ETAT GLOBAL

MU4IN403– Algorithmique Répartie

SCIENCES
SORBONNE
UNIVERSITÉ

Remerciements à Jean-Michel Busca et à Maria Potop-Butucaru pour leurs transparents originaux

Plan

I. Introduction
   1. Motivation
   2. Représentation et cohérence

II. Exemple d'état global

III. Algorithme de snapshot de Chandy et Lamport 85

Plan

I. Introduction

II. Exemple d'état global

1. Définition du système étudié
2. Graphe des états accessibles
3. Remarques

III. Algorithme de snapshot de Chandy et Lamport 85

Plan

I. Introduction

II. Exemple d'état global

III. Algorithme de snapshot de Chandy et Lamport 85

1. Principes
2. Analyse
3. Algorithme
4. Exemples
5. Conclusion

I. Introduction

1. Motivation

2. Représentation et cohérence

I.1 Motivation

Calcul de l'état global d'un système réparti

- instantané de l'état du système
- capture d'un moment donné de son exécution

État global d'un système réparti à un instant t

- Union des états de chacun des processus du système à t

ET

- Union des états de chaque canal de communication à t

I.1 Motivation

Pas de solution simple

- Collecte distante de données/états locaux
- **Pas de temps global**
- Intrusivité de la collecte/observation doit être minimale

Plusieurs utilisations possibles :

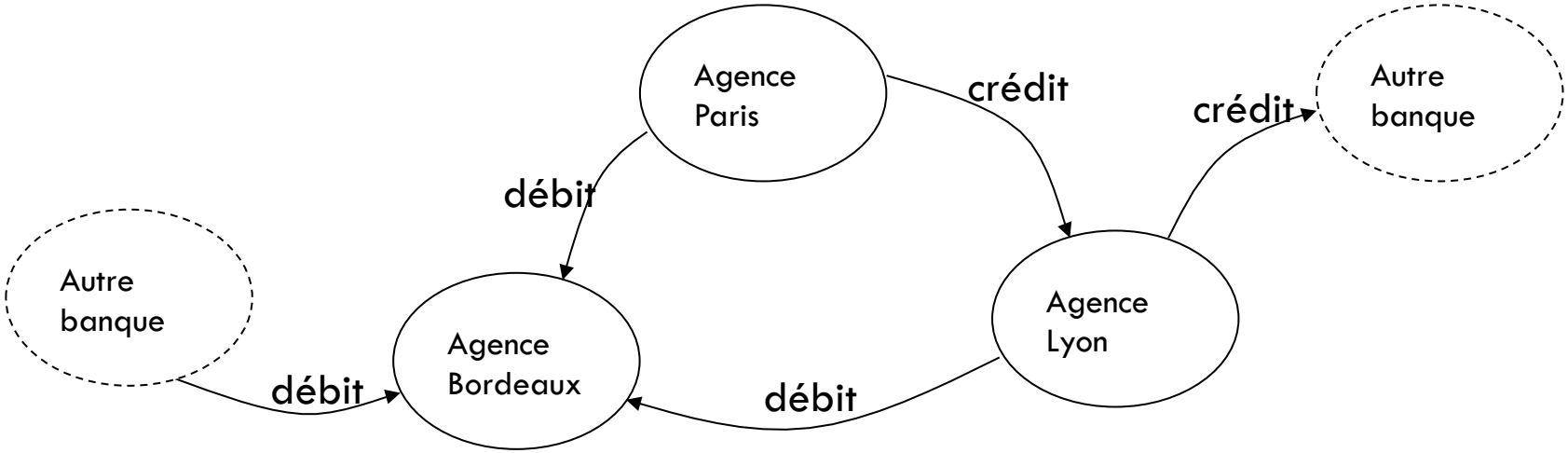
1. Exécution d'un algorithme centralisé sur des données distantes
2. Détection d'états globaux stables (*eg. interblocage, terminaison*)
3. Ramasse-miette distribué
4. Tolérance aux fautes

I.1 Motivation

Exécution d'un algorithme centralisé sur des données distantes

s'affranchir du caractère réparti du système.

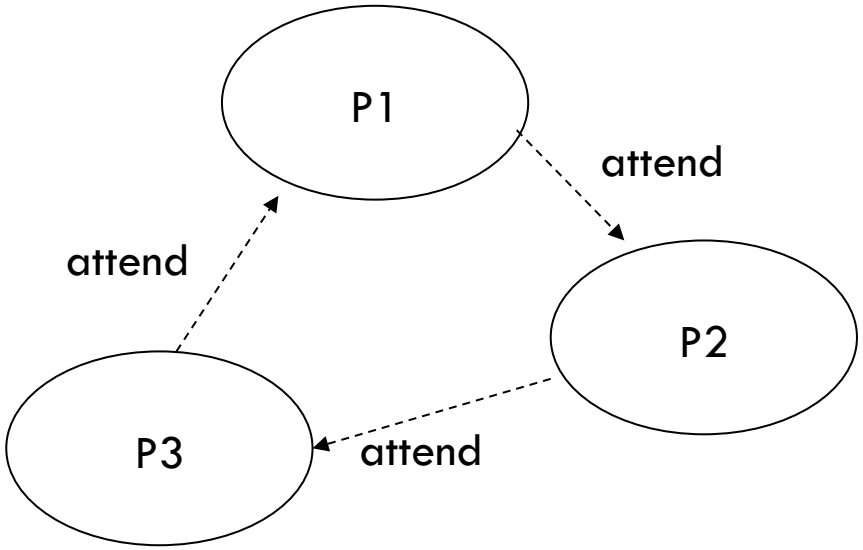
eg. calcul des avoirs d'une banque à partir de ceux de chacune de ses agences.



$$Avoirs_{banque} = Avoirs_{agence\ Paris} + Avoirs_{agence\ Lyon} + Avoirs_{agence\ Bordeaux}$$

I.1 Motivation

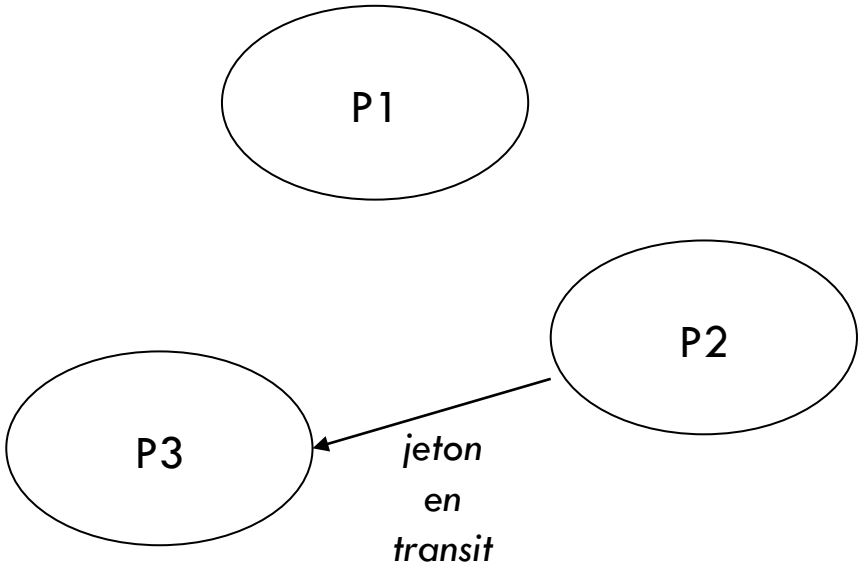
Détection d'états globaux stables



Inter-blocage ?

