Observing global states of asynchronous distributed applications

Jean-Michel Hélary

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 1042

Programme 3

OBSERVING GLOBAL STATES OF ASYNCHRONOUS DISTRIBUTED APPLICATIONS

Jean-Michel HELARY

Mai 1989

RR-1042

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
35042-RENNES CÉDEX
FRANCE
Téléphone: 99 36 20 00
Télex: UNIRISA 950 473 F
Télécopie: 99 38 38 32

Observing global states of asynchronous distributed applications.

Publication Interne n°468 - Avril 1989 - 24 Pages

Observation des états globaux d'applications réparties asynchrones.

Jean-Michel HELARY
IRISA-IFSIC - Campus de Beaulieu- 35042 RENNES CEDEX
E.mail: helary@irisa.fr

Abstract

Observing global states of an asynchronous distributed application is a difficult task due to arbitrary messages transfer delays. Notion of global states and some of their properties - consistency, being transitless - are precisely stated, and the problem, in both FIFO and non FIFO communication models, is solved in a progressive way : local synchronization allows neighbour processes to record mutually consistent local states, then a sequence of consistent global states is obtained by composition with global wave synchronization; computing some functions over consistent global states becomes easier and an example is displayed (number of messages in transit). Solution generalizes and improves known results, both in FIFO (relaxation of synchronization constraints) and non FIFO (absence of message storing) situations.

Résumé

L'observation d'états globaux d'une application répartie asynchrone est une tâche difficile, notamment à cause du délai arbitraire de transfert des messages. La notion d'états globaux et certaines de leurs propriétés - cohérence, sans transit - sont définies avec précision et le problème est résolu de manière progressive, dans les deux modèles de communication FIFO et non FIFO : une synchronisation locale permet aux processus voisins d'enregistrer des états locaux mutuellement cohérents, puis une séquence d'états globaux cohérents est obtenue par composition avec une synchronisation globale par train de vagues; le calcul de fonctions sur les états globaux cohérents en est facilité et un exemple est présenté (nombre de messages

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L. A. 227)
UNIVERSITÉ DE RENNES 1 I. N. S. A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

en transit). La solution présentée généralise et améliore les résultats déjà connus, auussi bien dans le cas FIFO (relâchement de contraintes de synchronisation) que non FIFO (absence de stockage des messages)

keywords : asynchronism, observation, global state, synchronization, wave sequence, algorithmic composition

Contents

1 Introduction

Computing global states of distributed systems or applications (also known as *computing snapshots*) is a difficult but major problem as previously pointed [CL85, CM88, FGL82, HPR89]. The difficulty is mainly from the lack of global clock and from non determinism inherent to arbitrary transfer delays of messages proper to asynchronism. The interest is yet multiple: on the one hand, it provides solution to a variety of control problems, e.g. detection of stable properties [CM86, Tel86, HJPR87], system recovery after failure [Mor85], etc.; on the other hand, its solution may give prominence to fundamental mechanisms for distributed application control. That's why several interesting results related to this problem have already been obtained. The first published snapshot algorithm was from Chandy and Lamport [CL85]; in this solution the communication channels need to be error-free, unidirectionnal and obey the first-in-first-out property. Synchronization rules use *markers* as control messages tranmitted on the main computation channels: reception of a *marker* on a channel is a synchronization event which, due to the FIFO assumption let processes record their local state consistently (we explain these terms in §2). Under the same communication hypotheses, Hélary et al. [HPR89] solved a stronger problem, namely repeated capture of consistent global states, allowing to detect those without messages in transit; the derivation is based upon a principle of algorithm composition, more precisely a *phase synchronization* (using *markers*) is combined with a *sequential wave synchronization* (implemented with a *token* ring), and the obtained control algorithm is superimposed to the observed application. Among other proposed solutions, Lai and Yang [LY87] gave an interesting solution dropping out the FIFO assumption; markers aren't used and few synchronization is needed but storage of messages sent and received on every node makes a serious implementation problem.Also free from FIFO assumptions, Mattern [Mat89] gives a nice generalization of Chandy-Lamport's, based upon a generalized virtual time concept.

In this paper, we are interested in deriving snapshot computations based upon general, implementation free *wave synchronization* and improving previously known solutions in two directions : firstly under FIFO assumptions, *marker synchronization* constraints are weakened; secondly, we drop out the FIFO model, assuming only that channels are reliable, and unlike Lai and Yang's, only local counters storage, periodically reset to zero, is needed; in that sense, our result is similar to Mattern's one, but unlike the latter we achieve termination of a snapshot computation independantly of observed

application behaviour, and our derivation and implementation are different.

In §2 we define precisely the notions of distributed application, its global state and some of their properties (consistency, no transit, etc.). In §3 we set synchronization rules allowing processes to record their local states consistently according to communication hypotheses; these rules define a distributed control algorithm superimposed to the underlying observed application, whereas §4 shows how a global synchronization technique – namely *wave sequence* – can be used to implement some computations on the set of recorded local states. An example of such a computation is given in §5 : number of intransit messages in a global state, allowing detection of global states without messages in transit, both in reliable FIFO and non-FIFO communication models.

2 Global state

2.1 Distributed application or system

Application (or system) to be observed consists of a set of n computing processes $X = \{P_1, P_2, \ldots, P_n\}$, interacting through message exchange only; each process is located on a node, named a *computing node* in the following; messages circulate along a set $U \subseteq X \mathbf{x} X$ of communication channels [1]. We shall denote by c_{ij} the channel connecting P_i to P_j.

