Problèmes fondamentaux dans les systemes RÉPARTIs

Introduction

– Un des problème fondamentaux des systèmes répartis : le consensus!

• Comment faire pour mettre d'accord plusieurs processus indépendants?

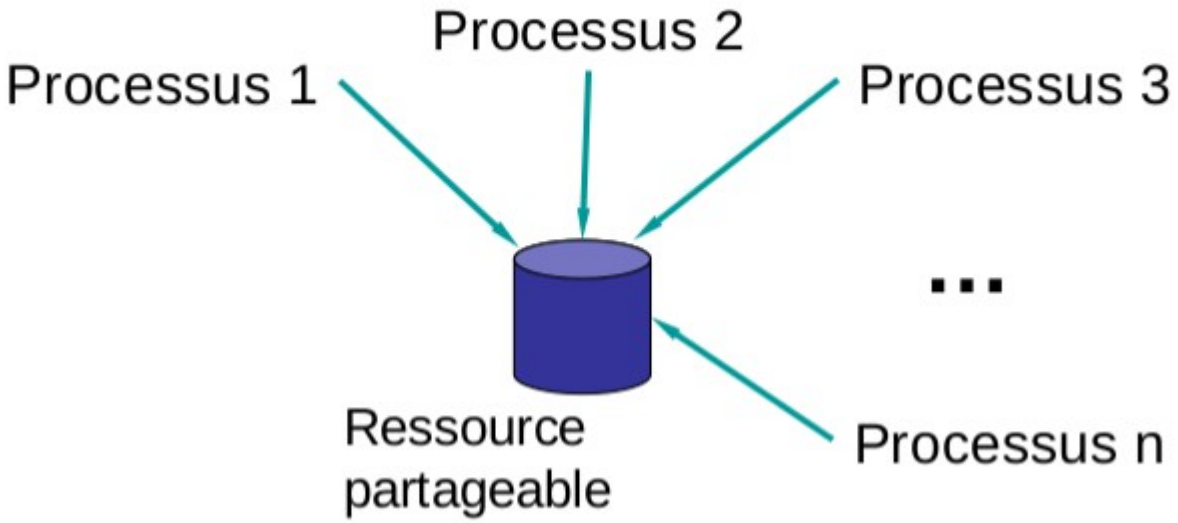
• Comment faire pour coordonner leurs actions?

– Comment faire pour résoudre ce problème?

• Une solution naïve est d'implémenter le modèle maître-esclave ;

– Est-ce que le problème du consensus est le même pour les systèmes synchrones et asynchrones?

Exclusion mutuelle répartie



– Exclusion mutuelle est nécessaire pour :

- Prévenir les interférences ;
- Assurer la cohérence en cas d'accès simultanés aux ressources.

Exclusion mutuelle répartie

- **Synchronisation par exclusion mutuelle répartie**

– Sûreté: un seul client à la fois obtient le verrou.

– Vivacité: personne n'est laissé pour compte, chaque client voit éventuellement sa requête satisfaite.

– Ordre: premier arrivé, premier servi, si un client A fait une demande, envoie un message à B, et B fait une demande, l'ordre logique dit que la demande de A arrive avant celle de B. Elle devrait être servie en premier.