I.1 Motivation

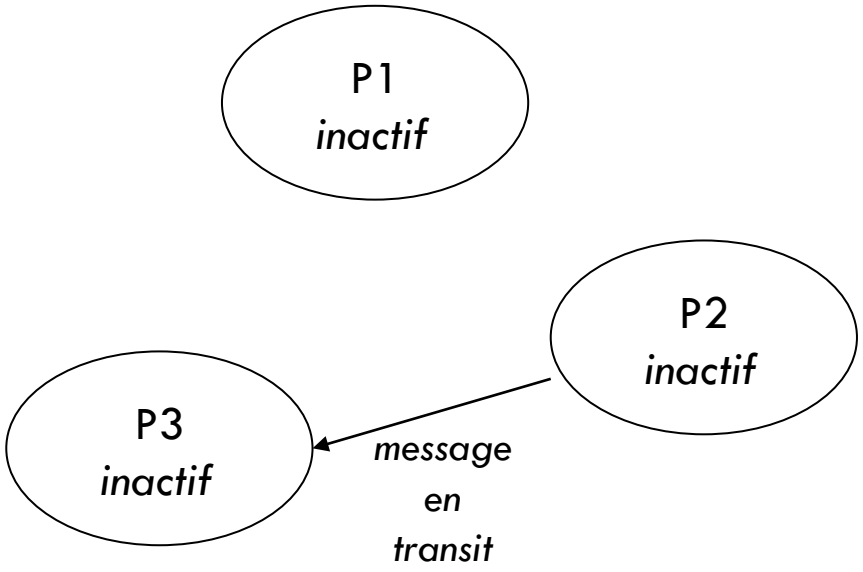
Détection d'états globaux stables



Perte du jeton ?

I.1 Motivation

Détection d'états globaux stables

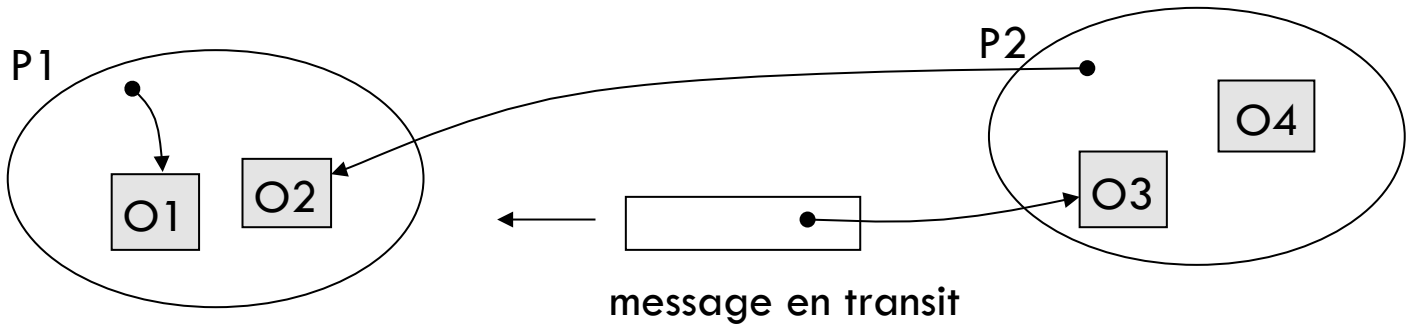


Terminaison ?

I.1 Motivation

Ramasse-miette distribué

élimination des objets inutilisables conservés en mémoire
détection des références locales/***distantes***
objet sans référence => suppression



04 n'est plus référencé dans le système réparti
03 n'est plus référencé que dans le message en transit (pour l'instant)

I.1 Motivation

Tolérance aux pannes

Sauvegarde de l'état global du système => points de reprise (*checkpoints*)
En cas de panne d'un des sites, procédure de recouvrement :

1. chaque site revient à son checkpoint (rollback)
2. reprise de l'exécution à partir de cet état.

Problème : garantir que le recouvrement est cohérent (duplication, causalité)

I.2 Représentation et Cohérence

Etat local d'un processus P_i

Valeur des variables locales de P_i

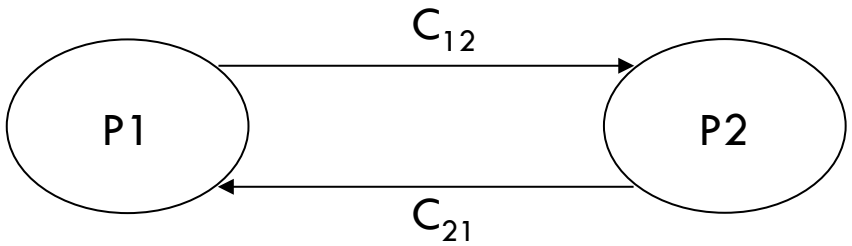
Déterminisme : état local de P_i = f(état initial de Pi, succession des événements sur P_i)

Etat d'un canal de communication C_{ij} (entre P_i et P_j)

Ensemble des messages en transit entre P_i et P_j

Canal FIFO => ensemble ordonné

Canaux sont considérés unidirectionnels (C_{ij}, de P_i vers P_j, et C_{ji} de P_j vers P_i)

