.02 | 329 | 491 |
| 40% | 372.53 | 34.20 | 326 | 495 | 369.89 | 37.56 | 328 | 505 |
| 50% | 405.17 | 109.65 | 329 | 985 | 377.95 | 37.67 | 330 | 494 |
| 60% | 369.28 | 35.66 | 328 | 504 | 388.07 | 39.99 | 331 | 509 |
| 70% | 520.08 | 169.99 | 327 | 951 | 395.12 | 38.41 | 333 | 507 |
| 80% | 566.98 | 185.31 | 333 | 996 | 403.00 | 39.71 | 331 | 512 |
| 85% | 558.62 | 184.64 | 334 | 992 | 406.38 | 35.67 | 334 | 511 |
| 90% | 1003.91 | 458.80 | 339 | 1974 | 411.46 | 36.72 | 340 | 512 |
| 95% | 2930.88 | 1419.39 | 389 | 5950 | 410.50 | 37.20 | 343 | 512 |

TAB. 6.5: *Statistiques des délais (Esclave 1)*

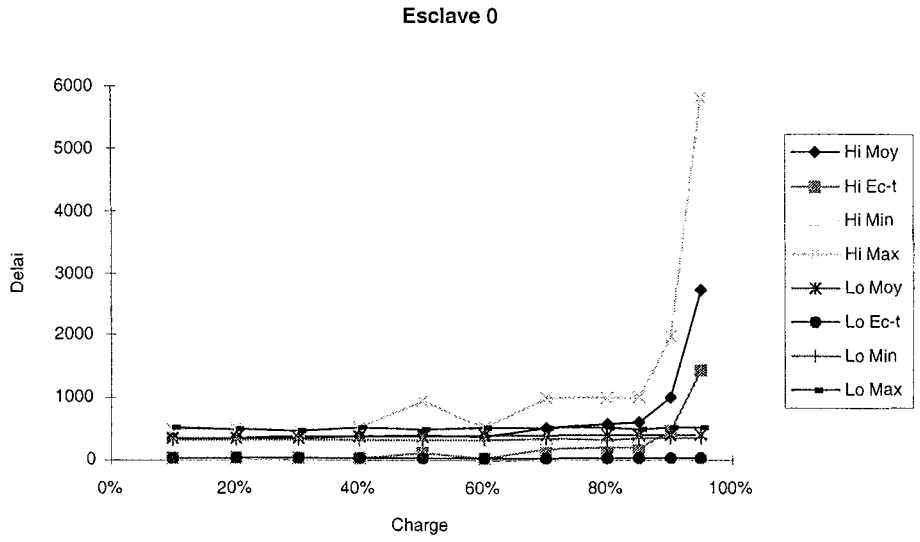



FIG. 6.8: *Statistiques des délais (Esclave 0)*

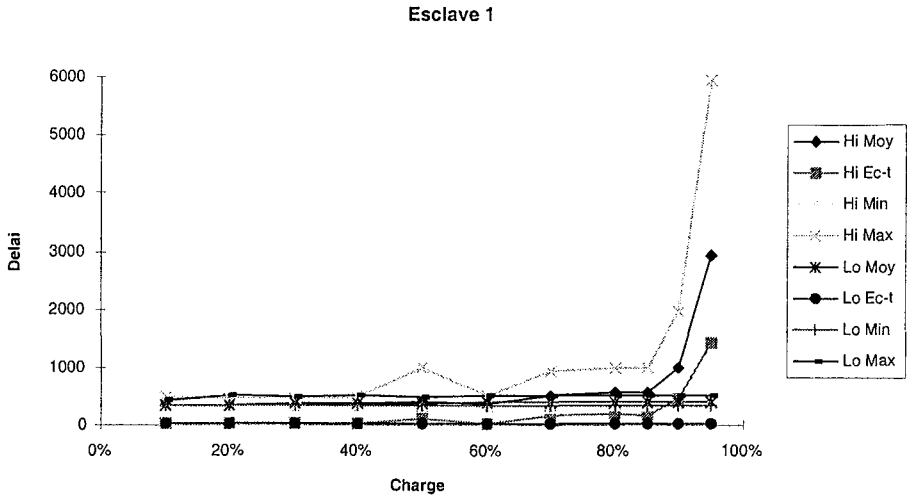



FIG. 6.9: *Statistiques des délais (Esclave 1)*

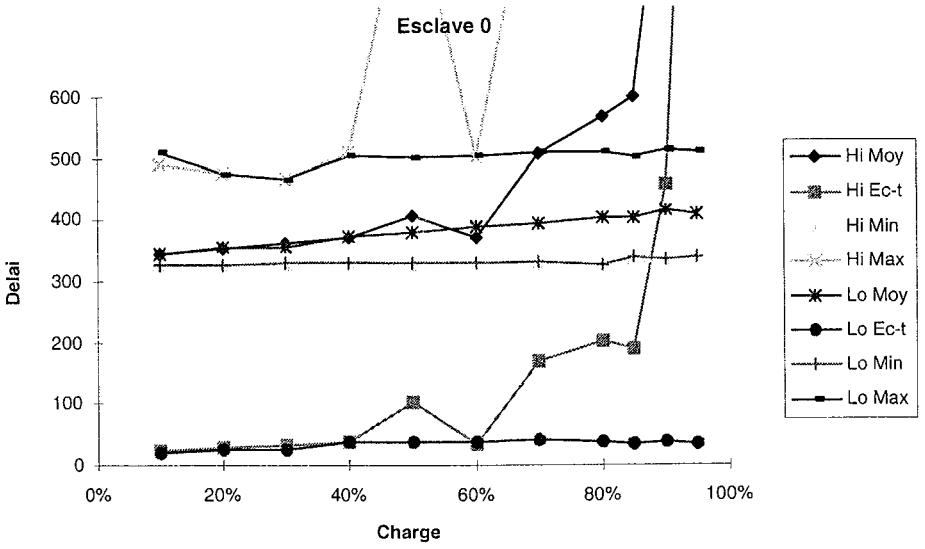



FIG. 6.10: *Statistiques des délais (Esclave 0)*

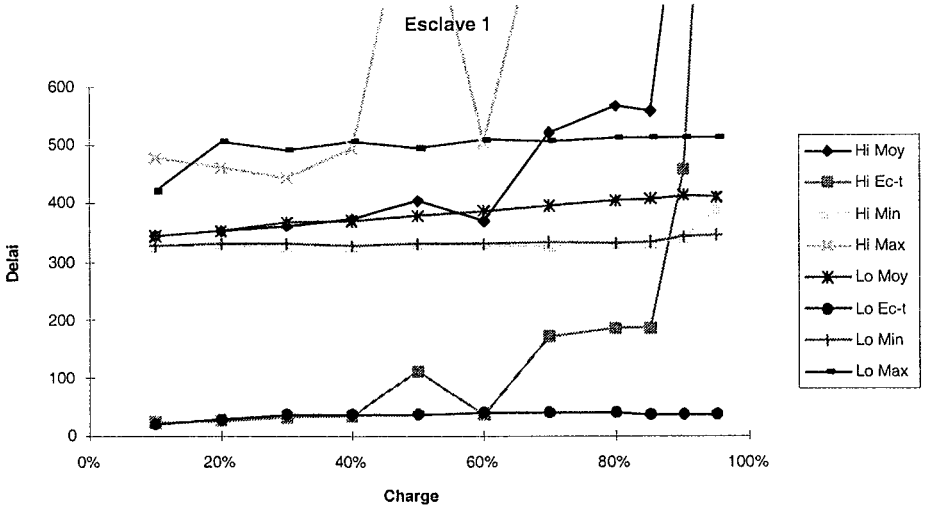



FIG. 6.11: *Statistiques des délais (Esclave 1)*

valeur moyenne monte légèrement.

Ceci est en accord avec ce que l'on attendait. Quand les identificateurs des messages de synchronisation sont inférieurs aux identificateurs de l'application, leur priorité est plus grande et l'envoi de ces messages se fait avec une perturbation minimale, même avec des valeurs de la charge imposée par l'application élevées. Par contre, si les valeurs des identificateurs des messages de synchronisation sont supérieures aux valeurs des identificateurs de l'application, le délai des messages de synchronisation est sensible à la charge. Dans ce cas, les messages de synchronisation ont une très basse priorité et, en cas de conflit d'accès au médium avec les messages de l'application, ils sont toujours perdants. Leur délai est, par conséquence, plus grand.

Remarquons aussi que la valeur minimale du délai, même pour le cas "hi", reste pratiquement constante pour toutes les valeurs de la charge. La seule exception est la valeur pour les IDs "hi" avec une charge de 95%.

On peut conclure donc que, si pour des valeurs petites pour la charge sur le réseau la valeur des identificateurs attribués aux messages de synchronisation n'influence pas fortement le comportement des délais du point de vue statistique, ceci n'est pas vrai pour toutes valeurs de la charge. Pour des valeurs élevées, le comportement des délais est fortement influencé par la charge.

## 6.3 Vérification du modèle avec un algorithme d'intervalle

Le théorème 4.1 nous permet de trouver les conditions locales qui sont suffisantes pour garantir une caractéristique globale du système. Ceci nous permet de nous concentrer sur les conditions locales qui sont plus faciles à calculer et à maîtriser. Le résultat est basé sur les hypothèses suivantes :

1. l'indépendance des erreurs
2. l'existence de la borne $\epsilon^{\text{tol}}$ qui garantit la précision $\delta^{\text{spec}}$
3. l'existence de la fonction $f^{\text{err}}$, le majorant pour la probabilité que l'erreur dans une étape soit supérieure à $\epsilon^{\text{tol}}$.

Cette section adressera la question de la validité de l'indépendance des erreurs et l'existence de la fonction $f^{\text{err}}$(ou la validité de la dérivation de l'expression de $f^{\text{err}}$pour un cas particulier).

### 6.3.1 Expérience 2 : Caractérisation des délais

Le modèle qui a été présenté utilise, pour la prévision du nombre de messages nécessaires, la représentation du délai comme une variable aléatoire. Dans cette section, nous présentons les résultats des expériences dont l'objectif est la caractérisation du délai comme une variable aléatoire. Les variables sont la charge sur le réseau et la valeur des identificateurs des messages de synchronisation. Les valeurs utilisées pour la charge étaient 10%, 70%, 85% et 95%. Les valeurs utilisées pour les identificateurs étaient les groupes Hi et Lo, définis comme auparavant. Chaque combinaison charge/ID définit un *cas*. L'identification de chaque cas est présentée dans le tableau 6.6. Les figure 6.12 à 6.19 présentent, pour chacun des cas, l'histogramme de la distribution des délais d'aller/retour. A partir de ces résultats, nous avons

fait l'hypothèse que la partie variable du délai est une variable avec distribution gamma. La moyenne et la variance pour cette distribution sont estimées à partir des données de l'expérience (eq. (4.50) et (4.51)). La fonction distribution de probabilité (f.d.p.) de cette variable gamma est présentée superposée à l'histogramme.

| | ID messages | |
|---|---|---|
| Charge | Lo | Hi |
| 10% | 01 | 11 |
| 70% | 02 | 12 |
| 85% | 03 | 13 |
| 95% | 04 | 14 |

TAB. 6.6: *Paramètres pour la caractérisation des délais*

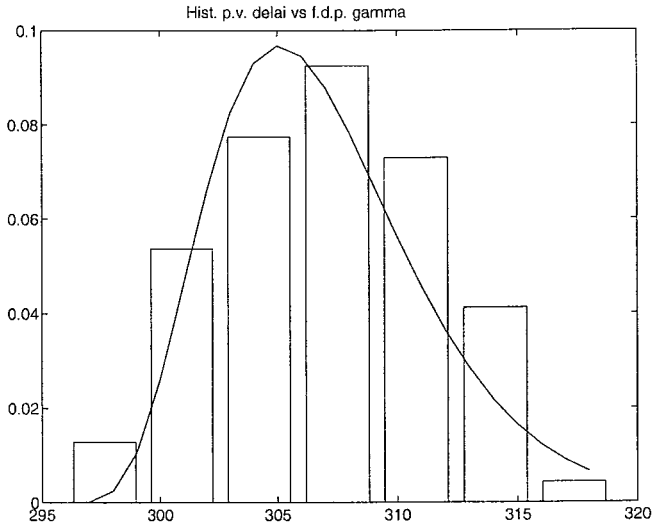



FIG. 6.12: *Histogramme du délai et f.d.p. gamma (cas 01)*

Les résultats nous permettent de vérifier que la distribution des délais dans l'expérience est proche d'une distribution gamma.

### 6.3.2 Expérience 3 : Existence de $f^{\text{err}}$

L'expression pour la fonction $f^{\text{err}}$dans les cas des algorithmes d'intervalle était fondée sur l'hypothèse que chaque essai de lecture de l'horloge distante était un événement aléatoire, dont le résultat est indépendant du résultat de tous les autres essais (hyp. 4.1). Dans ce cas, la probabilité que l'erreur après $r$ essais de lecture soit inférieure à une valeur $\epsilon^{\max}$ est $1-(1-p)^r$, où $p$ est la probabilité que la partie variable du délai soit inférieure à $2\epsilon^{\max}$ (eq. 4.29 et 4.30).

