RAIRO. INFORMATIQUE THÉORIQUE ET APPLICATIONS

BERNADETTE CHARRON-BOST
GÉRARD TEL

Calculs approchés de la borne inférieure de valeurs réparties

RAIRO. Informatique théorique et applications, tome 31, n° 4 (1997), p. 305-330

<http://www.numdam.org/item?id=ITA_1997__31_4_305_0>

© AFCET, 1997, tous droits réservés.

L'accès aux archives de la revue « RAIRO. Informatique théorique et applications » implique l'accord avec les conditions générales d'utilisation (http://www.numdam.org/conditions). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM
Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques
http://www.numdam.org/

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 31, n° 4, 1997, pp. 305-330)

CALCULS APPROCHÉS DE LA BORNE INFÉRIEURE DE VALEURS RÉPARTIES (*)

par Bernadette CHARRON-BOST ([1]) et Gerard TEL ([2]) ([3])

Résumé. – *Le problème de DIA (Distributed Infinimum Approximation) consiste à calculer une approximation de la borne inférieure de valeurs réparties sur les différents sites et dans les différents canaux d'un système réparti. Ces valeurs sont supposées pouvoir changer au cours du temps mais selon des règles précises. Nous montrons que l'approximation du temps virtuel (GVT) et la détection de terminaison répartie sont des cas particuliers de DIA. Différents algorithmes répartis résolvant le problème de DIA et correspondant à différentes hypothèses portant sur les communications (synchrones, FIFO, causalement ordonnées, quelconques) sont ensuite proposés. Tous ces algorithmes déterminent des périodes d'observation du calcul de base. Nous exhibons alors une condition suffisante pour qu'un algorithme de vague puisse être utilisé, à l'intérieur d'un tel algorithme de DIA, pour définir ces périodes d'observation.*

Abstract. – *The Distributed Infinimum Approximation (or DIA, for short) problem is to compute or approximate a lower bound on a set of values distributed over different sites and channels of a distributed system. The values themselves change in time, but according to some rules implying monotonicity of their lower bound. We demonstrate that approximation of the Global Virtual Time (GVT) and distributed termination detection are instances of DIA. We propose several classes of distributed solutions for the DIA problem, corresponding to different assumptions about the communication subsystem (synchronous, FIFO, causal order, unrestricted). All algorithms need to determine observation periods during which the local control algorithm observes the local basic computation (including its communications). It is demonstrated that wave algorithms can be used as a building block inside DIA algorithms for the determination of observation periods, for combining the local observation results, and for disseminating the subsequent approximations of the lower bound.*

1. INTRODUCTION

Dans les systèmes informatiques classiques centralisés, l'état du système est une donnée bien définie et accessible instantanément par le système lui-même. A l'inverse, dans un système réparti, un processus ne peut connaître l'état des autres processus que par échanges de messages et donc avec des

(*) Reçu février 1994. Version révisée juin 1997.

([1]) Laboratoire d'Informatique LIX, École Polytechnique, 91128 Palaiseau Cedex, France. E-mail: charron@lix.polytechnique.fr.

([2]) Département d'Informatique, Univ. de Utrecht, B.P. 80.089, 3508 TB Utrecht, Pays Bas. Email: gerard@cs.ruu.nl.

([3]) Le travail de cet auteur a été en partie financé par ESPRIT Basic Research Action No. 7141 (projet ALCOM II: *Algorithms and Complexity*).

délais plus ou moins longs. Par conséquent, dans un tel système, un processus ne peut pas calculer l'état global *courant* du système.

D'autre part, les différentes informations locales collectées par un site calculant un état global n'ont pas toutes la même antériorité et peuvent donc former, une fois rassemblées, une vue incohérente du système. Cependant, dans bien des cas, un processus a besoin d'informations précises concernant l'état global (même ancien) du système afin de prendre une décision locale pertinente (arrêt d'un calcul local, retour en arrière, ...).

Chandy et Lamport [1] décrivent un algorithme qui calcule une « image » (un instantané) cohérente mais ancienne du système, *i.e.*, un état global par lequel le système est passé ([1]). Cet ancien état global donne des informations concernant l'état courant du système à condition de ne s'intéresser qu'à des propriétés *stables* du système, *i.e.*, des propriétés qui, une fois vraies, ne sont pas violées par les actions exécutées ultérieurement par le système. Une telle propriété peut être vue comme une *fonction booléenne monotone* de l'état du système (la monotonie traduisant exactement la stabilité). En outre, un grand nombre de propriétés stables classiquement étudiées dans les systèmes répartis sont des conjonctions de propriétés locales (*CPL* en abrégé) aux processus et aux canaux. C'est le cas, par exemple, de la terminaison répartie et de l'interblocage (deadlock).

Tel [15] généralise la notion de propriété stable en considérant les fonctions monotones à valeurs dans un treillis quelconque. La classe des propriétés CPL admet alors comme généralisation la classe des fonctions dont la valeur ne dépend que des valeurs de variables locales aux processus et de valeurs « présentes » dans les canaux (une définition précise sera donnée dans la suite). La monotonie des fonctions calculées découlera de règles imposées à l'évolution des valeurs dont les fonctions dépendent. Les algorithmes permettant d'évaluer ces fonctions monotones sont appelés algorithmes de *DIA (Distributed Infinimum Approximation)* (*cf.* [15]).

La notion d'algorithme de DIA tire son intérêt du fait qu'elle permet de regrouper des algorithmes répartis bien connus et qui semblaient jusqu'à présent non reliés. Par exemple, les algorithmes de détection de terminaison répartie, les algorithmes calculant une approximation du temps virtuel global ainsi que certains algorithmes de ramasse-miettes sont tous des algorithmes de DIA.

([1]) Compte tenu des « phénomènes relativistes » dans les systèmes répartis (*cf.* [14]), il serait plus correct de parler d'état global par lequel le système *aurait pu* passer.

À partir d'un algorithme qui prend un instantané du système, il est facile de construire un premier exemple d'algorithme de DIA : il suffit d'itérer l'algorithme de prise d'instantané et de calculer, pour chaque état global cohérent construit par cet algorithme, la valeur de la fonction monotone dont on veut une approximation. Cette méthode n'utilise que très partiellement les propriétés des fonctions monotones considérées ([2]) ; on peut donc raisonnablement penser qu'il existe des solutions plus simples car mieux adaptées au problème de DIA. De fait, nous proposons dans cet article différents algorithmes résolvant ce problème qui ne nécessitent pas le calcul d'une succession d'états globaux cohérents. La méthode utilisée dans tous ces algorithmes consiste à observer le système, non pas dans un état global cohérent précis, mais durant un certain laps de temps. Chaque processus sera observé localement pendant un intervalle de temps appelé *période d'observation locale*. Nous montrerons que, contrairement aux clichés pris localement pour calculer un état global cohérent, ces différentes périodes d'observation locale ne doivent être que « faiblement » synchronisées pour que les algorithmes que nous décrivons soient corrects.

La définition du problème de DIA ainsi que deux des trois algorithmes que nous donnons pour résoudre ce problème figurent déjà dans [15]. La différence entre cet article et le chapitre 4 de [15] réside essentiellement dans les preuves de correction des algorithmes qui sont données. En effet, les preuves qui figurent dans [15] utilisent de façon fondamentale la notion de temps physique global (extérieur au système). Or, l'absence de référentiel de temps absolu rend discutable ce type de preuve. Lamport [9] a montré que les événements d'un calcul réparti ne sont pas (totalement) ordonnés par le temps mais par une relation de causalité. La notion de *coupe cohérente* doit être substituée à celle d'état global du système à un moment donné. Nous adoptons ici ce point de vue et présentons des preuves nouvelles qui n'utilisent que la structure causale des calculs. D'autre part, nous définissons des propriétés de sûreté et de vivacité des algorithmes locaux et globaux qui composent chacun des algorithmes de DIA que nous décrivons. Nous montrons alors que ces propriétés assurent à leur tour la sûreté et la vivacité des algorithmes de DIA. Cette présentation modulaire des critères de correction nous permet de

([2]) Plus précisément, on peut remarquer que les fonctions considérées dans le problème de DIA sont *fortement monotones* au sens de Lai et Yang [8]. On obtient aisément un minorant d'une telle fonction en calculant sa valeur pour un état global quelconque (cohérent ou non) du système. Nous ne donnons pas plus de détails concernant la propriété de forte monotonie car nous ne l'utilisons pas dans la suite de notre travail.

généraliser aux communications causalement ordonnées les résultats obtenus dans [15] pour les seules communications synchrones (*cf.* section 4.2).