Several communication models can be defined:

- either synchronous, e.g. :

  rendez-vous (instant communication)

  bounded delay (known maximum message transfer delay)

- or asynchronous, e.g. :

  reliable FIFO (finite but impredictable message transfer delay : each channel acts like an unbounded FIFO buffer)

  bounded FIFO (like FIFO but with limited amount of storage : each channel acts like a bounded buffer and thus can lost messages or cause sender's blocking in case of full buffer)

  reliable non FIFO (messages can be desequenced but neither lost nor duplicated nor altered)

[1] Channels are assumed to be *logically* directed but *physically* bidirectionnal

2.2 Global and local time

Whatever the model, some common features concerning time are to be noticed :

- There is no global clock, i.e. there is no concrete global time shared by the n processes,
- on the other hand, we will assume existence of an abstract global time (for purpose of reasonning only); this abstract time obeys to the *causality principle* : for any message m, emission precedes reception :

$$em(m) <_{agt} rec(m)$$

  where $<_{agt}$ is the precedence relation in this abstract global time,

- each process is endowed with a local clock; local events (message sending or receiving and internal events) are sequential and totally ordered in the local time; this ordering is compatible with the abstract global time. In the following we shall use the term *time* for *local time.*

Several well-known implementations of such an abstract global time exist, e.g. [Lam78, Mat89]

2.3 Global and local states

For each process $P_i \in X$, the *local state* of P_i at a given time is defined by the local context of the distributed application (projection onto P_i of the global context). A *global state* is a set $GS = \{LS_1, LS_2, \ldots, LS_n\}$ containing processes local states. We shall denote by $\tau(LS_i)$ (τ_i in brief) the time when P_i has recorded its local state LS_i. For a message m_{ij}, sent from P_i to P_j (along channel c_{ij}) we shall say that :

- its emission belongs to LS_i **iff** $em(m_{ij}) < \tau(LS_i)$
- its reception belongs to LS_j **iff** $rec(m_{ij}) < \tau(LS_j)$

For a message m_{ij} and a pair of local states (LS_i, LS_j), we denote the four possible configurations in the following way:

(i)	$em(m_{ij}) \in LS_i \wedge rec(m_{ij}) \in LS_j$	$\rightarrow$	consumed message $m_{ij} \in consumed(LS_i, LS_j)$
(ii)	$em(m_{ij}) \in LS_i \wedge rec(m_{ij}) \notin LS_j$	$\rightarrow$	intransit message $m_{ij} \in transit(LS_i, LS_j)$
(iii)	$em(m_{ij}) \notin LS_i \wedge rec(m_{ij}) \in LS_j$	$\rightarrow$	inconsistent message $m_{ij} \in inconsistent(LS_i, LS_j)$
(iv)	$em(m_{ij}) \notin LS_i \wedge rec(m_{ij}) \notin LS_j$	$\rightarrow$	inexistent message $m_{ij} \in inexistent(LS_i, LS_j)$

The state of a channel c_{ij} *with respect to a pair of local states* (LS_i, el_j) can be defined as the set of messages

$$CS_{ij} = transit(LS_i, LS_j) \bigcup inconsistent(LS_i, LS_j)$$

Stress mut be laid on the fact that, in a distributed system, only processes local states are (locally) observable at a given time; the state of a given channel is defined formally, from both states of its end processes.

We now give the following definitions concerning global states :

Definition 1. Global state $GS = \{LS_1, LS_2, \ldots, LS_n\}$ is *consistent* if, and only if,

$$\forall i \; \forall j \; inconsistent(LS_i, LS_j) = \emptyset$$

Definition 2. Global state $GS = \{LS_1, LS_2, \ldots, LS_n\}$ is *transitless* if, and only if,

$$\forall i \; \forall j \; transit(LS_i, LS_j) = \emptyset$$

Definition 3. A global state is *strongly consistent* if, and only if, it is consistent and transitless [HPR89]

This last property means that, for any pair LS_i, LS_j, a message m_{ij} is received in LS_j if, and only if, it is sent in LS_i.

3 Synchronization rules for local mutual consistency

3.1 The problem

When a process P_i decides to record a local state LS_i it is necessary that it proceeds in cooperation with its neighbours, more precisely with processes

P_j such that $(P_j, P_i) \in U$; in fact, inconsistent messages with respect to the pair (LS_j, LS_i) should be avoided, since they are messages having been received but not yet sent (in the global state $GS = \{\ldots, LS_j, \ldots, LS_i, \ldots\}$). The problem to be considered here can be stated more precisely : consider a pair of connected processes $(P_j, P_i) \in U$, each of them decides to record its local state respectively at times τ_j and τ_i. How to synchronise τ_j and τ_i in order to have $inconsitent(LS_j, LS_i) = \emptyset$?

The answer depends on the communication model; we will not be concerned by synchronous models (rendez-vous, bounded delays) since problem here is obvious; we'll focus, in what follows, on two asynchronous models : FIFO (unbounded or not) and non FIFO. In order to implement a synchronization control which doesn't disturb the observed application, we will assume that on every computing node, there exists a control process CTL which performs the following tasks (let P be the application process associated to CTL on the same node):

i) monitor message sending and receiving on the account of P (this action must neither create nor modify nor desequence messages : it should be limited to observation and synchronisation),

ii) handle control messages, transparent to P,

iii) record local states on the account of P.

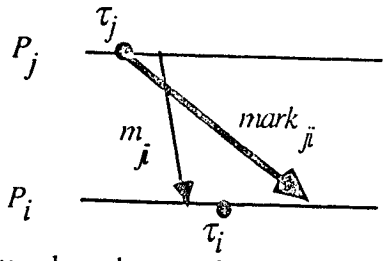
3.2 FIFO communication model

This model is characterized by the following property : let m and m' two messages circulating over the same channel; then

$$em(m) < em(m') \Rightarrow (rec(m) < rec(m') \vee rec(m) = \infty)$$

(The second term handles with loss of messages in non reliable or bounded buffer models). The set of messages sent over a given channel has an ordered list structure which leads to use control messages acting like *end-of-(sub)file markers*.