Pour valider cette hypothèse, l'algorithme de Cristian était mis en marche sur un Maître et deux Esclaves. Les paramètres du Maître sont présentés dans le tableau 6.7. La valeur 0 pour

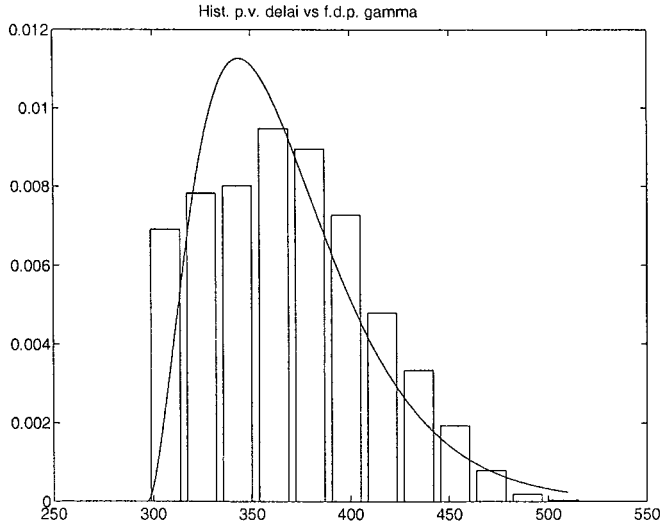



FIG. 6.13: *Histogramme du délai et f.d.p. gamma (cas 02)*

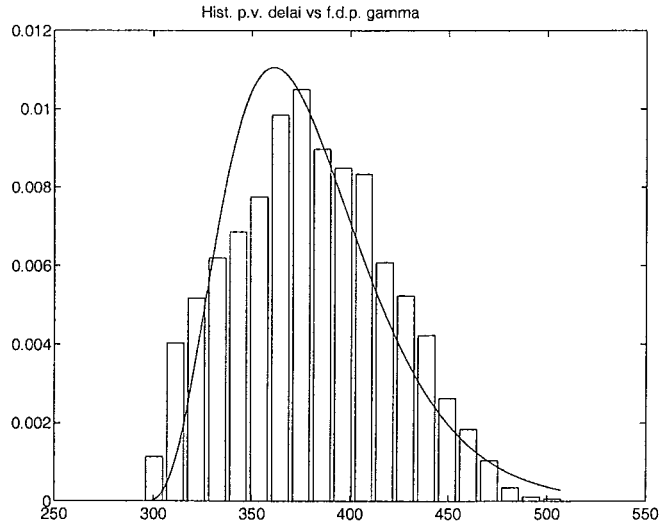



FIG. 6.14: *Histogramme du délai et f.d.p. gamma (cas 03)*

| Paramètre | Valeur |
|---|---|
| $InterRoundTime$ | 15s |
| $DtoNextAttempt$ | 0.1s |
| $InterSlavePhase$ | 4s |
| $Dmax$ | 0 |
| $NMaxTries$ | 40 |

TAB. 6.7: *Paramètres du Maître pour la vérification de* $f^{\text{err}}$

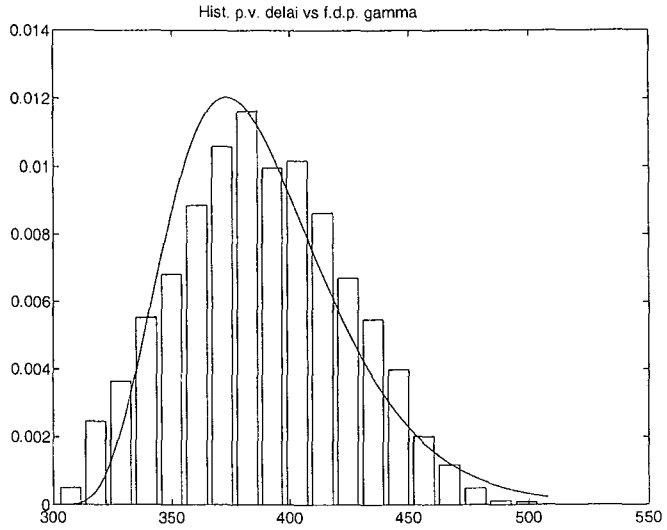



FIG. 6.15: *Histogramme du délai et f.d.p. gamma (cas 04)*

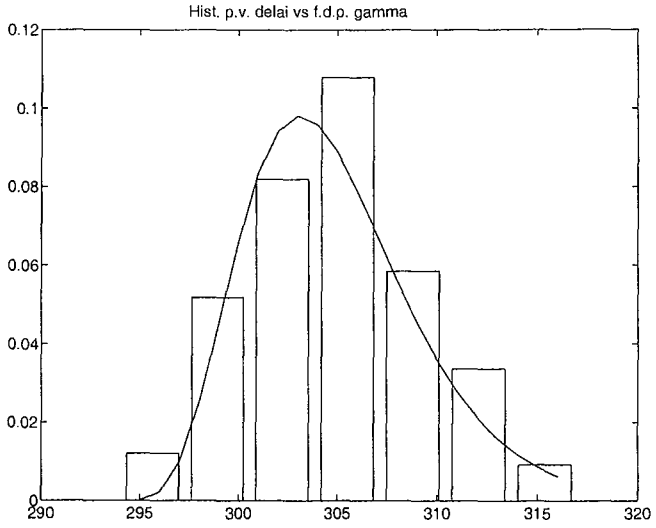



FIG. 6.16: *Histogramme du délai et f.d.p. gamma (cas 11)*

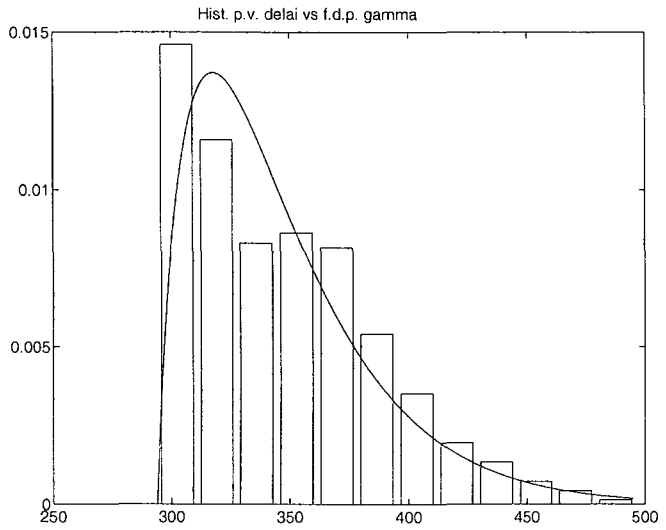



FIG. 6.17: *Histogramme du délai et f.d.p. gamma (cas 12)*

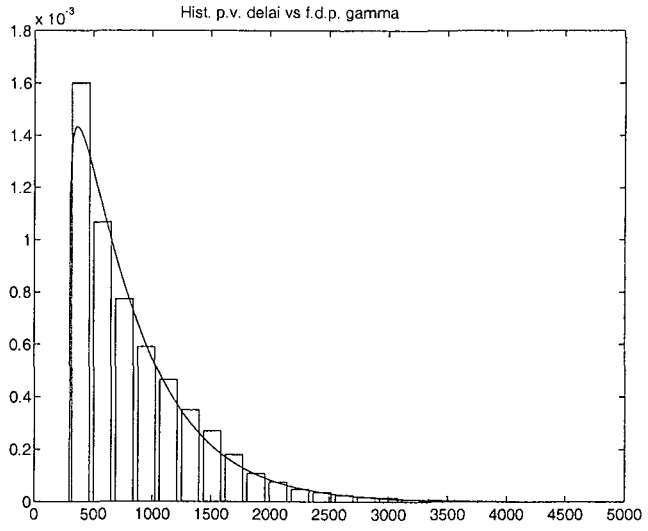



FIG. 6.18: *Histogramme du délai et f.d.p. gamma (cas 13)*

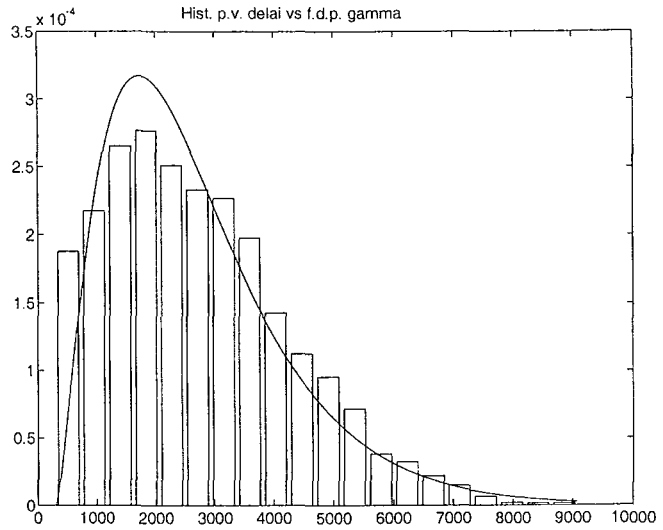



FIG. 6.19: *Histogramme du délai et f.d.p. gamma (cas 14)*

*Dmax* signifie que le Maître ne considérera jamais le succès de la lecture de l'Esclave et donc il essayera jusqu'à $NMaxTries$ tentatives (à ce moment il abandonne). Nous obtenons une suite de 40 lectures de l'horloge distante, en rafale, dont le délai d'aller-retour est enregistré. Ces valeurs composent la matrice **D** des délais :

$$\mathbf{D} \quad : \quad D_{i,j} = \left[ \begin{array}{l} \text{Délai aller-retour} \\ \text{du message } j \\ \text{dans la rafale } i \end{array} \right] \tag{6.2}$$

Nous sommes intéressés par la valeur minimale dans la rafale (la probabilité qu'au moins un délai dans une rafale soit inférieur à $X$ est la probabilité que le délai minimal de la rafale soit inférieur à $X$). Il faut donc calculer la matrice **M** des délais minimaux, où $M_{i,r}$ est le délai minimal après une rafale de $r$ essais.

$$\mathbf{M} \quad : \quad M_{i,r} = \min_{k=1..r} \{D_{i,k}\} \tag{6.3}$$

La probabilité à calculer est fonction de deux paramètres : le délai d'aller retour souhaité (dont dépend l'erreur) et le nombre de messages $r$. Ceci définit un plan sur un espace à trois dimensions, où les coordonnées sont $D$ (la valeur maximale pour le délai), $r$ (le nombre d'essais) et $P = \mathbf{P}[D(r) \leq D]$ (la probabilité que le délai d'aller-retour minimal après $r$ essais soit inférieur à $D$) (fig. 6.20). Cette probabilité peut être calculée de deux façons : soit en utilisant les données expérimentales pour faire une estimation, soit en utilisant l'hypothèse de l'indépendance. Dans le premier cas :

$$\mathbf{P}[D(r) \leq D] = \frac{\#\{M_{i,r} \ : \ M_{i,r} \leq D\}}{\#\{M_{i,r}\}} \tag{6.4}$$

où $D(r)$ est le délai d'aller-retour après $r$ essais et le symbole $\#$ indique le cardinal d'un ensemble.

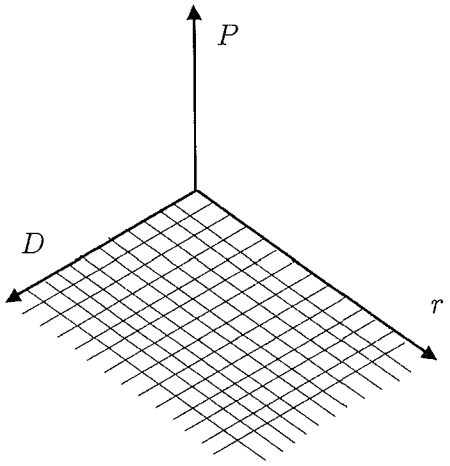



FIG. 6.20: *Espace $D, r, P$*

Pour le deuxième cas, nous utilisons l'hypothèse de l'indépendance des essais et l'hypothèse que les délais sont représentés par une variable aléatoire avec distribution gamma :

$$\mathbf{P}\left[D(r) \leq D\right] = 1 - \left(1 - \mathrm{F}_{\Gamma(\alpha,\lambda)}(D)\right)^r \tag{6.5}$$

où les valeurs $\alpha$ et $\lambda$ sont calculées à partir de la moyenne et variance du délai estimées sur toute la matrice $\mathbf{D}$. Nous présentons les résultats de 4 cas de cette expérience qui correspondent aux identificateurs "hi" et "lo" et à une charge de 0% et 95% (tab. 6.8).

| Situation | Esclave 1 | | Esclave 2 | |
|---|---|---|---|---|
| | Moyenne | Écart-type | Moyenne | Écart-type |
| Lo, 0% | 307.40 | 4.44 | 306.41 | 4.38 |
| Lo, 95% | 388.41 | 35.71 | 387.72 | 35.94 |
| Hi, 0% | 305.25 | 4.31 | 304.46 | 4.40 |
| Hi, 95% | 2713.02 | 1548.54 | 2715.85 | 1544.27 |

TAB. 6.8: *Moyenne et écart-type pour les 4 cas considérés.*

Les figures représentent la projection sur le plan $r \times D$ des lignes de iso-probabilité pour les probabilités de succès égales à 0.1, 0.9 et 0.95. Chaque figure contient les courbes pour la charge de 0% (sur la droite) et 95%(sur la gauche) (figs. 6.21 à 6.24). Les lignes continues représentent la probabilité vérifiée expérimentalement et les lignes pointillées la probabilité calculée en utilisant l'expression de la fonction d'erreur.

Le modèle basé sur l'indépendance des délais et sur la représentation du délai comme une variable aléatoire de distribution gamma suit les tendances présentées par le système et fournit, dans la plupart des cas, une estimation pessimiste, c.-à-d., une condition suffisante pour que la lecture réussisse avec la probabilité souhaitée.