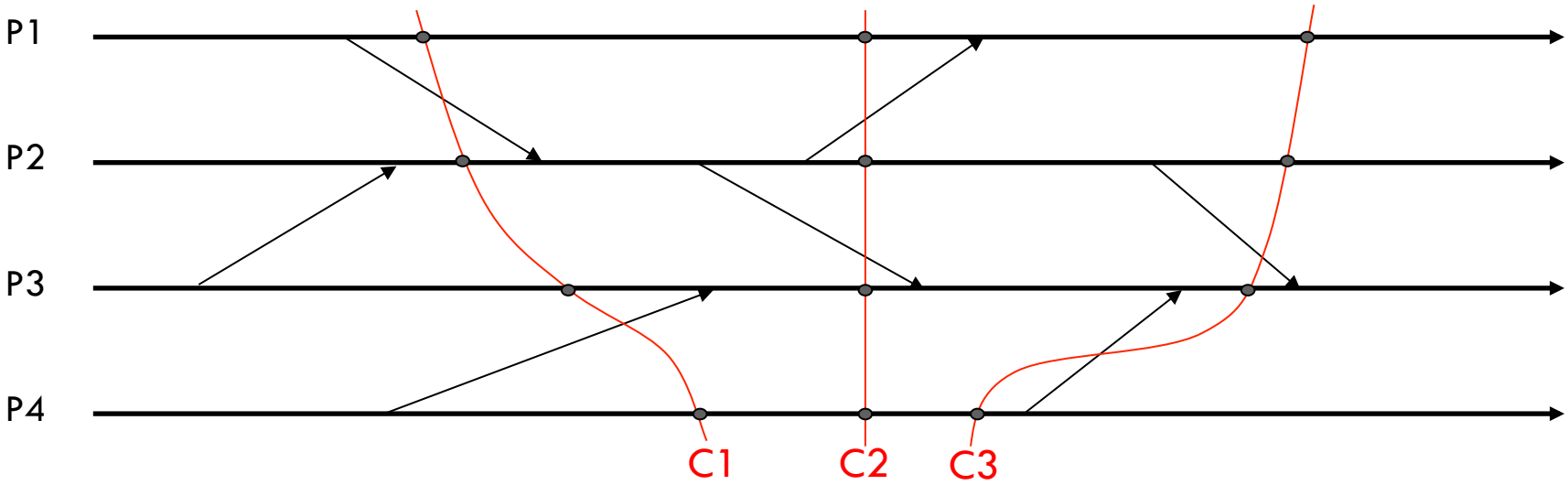


I.2 Représentation et Cohérence

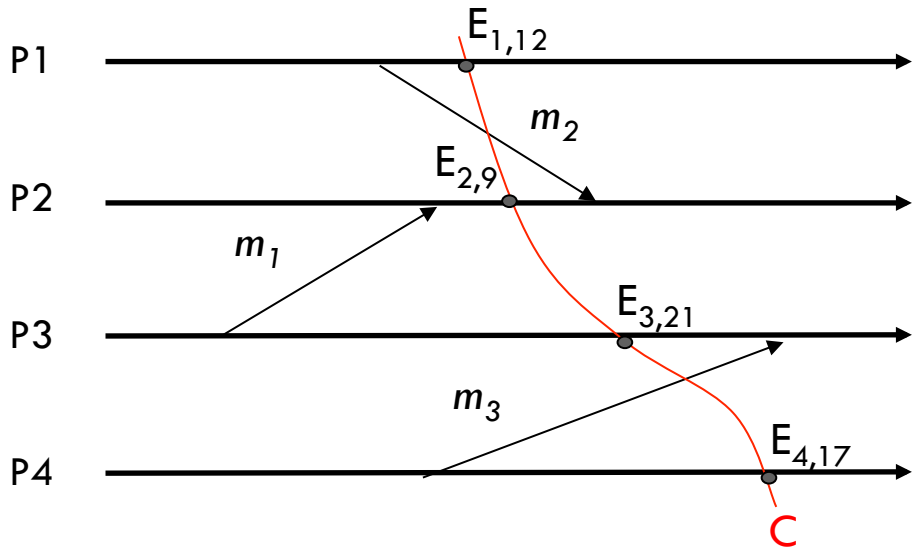
Etat global d'un système réparti

Union des états de l'ensemble des P_i et des C_{ij}

Un état global peut être représenté par une courbe (***coupure***) sur le diagramme temporel



I.2 Représentation et Cohérence



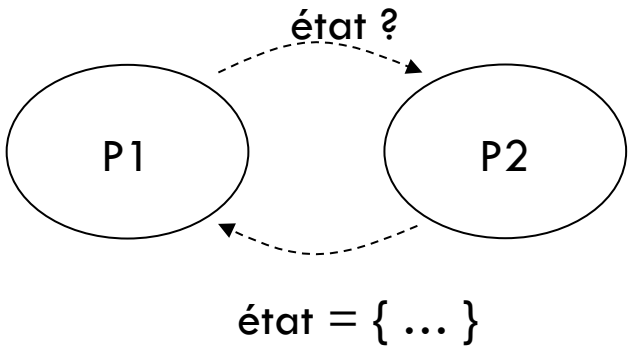
$$\mathbf{E_C(SR) = E(P_1) \cup E(P_2) \cup E(P_3) \cup E(P_4) \cup E(C_{12}) \cup E(C_{43})}$$
$$\mathbf{= E_{1,12} \cup E_{2,9} \cup E_{3,21} \cup E_{4,17} \cup \{m_2\} \cup \{m_3\}}$$

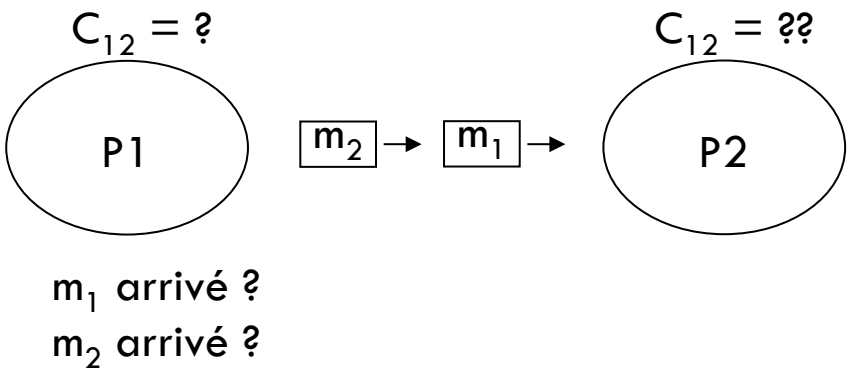
I.2 Représentation et Cohérence

Problème de la **collecte** des états

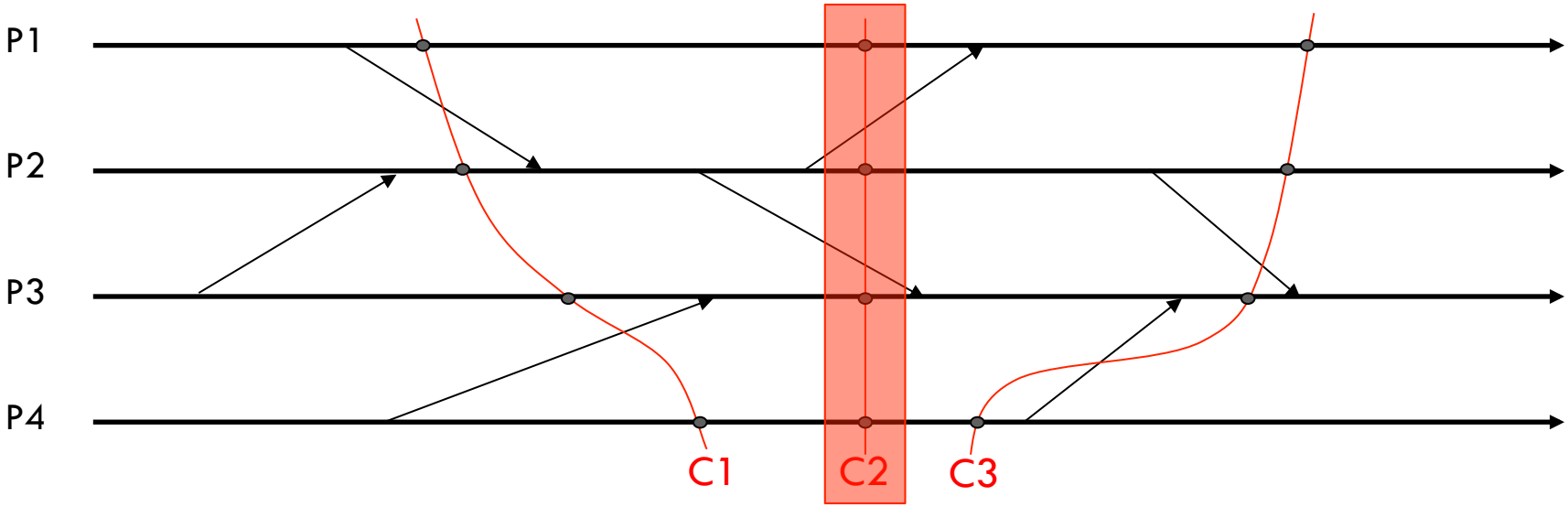
$E(P_i)$ n'est directement et immédiatement observable que sur P_i