L'article est organisé de la façon suivante. Dans la section 2, nous décrivons formellement un modèle de calcul réparti et rappelons quelques définitions de base de ce modèle. Nous donnons ensuite une spécification précise du problème de DIA et nous montrons que des problèmes bien connus dans les systèmes répartis, comme le problème de la terminaison répartie ou celui du calcul du temps global virtuel sont des cas particuliers de ce problème. Dans la section 3, nous décrivons la structure générale des algorithmes de DIA que nous construisons : chacun de ces algorithmes est composé d'un algorithme *global* (ou *de communication*), déterminant des périodes pendant lesquelles le calcul de base est observé, et d'un algorithme *local* (ou encore *d'observation*) qui observe le calcul de base. Nous établissons ensuite un critère assurant la correction d'un tel algorithme. Différents algorithmes locaux sont proposés en section 4. Dans la section 5, nous montrons comment implanter des algorithmes globaux à partir d'algorithmes répartis classiques appartenant à la famille des algorithmes *de vague* (*cf.* [16, Chapitre 6]). Nous revenons ensuite au problème particulier de la détection de terminaison répartie dans la section 6.

2. LE PROBLÈME DE DIA

2.1. Calculs d'un Système Réparti

Dans ce paragraphe, nous décrivons le modèle de système réparti que nous considérerons dans la suite de cet article et nous rappelons brièvement certaines définitions de base pour ce modèle ; voir [2], [14].

Un *système réparti* est constitué d'un ensemble fini $\mathbb{P}$ de processus. Chaque processus exécute un programme séquentiel ; un calcul local effectué par un processus est donc modélisé par une suite d'événements. Un *événement* est une émission de message, une réception de message ou une action interne, *i.e.*, un événement qui ne modifie que l'état local du processus qui l'exécute. Les communications entre processus sont supposées point à point (chaque message n'a qu'un seul émetteur et qu'un seul récepteur) et asynchrones. On supposera donc seulement ([3]) que tout message reçu a été envoyé (la réciproque peut être fausse pour les messages en transit).

([3]) Contrairement au cas des communications synchrones pour lesquelles on doit supposer de plus que tout message émis est aussi reçu.

Aucune notion temporelle n'apparaît dans notre modèle. Cependant, deux événements peuvent être reliés l'un à l'autre par le fait que l'un est une cause de l'autre. Dans un système réparti, la dépendance causale entre événements résulte d'une part de l'ordre total dans lequel chaque processus exécute les événements et d'autre part du fait que l'émission d'un message est une cause de sa réception. Plus formellement, on définit la relation de causalité de la façon suivante :

DÉFINITION 2.1 : *La* relation de causalité $\prec$ *est la plus petite relation transitive définie sur l'ensemble des événements produits par un système réparti satisfaisant aux deux conditions suivantes :*

1. *si a et b sont deux événements différents exécutés par le même processus et si a a lieu avant b alors $a \prec b$;*
2. *si s est l'émission d'un message et r sa réception alors $s \prec r$.*

DÉFINITION 2.2 : *Un* calcul réparti *est une famille de calculs locaux (un par processus) pour laquelle la relation de causalité $\prec$ est une relation acyclique.*

Dans ce cas, la relation de causalité est une relation d'ordre; on note $\preceq$ sa fermeture réflexive. Si c est un calcul du système réparti $\mathbb{P}$ et si p est un processus de $\mathbb{P}$, c_p désigne le calcul local exécuté par p au cours de c. Le calcul c s'écrit alors $c = \{c_p : p \in \mathbb{P}\}$. La restriction de la relation $\prec$ (resp. $\preceq$) au calcul local c_p est notée $\prec_p$ (resp. $\preceq_p$).

Dans la suite, nous utiliserons les *diagrammes espace-temps* proposés par Lamport [9] pour représenter un calcul réparti et la relation de causalité $\prec$. Dans ces diagrammes, les lignes horizontales représentent les calculs locaux exécutés par les différents processus. Les événements sont repérés par des points le long de ces lignes. Un message est représenté par un vecteur dont l'origine est l'émission du message et dont l'extrémité est sa réception. La figure 1 représente le diagramme espace-temps d'un calcul réparti sur trois processus p, q et r.

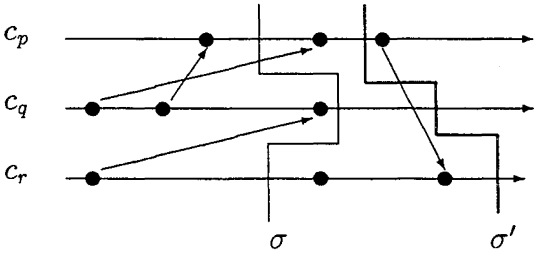

Figure 1. – Diagramme espace-temps.

On dit qu'un processus p est un *initiateur* du calcul réparti c si le premier événement de c_p est une émission ou une action interne. Par exemple, q et r sont les deux initiateurs du calcul décrit dans la figure 1.

2.2. Coupes et Coupes Cohérentes

A chaque étape d'un calcul, un processus est dans un état local qui est déterminé par son état initial et la suite des événements qu'il a exécutés jusqu'à cette étape. Réciproquement, on peut supposer que l'état local d'un processus contient la suite des événements qu'il a exécutés pour atteindre cet état. Par suite, l'état local d'un processus peut être entièrement modélisé par le calcul local qui l'a conduit dans cet état. L'état global d'un système réparti est défini comme l'ensemble des états locaux des processus qui composent le système. Par suite, les états globaux d'un système dans un calcul réparti c sont modélisés par les *coupes* de c, définies comme suit :

DÉFINITION 2.3 : *Une* coupe *d'un calcul réparti* $c = \{c_p : p \in \mathbb{P}\}$ *est une famille* $\{\sigma_p : p \in \mathbb{P}\}$ *telle que, pour tout processus* p, σ_p *est un préfixe de* c_p.

Il est immédiat de vérifier qu'un sous-ensemble d'événements de c est une coupe si et seulement si il est fermé à gauche pour toutes les relations $(\prec_p)_{p\in\mathbb{P}}$. Par contre, un tel ensemble n'est pas nécessairement fermé à gauche pour la relation $\prec$. En effet, l'ensemble des événements d'une coupe peut contenir la réception d'un message et ne pas inclure son émission. Ces coupes, qui ne sont pas cohérentes vis-à-vis de la relation de causalité, correspondent à des états globaux incohérents, *i.e.*, qui ne peuvent pas être atteints par le système. Ceci conduit alors à la définition suivante :

DÉFINITION 2.4 : *Une coupe* σ *d'un calcul* c *est* cohérente *si elle est fermée à gauche pour la relation de causalité* $\prec$, i.e.,

$$\forall (a, b) \in c^2,\ (b \in \sigma \text{ et } a \prec b \Rightarrow a \in \sigma).$$

Par exemple, dans la figure 1, la coupe σ est cohérente alors que σ' ne l'est pas (le message que p envoie à r est reçu mais non envoyé dans σ'). Notons qu'une coupe cohérente d'un calcul réparti est elle-même un calcul réparti. Un calcul réparti c est dit *maximal* s'il n'existe pas de calcul différent de c dont c est une coupe cohérente.

Un exemple fondamental de coupe cohérente est celui du *passé causal* d'un événement a, noté $(\downarrow a)$ et défini comme suit :

$$(\downarrow a) = \{x \in c : x \preceq a\}.$$

Notons que, étant donnée la définition de la relation $\prec$, le passé d'un événement est un ensemble fini.