### 6.3.3 Expérience 4 : Indépendance entre étapes

La validité du modèle présenté dans le chapitre 4 était aussi basée sur l'indépendance du succès de la synchronisation entre étapes (hyp. 4.9). Pour l'algorithme de Cristian, le

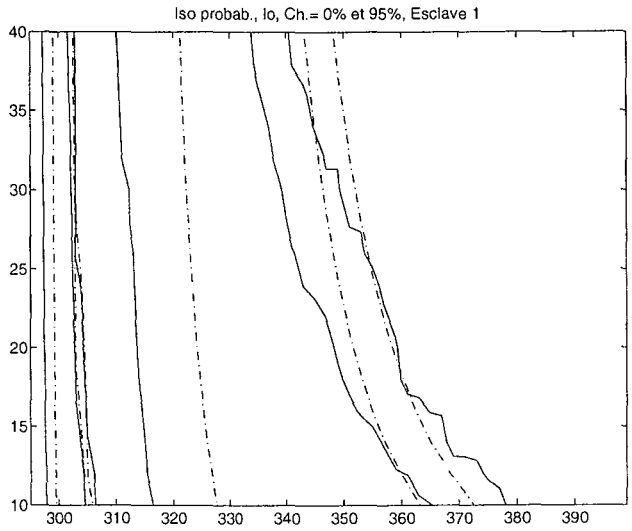



FIG. 6.21: *Expérience 3 : Lignes de iso-probabilité (Io, esclave 1)*

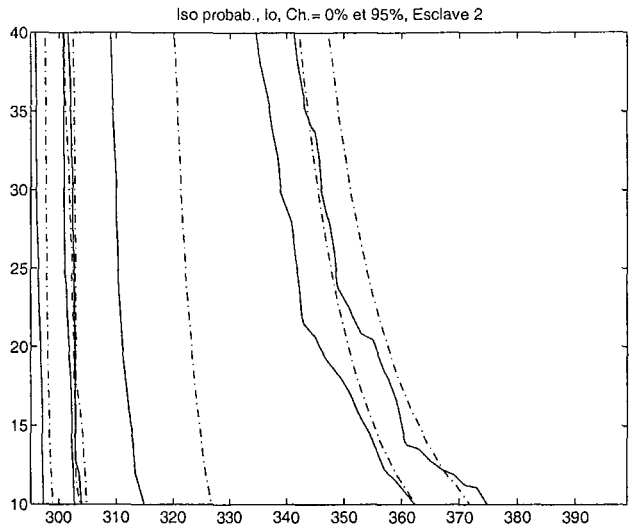



FIG. 6.22: *Expérience 3 : Lignes de iso-probabilité (Io, esclave 2)*

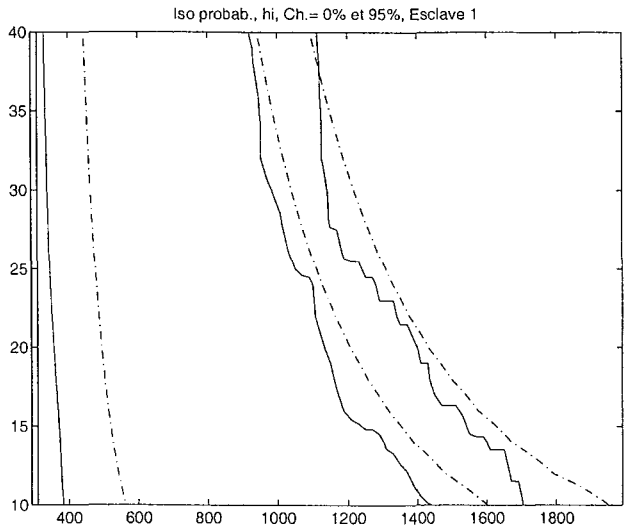



FIG. 6.23: *Expérience 3 : Lignes de iso-probabilité (hi, esclave 1)*

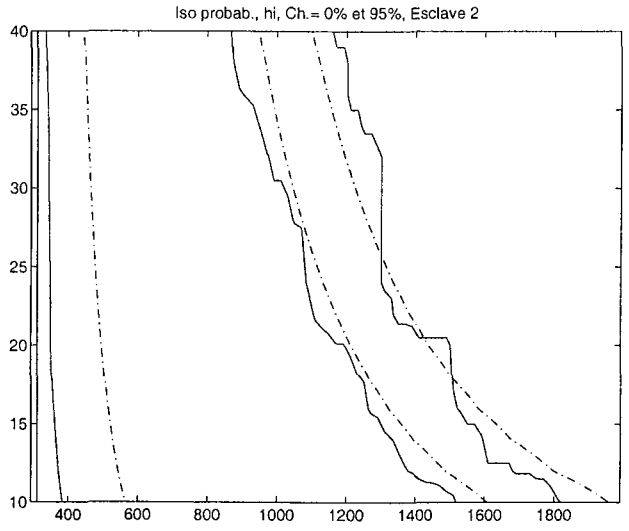



FIG. 6.24: *Expérience 3 : Lignes de iso-probabilité (hi, esclave 2)*

succès de la lecture signifie que le délai minimal est inférieur à un seuil $D$, qui est fonction de l'erreur tolérée, $\epsilon^{\text{tol}}$. Sous l'hypothèse de l'indépendance entre étapes, la probabilité de succès de toutes les étapes est égale au produit de la probabilité de succès de chaque étape.

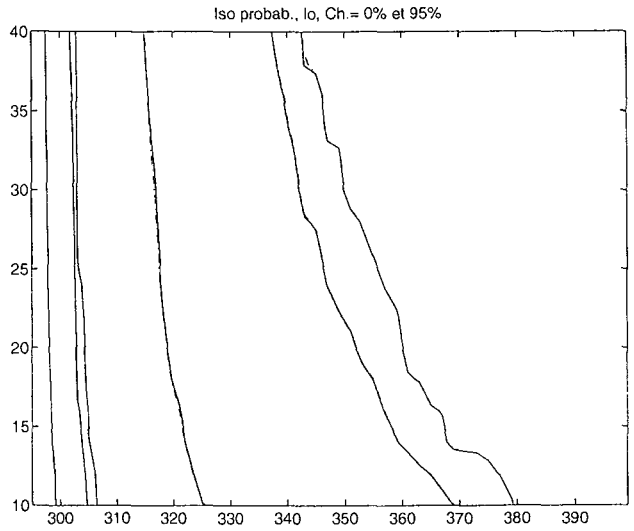



FIG. 6.25: *Expérience 4 : Occurrence d'un délai inférieur à D sur les deux esclaves: lignes de iso-probabilité (lo)*

Pour tester cette hypothèse, les données de l'expérience antérieure étaient utilisées pour calculer la probabilité de succès dans toutes les étapes en fonction de $r$ et $D$. Dans ce cas, le nombre d'étapes est égal à deux. Les résultats sont présentés dans les figures 6.25 et 6.26. Les lignes continues indiquent les lignes de iso-probabilité vérifiées expérimentalement pour la probabilité de succès dans les deux étapes. Les lignes pointillées correspondent à la valeur de cette probabilité calculée en utilisant l'hypothèse d'indépendance.

Les lignes de iso-probabilité sont presque toujours superposées, c'est à dire, la probabilité de succès dans les deux étapes est identique au produit des probabilités de succès dans chacune des étapes. Ceci indique que la réussite de la synchronisation dans une étape (avec un esclave) est indépendante de la réussite dans une autre étape (avec un autre esclave).

### 6.3.4 Expérience 5 : Garantie de la synchronisation

Les résultats précédents supportent la validité des hypothèses associées au modèle pour la performance des ANDSH. Ce modèle a permis d'établir la Condition de Garantie de la Synchronisation (théorème 4.1). Après avoir expérimentalement vérifié la validité de ces hypothèses, il faut encore vérifier la validité de ce théorème.

La théorème 4.1 garantit que l'algorithme de synchronisation non-déterministe réussit à synchroniser les horloges avec une probabilité $P \geq \mathrm{P_{spec}}$ avec une erreur $\epsilon^{\text{tol}}$ et en utilisant $r$ messages si :

$$f^{\text{err}}(\epsilon^{\text{tol}}, r) \leq \frac{1 - \mathrm{P_{spec}}}{n_p n_e n_c} \tag{6.6}$$

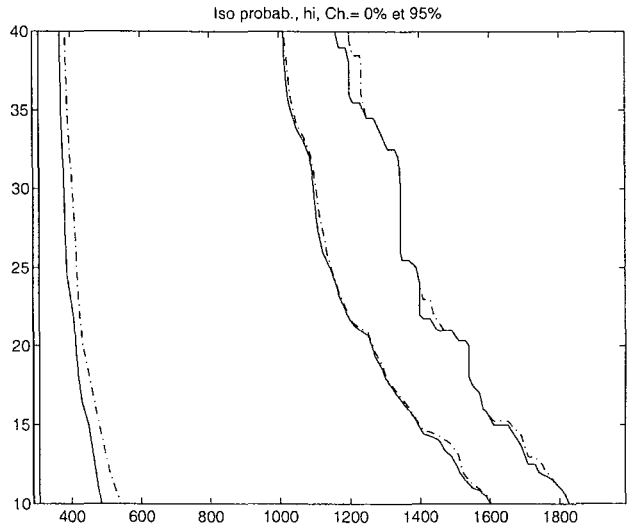



FIG. 6.26: *Expérience 4 : Occurrence d'un délai inférieur à D sur les deux esclaves: lignes de iso-probabilité (hi)*

$f^{\text{err}}$est la fonction d'erreur et, dans notre cas, $f^{\text{err}}$est la probabilité que le délai minimal d'aller-retour après $r$ essais, $\Gamma_{ar,min}(r)$, soit supérieur à un seuil $D$ qui définit l'erreur :$D = 2\epsilon^{\text{tol}}$.

$$\begin{aligned} f^{\text{err}}(\epsilon^{\text{tol}}, r) &= \mathbf{P}\left[\Gamma_{ar,min}(r) > D\right] \\ &= 1 - \mathbf{P}\left[\Gamma_{ar,min}(r) \leq D\right] \\ &= 1 - \left[1 - (1 - \mathrm{F}_{\Gamma(\alpha,\lambda)}(D))^r\right] \\ &= (1 - \mathrm{F}_{\Gamma(\alpha,\lambda)}(D))^r \end{aligned} \tag{6.7}$$

où $\mathrm{F}_{\Gamma(\alpha,\lambda)}(x)$ est la fonction de distribution du délai.

De (6.6) et (6.7), nous avons :

$$\mathrm{P}_{\text{spec}} \leq 1 - (1 - \mathrm{F}_{\Gamma(\alpha,\lambda)}(D))^r n_p n_c n_e \tag{6.8}$$

Pour l'algorithme de Cristian s'exécutant dans le cas d'une communication point-à-point et avec une structure Maitre-Esclave nous avons :

$$\begin{aligned} n_p &= 1 \\ n_c &= 1 \\ n_e &= n_{\text{esclaves}} \end{aligned}$$

où $n_{\text{esclaves}}$ est le nombre d'esclaves (2 dans les expériences). Finalement, on obtient :

$$\mathrm{P}_{\text{spec}} \leq 1 - (1 - \mathrm{F}_{\Gamma(\alpha,\lambda)}(D))^r n_{\text{esclaves}} \tag{6.9}$$

La moitié droite de l'équation (6.9) est (par hypothèse) un minorant de la probabilité de succès dans la synchronisation (sous les conditions spécifiées).

Pour vérifier la validité de la condition de garantie de synchronisation nous avons mis l'algorithme de Cristian en marche et nous avons comparé la valeur de l'expression pour le minorant de l'équation (6.9) avec l'estimation expérimentale pour la probabilité de réussite. Les variables d'expérience sont : le délai maximal, $D$, le nombre de messages $r$, la charge du réseau (dont dépendent les paramètres $\alpha$ et $\lambda$) et l'ID des messages ("lo" ou "hi"). Chaque combinaison de ces variables définit un cas.

Les cas considérés sont présentés dans le tableau 6.9. La dernière colonne présente aussi la figure avec les résultats. Dans les figures, la ligne correspond aux résultats théoriques et les points aux résultats expérimentaux. Nous présentons seulement les cas où la probabilité de réussite change significativement avec $r$. Les autres cas sont des situations où la probabilité de réussite est quasiment nulle ou alors où cette probabilité est égale à 1 (exemple du cas numéro 8, où la synchronisation réussit toujours)

Les figure 6.27 à 6.33 permettent de vérifier que les points correspondants aux résultats expérimentaux sont toujours au dessus de la courbe du minorant théorique. Dans les cas où le modèle indique une probabilité de réussite égale à 1, le système se synchronise toujours.

| Cas | ID | $D$ | Charge | Figure |
|---|---|---|---|---|
| 01 | lo | 310 | 70% | 6.27 |
| 04 | lo | 350 | 70% | 6.28 |
| 06 | lo | 350 | 95% | 6.29 |
| 08 | lo | 400 | 70% | 6.30 |
| 12 | hi | 350 | 85% | 6.31 |
| 15 | hi | 500 | 85% | 6.32 |
| 31 | hi | 1000 | 95% | 6.33 |

TAB. 6.9: *Description des cas de l'expérience 5*

Les résultats permettent de conclure que, pour la situation considérée (algorithme de Cristian en structure Maitre-Esclave sur le réseau CAN), la Condition de Garantie de Synchronisation est valide : la probabilité de réussite associée à un certain ensemble de variables a été toujours garantie même dans des situations "dégénérées", comme celles où la probabilité de réussite est égale à 1.

## 6.4 Conclusions

Dans ce chapitre, nous avons repris les questions qui étaient laissées ouvertes après la présentation du modèle pour le comportement des algorithmes de synchronisation non-déterministes. Notre approche a été d'identifier les questions fondamentales sur lesquelles le modèle se basait :

1. Quelle distribution (si elle existe) peut représenter le délai comme une variable aléatoire?

2. Est-ce que les essais consécutifs de lecture de l'horloge d'un site distant sont des événements statistiquement indépendants (indépendance temporelle)?