Let two processes $(P_j, P_i) \in U$ having recorded local states LS_j, LS_i at respective times τ_j, τ_i. Let m_{ji} a message sent over chanel c_{ji} and $\tau_j < em(m_{ji})$. To insure $inconsistent(LS_j, LS_i) = \emptyset$ it is sufficient that $\tau_i < rec(m_{ji})$. Suppose CTL_j sends a control message $mark_{ji}$ over c_{ji} at time τ_j. We have then $em(mark_{ji}) = \tau_j < em(m_{ji})$ whence, from FIFO assumption : $rec(mark_{ji}) < rec(m_{ji})$; from this follows that the relation $\tau_i \leq rec(mark_{ji})$



m_{ji} overpassed marker and $m_{ji} \in inconsistent(LS_j, LS_i)$

Figure 1: non-FIFO communication

insures the desired consistency property. We have obtained the following lemma :

Lemma 3.1. Let $mark_{ji}$ a control message sent over c_{ji} when CTL_j records the local state LS_j of P_j. Then, for every local state LS_i of process P_i recorded by CTL_i not later than the reception of $mark_{ji}$, the property $inconsistent(LS_j, LS_i) = \emptyset$ holds.

Remark. This result remains true in case of application message loss, since the relation $rec(m_{ji}) = \infty$ implies $rec(m_{ji}) > \tau_i$. On the other hand, loss of control message $mark_{ji}$ makes the lemma meaningless since the event $rec(mark_{ji})$ never happens!

3.3 non-FIFO communication model

In that case, the set of messages sent over a given channel is no more totally ordered and using markers is no more sufficient to insure the desired consistency property. Figure 1 shows that LS_i's recording occurs too late with respect to LS_j's recording, despite the marker's synchronization. Time τ_j has to be delayed in order that messages m_{ji} sent after τ_j cannot reach P_i before τ_i. To this end, an *acknowledgement* technique can be used : when CTL_j decides to record P_j's local state LS_j, it sends a marker $mark_{ji}$ over channel c_{ji}; when CTL_i receives this marker, it sends back an acknowledgement message ack_{ij} to $CTLj$. The latter cannot record LS_j before the reception of this ack (figure 2). Let's show that this technique leads to the following result :

Lemma 3.2. Let $mark_{ji}$ a control message sent over c_{ji} by $CTLj$ and ack_{ij} its acknowledgement of receipt. For any local state LS_j such that

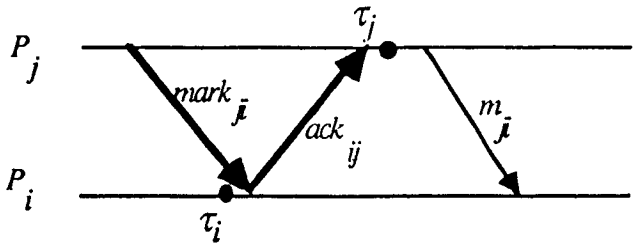


Figure 2: Acknowledgement technique

$\tau(LS_j) \geq rec(ack_{ij})$ and every local state LS_i such that $\tau(LS_i) \leq rec(marq_{ji})$ the property $inconsistent(LS_j, LS_i)$ holds.

Proof. Let m_{ji} such that $em(m_{ji}) > \tau_j$. The following relations hold :

$rec(m_{ji}) >_{agt} em(m_{ji})$	(causality)
$> \tau_j$	(by construction)
$\geq rec(acq_{ij})$	(hypothese)
$>_{agt} em(acq_{ji})$	(causality)
$= rec(marq_{ji})$	(by ack definition)
$\geq \tau_i$	(hypothese)

□

3.4 Conclusion

This section was devoted to local synchronization between adjacent nodes in order to avoid inconsistant messages on incoming channels. We didn't handle the problem of deadlock occurence which could result from delays involved in these local synchronizations; the next section will be devoted to global synchronization techniques, leading to consistent and deadlock free global states recording.

4 Global synchronization for global state consistency

This section deals with the following problem : each process of the observed application has to record a sequence of local states $(LS_i^{(\mu)})_{\mu \in N}$ such

that $\forall\mu\ \forall(P_i, P_j) \in U$: $inconsistent(LS_i^{(\mu)}, LS_j^{(\mu)}) = \emptyset$, in other words : $\forall\mu\ GS(\mu) = \{LS_1^{(\mu)}, LS_2^{(\mu)}, \ldots, LS_n^{(\mu)}\}$ is a consistent global state. This problem will be treated here by composing two distributed algorithmic techniques :

1. Markers (resp. markers and acknowledgements) synchronization depending whether the communication model is FIFO or not, as seen in the preceeding section,

2. Wave sequence synchronization.

The first synchronization insures local mutual consistency for every pair (LS_i, LS_j) whereas the second one insures global state consistency and controls transition from a global state computation $GS(\mu)$ to the next one $GS(\mu + 1)$; moreover, data carried over by waves make possible to perform some computations on recorded global states such that, for instance, "$GS(\mu)$ is transitless,..."