$E(C_{ij})$ n'est jamais directement observable, ni sur P_i, ni sur P_j





I.2 Représentation et Cohérence



Problème de la **cohérence** des états collectés

Solution idéale : figer l'état de tous les P_i et C_{ij} au même instant absolu

Mais pas de temps global (ie. pas de référence temporelle commune)

Solution pragmatique : assouplir le critère de cohérence ***sans rendre l'état global inexploitable***

=> Relation de précédence causale

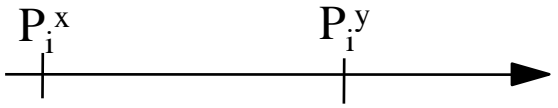
I.2 Représentation et Cohérence

Relation de précédence causale

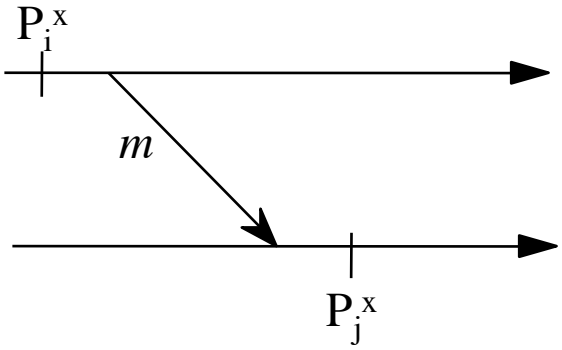
Relation d'ordre entre événements dans le temps

e *précède causalement* e' est noté e → e'

2 événements d'une même exécution ne sont pas forcément liés causalement

a) 

devient $P_i^x \rightarrow P_i^y$

b) 

devient $P_i^x \rightarrow P_j^x$

I.2 Représentation et Cohérence

Coupure cohérente

Définit un passé fermé pour la relation de précédence causale
Pour tout événement survenu avant la coupure,
les événements qui le précèdent causalement sont également survenus avant la coupure

Formellement

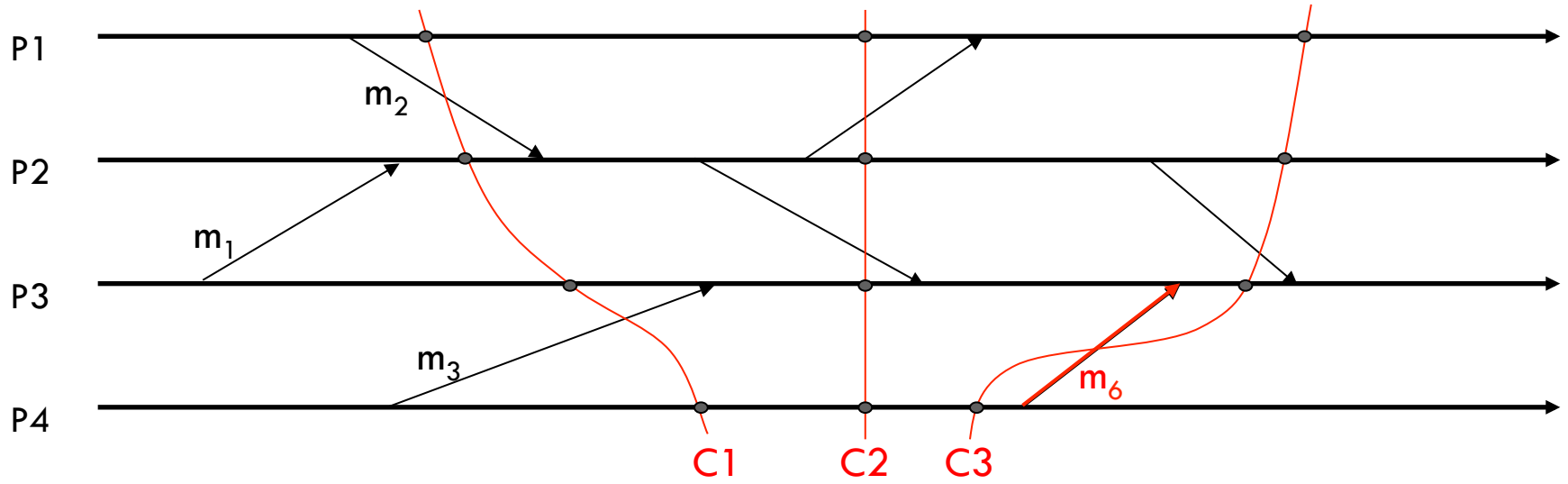
C coupure cohérente $\Leftrightarrow$ $\forall e \in \text{Passé}(C),\ e' \rightarrow e \Rightarrow e' \in \text{Passé}(C)$

Intuitivement
La cause d'un événement ne peut être dans son futur (de la coupure).

Par définition
un **état global cohérent** est un état global obtenu selon une coupure cohérente

I.2 Représentation et Cohérence

Pour déterminer si une coupure est cohérente, il suffit d'examiner les messages en transit.
Aucun message ne doit aller du futur vers le passé.



C1 et C2 sont cohérentes

C3 **n'est pas** cohérente : réception(m_6) ∈ Passé(C3) et émission(m_6) ∉ Passé(C3)

II. Exemple d'état global

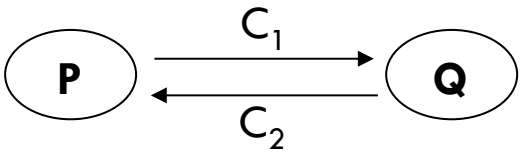
1. Définition du système étudié
2. Graphe des états accessible
3. Remarques

II.1 Définition du système étudié

Définissons le système réparti SR :

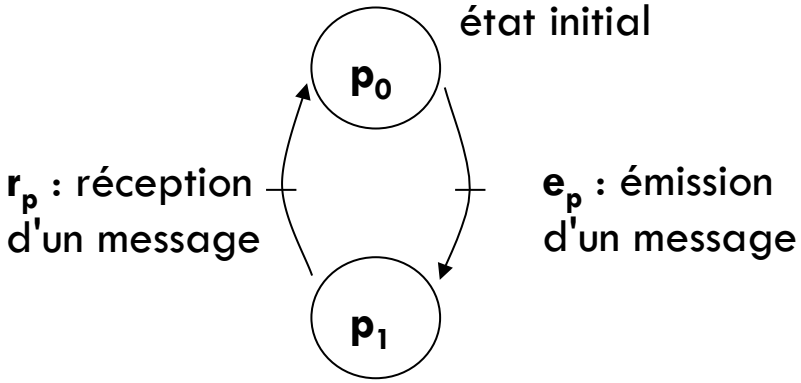
2 processus P et Q

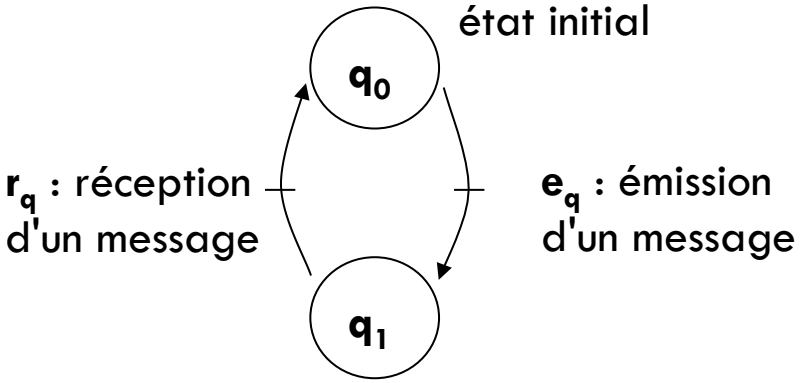
2 canaux unidirectionnels C_1 (P vers Q) et C_2 (Q vers P)



Chacun des processus P et Q envoie puis recoit un message, alternativement.

Automates des processus :





Etat global du système : quadruplet <état local P, état C1, état C2, état local Q>

II.2 Graphe des états accessibles

Définition

Ensemble des états que peut prendre le système réparti

Chaque nœud du graphe correspond à un état global du système réparti

Chaque arête correspond à une transition d'un état à un autre suite à un événement

Une exécution particulière du système

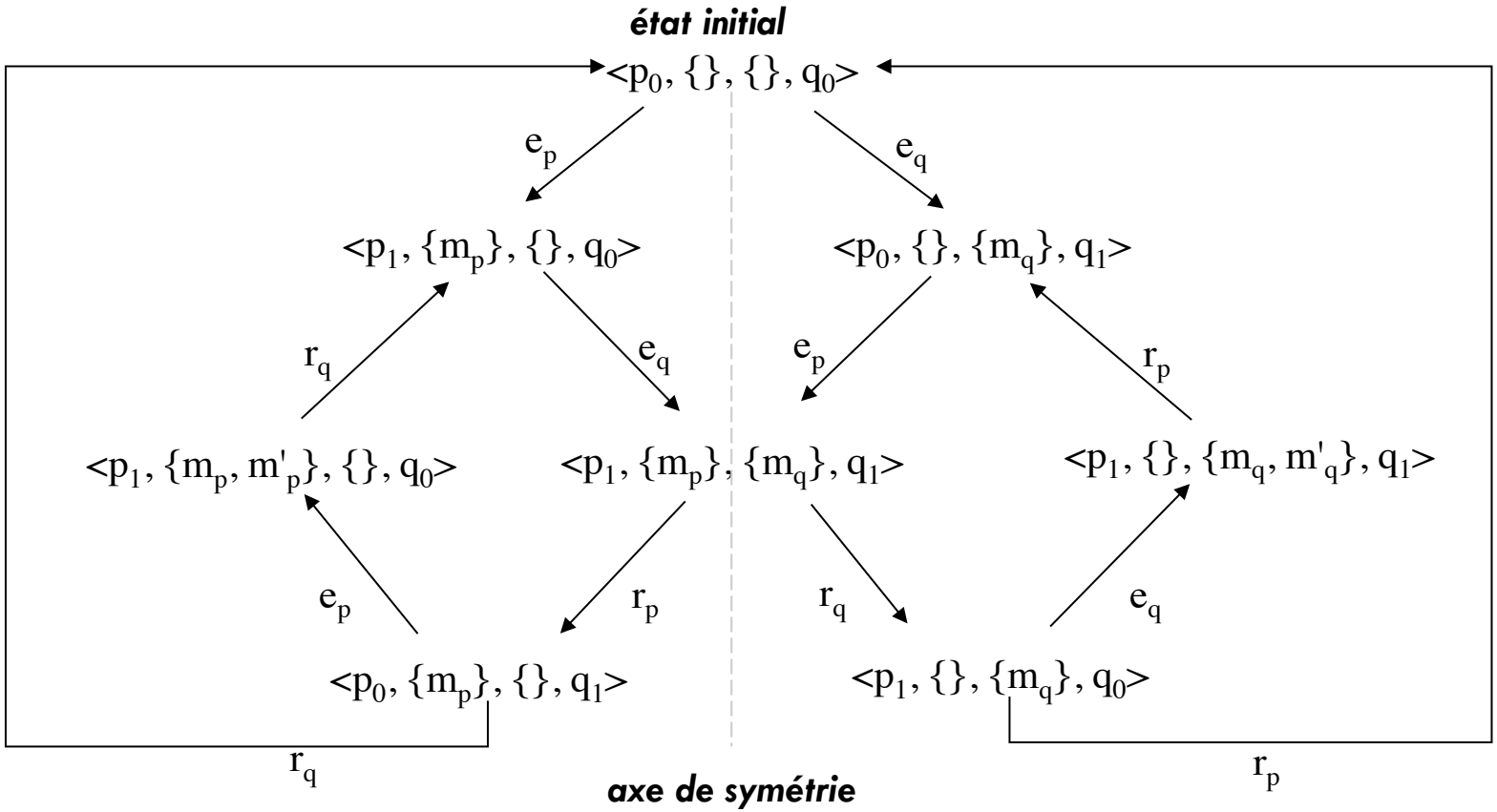
Un cheminement dans ce graphe à partir de l'état initial.

Dans SR

un nœud est un quadruplet <état local P, état C1, état C2, état local Q>

les arêtes sont e_p, r_p, e_q *ou* r_q

II.2 Graphe des états accessibles



II.3 Remarques

Le graphe des états accessibles permet :

1. de détecter les deadlocks
   état puits : état sans arête sortante
2. de dimensionner les canaux
   visibilité du nombre maximal de messages dans chaque canal
   eg. sur SR, au plus deux messages dans les canaux C1 et C2

Chacun des états du graphe est cohérent : il correspond à un état réellement accessible.

Si le système réparti est symétrique, son graphe l'est également.

Pb : la taille du graphe croît **très** rapidement
eg. bien que SR soit simple, son graphe comprend déjà 8 états

III. Snapshot algorithm (Chandy & Lamport 85)

1. Principes
2. Analyse
3. Algorithme
4. Exemples
5. Conclusion

III.1 Principes

Objectif

Calculer **dynamiquement** un **état global cohérent** d'un système réparti en cours d'exécution

Minimiser la complexité et le caractère intrusif de la solution

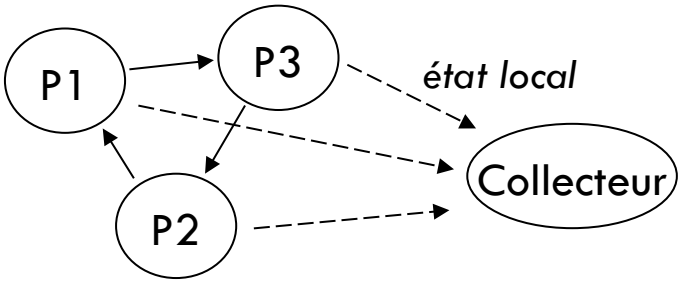
Modèle système

N processus répartis sur un **réseau fortement connexe**

Canaux de communication **fiables** et **FIFO**

Un processus (collecteur) désigné à l'avance et connu de tous les autres

Aucune panne de processus au cours du déroulement de l'algorithme



III.2 Analyse

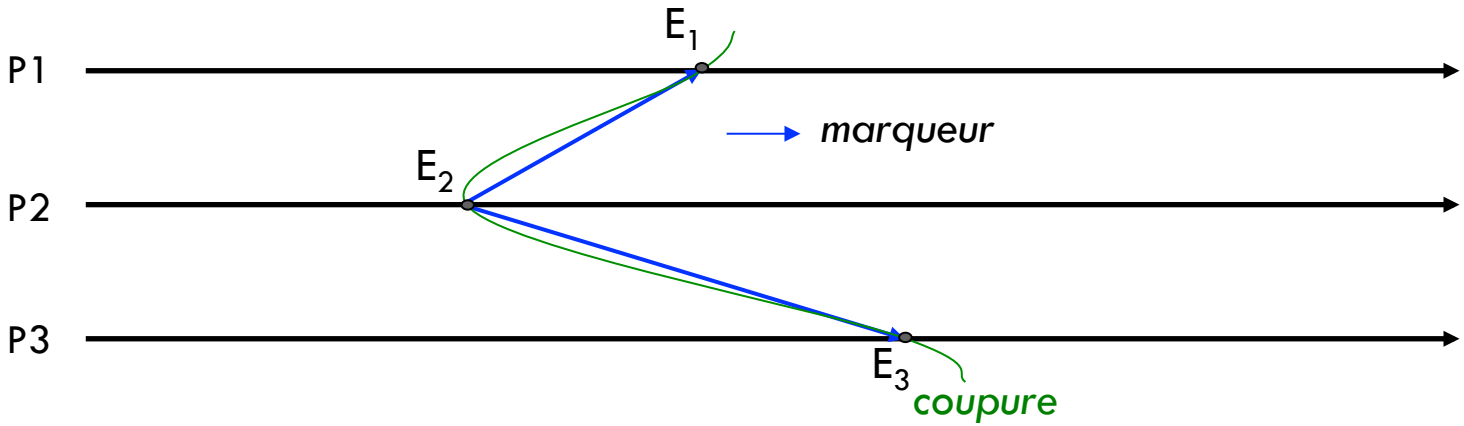
On part de l'algorithme le plus simple pour lister les problèmes à résoudre.
Un processus déclenche le calcul d'un état global

il sauvegarde sont état local

puis demande aux autres pcs d'en faire autant par un message spécial (marqueur)

2 propriétés à assurer :

1. la **cohérence** de la coupure,
2. et la **complétude** l'état sauvegardé pour les canaux de communication.

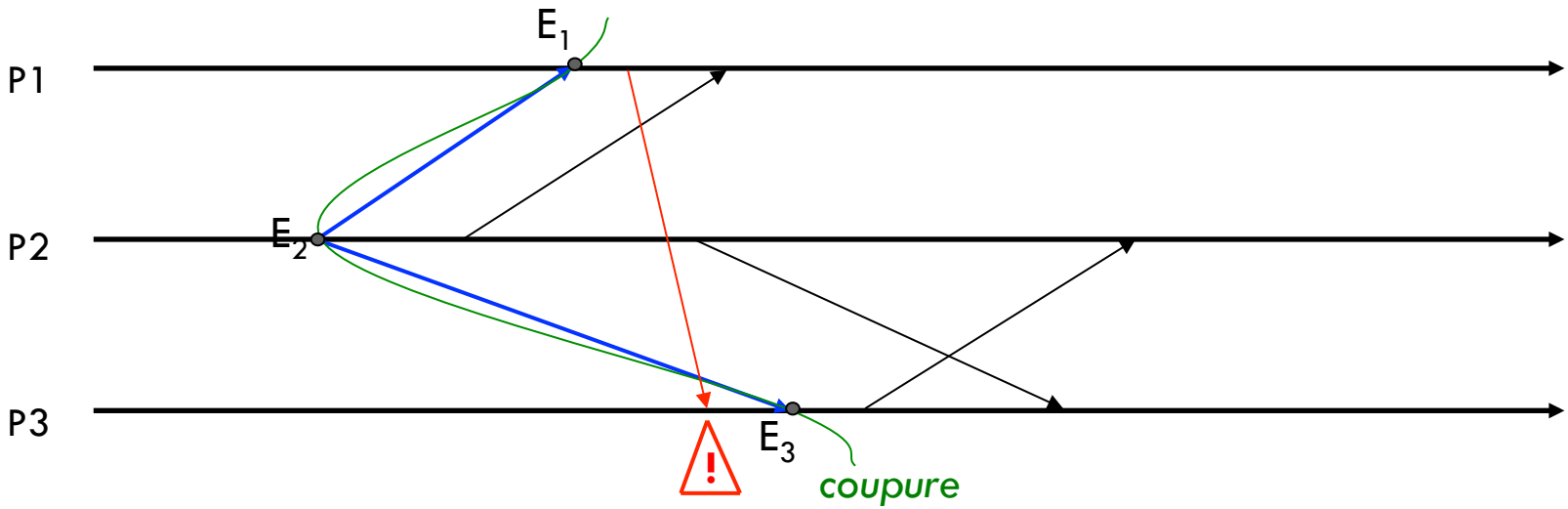


III.2 Analyse

Cohérence

Un message reçu **avant** la coupure doit être émis **avant** la coupure

- de P_2 vers P_1 et P_3 : pas de problème puisque les canaux sont FIFO
- de P_3 ou P_1 vers P_2 : pas de problème en raison de la causalité
- aucune garantie pour les autres couples de processus

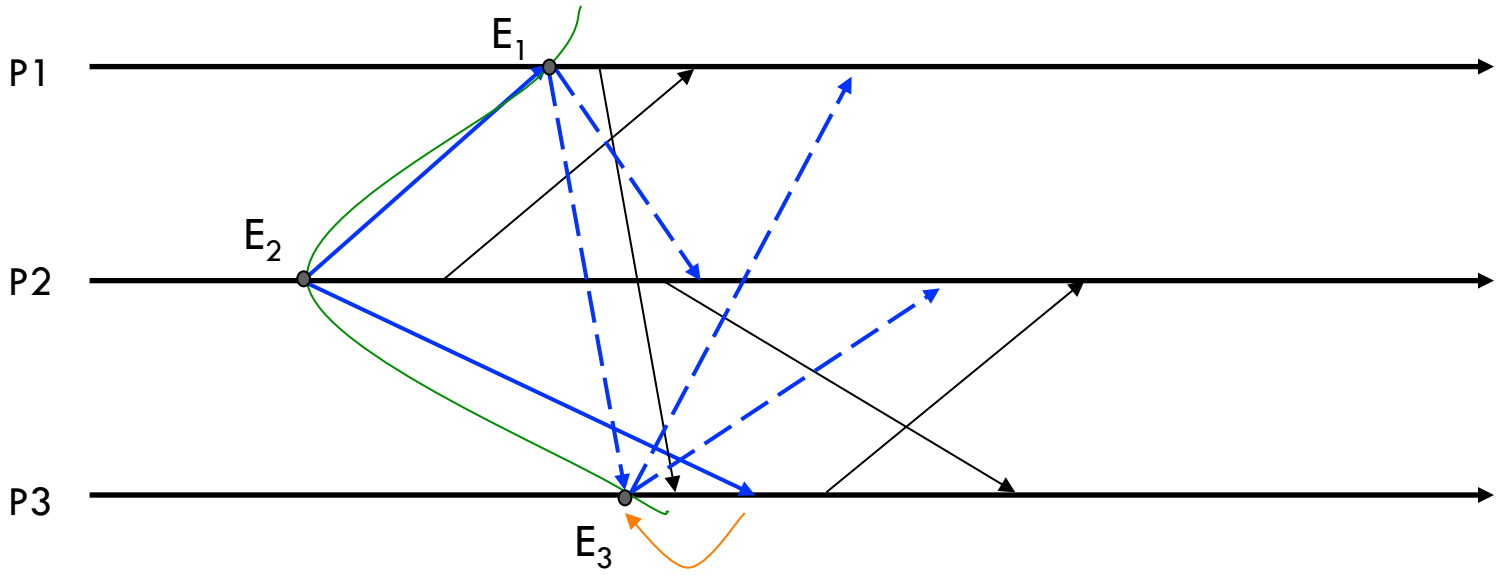


III.2 Analyse

Solution pour la cohérence : propager le marqueur.

Chaque sauvegarde provoque la diffusion du marqueur.

Le premier marqueur reçu provoque une sauvegarde, mais pas sur les suivants.



III.2 Analyse

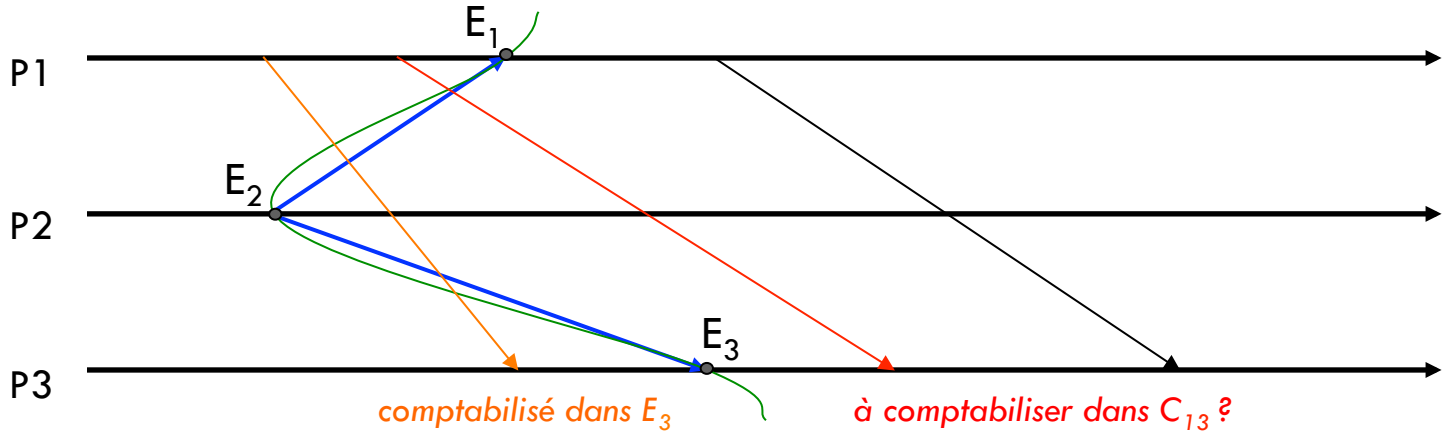
Complétude

Tout message émis **avant** une sauvegarde doit être comptabilisé

- Soit dans la sauvegarde du destinataire en étant reçu **avant** cette sauvegarde
- Soit dans l'état du canal parce qu'il a été reçu après cette sauvegarde

Problème

- quand débuter et arrêter l'enregistrement des messages en transit ?
- Quels sont les messages émis qui ont été reçus avant la sauvegarde du destinataire ?



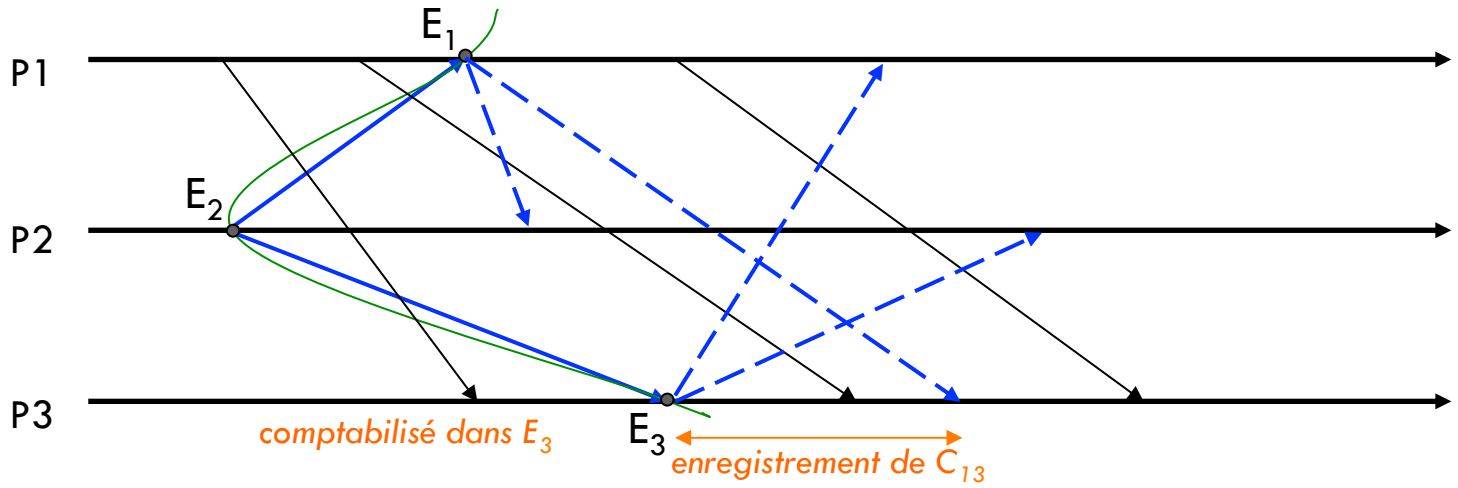
III.2 Analyse

Solution pour la complétude : partir des messages reçus

Utiliser les marqueurs propagés

Message en transit : reçu pendant la phase de sauvegarde

(ie. entre le premier marqueur et celui associé au canal)



Note : C_{23} est par définition vide car P_3 reçoit son premier marqueur du collecteur (P_2)

III.2 Analyse

Résumé

Utilisation de messages **marqueurs**

messages de contrôle, *orthogonaux* aux messages applicatifs (coloration)

Mise en œuvre de 2 règles

1. Règle d'envoi du marqueur
   sauvegarde de l'état local P_i => envoi du marqueur sur tous les canaux sortants C_{ix}
2. Règle de réception du marqueur
   sur réception du premier marqueur, P_i
   a. sauvegarde son état
   b. débute l'enregistrement des messages reçus sur chacun de ses canaux
   c. clôt l'enregistrement sur réception du marqueur associé à chaque canal

Le processus initiateur agit comme sur réception d'un marqueur (fictif).

III.3 Algorithme

Variables locales :
$enreg_i$ = faux;
$etatLocal_i$ = ∅;
$etatCanal[N]_i$ = {∅, ∅, …};
$marqRecu[N]_i$ = {faux, faux, …};

sauvegarder() :
$enreg_i$ = vrai;
$etatLocal_i$ = etatLocal();
POUR j = 1 .. N SAUF i
  envoyer(MARQ, P_j)
  $etatCanal[j]_i$ = ∅;
  $marqRecu[j]_i$ = faux;
FPOUR

recevoir(m, P_j) :
SI m == MARQ ALORS
  SI ! $enreg_i$ ALORS
    sauvegarder()
  FSI
  $marqRecu[j]_i$ = vrai;
  SI toutRecu() ALORS
    envoyer(<$etatLocal_i$, $etatCanal[]_i$, Collecteur);
    $enreg_i$ = faux;
    $marqRecu[]_i$ = {faux, faux, …};
  FSI
  RETOURNER
FSI
SI $enreg_i$ ET ! $marqRecu[j]_i$ ALORS
  $etatCanal[j]_i$ ∪= { m };
FSI

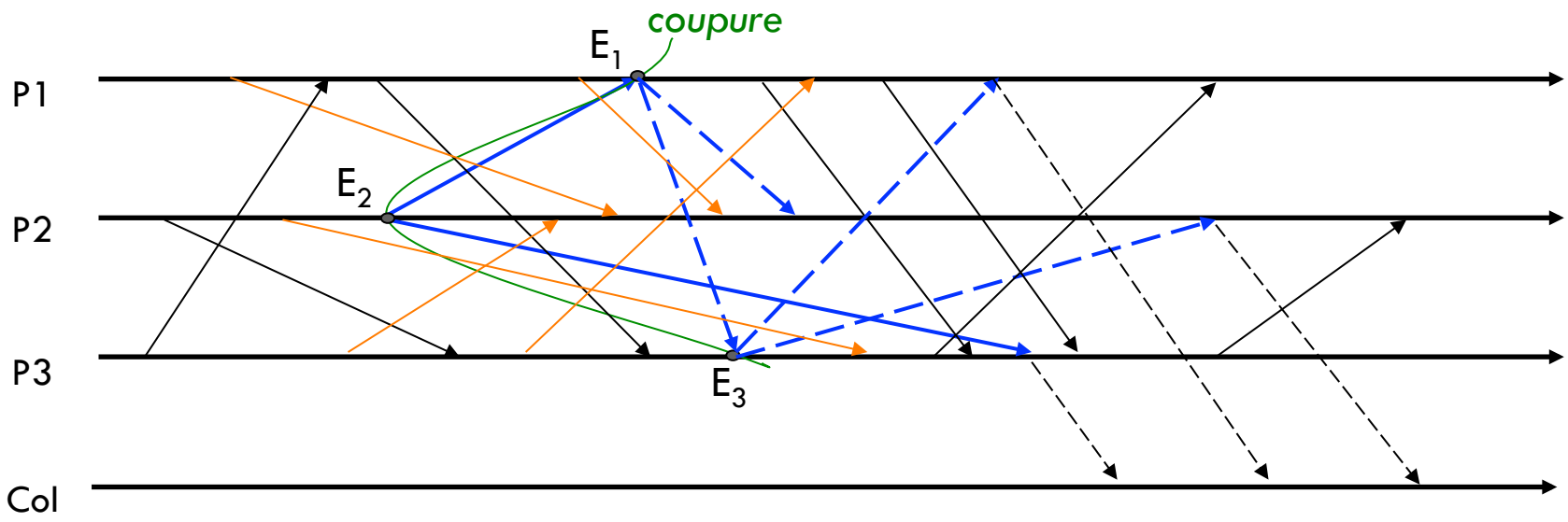
III.3 Algorithme (2)

etatLocal() :
RETOURNER variables locales du processus

toutRecu() :

```
POUR j = 1 .. N SAUF i
    SI ! marqRecu[j]i ALORS
        RETOURNER faux;
    FSI
FPOUR
RETOURNER vrai;
```

III.4 Exemples



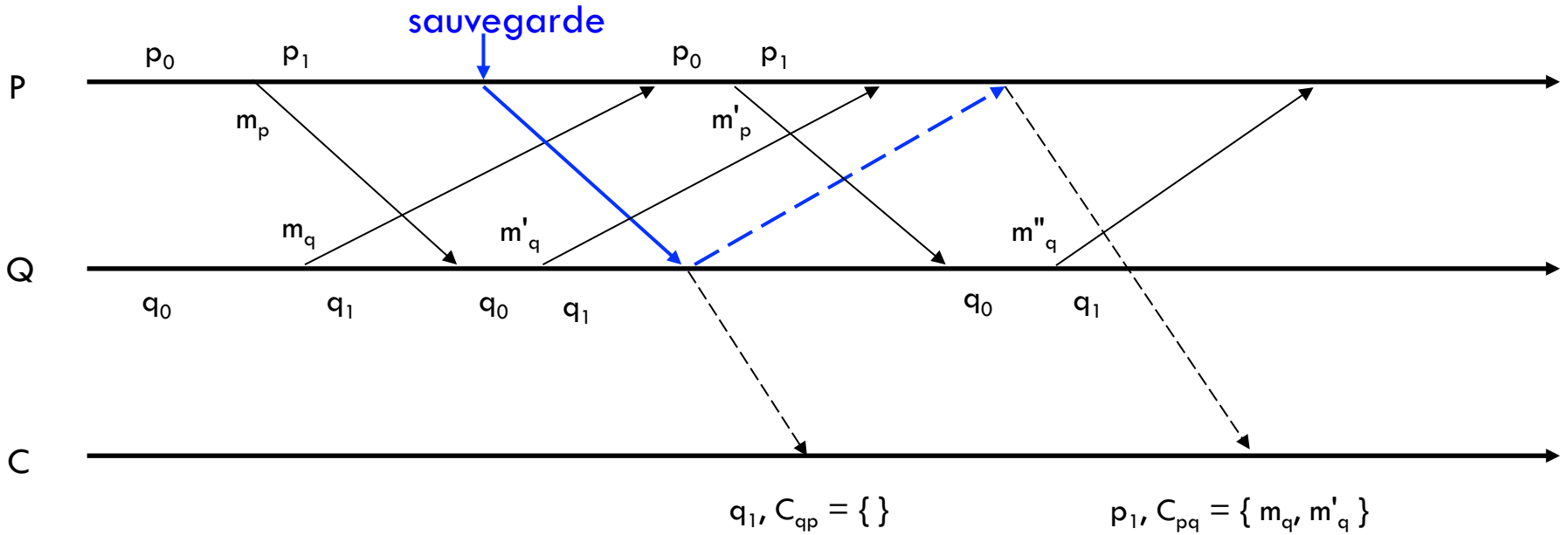
état global collecté

$E_2 \cup C_{12} = \{ m_{12,1}, m_{12,2} \} \cup C_{32} = \{ m_{32,1} \}$ *(par P2)*

$\cup E_3 \cup C_{13} = \{ \} \cup C_{23} = \{ m_{23,2} \}$ *(par P3)*

$\cup E_1 \cup C_{21} = \{ \} \cup C_{31} = \{ m_{31,2} \}$ *(par P1)*

III.4 Exemples



Etat global collecté : < p_1, { }, { m_q, m'_q }, q_1 >

III.5 Conclusion

Propriétés

L'algorithme se termine.

L'état global enregistré est complet.

L'état global enregistré est cohérent.

Remarque

L'état enregistré peut ne pas correspondre à un état global effectivement atteint par le système au cours de la même exécution. Mais on peut montrer qu'il correspond à un état global que le système aurait atteint en réordonnant les événements de façon compatible avec la causalité (i.e. en réordonnant les seuls événements concurrents)

III.5 Conclusion

Extension à des canaux non FIFO [Lai-Yang 87]