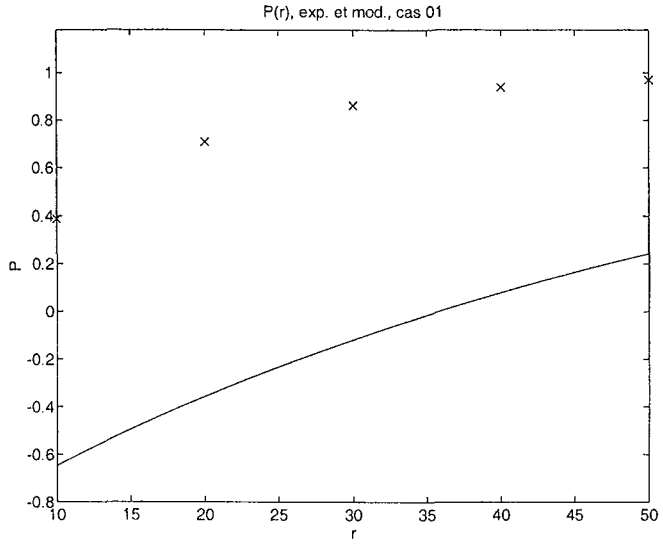



FIG. 6.27: *Expérience 5: cas 01*

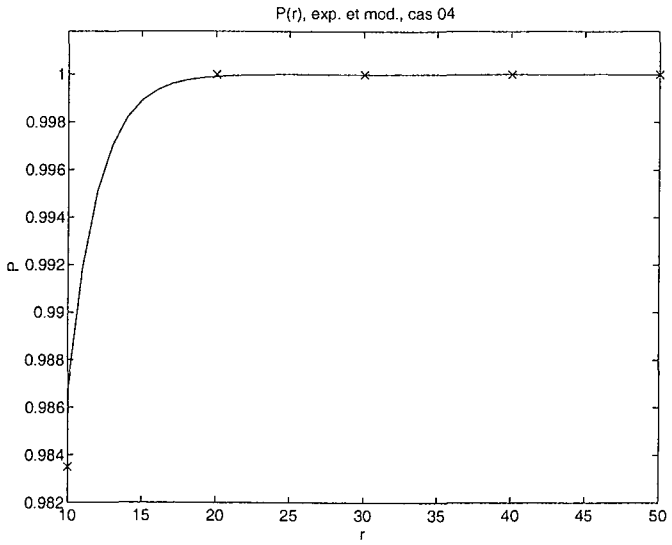



FIG. 6.28: *Expérience 5: cas 04*

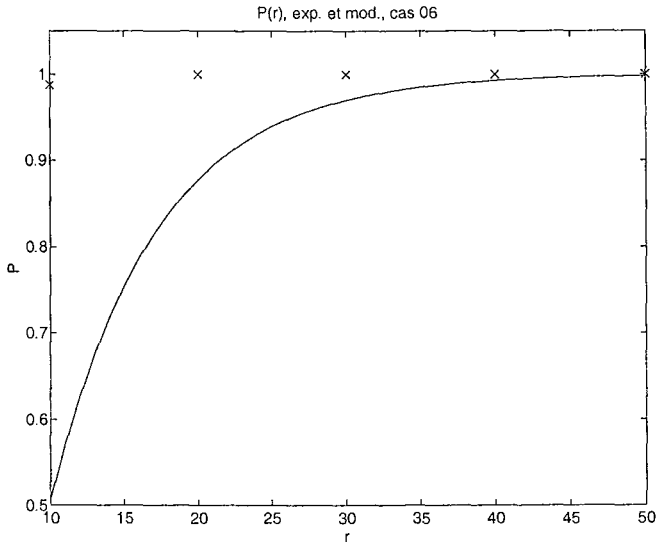



FIG. 6.29: *Expérience 5: cas 06*

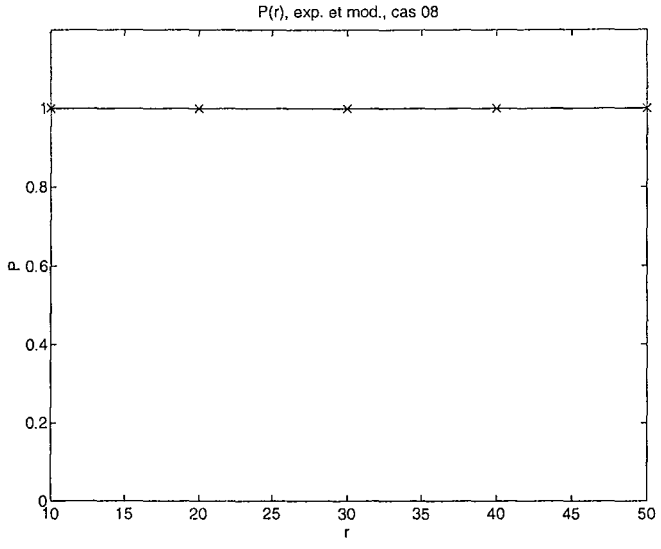



FIG. 6.30: *Expérience 5: cas 08*

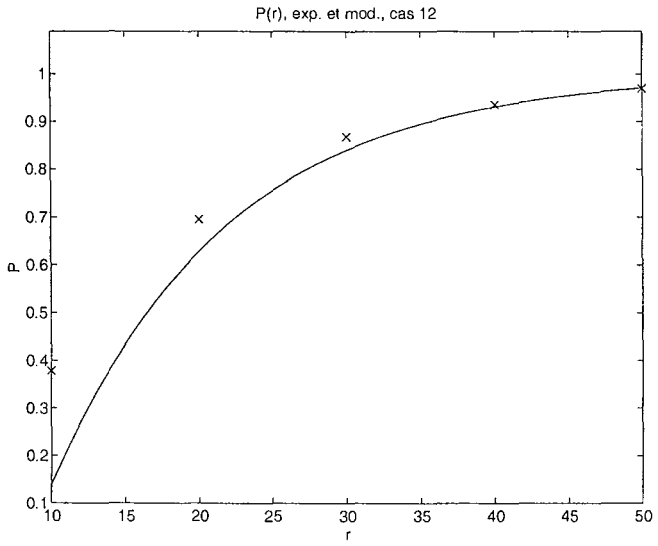



FIG. 6.31: *Expérience 5: cas 12*

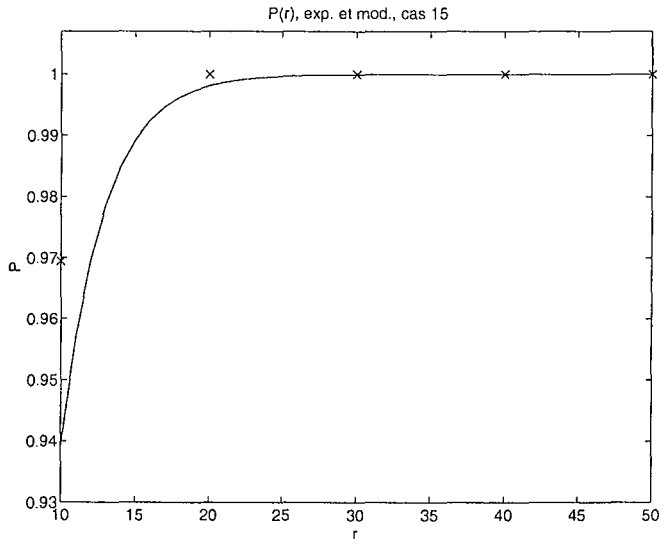



FIG. 6.32: *Expérience 5: cas 15*

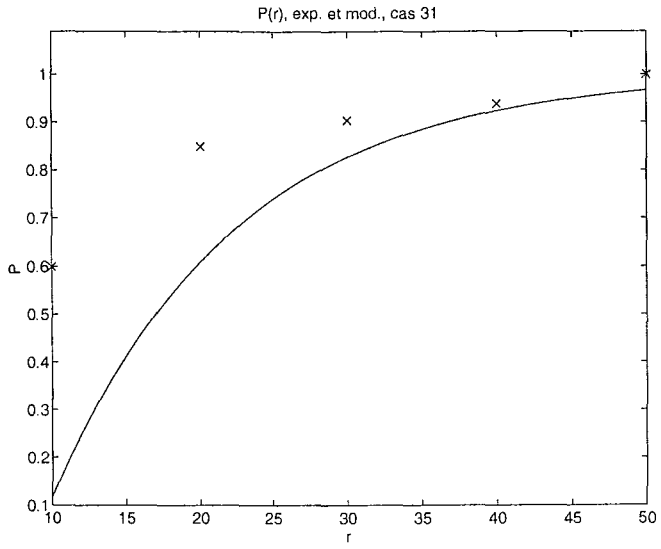



FIG. 6.33: *Expérience 5: cas 31*

3. Est-ce que les essais de lecture de l'horloge de deux sites distants sont des événements statistiquement indépendants (indépendance spatielle)?

Ces questions permettraient de valider expérimentalement les hypothèses qui valident le modèle. L'étape suivante a été de vérifier la validité du modèle lui-même.

Les expériences réalisées se sont limitées à un type d'algorithme et un type de réseau. Les raisons pour des expériences aussi dirigées ont été principalement deux et elles se rapportent au temps qui serait nécessaire pour réaliser un autre ensemble d'expériences. La première raison a été le temps qui serait nécessaire pour aboutir à une version correcte (ou vérifiée comme correcte d'une façon aussi extensive que possible) d'un autre algorithme. Les algorithmes sont présentés d'une façon informelle. Ce type de représentation permet que le lecteur saisisse rapidement le fonctionnement de l'algorithme mais, par contre, peut être difficile à traduire dans un langage séquentiel, comme la plupart des langages utilisés dans la programmation. Ceci signifie que ce processus de traduction d'une description informelle dans un programme en langage C, par exemple, est un processus qui demande un grand nombre de choix et où des erreurs peuvent facilement s'introduire. À la fin, le résultat est souvent un programme dont le concepteur n'est pas toujours sûr qu'il représente la spécification originale. Seulement le test pourra apporter quelques indications sur la validité des solutions adoptées, mais toujours avec les limitations associées a ce forme de validation.

La deuxième raison provient du type de modèle proposé. Le modèle est un modèle probabilistique et vérifier expérimentalement ses prévisions d'une façon aussi correcte que possible demande l'obtention d'échantillons d'une taille considérable. En conséquence, les expériences à réaliser sont longues et prennent forcément du temps.

Le fait que les expériences ont été réalisées avec un seul algorithme sur un seul type de réseau a donc été imposé par des contraintes de temps. Nous devons faire la remarque qu'une telle limitation n'est pas imposée par le modèle, car celui est générique : il est destiné à l'ensemble des algorithmes non-déterministes de synchronisation des horloges et n'impose

pas de restrictions sur le type de réseau.

Les résultats des expériences décrites dans ce chapitre nous ont permis de vérifier que le délai peut être représenté par une variable aléatoire avec une distribution gamma et que les essais de lecture sont indépendants dans le cas considéré (exécution de l'algorithme de Cristian sur un réseau CAN). Finalement, l'essai des prévisions issues de la Condition de Garantie de la Synchronisation a été conforme aux résultats expérimentaux. Ces observations permettent donc de valider le modèle présenté.

# Chapitre 7

# Conclusion générale

Au cours de ce travail, nous avons étudié le problème de la synchronisation des horloges dans les systèmes distribués. La synchronisation garantit que les horloges d'un tel système ne s'écartent pas plus d'une certaine valeur (la précision de synchronisation). Dans certains cas, il faut non seulement garantir l'accord des horloges du système entre elles (précision) mais aussi entre ces horloges et une référence extérieure, que l'on considère correcte (exactitude). La synchronisation des horloges est fondamentale pour la bonne marche d'un système distribué qui doit exécuter ses actions sous des contraintes temporelles (un système distribué temps-réel). Malgré son importance et le fait que ce problème ait été traité par des chercheurs depuis deux décennies, la discussion sur ce sujet n'est pas encore terminée. Nous ne sommes pas encore arrivés aux réponses absolues (est-ce qu'on y arrivera?...). Nous pouvons citer, par exemple, les opinions de Mills ([Mills, 1994]) et de Suri, Hughe et Walter ([Suri et al., 1994]) sur la façon d'intégrer la synchronisation des horloges comme un service de base dans les systèmes distribués :

> « In a perfect world, the NTP PLL model would be implemented as an intrinsic feature of the kernel with standardized interfaces for the user and daemon processes and with a precision local clock oscillator available as a standard option. However, during the development and deployment of NTP technology, there was considerable reluctance to intrude on kernel hardware or software features since this would impede portability, maitainability and, perhaps, reliability. » ([Mills, 1994])

> « Ideally, synchronization should be inherent in a system's control or communication primitives. However, the need to synchronize services in the presence of faults requires additional support features in the system model, sufficient to require that synchronization constitute a necessary and discrete system primitive. That is, synchronization cannot be treated as an add-on feature. » ([Suri et al., 1994])

L'analyse des propriétés des horloges est faite en utilisant un modèle mathématique qui est, à notre connaissance, original. Le modèle que nous proposons est utilisé pour établir toute une collection de résultats concernant les propriétés des horloges d'une façon systématique et unifiée. En particulier, nous présentons des résultats sur l'établissement de la séquence d'événements dans un système distribué basé sur l'estampillage local avec horloges synchronisées. Ces résultats sont en accord avec la littérature ([Veríssimo, 1994, Kopetz, 1997]) mais sous

des hypothèses moins restrictives et proposent un cadre plus élargi pour la validité de ces conditions. Notamment, nos résultats permettent d'inclure l'effet de la dérive des horloges, ce qui n'est pas présent dans les résultats que nous avons trouvé publiés.

En ce qui concerne les moyens qui sont disponibles au concepteur d'un système informatique pour synchroniser les horloges de son système, nous avons repris la classification proposée dans [Olson et Shin, 1994], qui considère trois grands groupes : les solutions basées sur la réception d'un signal de référence, celles basées sur le matériel et celles basées sur l'échange de messages. Bien que, comme souvent dans la recherche de classification, certaines solutions se trouvent à la frontière entre deux classes, la classification proposée permet, à notre avis, de mettre en évidence la valeur actuelle de solutions comme celles qui font l'objet de cette thèse (basées sur l'échange de messages entre les sites du système à travers le réseau) par rapport à d'autres solutions qui sont, nous aurions l'envie de dire, « très à la mode », telles que la synchronisation par GPS. Comme nous l'avons démontré durant la présentation de notre classification, les deux types de solutions correspondent à des cas distincts et complémentaires et, pour le moment, l'une n'est pas le remplaçant de l'autre.

Dans le cas des solutions pour la synchronisation basées sur l'échange de messages sur le réseau, notre intérêt s'est porté sur les algorithmes de synchronisation des horloges non-déterministes. Les algorithmes déterministes, selon ce qui a été démontré par Lundelius et Lynch ([Lundelius-Welch et Lynch, 1984]), subissent une limitation en ce qui concerne la meilleure précision qui peut être atteinte. Les algorithmes non-déterministes essayent de proposer des solutions auxquelles cette limite ne s'applique pas. Le prix à payer est une (petite) probabilité que le système ne réussisse pas à se synchroniser avec la précision souhaitée ; en principe, cette probabilité peut être choisie aussi proche de zéro qu'on le désire en utilisant un nombre suffisamment grand de messages. Il y a donc trois paramètres qui sont interdépendants et que l'utilisateur doit pouvoir maîtriser : la précision souhaitée, la probabilité de réussite et le nombre de messages échangés. En principe, l'utilisateur est intéressé par un nombre de messages le plus petit possible en garantissant toujours la synchronisation avec la probabilité et la précision souhaitées.

Avec ces objectifs à l'esprit, nous avons développé un modèle pour l'analyse de la performance des algorithmes de synchronisation des horloges non-déterministes. Pour la construction de ce modèle, il a fallu introduire un ensemble d'hypothèses, a savoir :

- l'indépendance des erreurs
- l'existence de la borne $\epsilon^{\text{tol}}$ qui garantit la précision $\delta^{\text{spec}}$
- l'existence de la fonction $f^{\text{err}}$, le majorant pour la probabilité que l'erreur dans une étape soit supérieure à $\epsilon^{\text{tol}}$.

Finalement, le modèle a permis l'établissement de la Condition de Garantie de la Synchronisation, qui nous permet de déterminer les paramètres du système qui garantissent que la précision souhaitée est atteinte avec la probabilité spécifiée. Cette condition, qui est une condition globale du système, est exprimée en termes de paramètres locaux d'un site ou en termes de paramètres du système facilement calculables, tels que le nombre de sites.

Un des objectifs dans la construction de ce modèle a été de rendre son utilisation la plus simple possible, même si la déduction des formules nécessaires pour y aboutir est d'une certaine façon compliquée. Comme nous l'avons démontré, la classification proposée regroupe les algorithmes publiés en classes qui présentent une formulation commune pour les grandeurs

et variables d'intérêt. Une fois déduite l'expression de la fonction de l'erreur ou du rapport entre l'erreur tolérée et la précision souhaitée pour un algorithme, ces valeurs pour les autres algorithmes de la classe peuvent être obtenues par mimétisme. Le résultat est une expression simple, dont l'utilisation est facile et claire. L'utilisation est simple ; la complexité est confinée aux étapes de déduction des expressions mathématiques, ce qui est fait (dans le pire des cas) une fois pour chaque algorithme.

Dans la littérature, la proposition de nouveaux algorithmes est souvent accompagnée de l'énumération des avantages du nouveau par rapport à l'ancien. Mais souvent cette comparaison est faite en changeant l'algorithme et modifiant les conditions sous lesquelles l'exécution se réalise (structure, type de réseau, ...). Notre modèle propose aussi l'analyse du comportement des algorithmes de synchronisation indépendemment de la structure ou du type de réseau utilisés. La comparaison entre la performance des différents algorithmes est donc plus facile, puisque l'impact de l'environnement d'exécution sur la performance de l'algorithme (c'est-á-dire, si le réseau la communication est par diffusion ou point-à-point, si la structure est distribuée ou centralisée, etc.) est représenté à travers certains paramètres. En plus de permettre d'analyser la performance de différents algorithmes sur un base commune, cette paramétrisation permet aussi d'évaluer l'effet de l'environnement sur la performance des algorithmes, aidant aussi à trouver quelles sont les solutions qui s'adaptent le mieux à un système donné.

La validation du modèle présenté a été faite sur un système distribué basé sur le réseau CAN. Ce système (tant la partie matérielle que la partie logicielle) a été totalement développé au long des travaux de cette thèse. Le développement intégral d'une telle structure présente, pour le cas d'un travail de thèse, des avantages et des inconvénients. L'incovénient principal est celui de l'effort qui doit être consacré à la mise en œuvre d'une telle solution. Dans notre cas, ceci correspond à un an de travail pour mettre en œuvre le système de base, c'est-à-dire, tout le matériel et un ensemble de fonctions logicielles qui fournit une première couche d'abstraction au dessus du niveau matériel (faire la programmation des compteurs, envoyer et recevoir de messages, etc.). La programmation des algorithmes de synchronisation a été faite à partir de cette base.

Du côté des avantages, le choix du développement intégral a permis de bien maîtriser toutes les étapes des algorithmes de synchronisation et, quand cela était nécessaire, d'optimiser le code de façon à améliorer la performance autant que possible. Ceci est important puisque l'objectif premier était la vérification d'un modèle théorique. Dans ce cas, il est indispensable de bien connaître toutes les déviations de l'implantation par rapport au modèle. Dans un système où il faut utiliser des « boites noires » (telles qu'un système d'exploitation) ce type de connaissances est forcément réduit. Un autre avantage qui n'est pas négligeable est celui du prix : notre système présente un coût réduit par rapport à d'autres systèmes qui sont offerts commercialement. En plus, le système développé présente des caractéristiques qui rendent le processus de mise en marche d'un algorithme distribué plus facile.

Les essais expérimentaux ont permis de valider les hypothèses qui avaient été faites et de vérifier que le modèle est conforme à la réalité. Si, d'un côté, les expériences se rapportent seulement à l'algorithme de Cristian, de l'autre, elles servent à démontrer que les hypothèses de base du modèle ne sont pas fantaisistes et que des expressions simples, comme la Condition de Garantie de la Synchronisation, peuvent aider à la conception des systèmes distribués.

La mise en marche d'un algorithme de synchronisation demande un temps, qui n'est pas négligeable, de mise au point de son implémentation. Quelqu'un qui a regardé la description

d'un algorithme de synchronisation peut vérifier que sa traduction dans un langage séquentielle (comme C) n'est pas immédiate. Étant donné un algorithme qui est destiné aux systèmes distribués, celui-ci nécessite l'implantation de processus concurrents et la gestion du temps. Une simple phrase comme « si le maître n'obtient pas une réponse après $x$ seconds, il décide de... » peut se traduire dans un programme dont la structure peut ne pas être du tout évidente au premier coup. Si l'on considère que ce maître peut être en train d'attendre plusieurs réponses simultanément (avec, probablement, différentes options selon l'endroit d'origine de la réponse), la chose se complique...

Nous considérons donc que nos objectifs ont été atteints. Nous avons proposé un modèle pour la performance des algorithmes non-déterministes de synchronisation des horloges qui permet d'obtenir une condition analytique facile à mettre en œuvre et qui garantit la réussite de la synchronisation sous les conditions spécifiées. Le modèle permet d'évaluer la performance de ces algorithmes indépendamment du réseau ou de la structure utilisés. Nous avons aussi proposé une formulation mathématique pour certains problèmes qui résultent de l'utilisation des horloges dans les systèmes distribués qui élargit son champ d'application, par rapport à l'existant.

La plate-forme développée est un sous-produit de la thèse qui continuera a être utilisée après la conclusion de ces travaux. Elle est déjà utilisée comme un outil de recherche dans le laboratoire de l'auteur. Du point de vue pédagogique (un aspect non-négligeable, puisque l'auteur a réalisé cette thèse en tant qu'enseignant-chercheur), elle peut être aussi bien utilisée pour l'enseignement que pour la recherche.

## 7.1 Perspectives et travaux futurs

Les perspectives de nos travaux se situent dans plusieurs domaines. Sur l'aspect strict de la continuation de ce qui est proposé dans cette thèse, il reste la question de la vérification du modèle avec d'autres algorithmes et avec d'autres réseaux. Il faudra reprendre les expériences décrites et les étendre aux autres algorithmes publiés et à d'autres réseaux. Un autre aspect à améliorer dans le modèle est de le rendre moins pessimiste.

L'extension du modèle à d'autres types de réseaux et la vérification des hypothèses sous-jacentes permettrait d'identifier quels types de réseaux pourront être les plus adéquats à l'applications des algorithmes non-déterministes. Mills, par exemple, affirme qu'il n'est pas possible de caractériser les délais comme des processus aléatoires stationnaires, car la taille des files d'attente peut augmenter et diminuer d'une façon chaotique et le trafic se fait souvent par des rafales ([Mills, 1991a]). Dans le cas particulier de l'Ethernet, les travaux de Leland *et al* ([Leland et al., 1994]) indiquent un comportement du type « *self-similar* » dans le trafic des messages.

Un aspect à prendre en compte dans le domaine de la synchronisation des horloges est celui de l'implantation, non seulement de l'algorithme mais de l'ensemble du système d'une façon intégrée. Il faut tenir compte que, pour les systèmes temps réel, le problème de la synchronisation des horloges n'est qu'un cas particulier d'un ensemble de problèmes qui doivent être résolus pour garantir sa bonne marche. La valeur de toute solution dépend de sa contribution pour l'aboutissement des objectifs auxquels les systèmes temps-réel se proposent. Notamment, il est impératif, pour chaque solution proposée, de mesurer et comparer non seulement les avantages, mais aussi le coût de sa mise en œuvre et son impact sur la performance du système.

La synchronisation des horloges offre des avantages évidents et ceci l'a rendu l'objet de l'attention de plusieurs chercheurs dans ces deux décennies. D'un côté, il n'est pas probable qu'on assiste à proposition de nouveaux algorithmes. Par contre, du côté de l'intégration de la synchronisation dans les systèmes, nous sommes d'avis qu'il y a encore des raisons pour continuer à faire de la recherche sur ce domaine. La mise en marche d'un service de synchronisation demande l'exécution d'un ensemble de tâches dans chaque site (identiques ou pas) et représente une surcharge pour le système de communication. On doit donc pouvoir maîtriser l'impact de la synchronisation sur la performance du système, de façon à bien évaluer les avantages apportés par rapport à son coût. Par exemple, les systèmes du type « time-triggered » se basent sur l'existence d'une base de temps commune à tous les sites du système distribué (horloges synchronisées). L'existence de cette base de temps permet, selon les avocats de cette solution, de présenter des meilleures caractéristiques par rapport aux systèmes « event-triggered ». La mesure de la dégradation de performance dûe à l'introduction de la synchronisation des horloges peut permettre de mieux comparer les avantages de chaque type de système. Éventuellement, l'obtention d'un modèle analytique pour la performance du système permettrait de déterminer des situations où un type de système présente un avantage clair sur l'autre et quelles sont les régions de décision.

Un aspect pour lequel les algorithmes non-déterministes pourraient contribuer est celui de la tolérance aux fautes et, en particulier, à la question de la dégradation de la performance. En se basant sur des hypothèses moins restrictives (notamment, en ne demandant pas des bornes pour le délai de transmission d'un message) les algorithmes non-déterministes pourront mieux s'adapter à une dégradation de la qualité du service de transmission de messages. Pour ces algorithmes, il n'existe pas une transition abrupte entre la région où les prémisses sont valides et une région où ces prémisses ne le sont plus. Par contre, le changement se fait d'une façon continue, par la diminution de la probabilité que la réussite se vérifie. Ce type de comportement pourrait présenter quelques avantages dans le domaine de la tolérance à la dégradation de performance.

Puisque la qualité de service offert par les algorithmes non-déterministes dépend du nombre de messages envoyés et qui est, à son tour, fonction de la description du délai de transmission comme une variable aléatoire, on peut envisager la possibilité d'avoir des algorithmes « adaptatifs ». Ces algorithmes seraient capables d'estimer des paramètres comme la valeur moyenne ou la variance du délai et de décider sur le nombre de messages à envoyer, de façon à garantir la précision et la probabilité de réussite spécifiées par l'utilisateur.

Un autre aspect que nous avons vérifié se rapporte à l'utilisation des algorithmes de régression. Ces algorithmes se basent sur l'estimation de la dérive entre deux horloges (la pente de la droite de régression). Cette valeur doit être obtenue avec une très grande précision ; il s'agit de calculer une valeur proche de 1 avec une erreur de l'ordre de quelques p.p.m.. Au départ, c'est un objectif ambitieux, surtout quand on considère que la mesure est faite en échangeant des messages sur le réseau. Pendant des essais préliminaires au système, nous avons comparé la valeur de la dérive mesurée par un dispositif spécialisé (une carte électronique qui recevait des signaux directement des différents sites du système) avec la valeur mesurée à travers les messages échangés sur le réseau. Les résultats ont montré que les deux valeurs sont très proches et que chaque site réussit effectivement à calculer sa dérive par rapport aux autres sites en utilisant le réseau. Ceci est permis par le fait que la dérive entre deux sites semble être stationnaire (où proche de stationnaire). La différence entre la fréquence de deux horloges peut être significative: nous avons trouvé des valeurs supérieures à $10^{-4}$ pour la dérive (ce

qui, en passant, est en accord avec les commentaires de Mills ([Mills et Thyagarajan, 1994])). Par contre, cette différence maintient sa valeur pour des intervalles de plusieurs heures. Ceci, allié au fait que le calcul de la dérive peut être fait de façon différentielle, permet d'estimer la dérive avec précision. Ces expériences ne sont pas présentées dans cette thèse car pour le moment elles ne sont plus que des indices et demandent un traitement plus soigneux. Un site qui connaît la valeur de la dérive de l'horloge locale peut, par exemple, se synchroniser à partir d'une horloge de référence. Si la communication avec le site où se trouve cette horloge échoue, il peut utiliser l'estimation locale de sa dérive pour s'« auto-synchroniser » : il utilise son horloge locale, la dernière indication de la référence et l'estimation de la dérive pour estimer la valeur du temps de référence. L'estimation locale du temps de référence possède ainsi une certaine « inertie » qui permettait au site de faire quelques estimations « en aveugle ».

Bien qu'en principe les algorithmes de synchronisation non-déterministes soient capables d'atteindre une précision aussi proche de zéro qu'on le souhaite, les résultats expérimentaux indiquent que ceci n'est pas vérifié dans la pratique ([Strong, 1996]). Les raisons se trouvent dans la capacité finie des moyens de transmission (il y a une limite pour le nombre de messages dans un intervalle de temps fini), la capacité finie des tampons de réception (le nombre de messages qu'un site peut recevoir est limité), la vélocité de traitement des messages et données au niveau des récepteurs, etc. Sur l'aspect de la garantie de qualité de service offert par ces algorithmes, il serait intéressant de rapporter les caractéristiques du système (en termes de capacité du réseau, type et nombre des tampons d'émission et réception, ...) au niveau de l'algorithme de synchronisation. Ceci correspondrait à faire intervenir, dans l'estimation de la performance, non seulement les paramètres intrinsèques de l'algorithme (tels que le nombre de messages échangés, la description des délais, ...) mais aussi une description de l'environnement dans lequel l'algorithme est exécuté. Éventuellement, on pourrait tomber sur la spécification d'une limite minimale pour la précision atteinte par les algorithmes non-déterministes.

En conclusion, nous croyons que le problème de la synchronisation des horloges, et notamment le cas des algorithmes non-déterministes, offre un champ de recherche qui n'est pas encore épuisé, soit dans une perspective centrée sur les algorithmes et leurs aspects spécifiques, soit sur l'aspect de la synchronisation comme une composante fondamentale des systèmes distribués temps-réel.

# Bibliographie

[Alari et Ciuffoleti, 1992] ALARI, G. et CIUFFOLETI, A. (1992). « Improving the Probabilistic Clock Synchronization Algorithm ». *Microprocessing and Microprogramming*, 35:463–468.

[Alari et Ciuffoletti, 1997] ALARI, G. et CIUFFOLETTI, A. (1997). « Implementing a Probabilistic Clock Synchronization Algorithm ». *Real-Time Systems*, 13(1):25–46.

[Anderson et Lee, 1981] ANDERSON, T. et LEE, P. A. (1981). *Fault Tolerance Principles and Practice.* Prentice-Hall International, Englewood Cliffs, NJ.

[Arvind, 1989] ARVIND, K. (1989). « A new probabilistic algorithm for clock synchronization ». Dans *Proc. Real-Time Syst. Symp.*, pages 330–339.

[Arvind, 1994] ARVIND, K. (1994). « Probabilistic Clock Synchronization in Distributed Systems ». *IEEE Trans. Parallel and Distributed Systems*, 5(5):474–487.

[Attiya et al., 1996] ATTIYA, H., HERZBERG, A., et RAJSBAUM, S. (1996). « Optimal Clock Synchronization under Different Delay Assumptions ». *SIAM J. Comput.*, 25(2):369–389.

[Avizienis, 1997] AVIZIENIS, A. (1997). « Toward Systematic Design of Fault-Tolerant Systems ». *IEEE Computer*, 30(4):51–58.

[Badami et Chbat, 1998] BADAMI, V. V. et CHBAT, N. W. (1998). « Home Appliances Get Smart ». *IEEE Spectrum*, 35(8):36–43.

[Baek et al., 1994] BAEK, Y., LEE, H.-K., et RYU, K. (1994). « A new hardware based fault-tolerant clock synchronisation scheme for real-time multiprocessor systems ». *Microelectronics and reliability*, 34(2):335–349.

[Ben-Or, 1983] BEN-OR, M. (1983). « Another advantage of free-choice: completely assynchronous agreement protocols ». Dans *Proc. 2nd ACM Symposium on Principles of Distributed Computing*, pages 27–30, Montreal, Ontario, Canada.

[Berger et al., 1998] BERGER, T., JOHN, D., et NEUMANN, K. J. (1998). « Hybrid Priorisation of CAN Messages to Improve Average Case Behavior ». Dans *Proceedings of ICC'98 — 5th International CAN Conference*, pages 09–02 - 09–07, San Jose, California, USA.

[Boasson, 1995] BOASSON, M. (1995). « Dreams and Realities in Distributed Real-Time Systems ». *International Journal of Mini and Microcomputers*, 17(1):1–10.

[Bohannon et Heffernan, 1998] BOHANNON, A. et HEFFERNAN, D. (1998). « A Structured Approach to Real-Time Application Software Design for the CAL Environment ». Dans *Proceedings of ICC'98 — 5th International CAN Conference*, pages 09–08–09–16, San Jose, California. CAN in Automation.

[Bosch, 1991] BOSCH (1991). « CAN Specification Version 2.0 ». Rapport Technique, Robert Bosch GmbH, Stuttgart.

[Bradley, 1997] BRADLEY, A. (1997). « *DeviceNet Specifications, Release 2.0* ».

[Bradley et al., 1998] BRADLEY, S. et OTHERS (1998). « CAN-Based Architecutre for Free-Falling Deep-Ocean Lander Vehicles ». Dans *Proceedings of ICC'98 — 5th International CAN Conference*, pages 12–19–12–26, San Jose, California, USA. CAN in Automation.

[Burns et Wellings, 1990] BURNS, A. et WELLINGS, A. (1990). *Real-Time Systems and their programming languages.* Addison-Wesley.

[CAN in Automation, 1996] CAN IN AUTOMATION (1996). « *CAN Application Layer for Industrial Applications* ».

[Cardoso, 1990] CARDOSO, M. E. (1990). A Aventura da Pneumonia. Dans *As minhas aventuras na República Portuguesa.* Assírio e Alvim, Lisboa.

[Choi et al., 1990] CHOI, B.-R., PARK, K. H., et KIM, M. (1990). « An improved hardware implementation of the fault-tolerant clock synchronization algorithm for large multiprocessor systems ». *IEEE Transactions on Computers*, C-39(3):404–407.

[Clegg et Marzullo, 1996] CLEGG, M. et MARZULLO, K. (1996). « Clock Synchronization in Hard Real-Time Distributed Systems ». Rapport Technique CS96-478, University of California, San Diego.

[Couvet et al., 1991] COUVET, D., FLORIN, G., et NATKIN, S. (1991). « A Statistical Clock Synchronization Algorithm for Anisotropi Networks ». Dans *10th Symp. on Reliable Distr. Syst.* IEEE.

[Cristian, 1989a] CRISTIAN, F. (1989a). « A probabilistic approach to distributed clock synchronization ». Dans *9th Intl. Conference on Distributed Computing Systems*, pages 288–296, Newport Beach. IEEE.

[Cristian, 1989b] CRISTIAN, F. (1989b). « Probabilistic clock synchronization ». *Distributed Computing*, 3:146–158.

[Cristian, 1991] CRISTIAN, F. (1991). « Understanding Fault-Tolerant Computing Systems ». *Communications of the ACM*, 34(2):57–78.

[Cristian et al., 1986] CRISTIAN, F., AGHILI, H., et STRONG, R. (1986). « Clock Synchronization in the presence of omission and performance faults, and processor joins ». Dans *Proc. Intern. Conf. on Fault-Tolerant Computing*, pages 218–223.

[Cristian et Fetzer, 1994] CRISTIAN, F. et FETZER, C. (1994). « Probabilistic Internal Clock Synchronization ». Dans *Proc. 13th Symposium on Reliable Distributed Systems*, pages 22–31. IEEE.

[Dana, 1997] DANA, P. H. (1997). « Global Positioning System (GPS) Time Dissemination for Real-Time Applications ». *Real-Time Systems*, 12(1):9–40.

[Davies et Wakerly, 1978] DAVIES, D. et WAKERLY, J. F. (1978). « Synchronization and Matching in Redundant Systems ». *IEEE Transactions on Computers*, C-27(6):513–539.

[Dolev et al., 1986] DOLEV, D., HALPERN, H., et STRONG, R. (1986). « On the possibility and impossibility of achieving clock synchronization ». *Journal of Computer and System Sciences*, 32(2):230–250.

[Drummond et Babaoğlu, 1993] DRUMMOND, R. et BABAOĞLU, Ö. (1993). « Low-Cost clock synchronization ». *Distributed Computing*, 6:193–203.

[Duda et al., 1987] DUDA, A., HARRUS, G., HADDAD, Y., et BERNARD, G. (1987). « Estimating Global Time in Distributed Systems ». Dans *Proc. 7th Int. Conf. on Distributed Computing Systems*, pages 299–306. IEEE.

[Ellingson et Kulpinski, 1973] ELLINGSON, C. et KULPINSKI, R. J. (1973). « Dissemination of system time ». *IEEE Trans. Comm.*, COM-23(5):605–624.

[Elloy, 1988] ELLOY, J. (1988). « Le Temps Réel : Rapport Ètabli Par Le Groupe de Réflexion Temps-Réel Du CNRS ». *Techniques et Sciences Informatiques*, 7(5):493–498.

[Enslow, 1978] ENSLOW, P. H. (1978). « What is a Distributed Data Processing System? ». *Computer*, 11(1):13–21.

[Etschberger, 1997] ETSCHBERGER, K. (1997). « CAN-based higher layer protocols and profiles ». Dans *Proc. 4th International Can Conference*, Berlin. Can in Automation.

[Fischer et al., 1990] FISCHER, M. J., LYNCH, N. A., et MERRIT, M. (1990). « *Easy Impossibility Proofs for Distributed Consensus Problems* », page?? Dans [Simons et Spector, 1989].

[Fonseca et al., 1998a] FONSECA, J. A., FONSECA, P., MOTA, A., et SANTOS, F. (1998a). « Sistema para Ensaio de Unidades Baseadas em Microprocessador Interligadas por Barramento de Campo ». Patente de invenção 102 232.

[Fonseca et al., 1998b] FONSECA, J. A., MOTA, A., SANTOS, F., FONSECA, P., AZEVEDO, J. L., et CURA, J. L. (1998b). « Affordable tools for teaching embedded systems ». Dans *ICECS'98 — 5th IEEE International Conference on Electronics, Circuits and Systems*, Lisboa, Portugal. IEEE. Poster.

[Fonseca et al., 1998c] FONSECA, P., FONSECA, J. A., et MAMMERI, Z. (1998c). « Can we trust Internet? A case study on clock synchronization algorithms ». Dans *Proceedings of INDC'98 — 7th IFIP/ICCC Conference on Information Networks and Data Communications*, pages 69–82, Aveiro, Portugal.

[Fonseca et Mammeri, 1996] FONSECA, P. et MAMMERI, Z. (1996). « A Framework for the Analysis of Non-Deterministic Clock Synchronisation Algorithms ». Dans BABAOĞLU, Ö. et MARZULLO, K., éditeurs, *Proceedings 10th Workshop on Distributed Algorithms (WDAG'96)*, Lecture Notes in Computer Science, pages 159–174. Springer-Verlag.

[Fonseca et al., 1998d] FONSECA, P., SANTOS, F., MOTA, A., et FONSECA, J. A. (1998d). « A Dinamically Reconfigurable CAN System ». Dans *Proc. 5th International CAN Conference*, San José, California, USA. Can In Automation.

[Fonseca et al., 1998e] FONSECA, P., SANTOS, F., MOTA, A., et FONSECA, J. A. (1998e). « *User Manual to the CANivete board* ». DET/Universidade de Aveiro, Aveiro, Portugal.

[Frerking, 1978] FRERKING, M. E. (1978). *Crystal Oscillator Design and Temperature Compensation.* Van Holland Reinhold.

[Gergeleit et Streich, 1994] GERGELEIT, M. et STREICH, H. (1994). « Implementing a Distributed High-Resolution Real-Time Clock Using the CAN-Bus ». Dans *Proceedings of the 1st International CAN-Conference*, Erlangen. CAN in Automation.

[Gifford et al., 1998] GIFFORD, D., KIRK, B., et LEISCH, B. (1998). « A Component Based Architecture for CAN Based Systems ». Dans *Proceedings of ICC'98 — 5th International CAN Conference*, pages 11–12–11–19, San Jose, California, USA. CAN in Automation.

[Gora et al., 1988] GORA, W., HERSOS, O., et TRIPATH, S. K. (1988). « Clock Synchronization in the Factory Floor ». *IEEE Transactions on Industrial Electronics*, 35(3):372–380.

[Guerraoui et Schiper, 1997] GUERRAOUI, R. et SCHIPER, A. (1997). « Software-Based Replication for Fault Tolerance ». *IEEE Computer*, 30(4):68–74.

[Gusella et Zatti, 1989] GUSELLA, R. et ZATTI, S. (1989). « The accuracy of the clock synchronization achieved by TEMPO in Berkeley Unix 4.3 BSD ». *IEEE Trans. on Software Engineering*, 15(7):847–853.

[Halpern et al., 1983] HALPERN, J., SIMONS, B., et STRONG, R. (1983). « An Efficient Fault-Tolerant Algorithm for Clock Synchronization ». Research Report RJ 4094, IBM Research Laboratory, San Jose, California.

[Hank, 1997] HANK, P. (1997). « PeliCAN: A New CAN Controller Supporting Diagnosis and System Optimization ». Dans *Proceedings of the 4th International CAN Conference*, pages 4–12 – 4–18.

[Hawking, 1988] HAWKING, S. (1988). *Breve História do Tempo.* Ciência Aberta. Gradiva, Lisboa.

[He et Mammeri, 1995] HE, J. et MAMMERI, Z. (1995). « Synchronisation des Horloges dans le Réseau FIP ». *Revue Électronique sur les Réseaux et l'Informatique Répartie*, 1(2):5–35.

[He et al., 1990] HE, J., MAMMERI, Z., et THOMESSE, J.-P. (1990). « Clock Synchronization in Real-Time Distributed Systems Based on FIP Field-Bus ». Dans *Proc. of the 2nd IEEE Conference on Dist. Comp. Syst.*, pages 135–141, Cairo. IEEE.

[He et al., 1994] HE, J., MAMMERI, Z., et THOMESSE, J.-P. (1994). « Modélisation des Techniques de Synchronization d'Horloges ». *Technique et science informatiques*, 13(2):185–221.

[He, 1993] HE, J.-Y. (1993). « *Modelisation de la synchronisation des horloges dans les systèmes répartis* ». Thèse de doctorat, Centre de Recherche en Informatique de Nancy, Nancy.

[Henderson et al., 1998] HENDERSON, W., KENDALL, D., ROBSON, A., et BRADLEY, S. (1998). « *Xrma*: An Holistic Approach to Performance Prediction to Distributed Real-Time CAN Systems ». Dans *Proceedings of ICC'98 — 5th International CAN Conference*, pages 09–17–09–24, San Jose, California, USA. CAN in Automation.

[Honeywell, 1996] HONEYWELL (1996). « *SDS Application Layer Protocol Specification — Version 2.0* ».

[Hopkins et al., 1978] HOPKINS, A. L., SMITH, T. B., et LALA, J. H. (1978). « FTMP - A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft ». *Proceedings of the IEEE*, 66(10):1221–1240.

[Intel Co., 1995] INTEL CO. (1995). « *82527 Serial Communications Controller* ». Intel Corporation, Santa Clara, CA, USA.

[ISO, 1993] ISO (1993). « ISO 11898 Road vehicles — Interchange of digital information — Controller Area Network (CAN) for high-speed communications ».

[ISO, 1994] ISO (1994). « ISO 11519-2 Road vehicles — Low-speed serial data communication — Part 2: Low speed controller area network (CAN) ».

[Kaplan, 1996] KAPLAN, E. D., éditeur (1996). *Understanding GPS: Principles and Applications*. The Artech House Mobile Communications Series. Artech House, Boston.

[Kartaschoff, 1991] KARTASCHOFF, P. (1991). « Synchronization in Digital Communication Systems ». *Proceedings of the IEEE*, 79(7):1019–1028.

[Kempthorne, 1952] KEMPTHORNE, O. (1952). *The Design and Analysis of Experiments*. John Wiley and Sons, New York.

[Kessels, 1984] KESSELS, J. L. W. (1984). « Two designs of a fault-tolerant clocking system ». *IEEE Trans. on Computers*, C-33(10):912–919.

[Khan, 1996] KHAN, A. R. (1996). « Workhorses of the Electronic Era ». *IEEE Spectrum*, 33(10).

[Klein, 1995] KLEIN, E. (1995). *Le Temps*. Dominos. Flammarion.

[Kopetz et Ramamritham, 1990] KOPETZ et RAMAMRITHAM (1990). « Distributed Real-Time Systems and Fault Tolerance ». Cours/Tutorial INRIA.

[Kopetz, 1989] KOPETZ, H. (1989). « Clock Synchronization Unit (CSU) Datasheet ». Research Report 22/89, Technische Universität Wien.

[Kopetz, 1992] KOPETZ, H. (1992). « Sparse Time versus Dense Time in Distributed Real-Time Systems ». Dans *Proc. 14th Intl. Conference on Distributed Computing Systems*, pages 460–467, Yokohama, Japan. IEEE.

[Kopetz, 1997] KOPETZ, H. (1997). *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, 1st edition.

[Kopetz et Ochsenreiter, 1987] KOPETZ, H. et OCHSENREITER, W. (1987). « Clock Synchronization in Distributed Real-time Systems ». *IEEE Trans. on Computers*, C-36(8):933–940.

[Krishna, 1990] KRISHNA, C. M. (1990). « Fault-Tolerant Synchronization Using Phase-Locked Clocks ». *Microelectronics and Reliability (UK)*, 30(2):275–287.

[Krishna et al., 1985] KRISHNA, C. M., SHIN, K. G., et BUTLER, R. W. (1985). « Ensuring Fault Tolerance of Phase-Locked Clocks ». *IEEE Trans. on Computers*, C34(8):752–756.

[Lamport, 1978] LAMPORT, L. (1978). « Time, clocks and the ordering of events in a distributed system ». *Communications of the ACM*, 21(7):558–65.

[Lamport et Melliar-Smith, 1984] LAMPORT, L. et MELLIAR-SMITH, P. M. (1984). « Byzantine Clock Synchronization ». Dans *Proc. 3rd Annual Conf. on Priciples of Distributed Computing*, pages 68–74, New York. ACM.

[Lamport et Melliar-Smith, 1985] LAMPORT, L. et MELLIAR-SMITH, P. M. (1985). « Synchronizing Clocks in the Presence of Faults ». *Journal of the ACM*, 32(1):52–78.

[Lawrenz, 1997] LAWRENZ, W. (1997). *CAN System Engineering: From Theory to Practical Applications*. Springer Verlag.

[Le Lann, 1990] LE LANN, G. (1990). « Synchronisation statistique d'horloges physiques: étude algorithmique ». Rapport Technique, INRIA.

[Le Lann, 1991] LE LANN, G. (1991). « Designing Real-Time Dependable Distributed Systems ». Rapport de recherche 1425, INRIA, Rocquencourt, France.

[Leland et al., 1994] LELAND, W. E., TAQQU, M. S., WILLINGER, W., et WILSON, D. W. (1994). « On the Self-Similar Nature of Ethernet Traffic ». *IEEE/ACM Transactions on Networking*, 2(1):1–15.

[Lennartsson, 1998] LENNARTSSON, K. (1998). « Design of Mixed Higher Layer Protocol Systems ». Dans *Proceedings of ICC'98 — 5th International CAN Conference*, pages 07–14 - 07–20, San Jose, California, USA. CAN in Automation.

[Lichtenecker, 1997] LICHTENECKER, R. (1997). « Terrestrial Time Dissemination ». *Real-Time Systems*, 12(1):41–61.

[Lindsey et al., 1985] LINDSEY, W. C., GHAZVINIAN, F., HAGGMAN, W. C., et DESSOUKI, K. (1985). « Network Synchronization ». *Proceedings of the IEEE*, 73(10):1445–1467.

[Lundelius et Lynch, 1984] LUNDELIUS, J. et LYNCH, N. (1984). « An upper and lower bound for clock synchronization ». *Information and Control*, 62:190–204.

[Lundelius-Welch et Lynch, 1984] LUNDELIUS-WELCH, J. et LYNCH, N. (1984). « A New Fault Tolerant Algorithm for Clock Synchronization ». Dans *Proc. 3rd Annual ACM Symp. on Principles of Distributed Computing*, pages 75–88, New York. ACM.

[Lundelius-Welch et Lynch, 1988] LUNDELIUS-WELCH, J. et LYNCH, N. (1988). « A New Fault Tolerant Algorithm for Clock Synchronization ». *Information and Computation*, 77(1):1–36.

[Maillet et Tron, 1995] MAILLET, E. et TRON, C. (1995). « On Efficiently Implementing Global Time for Performance Evaluation on Multiprocessor Systems ». *Journal of Parallel and Distributed Computing*, 28:84–93.

[Mammeri, 1997] MAMMERI, Z. (1997). « Synchronisation d'Horloges dans les Systèmes Répartis ». Dans *Ecole D'été Temps Réel '97*, pages 102–115, Futuroscope, Poitiers.

[Mammeri et He, 1996a] MAMMERI, Z. et HE, J. (1996a). « A Framework for Timing Performance Analysis of Clock Synchronization Techniques ». Rapport Technique, LIH Le Havre, Université du Havre, Le Havre, France.

[Mammeri et He, 1996b] MAMMERI, Z. et HE, J. (1996b). « Modeling and Timing Performance Analysis of Deterministic Clock Synchronization Algorithms ». Dans *Proc. 9th Int. Conference on Parallel and Distributed Systems*, pages 219–224, Dijon, France.

[Marzullo et Owicki, 1983] MARZULLO, K. et OWICKI, S. (1983). « Maintaining time in a distributed System ». Dans *Proceedings of the Second ACM Symposium on Principles of Distributed Computing*, pages 44–54.

[Mills, 1991a] MILLS, D. L. (1991a). « Internet Time Synchronization: The Network Time Protocol ». *IEEE Trans. on Commun.*, COM-39(10):1482–1493.

[Mills, 1991b] MILLS, D. L. (1991b). « On the Chronometry and Metrology of Computer Network Timescales and thir Application to the Network Time Protocol ». *Computer Communication Review*, 21(5):8–17.

[Mills, 1994] MILLS, D. L. (1994). « Improved Algorithms for Synchronizing Computer Network Clocks ». *Computer Communication Review*, 24(4):317–327.

[Mills et Thyagarajan, 1994] MILLS, D. L. et THYAGARAJAN, A. (1994). « Network Time Protocol Version 4 - Proposed Changes ». Rapport Technique 94-10-2, Electrical Engineering Department, University of Delaware.

[Mota, 1993] MOTA, A. (1993). « *Controlo Adaptativo — Um Caso Concreto* ». PhD thesis, Departamento de Electrónica, Universidade de Aveiro.

[Mullender, 1989] MULLENDER, S., éditeur (1989). *Distributed Systems.* Frontier Series. ACM Press, New York.

[Navet et Song, 1997] NAVET, N. et SONG, Y.-Q. (1997). « CAN Modeling : Towards Integrating Analytic Methods And Simulation ». Dans *OPnet European Users Group (OPEUG)*, Paris (France).

[Navet et Song, 1998] NAVET, N. et SONG, Y.-Q. (1998). « Design of Reliable Real-Time Applications Distributed over CAN (Controller Area Network) ». Dans *INCOM'98 - IFAC Int. Symp. On Information Control in Manufacturing*, pages 391–396, Nancy/Metz, France.

[Neumann, 1995] NEUMANN, P. G. (1995). *Computer-Related Risks.* ACM Press, New York.

[Olson et Shin, 1991] OLSON, A. et SHIN, K. G. (1991). « Probabilistic clock synchronization in large distributed systems ». Dans *Proc. Distributed Computing Systems*, pages 290–297.

[Olson et Shin, 1994] OLSON, A. et SHIN, K. G. (1994). « Fault-tolerant Clock Synchronization in Large Multicomputer Systems ». *IEEE Trans. Parallel and Distributed Systems*, 5(9):912–923.

[Olson et al., 1995] OLSON, A., SHIN, K. G., et JAMBOR, B. J. (1995). « Fault-Tolerant Clock Synchronization for Distributed Systems Using Continuous Synchronization Messages ». Dans *Proc. 25th International Symposium on Fault Tolerant Computing*, pages 154–163, Pasadena, California. IEEE.

[Paret, 1996] PARET, D. (1996). *Le bus CAN*. Réseaux Locaux. Dunod, Paris.

[Parzen, 1960] PARZEN, E. (1960). *Modern Probability Theory and its Applications*. John Wiley & Sons, Inc.

[Paterson, 1998] PATERSON, D. (1998). « Vulnerable Intel ». *The New York Times*.

[Patt-Shamir et Rajsbaum, 1994] PATT-SHAMIR, B. et RAJSBAUM, S. (1994). « A Theory of Clock Synchronization ». Dans *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 810–819, Montreal, Quebec, Canada. ACM.

[Perry, 1985] PERRY, K. J. (1985). « Randomized Byanztine Agreement ». *IEEE Transactions on Software Engineering*, SE-11(6):539–546.

[Philips, 1997] PHILIPS (1997). 80C51 Based 8-Bit Microcontrollers.

[Piettre, 1994] PIETTRE, B. (1994). *Philosophie et Science Du Temps*. Que Sais-Je? Presses Universitaires de France, Paris.

[Purdy, 1998] PURDY, D. R. (1998). « Target's Commom Digital Architecture ». Dans *Proceedings of ICC'98 — 5th International CAN Conference*, pages 03–08 - 03–15, San Jose, California, USA. CAN in Automation.

[Quinn, 1991] QUINN, T. J. (1991). « The BIPM and the Accurate Measurement of Time ». *Proceedings of the IEEE*, 79(7):894–905.

[Rabin, 1983] RABIN, M. O. (1983). « Randomized Byzantine Generals ». Dans *24th Annual Symp. on Foundations of Computer Science*, pages 403–409.

[Ramanathan et al., 1990a] RAMANATHAN, P., KANDHUR, D. D., et SHIN, K. G. (1990a). « Hardware assisted software clock synchronization for homogeneous distributed systems ». *IEEE Trans. on Computers*, C-39(4):514–524.

[Ramanathan et al., 1990b] RAMANATHAN, P., SHIN, K. G., et BUTLER, R. W. (1990b). « Fault-Tolerant clock synchronization in Distributed Systems ». *IEEE Computer*, 23(10):33–42.

[Rangarajan et Tripathi, 1991] RANGARAJAN, S. et TRIPATHI, S. K. (1991). « Efficient Synchronization of clocks in a distributed system ». Dans *Proceedings of Real-Time Systems Symposium*, pages 22–31.

[Reichenbach, 1958] REICHENBACH, H. (1958). *Philosophy of Space and Time*. Dover Publications, Inc., New York.

[Rodriguez et Campelo, 1998] RODRIGUEZ, F. et CAMPELO, J. (1998). « EDF Message Scheduling on a CAN Network ». Dans *Proceedings of ICC'98 — 5th International CAN Conference*, pages 10–27–10–34, San Jose, California, USA. CAN in Automation.

[Ross, 1987] ROSS, S. M. (1987). *Introduction to Probability and Statistics for Engineers and Scientists*. Wiley Series on Probability and Mathematical Statistics. John Wiley & Sons, New York.

[Rüdiger, 1998] RÜDIGER, R. (1998). « Evaluating the Temporal Behaviour of CAN Based Systems by Means of a Cost Functional ». Dans *Proceedings of ICC'98 — 5th International CAN Conference*, pages 10–09–10–26, San Jose, California, USA. CAN in Automation.

[Rufino et Veríssimo, 1995] RUFINO, J. et VERÍSSIMO, P. (1995). « A study on the inaccessibility characteristics of the Controller Area Network ». Dans *Proceedings 2nd International CAN Conference 95*, London. CAN in Automation.

[Schlett, 1998] SCHLETT, M. (1998). « Trends in Embedded-Microprocessor Design ». *IEEE Computer*, 31(8):44–49.

[Schmid, 1995] SCHMID, U. (1995). « Synchronized Universal Time Coordinated for Distributed Real-Time Systems ». *Control Engineering Practice*, 3(6):877–884.

[Schneider, 1986] SCHNEIDER, F. B. (1986). « A Paradigm for Reliable Clock Synchronization ». Dans *Proc. Advanced Seminar on Real-Time Local Area Networks*, pages 85–104, Bandol, France.

[Schneider, 1987] SCHNEIDER, F. B. (1987). « Understanding protocols for Byzantine clock synchronization ». Technical Report 87-859, Department of Computer Science, Cornell University, Ithaca, New York.

[Schossmaier et al., 1997] SCHOSSMAIER, K., SCHMIDT, U., HORAUER, M., et LOY, D. (1997). « Specification and Implementation of the Universal Time Coordinated Synchronization Unit (UTCSU) ». *Real-Time Systems*, 12(1):295–328.

[Schreiber, 1992] SCHREIBER, F. (1992). « Is Time a Real-Time? An Overview of Time Ontology in Informatics ». Dans *NATO Avanced Study Institute on Real Time Computing*, Sint Marten, Dutch Antilles.

[Schulze, 1998] SCHULZE, W. (1998). « Implementation of a CAN Gateway Between DeviceNet and CANopen ». Dans *Proceedings of ICC'98 — 5th International CAN Conference*, pages 03–02 - 03–07, San Jose, California, USA. CAN in Automation.

[Sedgewick, 1988] SEDGEWICK, R. (1988). *Algorithms*. Addison-Wesley Series in Computer Science. Addison-Wesley, 2 edition.

[Shin et Ramanathan, 1987] SHIN, K. G. et RAMANATHAN, P. (1987). « Clock Synchronization of a Large Multiprocessor Systems in the presence of Malicious Faults ». *IEEE Trans. on Computers*, C-36(1):2–12.

[Shin et Ramanathan, 1988] SHIN, K. G. et RAMANATHAN, P. (1988). « Transmission Delays in Hardware Clock Synchronization ». *IEEE Transactions on Computers*, 37(11):1465–1467.

[Simons et al., 1989] SIMONS, B., LUNDELIUS-WELCH, J., et LYNCH, N. (1989). An overview of clock synchronization. Dans [Simons et Spector, 1989], pages 84–96.

[Simons et Spector, 1989] SIMONS, B. et SPECTOR, A., éditeurs (1989). *Fault Tolerant Distributed Computing*. Numéro 448 dans Lecture Notes in Computer Science. Springer-Verlag.

[Singhal et Casavant, 1991] SINGHAL, M. et CASAVANT, T. L. (1991). « Distributed Computing Systems ». *IEEE Computer*, 24(8).

[Song et al., 1997] SONG, Y.-Q., SIMONOT-LION, F., et BELISSENT, P. (1997). « Validation des applications temps réel distribuées autour du réseau CAN à l'aide de l'évaluation de performances ». Dans *Real-Time and Embedded Systems, RTS'97*, pages 243–272, Paris (France).

[Srikanth et Toueg, 1987] SRIKANTH, T. et TOUEG, S. (1987). « Optimal Clock Synchronization ». *Journal of the ACM*, 34:626–645.

[Sterzbach, 1997] STERZBACH, B. (1997). « GPS-Based Clock Synchronization on a Mobile, Distributed Real-Time System ». *Real-Time Systems*, 12(1):63–76.

[Strong, 1996] STRONG, R. (1996). Communication personnelle.

[Sullivan et Levine, 1991] SULLIVAN, D. B. et LEVINE, J. (1991). « Time Generation and Distribution ». *Proceedings of the IEEE*, 79(7):906–914.

[Suri et al., 1994] SURI, N., HUGUE, M. M., et WALTER, C. J. (1994). « Synchronization Issues in Real-Time Systems ». *Proceedings of the IEEE*, 82(1):41–54.

[Tindell, 1995] TINDELL, K. (1995). « Analysis of hard real-time communications ». *Real-Time Systems*, 9(2):147–171.

[Tindell, 1998] TINDELL, K. (1998). « The Volcano System ». Dans *Proceedings of ICC'98 — 5th International CAN Conference*, pages 03–16 - 03–22, San Jose, California, USA. CAN in Automation.

[Tindell et Burns, 1994a] TINDELL, K. et BURNS, A. (1994a). « Guaranteeing message latencies for distributed safety-critical hard real-time control networks ». Rapport Technique YCS 229, Dept. of Computer Science, University of York, England.

[Tindell et Burns, 1994b] TINDELL, K. et BURNS, A. (1994b). « Guaranteeing Message Latencies on Control Area Network (CAN) ». Dans *Proc. 1st International CAN Conference*, Mainz, Germany. CiA — CAN in Automation.

[Turley, 1998] TURLEY, J. (1998). « 32- and 64- Bit Processors: An Embarrassment of Riches ». *Embedded Systems Programming*, 11(12):85–94.

[Vasanthada et Marinos, 1988] VASANTHADA, N. et MARINOS, P. N. (1988). « Synchronization of Fault-Tolerant Clocks in the Presence of Malicious Faults ». *IEEE Trans. on Computers*, C-37(4):440–448.

[Vasanthada et al., 1986] VASANTHADA, N., MARINOS, P. N., et MERSTEN, G. S. (1986). « Design and Performance Evaluation of Mutually Synchronized Fault-Tolerant Clock Systems ». Dans *Proc. 16th Symposium on Fault Tolerant Computing*, pages 206–211. IEEE.

[Vega, 1996] VEGA, L. (1996). « *Modèles de Coopération et de Communication Entre Processus Temps-Réel Répartis* ». PhD thesis, Insttut National Polytechnique de Lorraine, Nancy.

[Veríssimo, 1994] VERÍSSIMO, P. (1994). « Ordering and Timeliness Requirements of Dependable Real-Time Programs ». *Real-Time Systems*, 7(2):105–128.

[Verissimo et Rodrigues, 1992] VERISSIMO, P. et RODRIGUES, L. (1992). « A posteriori agreement for fault-tolerant clock synchronization on broadcast networks ». Dans *Proc. of FTCS*, pages 527–536.

[Veríssimo et al., 1997] VERÍSSIMO, P., RODRIGUES, L., et CASIMIRO, A. (1997). « CesiumSpray: A Precise and Accurate Global Time Service for Large-scale Systems ». *Real-Time Systems*, 12(1):243–294.

[Vig, 1992] VIG, J. R. (1992). « Introduction to Quartz Frequency Standards ». Rapport Technique SLCET-TR-92-1, US Army Communications-Electronics Command.

[Wannenmacher et Halang, 1994] WANNENMACHER, M. et HALANG, W. A. (1994). « GPS-based timing and clock synchronization for real time computers ». *Electronics Letters*, 30(20):1653–1654.

[Wensley et al., 1978] WENSLEY, J. W., LAMPORT, L., GOLDBERG, J., GREEN, M., LEVITT, K. N., MELLIAR-SMITH, P. M., SHOSTAK, R. E., et WEINSTOCK, C. B. (1978). « SIFT: Design and analysis of a fault-tolerant computer for aircraft control ». *Proceedings of the IEEE*, 66(10):1240–1255.

[Yang et Marsland, 1993] YANG, Z. et MARSLAND, T. A. (1993). « Annotated Bibliography on Global States and Times in Distributed Systems ». *ACM Operating Systems Review*, 27(3):55–74.

INSTITUT NATIONAL
POLYTECHNIQUE
DE LORRAINE

# AUTORISATION DE SOUTENANCE DE THESE DU DOCTORAT DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

o0o

VU LES RAPPORTS ETABLIS PAR :

**Monsieur COTTET Françis, Professeur, Ecole Nationale Supérieure de Mécanique et d'Aérotechnique Futuroscope,**

**Monsieur CARDEIRA Carlos, Professeur, Instituto Superior Técnico Lisbonne (Portugal),**

**Monsieur BOUDJLIDA Nacer, Professeur, LORIA/Université H.Poincaré Nancy I.**

Le Président de l'Institut National Polytechnique de Lorraine, autorise :

**Monsieur FARIA DA FONSECA Pedro Nicolau**

à soutenir devant un jury de l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE, une thèse intitulée :

**"Modélisation et validation des algorithmes non-déterministes de synchronisation des horloges"**

en vue de l'obtention du titre de :

**DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE**

Spécialité : **"INFORMATIQUE"**

Fait à Vandoeuvre le, **6 Avril 1999**

Le Président de l'I.N.P.L.,

J. HARDY

INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE NANCY

NANCY BRABOIS
2, AVENUE DE LA
FORET-DE-HAYE
BOITE POSTALE 3
F - 5 4 5 0 1
VANDŒUVRE CEDEX

TEL. 33/03.83.59.59.59
FAX. 33/03.83.59.59.55

# Resumé

## Modélisation et validation des algorithmes non-déterministes de synchronisation des horloges

Cette thèse traite le problème d'analyse et de conception des algorithmes de synchronisation non-déterministes dans les systèmes répartis. Les algorithmes non-déterministes constituent une solution intéressante pour le problème de la synchronisation des horloges, ce qui est témoigné par l'intérêt qui leur a été porté pendant les dernières années.

Les algorithmes de synchronisation non-déterministes utilisent des techniques statistiques ou probabilistes et permettent d'obtenir une meilleure précision par rapport aux algorithmes déterministes, le prix à payer étant la probabilité que le système ne réussisse pas à se synchroniser avec la précision souhaitée. Cette probabilité d'échec tend vers zéro avec le nombre de messages échangés et elle peut être réduite à une valeur aussi petite qu'on le souhaite avec un nombre de messages suffisamment grand. Malheureusement, l'évaluation des différentes propositions publiées dans la littérature est difficile, principalement à cause de l'inexistence d'une base commune pour établir les comparaisons.

Nous proposons un modèle analytique pour le fonctionnement des algorithmes de synchronisation non-déterministes. Le but est de trouver une expression que le concepteur d'un système distribué puisse utiliser pour calculer le nombre de messages requis pour un certain algorithme, de façon à fournir la synchronisation avec la précision et la probabilité spécifiées. Ce résultat est la Condition de Garantie de Synchronisation, qui définit une condition suffisante qui garantit le succès de la synchronisation non-déterministe sous les conditions spécifiées. Cette condition est établie à partir de paramètres locaux d'un site et de paramètres du système qui sont facilement calculables, tels que le nombre de sites et les paramètres qui décrivent le délai comme une variable aléatoire.

Les conditions sous-jacentes au modèle proposé sont vérifiées expérimentalement à l'aide d'une plate-forme basée sur le réseau CAN que nous avons développée. Le test des algorithmes sur cette plate-forme a permis de vérifier la validité des hypothèses qui sont associées au modèle.



# Abstract

## Modelling and validation of non-deterministic clock synchronization algorithms

This thesis addresses the problem of analysing and designing non-deterministic clock synchronisation algorithms in distributed systems. Non-deterministic algorithms are a promising solution to the clock synchronisation problem, which can be testified by the attention they have received in recent years.

Non-determinist clock synchronisation algorithms use statistical or probabilistic techniques and they allow a better precision than with deterministic ones; the price to pay is a small probability that the system will fail to synchronise to the desired precision. This probability of failure can be made as small as desired by sending a sufficiently large number of messages. Unfortunately, assessing different algorithms is a difficult task, specially because we lack a common ground to establish comparisons.

We propose an analytical model for the behaviour of non-deterministic clock synchronisation algorithms. The aim is to find an expression that the distributed systems designer can use to estimate the required number of messages, in order to guarantee that a certain algorithm will synchronise to the desired precision and probability of success. This result is a Sufficient Condition for Synchronisation, which states the conditions that must be fulfilled in order to guarantee the desired precision and probability of success. This condition is based on local parameters and on system parameters that are easily computed, such as the number of sites and the description of the delay as a random variable.

The underlying assumptions are experimentally verified. For this purpose, we developed a test-bed based on the Controller Area Network. The results validate the assumptions used for the model.

# Resumé

## Modélisation et validation des algorithmes non-déterministes de synchronisation des horloges

Cette thèse traite le problème d'analyse et de conception des algorithmes de synchronisation non-déterministes dans les systèmes répartis. Les algorithmes non-déterministes constituent une solution intéressante pour le problème de la synchronisation des horloges, ce qui est témoigné par l'intérêt qui leur a été porté pendant les dernières années.

Les algorithmes de synchronisation non-déterministes utilisent des techniques statistiques ou probabilistes et permettent d'obtenir une meilleure précision par rapport aux algorithmes déterministes, le prix à payer étant la probabilité que le système ne réussisse pas à se synchroniser avec la précision souhaitée. Cette probabilité d'échec tend vers zéro avec le nombre de messages échangés et elle peut être réduite à une valeur aussi petite qu'on le souhaite avec un nombre de messages suffisamment grand. Malheureusement, l'évaluation des différentes propositions publiées dans la littérature est difficile, principalement à cause de l'inexistence d'une base commune pour établir les comparaisons.

Nous proposons un modèle analytique pour le fonctionnement des algorithmes de synchronisation non-déterministes. Le but est de trouver une expression que le concepteur d'un système distribué puisse utiliser pour calculer le nombre de messages requis pour un certain algorithme, de façon à fournir la synchronisation avec la précision et la probabilité spécifiées. Ce résultat est la Condition de Garantie de Synchronisation, qui définit une condition suffisante qui garantit le succès de la synchronisation non-déterministe sous les conditions spécifiées. Cette condition est établie à partir de paramètres locaux d'un site et de paramètres du système qui sont facilement calculables, tels que le nombre de sites et les paramètres qui décrivent le délai comme une variable aléatoire.

Les conditions sous-jacentes au modèle proposé sont vérifiées expérimentalement à l'aide d'une plate-forme basée sur le réseau CAN que nous avons développée. Le test des algorithmes sur cette plate-forme a permis de vérifier la validité des hypothèses qui sont associées au modèle.



# Abstract

## Modelling and validation of non-deterministic clock synchronization algorithms

This thesis addresses the problem of analysing and designing non-deterministic clock synchronisation algorithms in distributed systems. Non-deterministic algorithms are a promising solution to the clock synchronisation problem, which can be testified by the attention they have received in recent years.

Non-determinist clock synchronisation algorithms use statistical or probabilistic techniques and they allow a better precision than with deterministic ones; the price to pay is a small probability that the system will fail to synchronise to the desired precision. This probability of failure can be made as small as desired by sending a sufficiently large number of messages. Unfortunately, assessing different algorithms is a difficult task, specially because we lack a common ground to establish comparisons.

We propose an analytical model for the behaviour of non-deterministic clock synchronisation algorithms. The aim is to find an expression that the distributed systems designer can use to estimate the required number of messages, in order to guarantee that a certain algorithm will synchronise to the desired precision and probability of success. This result is a Sufficient Condition for Synchronisation, which states the conditions that must be fulfilled in order to guarantee the desired precision and probability of success. This condition is based on local parameters and on system parameters that are easily computed, such as the number of sites and the description of the delay as a random variable.

The underlying assumptions are experimentally verified. For this purpose, we developed a test-bed based on the Controller Area Network. The results validate the assumptions used for the model.