4.1 Wave sequences

A wave sequence is an abstract control scheme expressing the concept of global distributed iteration [HR89]. A wave [Sch85] is a control flow visiting every process in the system once, and only once, and allowing at least one process, predetermined or not, to know when this traversal is over. A wave defines the following events :

$\forall P_i$ $visit(i) \equiv$ process P_i is visited by the wave
$return \equiv \forall P_i visit(i)$ happened

The set of visits is called a *traversal.*
As data flow, a wave has two functions : it broadcasts a datum $DIFF$ and collects a datum $COLL.P_i$ from each process P_i (these functions are performed on each process upon the visit); moreover, data collected allow to perform computation $COLL = f(COLL.P_1, COLL.P_2, \ldots, COLL.P_n)$. The logic associated to a wave is thus the following :

$$\% \ \forall i \ P_i \text{ knows } COLL.P_i \wedge$$
$$\exists I \subseteq X, I \neq \emptyset \wedge (P \in I \Rightarrow P \text{ knows } DIFF) \ \%$$
$$\left(\begin{array}{c} \| visit(i) \\ P_i \in X \end{array} \right);$$
$$return$$

$\%\ \forall i\ P_i$ knows $DIFF \wedge$
$\exists J \subseteq X, J \neq \emptyset \wedge (P \in J \Rightarrow P$ knows $f(COLL.P_1, \ldots, COLL.P_n))\ \%$

Moreover, following temporal relations hold (with straightforward notations) :

$$\forall \mu(\forall i\ visit(i,\mu)) <_{agt} return(\mu) <_{agt} (\forall i\ visit(i,\mu+1))$$

A wave sequence ensures a strong global synchronization since the event $return(\mu)$ separates the two sets of events

$$\{visit(i,\mu) | P_i \in X\} \text{ and } \{visit(i,\mu+1) | P_i \in X\}$$

In other words, a global state exists for which all the events of the first set and none of the second set occured [Tel86].

Wave sequences may be implemented by various traversal structures : directed or undirected spanning trees, rings, etc.. They lead to sequential or parallel structuration of wave events, centralized or distributed return control, and so on [HR88b, Tel86]

4.2 Synchronizations involved

Wave sequence will be useful to :

- Global synchronization of local states recordings; global state $GS(\mu)$ will be recorded along with wave μ,
- collection (resp diffusion) of informations related to global states recorded by the current wave (resp. previous wave).

Thus the first synchronization rule can be stated as follows :

R1) A controller CTL_i records local state $LS_i^{(\mu)}$ of process P_i upon the event $visit(i,\mu)$ (This event can possibly be delayed by CTL_i until the actual recording of $LS_i^{(\mu)}$).

We now state and proof rules to insure consistency of global state $GS(\mu)$, according to the communication model

4.2.1 Case of FIFO model

R2) When CTL_i records $LS_i^{(\mu)}$, it sends a marker $mark_{ij}(\mu)$ over each outgoing channel c_{ij}

R3) (freezing rule) If CTL_i receives the marker $mark_{ji}(\mu)$ on the incoming channel c_{ji} before the occurence of event $visit(i,\mu)$, it prevents process P_i from receiving computations messages or markers on that channel until local state $LS_i^{(\mu)}$ is recorded.

We state the main result of this subsection :

Theorem 4.1. If communications obey FIFO hypotheses and don't loose control messages (markers and wave implementation messages), rules R1) to R3) imply

$$\forall \mu \; GS(\mu) \text{ is consistent}$$

Proof. Let P_i be a process, $\tau_i(\mu)$ the time when CTL_i records $LS_i^{(\mu)}$ and $\sigma_{ji}(\mu)$ the time when CTL_i receives marker $mark_{ji}$ on an incoming channel c_{ji} (sent by CTL_j at time $\tau_j(\mu)$). By rule R3) : $\tau_i(\mu) \leq \sigma_{ji}(\mu)$ whence, from lemma 3.1 : $inconsistent(LS_j^{(\mu)}, LS_i^{(\mu)}) = \emptyset$ □

Remark on the absence of deadlock. This point is straightforward since record of local state depends only of wave progression.

Remark on markers and waves identification. Due to global wave sequence synchronization, any marker $mark_{ji}(\nu)$ arriving on incoming channel c_{ji} before time $\tau_i(\mu)$ has a number ν less or equal to μ (in the case $\nu = \mu$, and only in that case, channel c_{ji} is "frozen" until time $\tau_i(\mu)$). Indeed, $mark_{ji}(\mu+1)$ is sent at time $\tau_j(\mu+1)$; but, from abstract global time relations due to wave sequence synchronization : $\tau_j(\mu+1) >_{agt} \tau_i(\mu)$ and thus $\sigma_{ji}(\mu+1) > \tau_i(\mu)$. This fact is noteworthy as far as implementation is concerned : markers and wave numbers needn't to be explicit. Each controller CTL_i handles a local counter array C_i, indexed by incoming channels, such that :

$\forall j \; C_i[j] \leftarrow C_i[j] + 1$ upon recording of a new local state
$C_i[j] \leftarrow C_i[j] - 1$ upon reception of a new marker on c_{ji}

If $C_i[j]$ is nul when a marker is received on c_{ji} then this channel is "frozen" until the next local state recording (at this time the value $C_i[j]$ is kept unchanged since it is increased then immediatly decreased by one).

4.2.2 Case of non FIFO communication model

R'2) CTL_i cannot record local state $LS_i^{(\mu)}$ until it has received, on each incoming channel c_{ji}, the acknowledgemenet $ack_{ji}(\mu - 1)$ related to marker $mark_{ij}(\mu - 1)$ (prior to the first visit we assume that every process sends an $ack(0)$ message on every outgoing channel),

R'3) When CTL_i records local state $LS_i^{(\mu)}$, it sends a marker $mark_{ij}(\mu)$ over each outgoing channel c_{ij}

R'4) (freezing rule) Once CTL_i received the marker $mark_{ji}(\mu - 1)$ on the incoming channel c_{ji} it prevents process P_i from receiving computations messages or markers on that channel until local state $LS_i^{(\mu)}$ is recorded.