Exclusion mutuelle répartie

- **Serveur central**

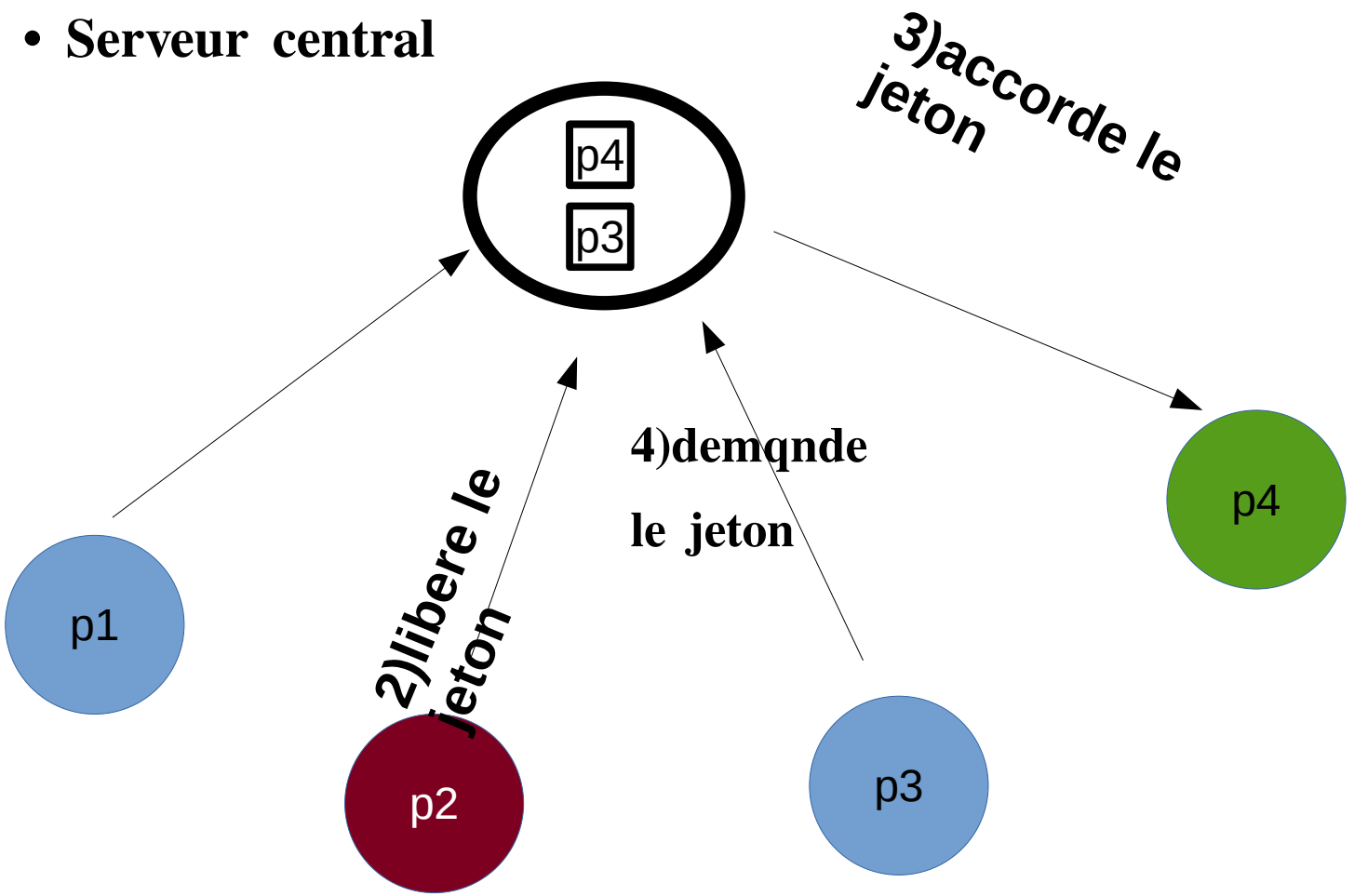
– Un client envoie une demande de verrou, le serveur attend que le verrou soit libre, le donne au client, et le client le relâche éventuellement, ce qui permet au serveur de le donner à un autre client.

– Si le serveur est en panne, élection d'un nouveau serveur (majorité de clients), et vérification de tous les clients pour voir qui possède des verrous ou a soumis une requête (problème si clients non rejoignables en raison de division du réseau).

– Exemple: serveur lockd sous NFS.

Exclusion mutuelle répartie

- **Serveur central**



Exclusion mutuelle répartie

- **Anneau à jeton**

– Le verrou passe de processus en processus constamment.