L'ensemble Γ_c des coupes cohérentes d'un calcul réparti c, muni de la relation d'inclusion $\subseteq$, est un ensemble partiellement ordonné. Cet ensemble admet un plus petit élément (la coupe vide) et un plus grand élément (la coupe c). Il est immédiat de vérifier que si σ et σ' sont deux coupes cohérentes de c alors $\sigma \cup \sigma'$ et $\sigma \cap \sigma'$ sont elles aussi deux coupes cohérentes de c et constituent respectivement la borne supérieure et la borne inférieure de σ et σ' dans l'ensemble partiellement ordonné $(\Gamma_c, \subseteq)$. En d'autres termes, l'ensemble $(\Gamma_c, \subseteq)$ est un treillis complet.

Il est possible, à l'intérieur de ce treillis, de « naviguer » d'une coupe cohérente à une autre. Plus précisément, étant données σ et σ' deux coupes cohérentes d'un même calcul telles que $\sigma \subseteq \sigma'$, il existe une chaîne de coupes cohérentes

$$\sigma_0 = \sigma \subseteq \sigma_1 \subseteq \cdots \subseteq \sigma_n = \sigma'$$

reliant σ à σ' par pas de un, *i.e.*, telle que, pour tout indice i, $\sigma_{i+1} \setminus \sigma_i$ est réduit à un événement. Pour montrer cela, il suffit de considérer l'ensemble $\sigma' \setminus \sigma$ et un ordre total prolongeant la restriction de la relation de causalité $\prec$ à cet ensemble. On pose alors $\sigma_0 = \sigma$ et chaque coupe σ_{i+1} est obtenue en rajoutant à σ_i le i-ème élément dans l'ordre total considéré. Une telle chaîne de coupes cohérentes est appelée *observation* du système de σ à σ'

2.3. Présentation du Problème de DIA

Dans toute la suite $(X, <)$ désigne un treillis. Considérons un système réparti asynchrone $\mathbb{P}$ dans lequel chaque processus p possède une variable locale x_p à valeurs dans X. Supposons d'autre part que l'évolution de ces différentes variables locales obéisse aux règles suivantes :

S : Chaque message envoyé par un processus p est estampillé par la valeur de x_p au moment de l'émission.

R : A la réception d'un message estampillé par la valeur x, le processus p modifie la valeur de sa variable locale x_p en lui affectant une valeur au moins égale à la borne inférieure ([4]) de x et de la valeur courante de x_p.

([4]) Si l'on remplace ici « borne inférieure » par « borne supérieure », on retrouve les horloges logiques scalaires définies par Lamport [9].

I : Au cours d'une action interne, un processus p peut accroître la valeur de x_p.

Ceci constitue les seules hypothèses portant sur le système de base $\mathbb{P}$. En particulier, nous ne faisons aucune hypothèse concernant la sémantique des différentes actions réalisées par les processus de ce système.

Pour toute coupe cohérente c d'un calcul de $\mathbb{P}$, notons $x_p(c)$ la valeur de x_p en la coupe c et $F(c)$ la borne inférieure des valeurs des variables locales x_p et des estampilles des messages en transit dans c :

$$F(c) = \inf(\{x_p(c) \ : \ p \in \mathbb{P}\} \cup \{x \ : < M, x > \text{ est en transit dans } c\}).$$

Il est clair que chaque variable x_p peut croître ou décroître au cours du temps. Notons qu'elle ne peut décroître que sur réception d'un message dont l'estampille est inférieure à sa valeur courante.

Dans la figure 2, un calcul réparti sur trois processus p, q et r illustre comment les variables locales x_p, x_q et x_r peuvent varier au cours du temps. Dans ce diagramme espace-temps, les entiers qui figurent à côté des vecteurs représentants les processus ou les messages désignent respectivement les valeurs des variables locales ou les estampilles des messages. Dans cet exemple, les variables locales sont toutes initialisées à 0 ; x_p et x_q augmentent et prennent la valeur 1 (événements a et b), puis décroissent en raison de la propagation de la valeur initiale de x_r (événements c, d, e et f).

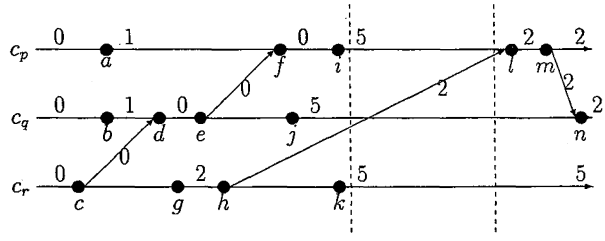

Figure 2. – Un calcul de base.

L'exemple donné dans la figure 2 montre que le minimum des variables locales n'est pas croissant. Pendant une période de temps arbitrairement longue (entre les deux lignes pointillées), toutes les variables locales valent 5 puis le minimum des variables locales passe à 2 juste après cette période. Les variables locales croissent et valent 5 (événements i, j et k), mais x_p décroît ensuite lors de la réception d'un "ancien" message (événements h et l). Cette plus petite valeur 2 est alors propagée au processus q (événements m et n).

Par contre, la proposition suivante montre que la fonction globale F est monotone.

PROPOSITION 2.5 : *Si c et c' sont deux coupes cohérentes d'un même calcul telles que $c \subseteq c'$ alors $F(c) \leq F(c')$.*

Preuve : Considérons une observation du système $\sigma_0 = c, \cdots, \sigma_p = c'$ allant de c à c'. Chaque coupe σ_{i+1} ne diffère de la précédente σ_i que par un seul événement et d'après les règles S, R et I nous avons :

$$F(\sigma_i) \leq F(\sigma_{i+1}).$$

Par suite, $F(c) \leq F(c')$. □

De même qu'il est impossible à un processus de connaître instantanément l'état global du système, un processus ne peut connaître la valeur courante de F. Cependant, dans bien des cas, on ne cherche à calculer qu'une valeur approchée de F par défaut. Nous appelons ce problème *problème de DIA* (Distributed Infinimum Approximation, *cf.* [15]).

Plus précisément, il s'agit de construire un algorithme de contrôle superposé à l'algorithme de base et qui calcule dans chaque processus p une *approximation locale* f_p de F. Ces algorithmes de contrôle sont appelés *algorithmes de DIA*. Nous noterons $f_p(c)$ la valeur de f_p en la coupe cohérente c. Pour que ces approximations soient significatives, il faut qu'elles satisfassent aux deux conditions suivantes :

Sûreté. Pour tout processus p et toute coupe cohérente c, $f_p(c) \leq F(c)$.

Vivacité. Si pour une coupe cohérente c d'un calcul maximal de base c' on a $k \leq F(c)$ alors il existe une coupe cohérente c'' de c' telle que $c \subseteq c'' \subseteq c'$ et

$$\forall \sigma \in \Gamma_{c'},\ \forall p \in \mathbb{P},\ (c'' \subseteq \sigma \subseteq c' \Rightarrow k \leq f_p(\sigma)).$$

Cette dernière propriété signifie intuitivement que si, à un moment donné d'une histoire du système, la fonction F devient plus grande que k, alors toutes les approximations locales calculées dans cette histoire seront, à partir d'un certain temps, plus grandes que k.

2.4. Applications des Algorithmes de DIA

Pour certains treillis $(X, <)$, le problème de DIA correspond à un problème bien connu dans la littérature :

1. **Détection de terminaison :** Lorsque $X = \{actif, passif\}$ et que $actif < passif$, la borne inférieure de toutes les valeurs (locales aux processus et présentes dans les canaux) est égale à *passif* si et seulement si toutes ces valeurs sont égales à *passif*. De plus, il est facile de vérifier que les règles S, R et I correspondent alors exactement aux règles qui régissent la terminaison répartie. Un algorithme de DIA sûr (resp. vivace) est, dans ce cas, un algorithme qui ne fait pas de fausse détection de terminaison (resp. qui détecte toute terminaison au bout d'un temps fini).
2. **Approximation du temps global virtuel :** Lorsque $X = \mathbb{R}$, le problème du DIA correspond au calcul d'une approximation du *Temps Global Virtuel* (GVT en abrégé) défini par Jefferson [7]. Ce problème est aussi connu sous le nom de « problème de *la Coupe Globale* » (*Global Cutoff*, [12]). La notion de GVT est fondamentale dans les systèmes répartis car on peut montrer que toute information concernant des états antérieurs au GVT est obsolète et peut, par conséquent, être détruite.
3. **Ramasse-miettes :** Dans l'algorithme de « ramasse-miettes » de Hughes [6], le problème du DIA joue un rôle essentiel. On affecte à certaines cellules une estampille et on considère certaines variables qui évoluent selon les règles S, R et I au cours du temps. On montre alors qu'une cellule dont l'estampille est inférieure à la borne inférieure des valeurs maintenues dans ces différentes variables est nécessairement inaccessible.