We state now the main result of this subsection :

Theorem 4.2. If communications are non FIFO and don't loose control messages (markers, acknowledgements and wave implementation messages), rules R1) and R'2 to R'4) imply

$$\forall \mu \; GS(\mu) \text{ is consistent}$$

Proof. Let P_i and P_j two processes such that $(P_i, P_j) \in U$ and m_{ij} a message such that $em(m_{ij}) > \tau_i(\mu)$. We must show that $rec(m_{ij}) > \tau_j(\mu)$. Let us denote $\alpha_{ji}(\mu - 1)$ the time when CTL_i received the acknowledgement $ack_{ji}(\mu - 1)$ related to marker $mark_{ij}(\mu - 1)$. By construction, rule R'2 and causality, the following relations hold :

$$em(m_{ij}) > \tau_i(\mu) \geq \alpha_{ji}(\mu - 1) >_{agt} \sigma_{ij}(\mu - 1)$$

thus, on process P_j : $rec(m_{ij} > \sigma_{ij}(\mu - 1)$ (causality).

Two situations may occur on channel c_{ij} :

i) $mark_{ij}(\mu - 1)$ is received by CTL_j before time $\tau_j(\mu)$ (it is not overpassed by $acq_{ij}(\mu - 1)$: fig 3). In that case, by rule R'4 receptions on channel c_{ij} are "frozen" by CTL_j in on the time interval $[\sigma_{ij}(\mu - 1), \tau_j(\mu)]$ whence $rec(m_{ij}) > \tau_j(\mu)$

ii) $mark_{ij}(\mu - 1)$ is received by CTL_j after time $\tau_j(\mu)$ (it is overpassed by $acq_{ij}(\mu - 1)$: fig 4). In that case, by causality, construction and rule R'2), the following relations hold :

$$rec(m_{ij}) >_{agt} em(m_{ij}) > \tau_i(\mu) \geq \alpha_{ji}(\mu - 1) >_{agt} \sigma_{ij}(\mu - 1) > \tau_j(\mu)$$

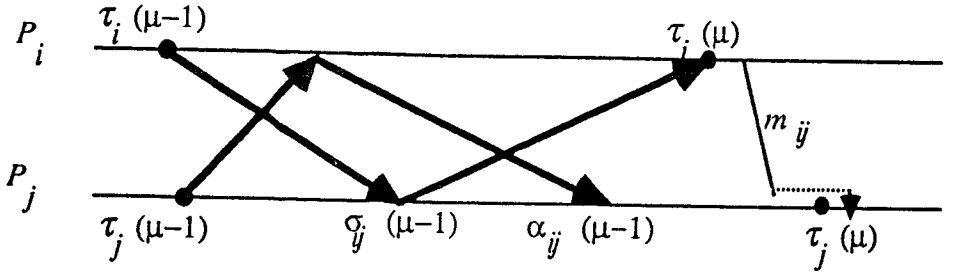


Figure 3: situation i)

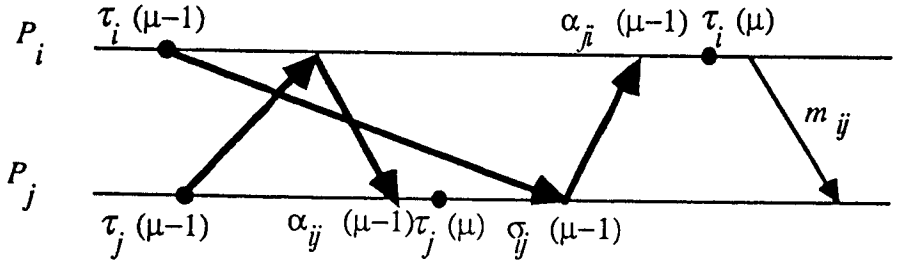


Figure 4: situation ii)

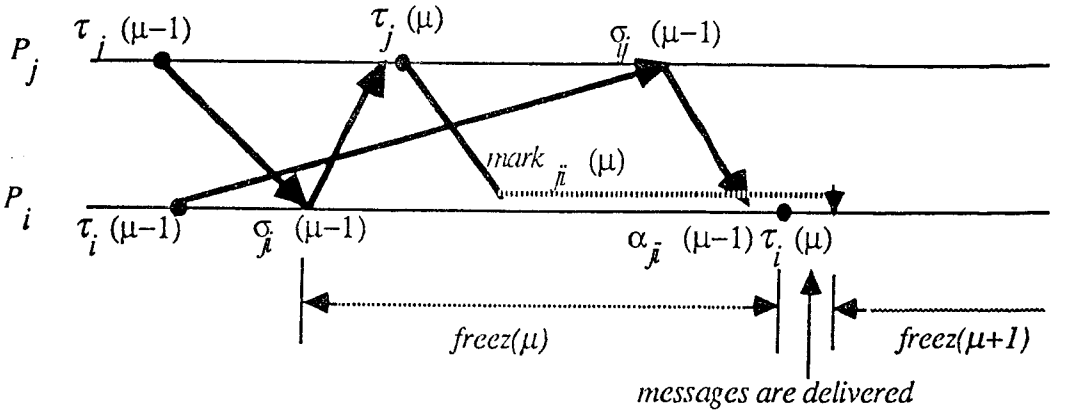


Figure 5: non overlapping of freezing periods

□

Remark on the absence of deadlock. Two asuumptions have been made :

i) When controllers start to observe underlying application, they send an $ack(0)$ message on every outgoing channel (prior to the first wave visit)

ii) Control messages have a finite transfer delay (they cannot be lost)

We show, by induction on the wave number, that $\forall \mu$ event $return(\mu)$ eventually occurs.