– Inutile lorsque personne ne requiert le verrou.

– Problème dès qu'un client fait défaut.

– Ne respecte pas premier arrivé, premier servi.

Exclusion mutuelle répartie

- **Exclusion par envoi à tous**
  - **Envoyer une requête à tous.**
  - **Recevoir le O.K. de tous.**
  - **Si on a le verrou, attendre d'en avoir terminé avant de répondre O.K.**
  - **Si on veut le verrou et notre demande est antérieure, attendre le verrou et d'en avoir fini avant de répondre O.K.**
  - **Demande n messages (ou 2n - 2 sans message à tous).**
  - **Tous les clients doivent être actifs.**
  - **Moins bon que serveur central.**

Election

- **Election hiérarchique**

– Lorsqu'un nouvel ordinateur est connecté, ou si le coordonnateur ne peut être rejoint, déclencher une élection.

– Envoyer un message d'élection à ceux de plus haute priorité.

– Pas de réponse, il se proclame coordonnateur et le signale à tous.

– Une réponse, il attend un message de proclamation qui devrait suivre.

– Demande n-1 messages dans le meilleur des cas, n*n si tous les processus déclenchent une élection en même temps.

Election

- **Election en anneau**

– Un participant envoie son message d'élection avec son ID.

– Le prochain participant envoie max(ID reçu, ID) à son voisin.

– Si un participant reçoit un message avec son ID, il se déclare élu et propage la nouvelle.

– Ceci prend un maximum de 3n - 1 messages (ID maximum n - 1, se découvrir élu n, propager la nouvelle n).

Le consensus en réparti

– Plusieurs processus, corrects ou fautifs, échangent des messages.

– Chaque processus doit prendre une décision (e.g. qui est le serveur maître).

– Terminaison: éventuellement tous les processus corrects arrivent à une décision.

– Consensus: tous les processus corrects terminent avec la même décision.

– Intégrité: si tous les processus corrects proposent la même décision, cette décision doit l'emporter.

– Chaque processus correct communique sa proposition et celle déjà connue d'autres processus au groupe. Chaque processus accumule les propositions des autres et se soumet à la proposition majoritaire.