3. ALGORITHMES LOCAUX ET GLOBAUX

La méthode que nous proposons pour résoudre le problème du DIA est l'observation répétée des différents processus. Les algorithmes de DIA que nous décrivons ici sont constitués de deux parties : d'une part un algorithme *local* ou *d'observation* et d'autre part un algorithme *global* ou *de communication*. Dans cette section, nous définissons précisément la fonction d'un algorithme local et celle d'un algorithme global. Chaque combinaison d'un algorithme local et d'un algorithme global donne un algorithme de DIA.

L'algorithme global a pour fonction de « couper » le calcul de base, de collecter les différentes observations effectuées par l'algorithme local et de diffuser une valeur d'approximation calculée par l'algorithme. Pour un processus p, la i-ème itération de l'algorithme global détermine un événement appelé *i-ème événement d'observation* et noté $a_p^{(i)}$. L'ensemble des événements produits par la i-ème itération de l'algorithme global

$W_i = \{a_p^{(i)} : p \in \mathbb{P}\}$ est appelé i-ème *vague d'observation*. L'événement $a_p^{(i)}$ marque le passage de la i-ème vague sur le processus p. On dit encore que la i-ème vague W_i *visite* le processus p en $a_p^{(i)}$.

D'autre part, la i-ème vague détermine une coupe (cohérente ou non) du calcul de base que l'on notera c_i. L'ensemble des événements du calcul de base situés entre les deux vagues consécutives W_{i-1} et W_i est appelé i-ème *intervalle d'observation* (*cf.* figure 3 où les lignes tracées en gras représentent les vagues et les cercles blancs marquent le passage des vagues sur les différents processus). Le premier intervalle d'observation commence à l'initialisation du système. Les vagues seront dites *causalement disjointes* (ou plus simplement *disjointes*) si :

$$\forall (p,q) \in \mathbb{P}^2,\ \forall i \in \mathbb{N},\ a_p^{(i)} \prec a_q^{(i+1)}.$$

Cette condition assure que la $(i+1)$-ème vague ne commence que lorsque chaque processus a déjà été visité i fois. Pendant le i-ème intervalle d'observation, l'algorithme local observe (localement) le comportement de l'algorithme de base et fournit lors du passage de la i-ème vague sur le processus p une valeur $r_p^{(i)}$. L'algorithme global collecte les différents $r_p^{(i)}$ et calcule la borne inférieure de ces valeurs. Cette borne inférieure, notée $f^{(i)}$, est alors prise comme nouvelle approximation de F. Le rôle de l'algorithme local est donc de calculer les différentes valeurs $r_p^{(i)}$ de telle manière que $f^{(i)} = \inf\{r_p^{(i)} : p \in \mathbb{P}\}$ soit une approximation correcte de F, *i.e.*, satisfasse aux deux conditions de correction (sûreté et vivacité) énoncées précédemment.

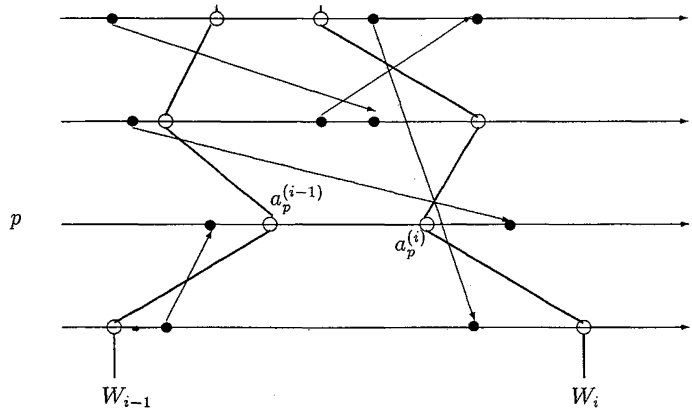

Figure 3. – i-ème intervalle d'observation.

Nous allons maintenant énoncer des propriétés de sûreté et de vivacité pour les algorithmes locaux et globaux. Nous montrerons que ces propriétés de sûreté (resp. de vivacité) suffisent à assurer la sûreté (resp. la vivacité) des algorithmes de DIA construits selon le schéma que nous venons de décrire. Dans la suite, les lettres V et S indiquent respectivement qu'il s'agit d'une propriété de vivacité ou de sûreté. Pour l'algorithme global, on pose :

V1. Toute vague commencée termine après un nombre fini d'événements du calcul de base et une suite infinie de vagues sera exécutée, *i.e.*, pour chaque coupe (cohérente ou non) c du calcul, il existe un indice i tel que $c \subseteq c_i$.

S1. Les vagues construites par l'algorithme global sont disjointes.

S2. Dans la vague W_i, une valeur notée $g^{(i)}$ et satisfaisant à $g^{(i)} = f^{(i)}$ (c'est à dire, $g^{(i)} = \inf\{r_p^{(i)} : p \in \mathbb{P}\}$) est calculée par au moins un processus.

S3. Dans tout processus p, la valeur $g^{(i)}$ est disponible et prise comme approximation locale de F au plus tard pendant la vague W_{i+1}.

Pour définir la sûreté et vivacité des algorithmes locaux, nous utiliserons le lemme suivant :

LEMME 3.1: *Dans tout calcul c de l'algorithme de base, pour toute vague W_i déterminant la coupe c_i de c, il existe une plus petite coupe cohérente σ_i contenant c_i. De plus, σ_i est contenue dans la coupe c_{i+1} déterminée par la vague W_{i+1},* i.e.,

$$c_i \subseteq \sigma_i \subseteq c_{i+1}.$$

Preuve : Notons c le calcul de base considéré et C_i^c l'ensemble des coupes cohérentes de c contenant c_i. Cet ensemble est non vide puisqu'il contient c et l'intersection de tous ses éléments est encore un élément de C_i^c. Par suite, C_i^c admet un plus petit élément que l'on note σ_i.

Comme $W_i = \{a_p^{(i)} : p \in \mathbb{P}\}$ on a

$$\sigma_i = \cup_{p \in \mathbb{P}} \left(\downarrow a_p^{(i)} \right), \tag{1}$$

où $(\downarrow a_p^{(i)})$ désigne le passé causal de $a_p^{(i)}$.

Considérons un événement quelconque e de σ_i. D'après (1), e est dans le passé d'un des événements $a_p^{(i)}$, *i.e.*, il existe un processus p tel que $e \preceq a_p^{(i)}$. Notons q le processus qui exécute l'événement e. Puisque les

vagues W_i et W_{i+1} sont disjointes, on a $a_p^{(i)} \prec a_q^{(i+1)}$. Par transitivité de la relation de causalité, on en déduit que $e \prec_q a_q^{(i+1)}$. Comme c_{i+1} est une coupe, cette dernière relation prouve que e est dans c_{i+1}. Par suite, σ_i est inclus dans c_{i+1}. □

La tâche de l'algorithme local est de fournir à l'algorithme global, lors de sa i-ème itération, les différentes valeurs $r_p^{(i)}$. Pour définir la sûreté et la vivacité des algorithmes locaux, on pose :

S4. Pour tout indice i, $\inf\{r_p^{(i+1)} : p \in \mathbb{P}\} \leq F(\sigma_i)$.

V2. Il existe un entier l tel que, si pour un indice i quelconque $k \leq F(\sigma_i)$, alors pour tout indice $j \geq i + l$, $k \leq f^{(j)}$.

PROPOSITION 3.2 : *Un algorithme de DIA construit par composition d'un algorithme local et d'un algorithme global satisfaisant aux propriétés de sûreté et de vivacité S1-S4 et V1-V2 est sûr et vivace.*

Preuve : (1) Montrons tout d'abord qu'un tel algorithme de DIA est sûr.

Soit c une coupe cohérente quelconque d'un calcul de base. Notons i l'indice de la vague la plus récente contenue dans c, *i.e.*,

$$c_i \subseteq c \text{ et } c_{i+1} \not\subseteq c.$$

Par définition de σ_i (lemme 3.1), nous avons $\sigma_i \subseteq c$.

Supposons que $c \not\subseteq c_{i+2}$, *i.e.*, il existe un événement e dans c et un processus p tels que $a_p^{(i+2)} \prec_p e$. Puisque les vagues sont disjointes (condition S1), nous en déduisons que, pour tout processus q, $a_q^{(i+1)} \prec e$. Autrement dit, c_{i+1} est inclus dans la coupe cohérente $(\downarrow e)$. Comme $(\downarrow e)$ est contenue dans c, ceci contredit le fait que $c_{i+1} \not\subseteq c$. Par suite, on a $c \subseteq c_{i+2}$.

Par S3, il résulte des inclusions $c_i \subseteq c \subseteq c_{i+2}$ que l'approximation de F disponible pour chaque processus p dans la coupe cohérente c est la valeur g calculée dans la vague W_{i-1} ou dans la vague W_i ou encore dans la vague W_{i+1}. Étant donné S2, pour tout processus, nous avons

$$f_p(c) = f^{(i-1)} \text{ ou } f_p(c) = f^{(i)} \text{ ou } f_p(c) = f^{(i+1)}.$$

Comme $\sigma_{i-2} \subseteq \sigma_{i-1} \subseteq \sigma_i \subseteq c$, la proposition 2.5 donne

$$F(\sigma_{i-2}) \leq F(\sigma_{i-1}) \leq F(\sigma_i) \leq F(c).$$

Les inégalités $f^{(j)} \leq F(\sigma_{j-1})$, impliquées par S4, assurent que $f_p(c) \leq F(c)$.

(2) Montrons maintenant la vivacité d'un tel algorithme de DIA.

Soit c une coupe cohérente quelconque d'un calcul maximal de base c'; supposons que $k \leq F(c)$. Soit i le premier indice tel que $c \subseteq \sigma_i$ (un tel indice existe d'après V1). Par V2, pour tout indice $j \geq i + l$, on a $k \leq f^{(j)}$. Par S2 et S3, pour toute coupe cohérente σ telle que $c_{i+l+1} \subseteq \sigma$, on a $k \leq f_p(\sigma)$. □

On peut aussi considérer d'autres conditions de sûreté pour les algorithmes locaux et globaux qui sont pertinentes vis-à-vis de la sûreté des algorithmes de DIA. Par exemple, on peut montrer que, si on affaiblit la condition S4 en

S'4. Pour tout indice i, $\inf\{r_p^{(i)} : p \in \mathbb{P}\} \leq F(\sigma_i)$,

tout en renforçant S3 en

S'3. Dans tout processus p, la valeur $f^{(i)}$ est disponible et prise comme approximation locale de F lorsque la vague W_{i+1} visite p,

les conditions S1-S2 et S'3-S'4 assurent que l'algorithme de DIA construit selon le schéma que nous avons décrit est sûr. Nous omettons la démonstration de ce dernier point, analogue à la première partie de la preuve de la proposition 3.2.

4. ALGORITHMES LOCAUX

Dans cette section nous décrivons différents algorithmes locaux. Nous étudions le cas des communications synchrones en section 4.1, celui des communications causalement ordonnées (notées CO) et FIFO en section 4.2 et enfin le cas général des communications asynchrones quelconques en section 4.3.

4.1. Communications Synchrones

Dans cette section, nous supposons que les communications sont synchrones. Dans ce cas, la relation de causalité, notée encore $\prec$, est définie dans un calcul c comme la plus petite relation transitive telle que :

1. si a et b sont deux événements différents de c exécutés par le même processus et si a a lieu avant b alors $a \prec b$;
2. si s est l'émission d'un message et r sa réception alors

$$\forall e \in c, \quad (e \prec s \Leftrightarrow e \prec r) \text{ et } (s \prec e \Leftrightarrow r \prec e)$$

(*cf.* [3]). Une coupe cohérente est alors une coupe dans laquelle tout message émis ou reçu est émis et reçu dans cette coupe. Une coupe cohérente ne

contient donc pas de message en transit. Par suite, pour toute coupe cohérente c, l'expression de F se simplifie de la façon suivante :

$$F(c) = \inf\{x_p(c) \ : \ p \in \mathbb{P}\}.$$

De plus, il est facile de vérifier que la proposition 3.2 est encore vraie si les communications sont synchrones.

On peut alors penser à l'algorithme local dans lequel la valeur $r_p^{(i)}$ fournie par p à la vague W_i est égale à la valeur de x_p au moment où la vague W_i visite p. En fait ceci conduit à un algorithme de DIA qui n'est pas sûr. En effet, considérons un système constitué de deux processus p et q tels que, initialement, $x_p = 0$ et $x_q = 5$. Supposons que la vague visite tout d'abord le processus q qui reporte alors la valeur 5. Le processus p envoie alors un message à q, ce qui conduit à un état dans lequel $x_q = 0$ (*cf.* règle R). Le processus p affecte alors la valeur 5 à x_p, puis est visité par la vague (*cf.* figure 4). L'approximation de F calculée dans cette vague est égale à 5 alors que F vaut 0.

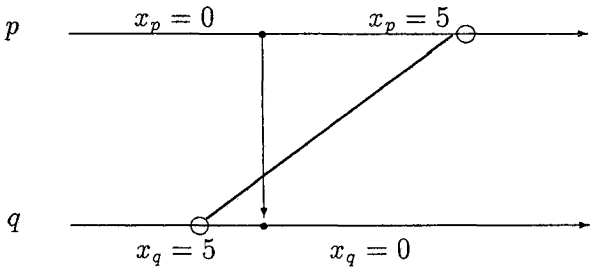

Figure 4. – $f_p(c) = f_q(c) = 5$ **et** $F(c) = 0$.

On peut modifier cet algorithme local de façon que chaque processus p reporte au passage de chaque vague non pas la valeur courante de x_p mais la valeur « la plus pessimiste », c'est-à-dire la plus petite valeur prise par x_p durant le dernier intervalle d'observation. Ceci conduit donc à considérer l'algorithme local suivant :

Algorithme local I : $r_{I,p}^{(i)} = \inf\left\{x_p(e) : a_p^{(i-1)} \preceq_p e \prec_p a_p^{(i)}\right\}$

Ici $x_p(e)$ désigne la valeur de x_p immédiatement après que le processus p ait exécuté l'événement e.

THÉORÈME 4.1 : *Si les communications de base sont synchrones, l'algorithme local I est sûr et vivace.*

Preuve : (1) Notons que

$$\forall i \in \mathbb{N},\ \forall p \in \mathbb{P},\ r_{I,p}^{(i)} \leq x_p(\sigma_{i-1}).$$

Par suite

$$\inf\left\{r_{I,p}^{(i)} : p \in \mathbb{P}\right\} \leq F(\sigma_{i-1}).$$

L'algorithme local I vérifie donc la propriété S4 et est donc sûr.

(2) Supposons que, pour un indice i, on ait $k \leq F(\sigma_i)$. Pour tout événement e exécuté par un processus p et tel que $e \notin c_i$ il existe une coupe cohérente σ passant par e et contenant σ_i (par exemple, $(\downarrow e) \cup \sigma_i$ convient). En utilisant la proposition 2.5, nous obtenons

$$x_p(e) \geq F(\sigma) \geq F(\sigma_i) \geq k.$$

Par conséquent, dès que $j \geq i+1$, on a $f^{(j)} \geq k$. L'algorithme local I satisfait donc la propriété V2 avec $l = 1$. □

Notons que la deuxième partie de cette preuve (propriété V2) reste correcte même si les communications de base ne sont pas supposées synchrones. L'algorithme local I est donc vivace pour des communications asynchrones quelconques.

Nous proposons maintenant deux autres algorithmes locaux qui, comme l'algorithme local I, sont corrects lorsque les communications sont synchrones. Dans l'algorithme local II, chaque processus p reporte dans la vague W_i la borne inférieure de la valeur de x_p au moment du passage de la vague et des estampilles de tous les messages que p a émis pendant le i-ème intervalle d'observation :

Algorithme local II :

$$r_{II,p}^{(i)} = \inf\left(\{x_p(a_p^{(i)})\} \cup \{x : \langle M, x\rangle \text{ émis par } p \text{ entre } W_{i-1} \text{ et } W_i\}\right).$$

THÉORÈME 4.2 : *Si les communications de base sont synchrones, l'algorithme local II est sûr et vivace.*

Preuve : (1) Supposons que $F(\sigma_i) < f^{(i)}$, *i.e.*, qu'il existe un processus q tel que $x_q(\sigma_i) < f^{(i)}$. Alors, d'après les règles S, R et I, q a reçu entre W_i et σ_i un message $< M, x >$ avec $x < f^{(i)}$. Ce message est émis dans σ_i (car σ_i est cohérente). D'autre part, comme $< M, x >$ est reçu après σ_{i-1} (qui est cohérente), $< M, x >$ est émis après σ_{i-1} et donc après W_{i-1}. Étant donnée

la définition de $r^{(i)}_{II,p}$ et puisque $x < f^{(i)}$, le message $< M, x >$ ne peut pas être émis avant W_i. Par suite $< M, x >$ est émis après W_i. En réitérant le raisonnement, on construit une chaîne infinie d'événements de base reliés par la relation de causalité et qui sont tous une cause de l'émission de $< M, x >$. Ceci contredit le fait que tout événement a un passé fini. Par suite, pour tout processus q, on a $x_q(\sigma_i) \geq f^{(i)}$ et donc la condition S'4 est vérifiée.

(2) Pour prouver la vivacité de l'algorithme local II, il suffit de remarquer que la valeur reportée par un processus p au passage de W_i dans l'algorithme local II est au moins égale à celle que p reporte dans l'algorithme local I (chaque événement $a^{(i)}_p$ étant un événement de contrôle – puisque produit par l'algorithme global – $a^{(i)}_p$ ne modifie pas la valeur de x_p et on a en fait $r^{(i)}_{I,p} = \inf\{x_p(e) : a^{(i-1)}_p \preceq_p e \preceq_p a^{(i)}_p\}$). La vivacité de l'algorithme local I prouve alors celle de l'algorithme local II. □

De façon duale, dans l'algorithme local III, chaque processus p reporte dans la vague W_i la borne inférieure de la valeur de x_p au moment du passage de la vague précédente W_{i-1} et des estampilles de tous les messages que p a reçus pendant le i-ème intervalle d'observation :

Algorithme local III :

$$r^{(i)}_{III,p} = \inf\Big(\{x_p(a^{(i-1)}_p)\} \cup \{x : \langle M, x\rangle \text{ reçu par } p \text{ entre } W_{i-1} \text{ et } W_i\}\Big).$$

THÉORÈME 4.3 : *Si les communications de base sont synchrones, l'algorithme local III est sûr et vivace.*

La preuve de ce théorème est analogue à celle du théorème 4.2.

4.2. La Propriété 1-W

Si les communications de base ne sont pas supposées synchrones, les algorithmes locaux I, II et III ne sont pas sûrs comme le prouve l'exemple décrit dans la figure 5. Cet exemple suggère d'étudier le cas où la réception d'un message ne peut être « trop longtemps » différée, sans toutefois supposer les communications synchrones. Plus précisément, on définit la propriété *1-W (One Wave Bounded)* de la façon suivante :

DÉFINITION 4.4 : *Les communications de l'algorithme de base satisfont à la propriété* 1-W *si tout message émis avant la vague W_{i-1} est reçu avant la vague W_i.*

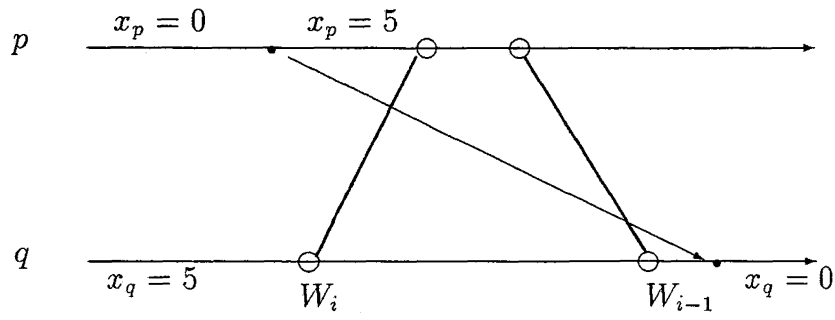

Figure 5. – Communications non synchrones.

THÉORÈME 4.5 : *Sous l'hypothèse 1-W, les algorithmes locaux I, II et III sont sûrs et vivaces.*

Preuve : Nous nous contenterons de faire la preuve pour l'algorithme local I.

(1) Nous avons remarqué que la vivacité (condition V2) de l'algorithme local I est assurée pour des communications quelconques.

(2) Montrons que l'algorithme local I vérifie la propriété S'4 sous l'hypothèse 1-W.

- Supposons qu'il existe un processus q tel que $x_q(\sigma_i) < f^{(i)}$. Alors, d'après les règles S, R et I, le processus q a nécessairement reçu après le passage de W_i un message dont l'estampille est strictement inférieure à $f^{(i)}$. En raison de la propriété 1-W, ce message est émis après le passage de la vague W_{i-1}. Il ne peut être émis pendant le i-ème intervalle d'observation car ceci contredit le fait que son estampille est strictement inférieure à $f^{(i)}$ (*cf.* règle S). Il est donc émis après W_i. En réitérant le raisonnement, on construit une chaîne infinie d'événements de base reliés par la relation de causalité et qui sont tous situés entre W_i et σ_i. Ceci contredit le fait que tout événement a un passé fini. Par suite, pour tout processus q, on a $x_q(\sigma_i) \geq f^{(i)}$.
- Supposons qu'il existe un message en transit dans σ_i dont l'estampille est strictement inférieure à $f^{(i)}$. D'après ce qui précède, ce message ne peut pas être émis après W_i. Il ne peut pas être émis pendant le i-ème intervalle d'observation d'après la définition de $f^{(i)}$. Enfin la propriété 1-W interdit qu'il soit émis avant W_{i-1}. Par conséquent, tout message en transit dans σ_i a une estampille au moins égale à $f^{(i)}$.

Ceci prouve que $f^{(i)} \leq F(\sigma_i)$ et donc que l'algorithme local I est sûr. □

Il est facile de vérifier que si les communications de base sont synchrones alors elles vérifient la propriété 1-W pourvu que les vagues soient supposées disjointes.

Le théorème 4.5 ne revêt réellement d'intérêt que si l'on peut affaiblir l'hypothèse de communications synchrones tout en conservant la propriété 1-W. Charron-Bost, Mattern et Tel [3] montrent que, si toutes les communications (de base et de contrôle) sont CO ([5]) (*cf.* [13]), la disjonction des vagues implique la propriété 1-W. Dans un système où les communications sont CO, les algorithmes locaux I, II et III sont donc corrects.

Si les communications de base et de contrôle sont seulement supposées FIFO, il est facile de vérifier que les algorithmes locaux I, II et III ne sont plus sûrs. Cependant, comme le font Chandy et Lamport [1], on peut penser à utiliser la technique du *dégorgement* ([6]) avec des canaux FIFO. Cette technique consiste à imposer à chaque processus d'envoyer à certains moments un message de contrôle (appelé généralement *marqueur*) le long de chaque canal sortant. Puisque les communications sont FIFO, grâce à ces marqueurs, un processus qui reçoit un message de base peut dater ce message, c'est-à-dire situer (dans le temps) son émission par rapport à celle du marqueur reçu sur le même canal. Une autre fonction des marqueurs est de vider le canal des messages de base situés en aval (d'où le terme de dégorgement).

Plus précisément, dans le cas des algorithmes que nous étudions ici, au début de chaque intervalle d'observation, les processus envoient un marqueur le long de tous leurs canaux sortants. D'autre part, un processus ne peut commencer sa période d'observation suivante que s'il a reçu un marqueur sur chacun de ses canaux entrants.

THÉORÈME 4.6 : *Si les canaux sont FIFO , la technique du dégorgement assure que la propriété 1-W est vérifiée.*

Preuve : Supposons qu'un processus p envoie un message m au processus q avant d'être visité par la vague W_{i-1}. Il existe donc un canal dirigé de p

([5]) Les communications sont dites *causalement ordonnées* (CO) si deux messages quelconques émis dans un certain ordre (pour la relation de causalité) vers un même processus p sont reçus dans le même ordre par p. Les communications CO sont nécessairement FIFO mais la réciproque est fausse puisque les messages considérés peuvent être émis par deux processus différents.

([6]) En Anglais, cette technique est appelée *flushing*.

vers q et p envoie à q le long de ce canal un $(i-1)$-ème marqueur (que l'on note M_{i-1}) au moment où p est visité par W_{i-1}. Comme les canaux sont FIFO, q reçoit m avant de recevoir M_{i-1} et donc avant d'être visité par W_i. □

Notons que cette technique ne peut pas falsifier la vivacité d'un algorithme global si l'on suppose que, comme tout message de base, un marqueur émis est reçu au bout d'un temps fini.

4.3. Cas Général

Notons que, dans les algorithmes locaux I, II et III, l'information reportée par un processus dans une vague W_i est constituée d'un simple élément de l'ensemble X et ne dépend que de l'observation effectuée par le processus durant le i-ème intervalle d'observation. Dans le cas de communications asynchrones quelconques où un message émis est reçu avec un délai fini mais arbitraire, l'information reportée par un processus dans une vague a un format beaucoup plus complexe et prend en compte des messages émis bien avant le dernier intervalle d'observation.

L'algorithme local II et la preuve de sa correction se généralisent dans le cas de communications asynchrones quelconques de la façon suivante :

Algorithme local IV : Chaque processus p rapporte dans la vague W_i le triplet $\left(x_p(a_p^{(i)}), Sent_p^{(i)}, Rec_p^{(i)}\right)$ où $Sent_p^{(i)}$ (resp. $Rec_p^{(i)}$) est l'ensemble des messages émis (resp. reçus) par p avant que celui-ci soit visité par la vague W_i.

Pendant la vague W_i, l'algorithme global calcule l'ensemble des messages en transit dans W_i

$$Trans_i = \bigcup_{p \in \mathbb{P}} Sent_p^{(i)} \setminus \bigcup_{p \in \mathbb{P}} Rec_p^{(i)}$$

ainsi que l'approximation

$$f^{(i)} = \inf\left(\{x_p(a_p^{(i)}) : p \in \mathbb{P}\} \cup \{x :< M, x >\in Trans_i\}\right).$$

Notons que l'on ne peut pas se contenter de raisonner sur les estampilles des messages : une vague W_i pouvant engendrer une coupe c_i incohérente, l'estampille d'un message en transit dans W_i peut être compensée par celle d'un message émis après W_i mais reçu avant W_i. Pour que ces phénomènes de "compensation" n'apparaîssent pas, il est donc nécessaire de supposer que tous les messages sont distincts.

5. ALGORITHMES GLOBAUX

Dans cette section, nous montrons comment on peut implanter les algorithmes globaux servant à la construction d'algorithmes de DIA. Tous les algorithmes globaux présentés ici sont en fait des algorithmes *de vague* (*cf.* [16, Chapitre 6] ou [15]).

5.1. Algorithmes de Vague

Suivant [16, Chapitre 6], on définit un algorithme de vague comme un algorithme dont tous les calculs maximaux (*cf.* section 2.2) sont finis, contiennent au moins un événement spécial marqué *décision*, et satisfont une propriété de *dépendance*. Cette propriété exige que tout événement de décision soit précédé causalement d'un événement dans chaque processus. Si d_p est un événement de décision du processus p dans le calcul c, ceci s'écrit encore :

$$\forall q \in \mathbb{P},\ \exists e \in c_q,\ e \preceq d_p.$$

En fait, Tel montre [16, Lemme 6.4] que l'on peut choisir l'événement e de manière que e soit une émission ou un événement de décision. Un processus qui exécute un événement de décision est appelé *décideur*.

Dans la figure 6, nous donnons un exemple très simple d'algorithme de vague, appelé algorithme de *l'anneau*, qui fonctionne sur un anneau unidirectionel ([7]). On note $Succ_p$ le successeur de p dans cet anneau. L'algorithme est centralisé, *i.e.*, il possède un unique initiateur. Cet initiateur,

```
if p = init
  then begin envoi < jeton > à Succ_p ;
             réception < jeton > ;
             décider
       end
  else begin réception < jeton > ;
             envoi < jeton > à Succ_p
       end
```

Algorithme 6. – Algorithme de l'anneau.

([7]) Un anneau unidirectionel est un graphe dirigé fortement connexe tel que tout sommet a exactement un voisin entrant et un voisin sortant. Deux tels graphes de même taille sont clairement isomorphes.

nommé *init*, initialise un jeton dans l'anneau. Lorsqu'un processus détient le jeton, il le transmet à son successeur. L'initiateur décide, *i.e.*, exécute l'événement de décision, lorsque le jeton lui revient.

Tel montre que les algorithmes de vague peuvent et même doivent être utilisés pour résoudre les problèmes fondamentaux suivants :

La Synchronisation. On assure qu'un certain événement b est exécuté après que chaque processus p ait exécuté un événement a_p en utilisant un algorithme de vague.

Le Calcul d'une Borne Inférieure. Une seule exécution d'un algorithme de vague permet de calculer la borne inférieure de valeurs présentes dans chaque processus au moment où le processus est visité par la vague.

La Diffusion. Une seule exécution d'un algorithme de vague permet de diffuser un message initialement donné à tout initiateur de l'algorithme.

5.2. Algorithmes de Vague et Algorithmes Globaux

Nous montrons ici comment un algorithme de vague peut être utilisé comme algorithme global dans la construction d'algorithmes de DIA.

Considérons des itérations répétées d'un algorithme de vague A. Pour tout processus p, notons $e_p^{(i)}$ le premier événement exécuté par le processus p dans la i-ème itération de A qui est une émission ou un événement de décision. On choisit alors de repérer le passage de la i-ème vague sur le processus p par l'événement $e_p^{(i)}$, c'est-à-dire $a_p^{(i)} = e_p^{(i)}$.

Pour assurer la disjonction des vagues, on introduit la règle suivante :

Règle de disjonction : Un processus n'initialise la $(i+1)$-ème itération de A qu'après avoir décidé dans la i-ème itération.

Le lemme qui suit montre la pertinence de la règle de disjonction. La preuve de ce lemme repose sur le fait que, dans un algorithme de vague, tout événement est précédé causalement par un événement exécuté par un initiateur [16, Lemme 6.2].

LEMME 5.1 : *La règle de disjonction implique que les vagues sont disjointes.*

Preuve : Soient p et q deux processus de $\mathbb{P}$. Soit $a_p^{(i)}$ (resp. $a_q^{(i+1)}$) le passage de la i-ème (resp. $(i+1)$-ème) vague sur le processus p (resp. q). Il existe un initiateur r de la $(i+1)$-ème vague et un événement f exécuté par r dans cette vague tels que $f \preceq a_q^{(i+1)}$. Par la règle de disjonction,

$d_r^{(i)} \preceq f$, où $d_r^{(i)}$ est l'événement de décision exécuté par r dans la i-ème vague. Etant donné que $a_p^{(i)} = e_p^{(i)}$ et d'après la dépendance $e_p^{(i)} \preceq d_r^{(i)}$, il vient $a_p^{(i)} \preceq a_q^{(i+1)}$. □

On peut donc, sans problème, utiliser des algorithmes de vague dans lesquels tout processus est décideur. C'est le cas, par exemple, de l'algorithme de *la phase* et de l'algorithme *bavard* (*voir* [16, Chapitre 6]). Dans le cas de l'algorithme de *l'anneau* ou de l'algorithme de *l'écho*, il n'y a qu'un seul décideur qui est l'unique initiateur de l'algorithme. La règle énoncée plus haut peut donc aussi être implantée aisément pour ces algorithmes de vague.

Nous donnons ici l'algorithme de DIA qui résulte de la composition de l'algorithme de l'anneau (comme algorithme global) et de l'algorithme local I. Chaque processus p possède deux variables locales de contrôle r_p et f_p initialisées respectivement à la valeur initiale de x_p et à $\perp$ ($\perp$ désigne le plus petit élément de X, s'il existe, ou est mis pour non défini). La variable locale f_p contient l'approximation locale à p de F.

L'algorithme local maintient r_p pour calculer la valeur $r_{I,p}^{(i)}$ spécifiée dans l'algorithme local I. Au début de chaque intervalle d'observation, on affecte à r_p la valeur courante de x_p. D'autre part, la valeur de r_p n'est modifiée que lorsque p reçoit un message de base : une réception peut faire décroître x_p, et on affecte alors à r_p la borne inférieure de la valeur courante et de la nouvelle valeur de x_p. Une action de réception d'un message de base M, exécutée par le processus p, s'écrit donc de la façon suivante :

R_p : **begin** réception de M; $r_p := \inf(r_p, x_p)$ **end**

Toute émission par p d'un message de base n'affecte pas x_p et tout événement interne exécuté par p ne peut que faire croître x_p; les émissions et les événements internes ne sont donc pas modifiés.

En parallèle avec le calcul de base (la réception de message étant modifiée comme nous venons de le décrire), chaque processus p exécute l'algorithme présenté dans la figure 7. Cet algorithme répète l'algorithme de l'anneau, présenté dans la figure 6, mais avec les modifications suivantes :

1. Le jeton contient deux champs notés r et f, l'un pour le calcul de la borne inférieure des valeurs $r_p^{(i)}$ fournies par les processus, l'autre pour diffuser la valeur calculée dans la vague précédente.
2. A l'initialisation de la i-ème vague, le premier champ du jeton contient la valeur de $r_{init}^{(i)}$. Chaque processus p, en transmettant le jeton,

remplace la valeur du premier champ r par $\inf(r, r_p^{(i)})$. Par conséquent, lorsque l'initiateur décide, la valeur de ce champ est $f^{(i)}$.

3. Suivant la règle de disjonction, l'initiateur ne commence la prochaine vague qu'après avoir décidé dans la vague précédente. Cette règle permet aussi de diffuser la valeur $f^{(i-1)}$ dans la i-ème vague (champ f du jeton).

```
if p = init then approx := ⊥ ;
f_p := ⊥ ;
while calcul de base n'est pas encore terminé
do if p = init
     then begin f_p := approx ;
                envoi < r_p, approx > à Succ_p ;
                r_p := x_p ;
                réception < r, f > ;
                (* décider *) approx := r
          end
     else begin réception < r, f > ;
                f_p := f ;
                envoi < inf(r, r_p), f > à Succ_p ;
                r_p := x_p
          end
```

Algorithme 7. – Algorithme de DIA.

6. DÉTECTION DE TERMINAISON

Lorsque le treillis $(X, <)$ a de « bonnes » propriétés, certaines optimisations peuvent être proposées dans la construction des algorithmes de DIA.

1. Si X a un plus petit élément $\bot$, la fonction F est égale à $\bot$ dès qu'un processus p a sa variable locale $x_p = \bot$. Un processus p peut donc différer sa participation à l'algorithme de DIA tant que $x_p = \bot$.
2. Si X a un plus grand élément $\top$, un message estampillé par $\top$ ne peut modifier la valeur de F. Un algorithme de DIA peut ignorer de ce fait ces messages.

Dans le cas de la détection de terminaison, $X = \{actif, passif\}$ a un plus petit élément *actif* et un plus grand élément *passif*. Les deux

optimisations proposées plus haut s'appliquent donc dans ce cas et se simplifient respectivement en :

1. Un processus peut différer sa participation à l'algorithme de détection de terminaison tant qu'il est actif.
2. Les messages autres que les messages d'activation sont ignorés.

Notons que, dans ce cas et avec les optimisations que nous venons de proposer, l'algorithme de détection de terminaison répartie obtenu en prenant pour algorithme de vague l'algorithme de l'anneau et pour algorithme local l'algorithme local II est une très légère variante de l'algorithme de détection de terminaison répartie de Dijkstra, Feijen et van Gasteren [4]. Par contre, l'algorithme de détection de terminaison répartie résultant de la composition de l'algorithme local I et de l'algorithme de l'anneau n'est autre que l'algorithme du « Sticky Flag » décrit par Mattern *et al.* [11].

7. CONCLUSION

L'approximation d'une borne inférieure répartie constitue un nouveau problème de contrôle (appelé problème de DIA) dans les systèmes répartis. Des problèmes bien connus comme le GVT ou la détection de terminaison répartie sont des cas particuliers de problème de DIA. Une classe particulière d'algorithmes de DIA a été étudiée ici : il s'agit des algorithmes basés sur une technique de vagues. L'intérêt de ces algorithmes est d'être modulaires et donc d'avoir une structure claire et une preuve relativement simple. Il existe des algorithmes de DIA qui n'appartiennent pas à cette classe. Certains d'entre eux redonnent dans le cas de la détection de terminaison répartie des algorithmes bien connus comme l'algorithme de Dijkstra et Scholten [5] ou celui du *compteur vectoriel* de Mattern [10]. Le lecteur pourra consulter [15] pour une étude détaillée de ces autres algorithmes de DIA.

RÉFÉRENCES

1. K. M. Chandy et L. Lamport, Distributed snapshots: Determining global states of distributed systems. *ACM Trans. Comput. Syst.*, 1985, *3*, 1, p. 63-75.
2. B. Charron-Bost, *Mesures de la Concurrence et du Parallélisme des Calculs Répartis*. Thèse de doctorat, Université Paris VII, 1989.
3. B. Charron-Bost, F. Mattern et G. Tel, Synchronous, asynchronous, and causally ordered communication, *Distributed Computing*, 1996, *9*, 4, p. 173-191.
4. E. W. Dijkstra, W. H. J. Feijen et A. J. M. van Gasteren, Derivation of a termination detection algorithm for distributed computations. *Inf. Process. Lett.*, 1983, *16*, 5, p. 217-219.

5. E. W. DIJKSTRA et C. S. SCHOLTEN, Termination detection for diffusing computations. *Inf. Process. Lett.*, 1980, *11*, 1, p. 1-4.
6. J. HUGHES, A distributed garbage collection algorithm. Dans *Functional Programming Language and Computer Architecture*, 1985, p. J. P. Jouannaud, Ed., tome 201 de *Lecture Notes in Computer Science*, Springer-Verlag, p. 256-272.
7. D. JEFFERSON, Virtual time. *ACM Trans. Program. Lang. Syst.*, 1985, *7*, 3, p. 404-425.
8. T. H. LAI et T. H. YANG, On distributed snapshots. *Inf. Process. Lett.*, 1987, *25*, p. 153-158.
9. L. LAMPORT, Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 1978, *21*, p. 558-564.
10. F. MATTERN, Algorithms for distributed termination detection. *Distributed Computing*, 1987, *2*, 3, p. 161-175.
11. F. MATTERN, H. MEHL, A. A. SCHOONE et G. TEL, Global virtual time approximation with distributed termination detection algorithms. Rapport RUU-CS-91-32, Dept d'Informatique, Université d'Utrecht, Pays Bas, Sept. 1991.
12. S. K. SARIN et N. A. LYNCH, Discarding obsolete information in a replicated database system. *IEEE Trans. Softw. Eng. SE-13*, 1987, p. 39-47.
13. A. SCHIPER, J. EGGLI et A. SANDOZ, A new algorithm to implement causal ordering. Dans *Int. Workshop on Distributed Algorithms*, 1989, p. J.-C. Bermond et M. Raynal, Eds., tome 392 de *Lecture Notes in Computer Science*, Springer-Verlag, p. 219-232.
14. R. Schwartz et F. MATTERN, Detecting causal relationships in distributed computations: In search of the holy grail. *Distributed Computing*, 1994, *7*, 3, p. 149-174.
15. G. TEL, *Topics in Distributed Algorithms*, tome 1 de *Cambridge Int. Series on Parallel Computation*. Cambridge University Press, Cambridge, 1991.
16. G. TEL, *Introduction to Distributed Algorithms*. Cambridge University Press, Cambridge, 1994.