• From i) and ii), every controller CTL_i receives messages $ack_{ji}(0)$ on each of its incoming channels c_{ji} after a finite time, thus $LS_i^{(1)}$ is eventually recorded whence wave 1 returns after a finite time

• Suppose $return(\mu - 1)$ has occured. Each controller CTL_i has recorded $LS_i^{(\mu-1)}$, and at that time, sent $mark_{ij}(\mu - 1)$ over outgoing channel c_{ij}; thus, from ii), it will receive $acq_{ji}(\mu - 1)$ a finite time later, whence $LS_i^{(\mu)}$ recording cannot be indefinitely delayed. Thus $return(\mu)$ eventually occurs

□

Remark on the non overlapping of freezing periods. To make sure that messages kept over by a controller CTL_i during "freezing" periods will be delivered to process P_i within a finite time, it is necessary (and sufficient) that freezing periods don't overlap. Let's denote by $freez(\mu) = [\sigma_{ji}(\mu - 1), \tau_i(\mu)]$ the eventual freezing period on channel c_{ji} preceeding the local state $LS_i^{(\mu)}$ recording (we will assume that it's not empty, otherwise the question is meaningless ; see fig 5) The freezing period $freez(\mu + 1)$, if not empty, is defined by the interval $[\sigma_{ji}(\mu), \tau_i(\mu + 1)]$. But $\sigma_{ji}(\mu) >$

$\sigma_{ji}(\mu-1)$, in other words $mark_{ji}(\mu)$ reaches CTL_i during or after the interval $freez(\mu)$, thus $\sigma_{ji}(\mu) > \tau_i(\mu)$ holds. When CTL_i records LS_i^μ, all messages which arrived during $freez(\mu)$ are delivered to P_i, afterwards, if marker $mark_{ji}(\mu)$ was among these messages, freezing period $freez(\mu+1)$ begins. Moreover, no marker $mark_{ji}(\nu), \nu \geq \mu+1$ reached CTL_i during $freez(\mu)$ since $mark(\mu+1)$ is sent at time $\tau_j(\mu+1) >_{agt} \tau_i(\mu)$ □

5 Computations over consistent global states

This section adresses the problem of computing functions defined over consistent global states recorded through a wave. The following example will be considered : compute the number $card(intransit(GS(\mu)))$ of intransit messages in the global state $GS(\mu)$ (for any μ). One of the applications of this computation is, for instance : is the recorded global state strongly consistent? The two communication models : reliable unbounded FIFO and reliable unbounded non FIFO will be considered. In the following, we say that a message m_{ji} is sent (resp. received) in $GS(\mu)$ if, and only if : $\tau_j(\mu-1) < em(m_{ji}) < \tau_j(\mu)$ (resp. $\tau_i(\mu-1) < rec(m_{ji}) < \tau_i(\mu)$), in other words, its emission (resp. reception) belongs to $GS(\mu)$ but not to $GS(\mu-1)$.

5.1 FIFO communication

5.1.1 Abstract global definitions

With every wave number μ and any node identity i let's associate the following sets :

$$
\begin{aligned}
E_i(\mu) &= \bigcup_{j\neq i}\{m_{ji} | em(m_{ji}) < \tau_j(\mu)\} \\
R_i(\mu) &= \bigcup_{j\neq i}\{m_{ji} | rec(m_{ji}) < \tau_i(\mu)\} \\
T_i(\mu) &= \bigcup_{j\neq i}\{m_{ji} | em(m_{ji}) < \tau_j(\mu) \wedge rec(m_{ji}) > \tau_i(\mu)\}
\end{aligned}
$$

Note that $T_i(\mu)$ is the set of intransit messages towards P_i in global state $GS(\mu)$. We have the following result :

Theorem 5.1. If $GS(\mu)$ is consistent, then $T_i(\mu) = E_i(\mu) - R_i(\mu)$

Proof. From $GS(\mu)$ consistency follows $R_i(\mu) \subseteq E_i(\mu)$; moreover, by construction, $T_i(\mu) \subseteq E_i(\mu)$, whence $R_i(\mu) \bigcup T_i(\mu) \subseteq E_i(\mu)$. Conversely relation $E_i(\mu) \subseteq R_i(\mu) \bigcup T_i(\mu)$ clearly holds, whence $E_i(\mu) = R_i(\mu) \bigcup T_i(\mu)$ and since $R_i(\mu) \bigcap T_i(\mu) = \emptyset$, it follows that $R_i(\mu)$ and $T_i(\mu)$ are a partition of $E_i(\mu)$, in other words, $T_i(\mu) = E_i(\mu) - R_i(\mu)$ □

Consider now the global abstract counter vector $MT(\mu)$ defined by

$$MT(\mu)[i] = \sum_{j \neq i} card(E_i(\mu)) - \sum_{j \neq i} card(R_i(\mu))$$

The following corollaries are immediate from theorem 5.1 :

Corollary 5.1. If $GS(\mu)$ is consistent, then $\forall i MT(\mu)[i] = \sharp$ messages in-transit towards P_i in $GS(\mu)$.