Le calcul de l'état global

Motivation

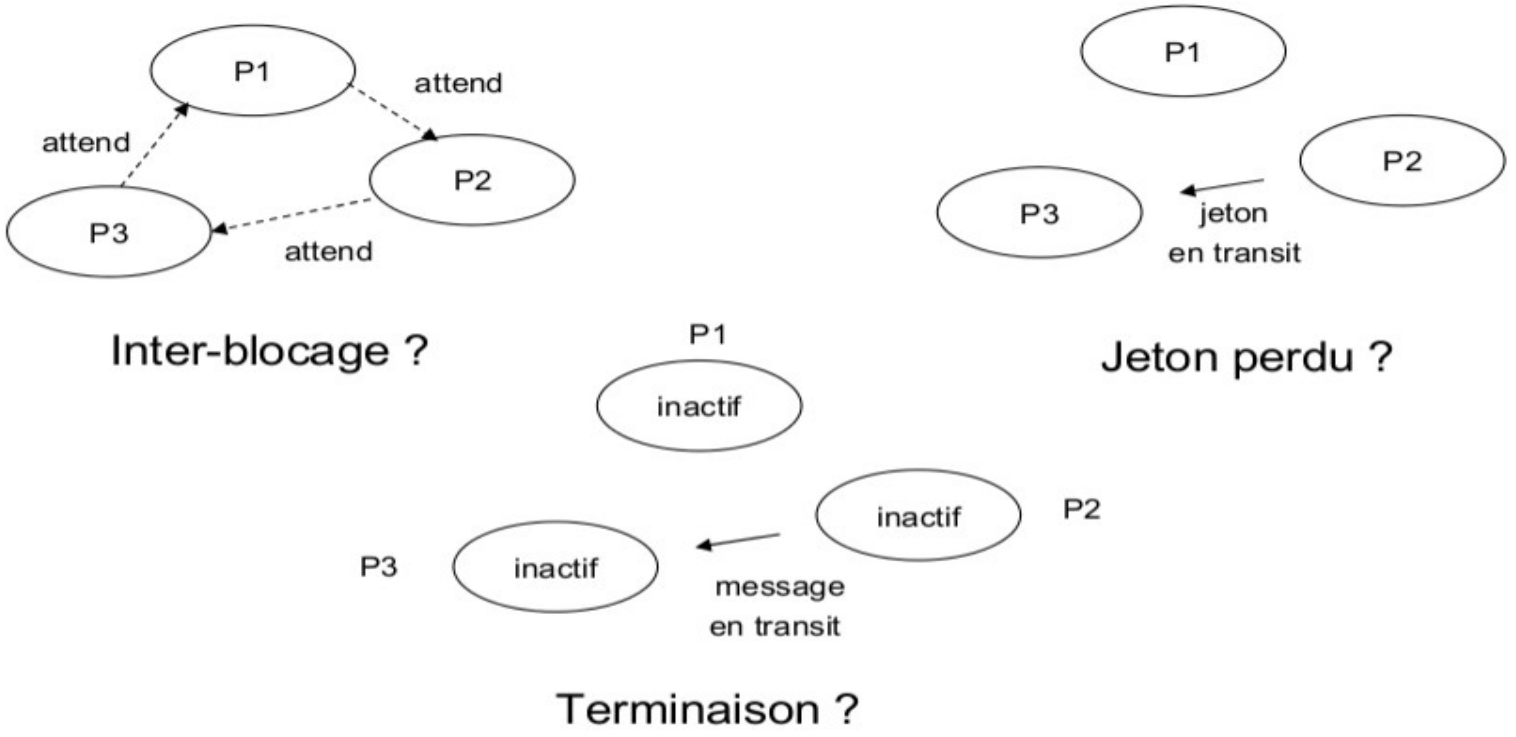
Le calcul de l'état global d'un système réparti consiste à prendre un instantané de l'état du système à un moment donné de son exécution. Plusieurs utilisations possibles :

1- Exécution d'un algorithme centralisé sur l'état global, pour s'affranchir du caractère réparti du système. Exemple : calcul des avoirs d'une banque à partir de ceux de chacunes de ses agences.

Le calcul de l'état global

Motivation(2)

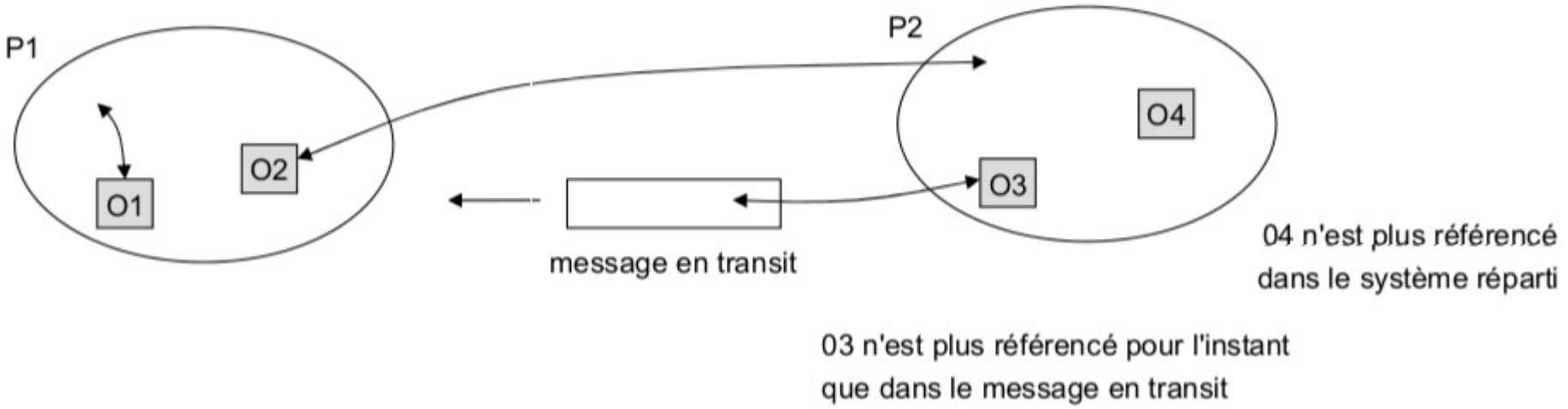
Détection d'états globaux stables, tels que l'inter-blocage ou la terminaison d'un algorithme distribué.