Corollary 5.2. If $GS(\mu)$ is consistent, then it is transitless if, and only if, $MT(\mu) \equiv 0$

5.1.2 Distributed implementation

The distributed computation of counters $MT(\mu)$ will be implemented using the wave sequence through which $GS(\mu)$ is recorded : informations $DIFF$ and $\forall i\ COLL.P_i$ have to be carried over by waves in order to maintain the following invariant :

Wave invariant. (When wave μ returns : $COLL(\mu) = MT(\mu)$)
$\wedge DIFF(\mu + 1) = COLL(\mu)$

Progression. By definition :

$$\begin{aligned}
\forall \mu\ \forall i\ MT(\mu)[i] = {} & MT(\mu - 1)[i] \\
& + \textstyle\sum_{j \neq i} card\{m_{ji} | \tau_j(\mu - 1) < em(m_{ji} < \tau_j(\mu)\} \\
& - \textstyle\sum_{j \neq i} card\{m_{ji} | \tau_i(\mu - 1) < rec(m_{ji} < \tau_i(\mu)\} \\
= {} & MT(\mu - 1)[i] && \text{(a)} \\
& + \textstyle\sum_{j \neq i} card\{m_{ji} | m_{ji} \text{ sent in } GS(\mu)\} && \text{(b)} \\
& - \textstyle\sum_{j \neq i} card\{m_{ji} | m_{ji} \text{ received in } GS(\mu)\} && \text{(c)}
\end{aligned}$$

Line (a) is nothing else, from the invariant, than $DIFF(\mu)[i]$. Each of lines (b) and (c) can be computed from local counters : each controller CTL_i is endowed with two counters vectors s_i and r_i such that :

$\forall j \neq i$: $s_i[j] = \sharp m_{ij}$ sent over c_{ij} since last visit

$\forall j \neq i$: $r_i[j] = \sharp m_{ji}$ received on c_{ji} since last visit
(where $\sharp \cdots$ denotes "the number of $\cdots$")
$s_i[j]$ contributes to $MT(\mu)[j](j \neq i)$ and $\sum_{j \neq i} r_i[j]$ to $MT(\mu)[i]$ when the event $visit(i, \mu)$ occurs.

Let's denote $s_i(\mu)[j], r_i(\mu)[j]$ the value of counter at time $\tau_i(\mu)$. The wave algorithm can now be stated :
For every wave μ:
wave μ carries over a counter vector $COLL$ and a boolean TL

→ **Initially** (wave μ)
$COLL(\mu) \leftarrow COLL(\mu - 1)$

→ **Upon** $visit(i, \mu)$
$\forall j \neq i \; COLL(\mu)[j] \leftarrow COLL(\mu)[j] + s_i(\mu)[j]$
$COLL(\mu)[i] \leftarrow COLL(\mu)[i] - \sum_{j \neq i} r_i(\mu)[j]$;
reset to 0 counters s_i, r_i ;
if TL **then** save last recorded state $LS_i^{(\mu-1)}$ **endif** ;
record $LS_i^{(\mu)}$

→ **Upon** $return(\mu)$
% $\forall i COLL(\mu)[i]$ contains the number of intransit messages towards P_i in $GS(\mu)$ %
$TL \leftarrow COLL(\mu) \equiv 0$

The result "$GS(\mu)$ is transitless" is diffused by wave $\mu + 1$ (through boolean TL) : liveness delay is equal to 1 wave.

5.2 non FIFO communication

5.2.1 Abstract global definitions

Consider the following global abstract quantities :

$$\begin{aligned} S(\mu) &= \sharp(\text{ messages sent in } GS(\mu)) \\ R(\mu) &= \sharp(\text{ messages received in } GS(\mu)) \\ NT(\mu) &= \sharp(\text{intransit messages in } GS(\mu)) \\ N(\mu) &= \sum_{\nu \leq \mu} S(\nu) - \sum_{\nu \leq \mu - 1} R(\nu) \end{aligned}$$

We have :

$$N(\mu) = \sum_{\nu \leq \mu-1} S(\nu) - \sum_{\nu \leq \mu-1} R(\nu) + S(\mu)$$

whence

$$N(\mu) = NT(\mu - 1) + S(\mu) \qquad \text{(a)}$$

Moreover, the following relation is straightforward from the definitions :

$$NT(\mu) = N(\mu) - R(\mu) \qquad \text{(b)}$$

5.2.2 Distributed implementation

Each controller CTL_i is endowed with two counters :

s_i = total number of messages sent since the last time local state has been recorded

r_i = total number of messages received since the last time local state has been recorded (actually, on each incoming channel, up to marker reception : cf. receptions freezing).

Moreover, upon each local state recording, CTL_i resets s_i, r_i to 0. Values obtained at time $\tau_i(\mu)$ will be denoted $s_i(\mu), r_i(\mu)$.

Wave μ performs the following tasks :

- saves $NT(\mu - 1)$ as $NTBACK$(computed by wave $\mu - 1$),
- computes sums $S(\mu) = \sum_i s_i(\mu), R(\mu) = \sum_i r_i(\mu)$.

Upon return, the following computations can be performed :

$$\begin{array}{lcl} N(\mu) & \leftarrow & NTBACK + S(\mu) \\ NT(\mu) & \leftarrow & N(\mu) - R(\mu) \\ NTBACK & \leftarrow & NT(\mu) \end{array}$$

Boolean information "$GS(\mu)$ is transitless" is equal to $NT(\mu) = 0$ and can be diffused by wave $\mu + 1$. Thus, liveness delay of 1 wave is necessary, as in the FIFO case; size of data carried over by a wave is equal to

- two integers (sums E, R),

- possibly one integer ($NTBACK$) if, due to the wave implementation, this result cannot be stored on a particular node,
- possibly one boolean $TL = (NTBACK = 0)$ if $NTBACK$ is not carried over by wave.