Le calcul de l'état global

Motivation (3)

Ramasse-miette distribué (élimination des objets qui ne sont plus référencés par aucun des processus du système réparti).



- Tolérance aux fautes : l'état global du système est sauvegardé pour servir de point de reprise. En cas de panne d'un des sites, le système réparti est replacé dans le dernier état global sauvegardé, et relancé à partir de cet état.

Définitions

Etat local d'un processus P i : valeur des variables locales de P i (et plus généralement de son contexte d'exécution).

Note : si P i est un processus déterministe, l'état local de P i est déterminé par son état initial et la succession des événements, notamment les événements réception s'étant produit sur P i .

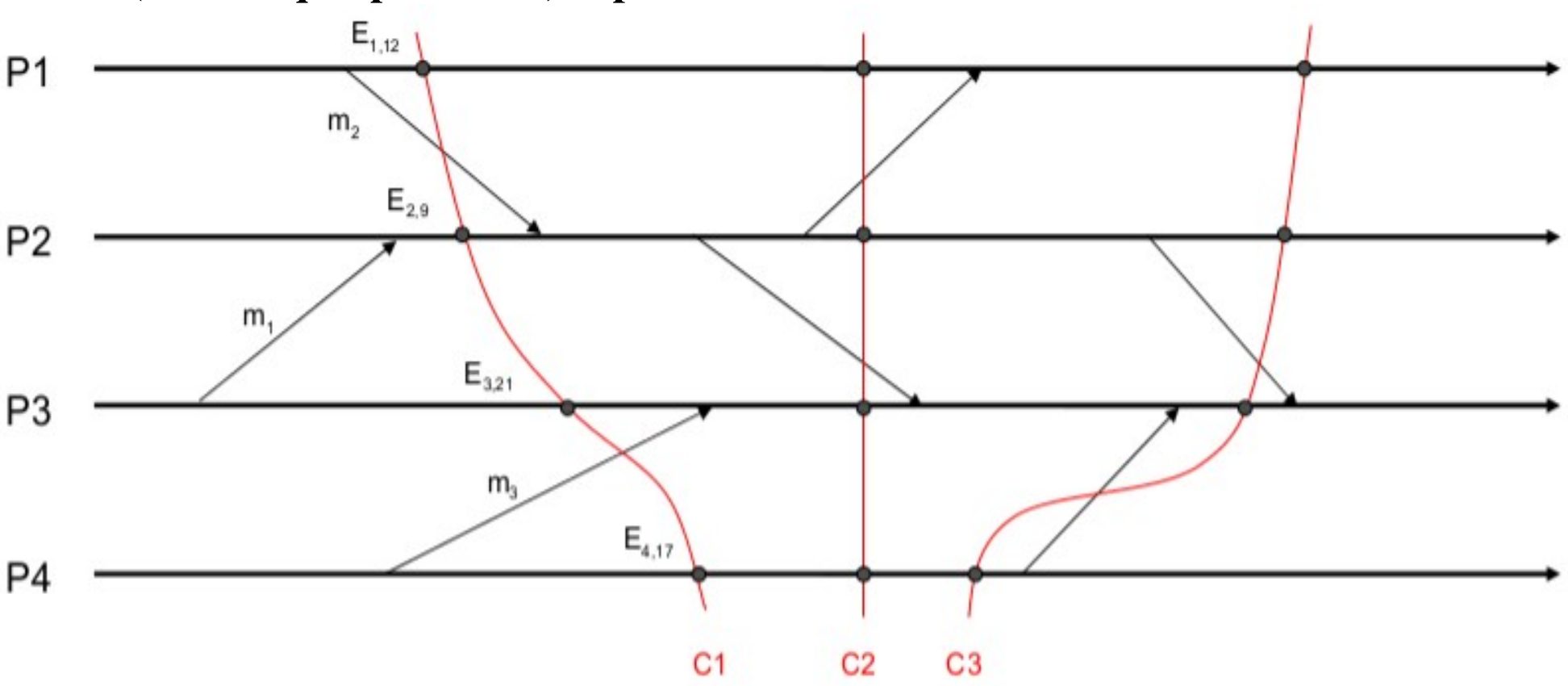
Etat du canal de communication C ij entre les processus P i et P j : ensemble (ordonné, si le canal est FIFO) des messages en transit entre P i et P j .

Note : on considère ici que les canaux de communication sont unidirectionnels ; il y a deux canaux de communication entre les processus P i et P j : C ij , de P i vers P j , et C ji , de P j vers P i .

Etat global d'un système réparti : union de l'ensemble des états locaux des processus P i et de l'ensemble des états des canaux C ij constituant le système réparti.

Définitions

Un état global peut être représenté par une courbe, appelée aussi coupure, sur le diagramme temporel du système réparti. La coupure divise le diagramme en deux zones (sur chaque processus) : passé et futur.

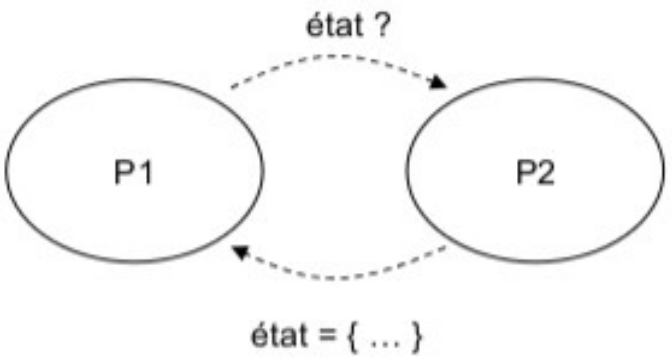


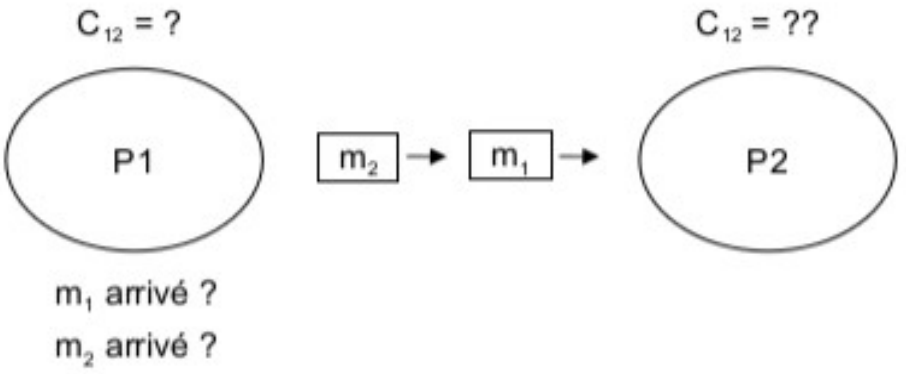
$$E_{C1}(SR) = E(P_1) = E_{1,12} \cup E(P_2) = E_{2,9} \cup E(P_3) = E_{3,21} \cup E(P_4) = E_{4,17} \cup E(C_{12}) = \{m_2\} \cup E(C_{43}) = \{m_3\}$$

Problèmes

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**





Problème de la cohérence des états collectés :

Idéalement, il faudrait figer l'état des différents éléments (processus et canaux) au même instant absolu (exemple : coupure C2) pour qu'ils soient tous en cohérence. Mais il n'y a pas de référence temporelle commune : il faut définir un critère de cohérence plus souple, mais qui donne néanmoins un état global exploitable.

Coupure cohérente

Une coupure est cohérente si elle définit un passé fermé pour la relation de précédence causale. Tous les événements ayant potentiellement causé un événement quelconque se situant avant la coupure sont également avant la coupure. Formellement :

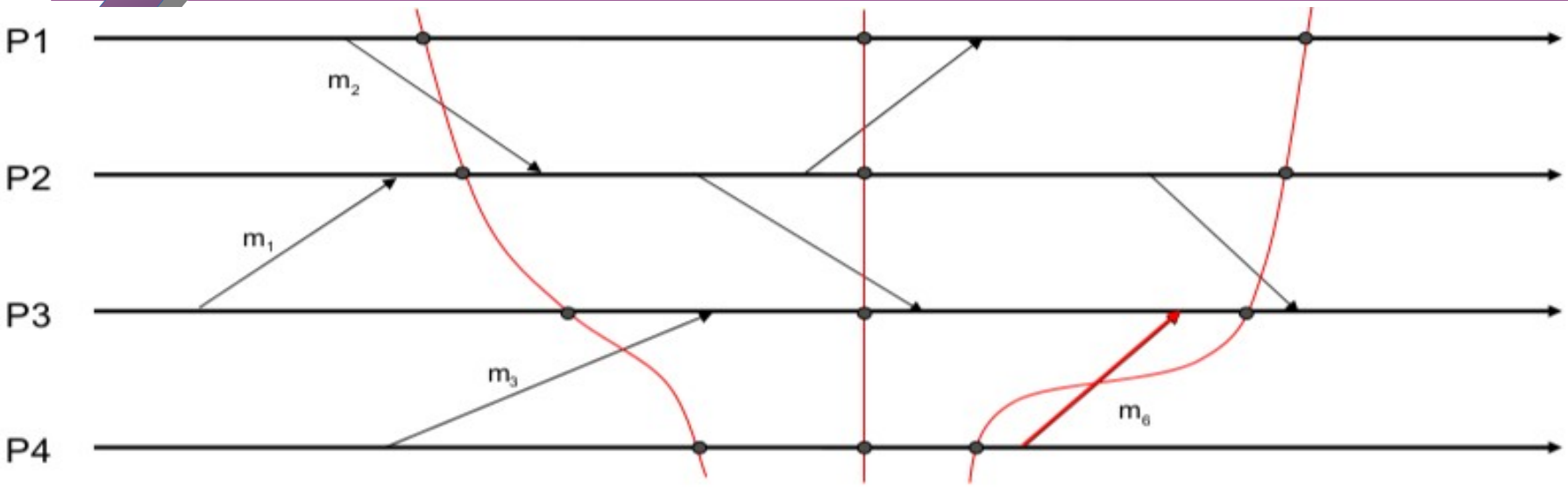
C coupure cohérente ⇔ ∀ e ∈ Passé(C), e' → e ⇒ e' ∈ Passé(C)

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

C'est la condition minimale pour qu'une coupure puisse correspondre à un état global réellement atteint lors de l'exécution du système réparti (sans que cet état global ait nécessairement été atteint).

Par définition, un état global cohérent est un état global obtenu selon une coupure cohrénte.

Coupure cohérente



C1 et C2 sont cohérentes

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

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.