6 Conclusion

Observing consistent global states is a fundamental problem in distributed systems, and a dificult one in asynchronous models. We gave a clear definition of global states, bringing out distinction between observable (processes) and not observable (channels) local states, and some of their properties (consistency, being transitless). The problem was tackled by steps : at first level, it was shown how a process can record its own local state consistently with its neighbours; the technique of markers, similar in the FIFO case to the one developped in [CL85, CM86], is generalized, through acknow- ledgements, to the non FIFO communication model. At second level, global synchronization requiring the wave sequence abstract control scheme solves the problem stated; from a methodological point of view, it displays yet another situation where this abstract network traversal scheme can be seen as a syntaxic scanner on which semantic actions are grafted : this approach has been previously enhanced in [HR88a]. Finally, at third level, a computation model is presented through an example : computing the number of intransit messages; clearly, other functions fit this model, e.g. set of intransit messages, or – in a bounded FIFO comunication context – number of lost messages due to full buffer (assuming that control messages have a specific storage buffer preventing them from loss), etc..

As conclusion, we lay stress on the fact that techniques developped lead to a whole family of algorithms devoted to observation of asynchronous distributed applications or systems.

References

[CL85] K. M. Chandy and L. Lamport. Distributed snapshots: determining global states of distributed systems. *ACM TOCS*, 63–75, February 1985.

[CM86] K. M. Chandy and J. Misra. An example of stepwise refinement of distributed programs: quiescence detection. *ACM TOPLAS*, 8(3), July 1986.

[CM88] K. M. Chandy and J. Misra. *Parallel program design : a foundation.* Addison-Wesley, 1988.

[FGL82] M. J. Fischer, N. D. Griffeth, and N. Lynch. Global states of a distributed system. *IEEE trans. on soft. eng.*, SE-8:3:198–202, may 1982.

[HJPR87] J.-M. Hélary, C. Jard, N. Plouzeau, and M. Raynal. Detection of stable properties in distributed applications. 6^{th} *ACM SIGACT-SIGOPS, Symp. Principles of Distributed Computing, Vancouver, Canada*, 125–136, August 1987.

[HPR89] J.M. Hélary, N. Plouzeau, and M. Raynal. A characterization of a particular class of distributed snapshots. In *Submitted to International Conference on Computing and Information (ICCI'89), Toronto*, may 23–27 1989.

[HR88a] J.M. Hélary and M. Raynal. Les parcours distribués de réseaux: un outil pour la conception de protocoles. In R. Castanet et O. Rafiq, editor, *CFIP'88 Ingénierie des protocoles*, pages 159–170, Eyrolles, 1988.

[HR88b] J.-M. Hélary and M. Raynal. *Synchronisation et contrôle des systèmes et des programmes répartis.* Eyrolles, Septembre 1988. English translation to appear, Wiley, 1990.

[HR89] J.-M. Hélary and M. Raynal. An abstract distributed iteration scheme: application to the computation of weighted shortest paths. *Technology and Science of Informatics, Vol. 7, No 3*, May 1989. (In French).

[Lam78] L. Lamport. Time, clocks and the ordering of events in a distributed system. *Communications. of the ACM*, 21(7):558–565, July 1978.

[LY87] T.H. Lai and T.H Yang. On distributed snapshots. *Inf. Proc. Letters*, 25:153–158, 1987.

[Mat89] F. Mattern. Virtual time and global states of distributed systems. In Cosnard, Quinton, Raynal, and Robert, editors, *Proc. Int. Workshop on Parallel and Distributed Algorithms, Bonas, France, oct. 1988*, North Holland, 1989.

[Mor85] C. Morgan. Global and logical time in distributed algorithms. *Inf. Proc. Letters*, 20:290–294, 1985.

[Sch85] F. B. Schneider. Paradigms for distributed programs. In *Distributed Systems*, pages 431–480, Springer Verlag, 1985. LNCS 190.

[Tel86] G. Tel. *Distributed Infimum Approximation.* Tech. report RUU-CS-86-12, University of Utrecht, 1986.

LISTE DES DERNIERES PUBLICATIONS INTERNES

PI 460 **FLOW ANALYSIS IN TANDEM QUEUES WITH FEEDFORWARD FLOWS**
Kamel SISMAIL
14 Pages, Février 1989.

PI 461 **NUMERICAL CONCERNS IN CONVOLUTION-TYPE ALGORITHMS**
Gerardo RUBINO, William STEWART
18 Pages, Février 1989.

PI 462 **ACCUMULATED REWARD OVER THE N FIRST OPERATIONAL PERIODS IN FAULT-TOLERANT COMPUTING SYSTEMS**
Gerardo RUBINO, Bruno SERICOLA
14 Pages, Mars 1989.

PI 463 **ELEMENTS FINIS C^1, POLYNOMIAUX DE DEGRE QUATRE PAR TRIANGLE, DANS UNE TRIANGULATION FORMEE DE TRIANGLES EQUILATERAUX**
Michel CROUZEIX, Miloud SADKANE
26 Pages, Mars 1989.

PI 464 **VERS UN MODELE D'ECLAIREMENT REALISTE**
Pierre TELLIER, Kadi BOUATOUCH
66 Pages, Avril 1989.

PI 465 **COMPILATION OF FUNCTIONAL LANGUAGES BY PROGRAM TRANSFORMATION**
Pascal FRADET, Daniel LE METAYER
28 Pages, Avril 1989.

PI 466 **MODEL BASED DIAGNOSIS : A CASE STUDY IN VIBRATION MECHANICS**
Michèle BASSEVILLE, Albert BENVENISTE
26 Pages, Avril 1989.

PI 467 **DELAI DE COMMUNICATION ENTRE NOEUDS VOISINS SUR L'iPSC/2**
Patrice BURGEVIN, André COUVERT, René PEDRONO
16 Pages, Avril 1989.

PI 468 **OBSERVING GLOBAL STATES OF ASYNCHRONOUS DISTRIBUTED**
Jean-Michel HELARY
24 Pages, Avril 1989.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique