Programmation répartie

Causalité et horloges

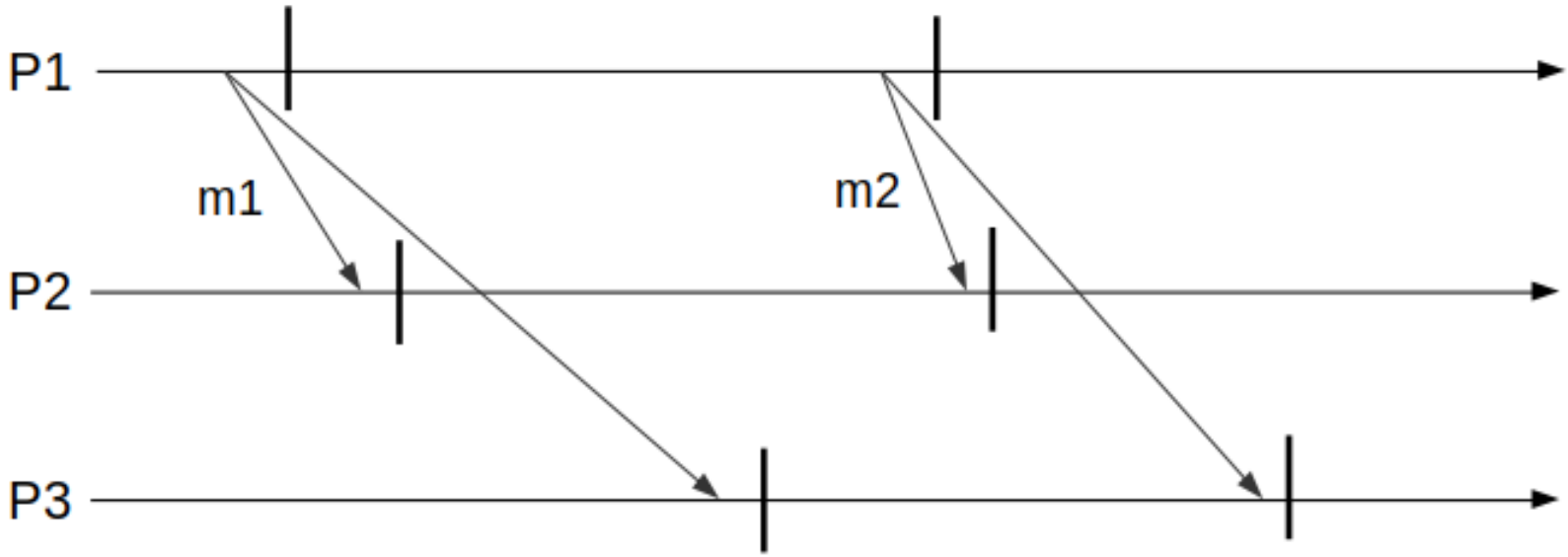
Hinde Bouziane

bouziane@lirmm.fr

Diffusion "best-effort" (1/2)

Propriétés

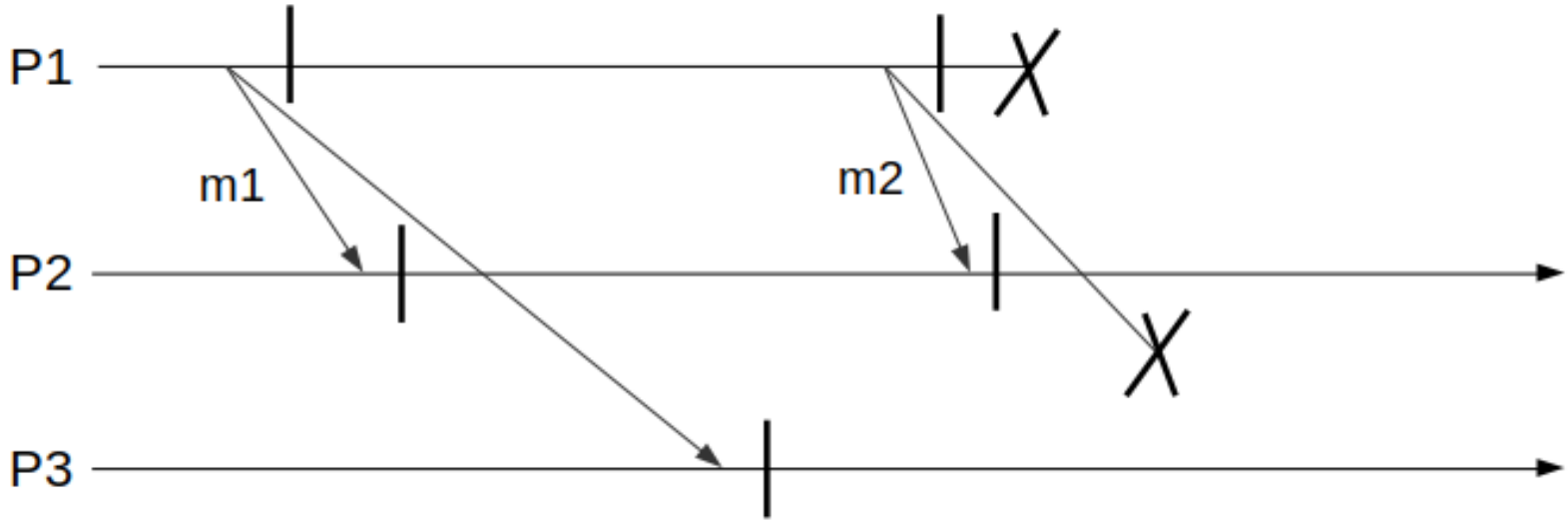
- Canaux de communication fiables.
- Si deux processus P_i et P_j sont corrects, alors, tout message diffusé par P_i sera inévitablement délivré par P_j (viabilité).
- Aucun message n'est délivré plus d'une fois (pas de duplication)
- Aucun message n'est délivré s'il n'a pas été diffusé (pas de création).



Diffusion "best-effort" (2/2)

Limitation

- Comment réagit cet algorithme en présence de pannes ?
- Exemple : que se passe-t-il si la source P_1 tombe en panne ?
  - Les précédentes propriétés disent : si un processus envoie un message, tous les autres processus reçoivent ce message. Mais si la source tombe en panne, rien n'est garanti.

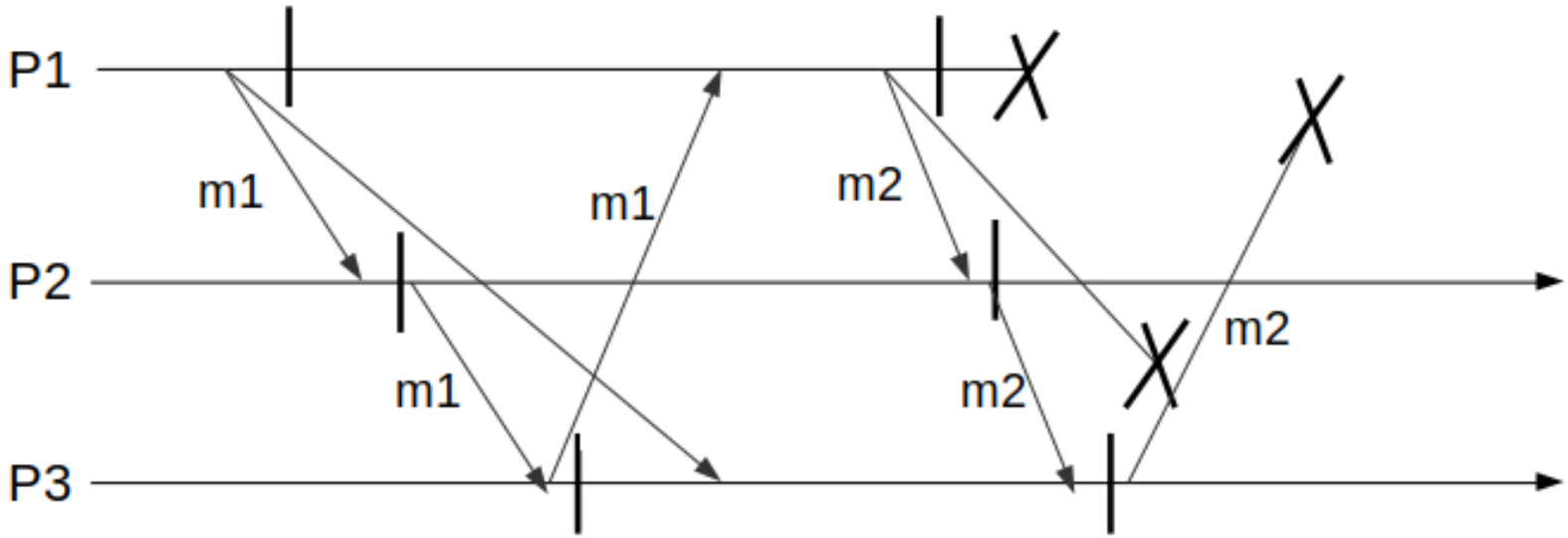


Diffusion "fiable"

Propriétés

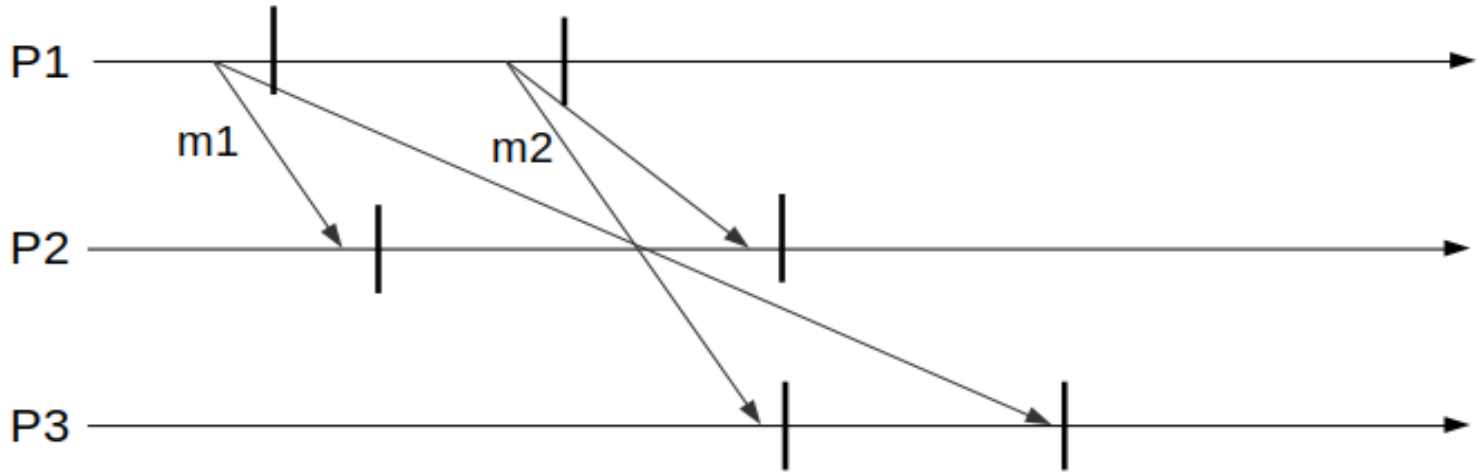
Propriétés précédentes + la suivante :

- Pour chaque message m, si un processus correct délivre m, alors tous les autres processus délivrent m.
  - Comment ? il suffit de demander à chaque processus qui reçoit un message m de le retransmettre.
  - Exemple d'algorithme vu dans l'introduction.

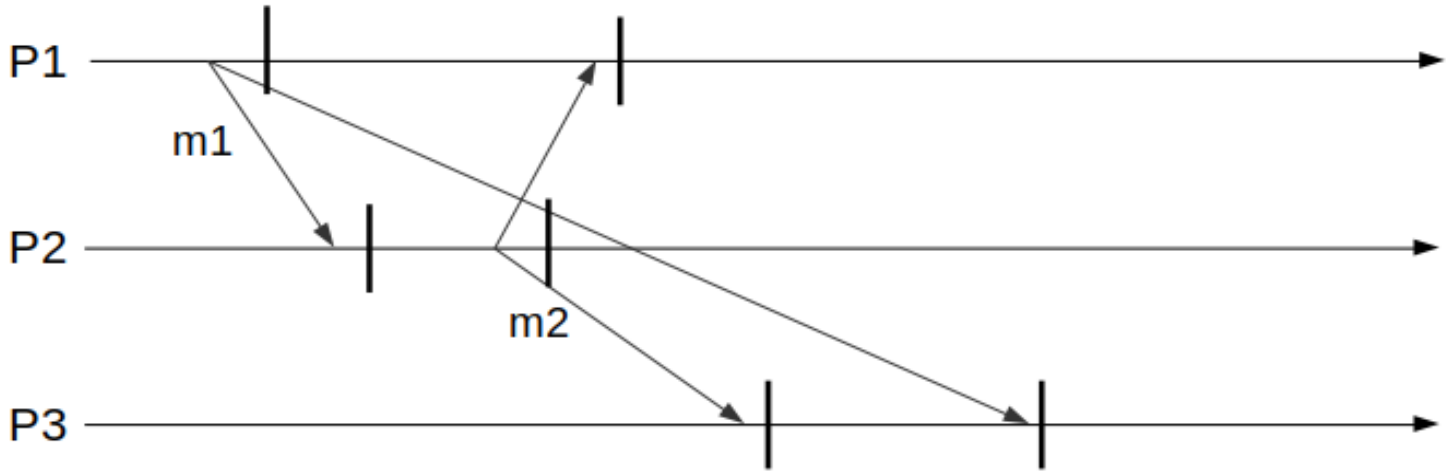


(In)dépendance des messages

Situation 1 :



Situation 2 :



Introduction

- Définie par Leslie Lamport en 1978.
- Définie une relation de dépendance entre les messages.
- Assurer que les messages sont délivrés dans leur ordre de dépendance.

Définition dans le cadre de la diffusion

Soit deux messages m_1 et m_2. m_1 précède causalement (ou précède ou implique) m_2 "$m_1 \rightarrow m_2$", si et seulement si (ssi) une des trois propriétés suivantes est satisfaite :

- un même processus P_i a diffusé m_1 puis a diffusé m_2.
- un même processus P_i a délivré m_1 ensuite a diffusé m_2.
- il existe un message m_3 tel que $m_1 \rightarrow m_3$ et $m_3 \rightarrow m_2$ (fermeture transitive).

Diffusion causale

Propriétés

Propriétés de la diffusion fiable + la suivante :

- Si un processus P_i délivre un message m_y, alors, P_i a délivré tout message m_x tel que $m_x \rightarrow m_y$.

Remarque

La plupart des réseaux sociaux mettent en oeuvre cette propriété de causalité.

Question

Comment assurer cette propriété ? Il y a un nombre conséquent de travaux autour de ce sujet et qui sont toujours d'actualité dans le monde de la recherche.

Deux grandes classes d'algorithmes

Classe 1

- Principe : on envoie avec un message son passé causal.
- Solution simple et directe (faisable d'un point de vu calculabilité).
- Solution inefficace.

Classe 2

- Principe : Envoyer un "code" avec un message m qui permet d'avertir le récepteur qu'il y a au moins un message dans le passé causal de m qu'il faut attendre avant de délivrer m (Lamport en 1978).
- Utilisé jusqu'à présent sous différentes formes.
- Représentation de ce code dans la suite ce chapitre.

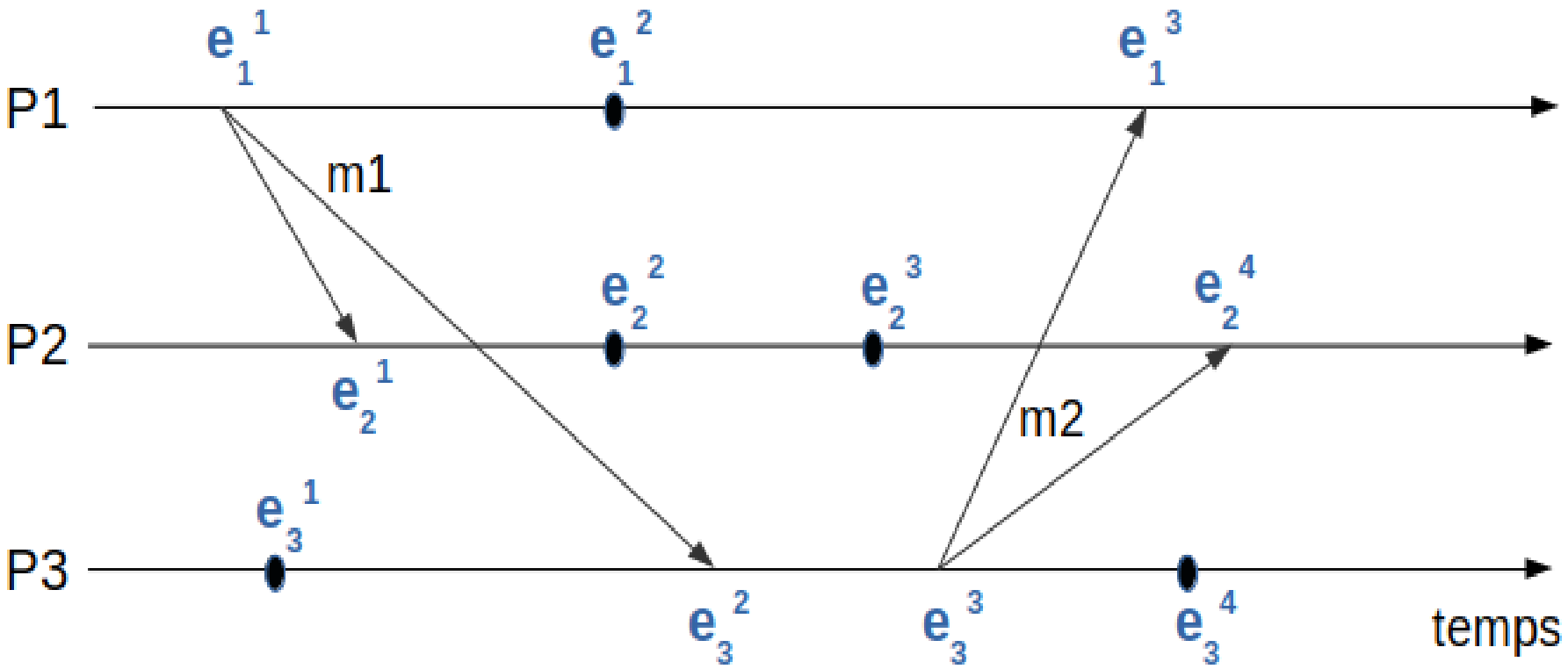
Remarque

La causalité est utilisé dans un cadre plus large d'algorithmes répartis

Retour sur la modélisation d'un système réparti (1/2)

- Soit $\Pi = \{P_1, P_2, ..., P_N\}$ l'ensemble des processus d'un système réparti S.
- Chaque P_i exécute une séquence ordonnée d'évènements $e_i^1, e_i^2, ..., e_i^k$
  - évènements locaux à P_i
  - une relation "intervient avant" notée "$<$" (exemple $e_i^1 < e_i^2$)
- Trois formes d'évènements :
  - Un calcul local ou interne (mise à jour d'une variable, etc.).
  - Une émission d'un message vers un autre processus.
  - Une réception d'un message émis par un autre processus.

Retour sur la modélisation d'un système réparti (2/2)



Causalité

Définition

- $e \rightarrow e'$ ssi :
  - e et e' sont locaux à un même processus et $e < e'$
  - $\exists m \mid e = emission(m)$ et $e' = reception(m)$
  - $\exists e'' \mid e \rightarrow e''$ et $e'' \rightarrow e'$

Exercice

A partir de la figure précédente, donner des exemples impliquant chacune de ces trois règles.

Horloge de Lamport

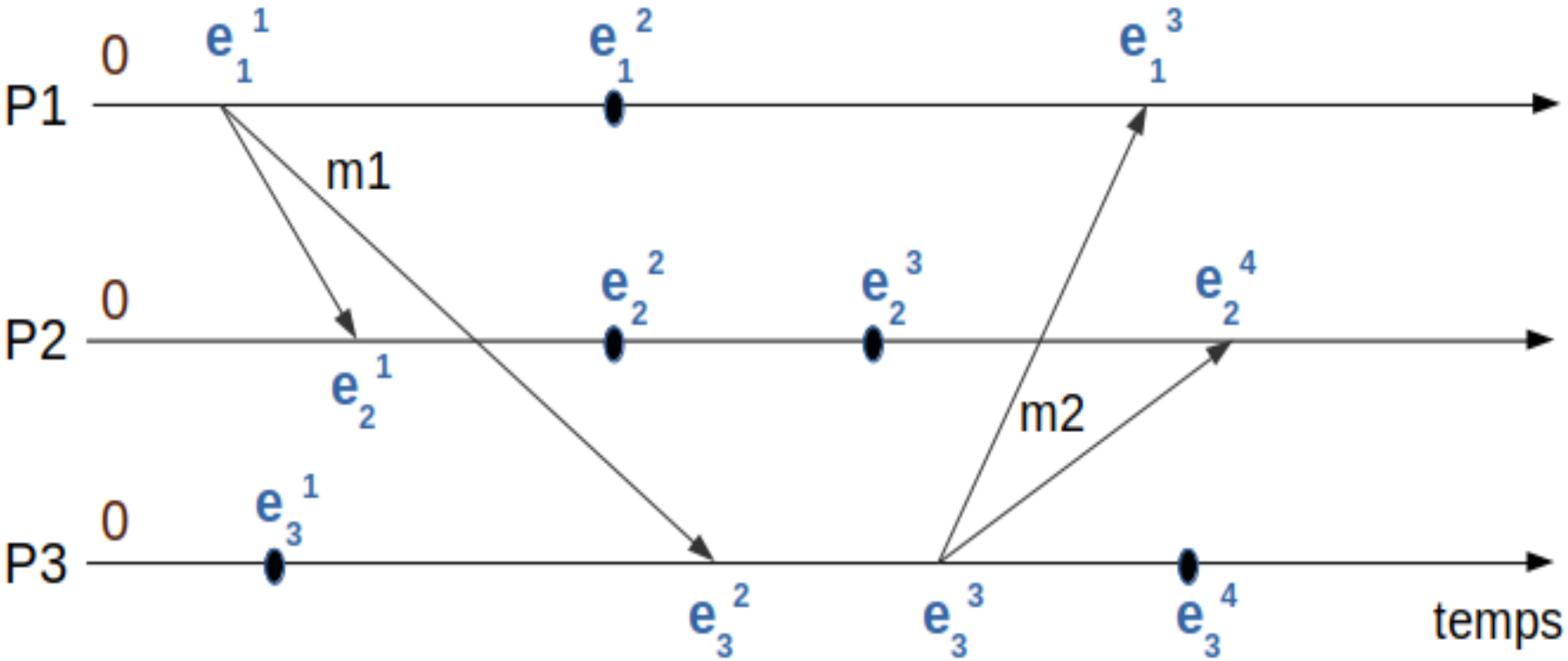
Principe et définitions

- Définie pour capturer la notion de causalité dans un système réparti.
- L'idée est que chaque évènement e se voit associé une horloge notée $H(e)$. Cette horloge est définie comme suit :
  - $e \rightarrow e' \implies H(e) < H(e')$
- En pratique, chaque processus P_i va estampiller ses évènements.
  - Initialisation : $H_{P_i} = 0$
  - Si e est un évènement interne : $H_{P_i} = H_{P_i} + 1$; $H(e) = H_{P_i}$.
  - Si e est l'émission d'un message m : $H_{P_i} = H_{P_i} + 1$; $H(e) = H_{P_i}$; ajouter dans le message m à envoyer une estampille $E(m) = H_{P_i}$.
  - Si e est la réception d'un message $< m, E(m) >$: $H_{P_i} = max(H_{P_i}, E(m)) + 1$; $H(e) = H_{P_i}$.

Exercice

Estampiller les évènements de la figure précédente.

Exemple


P1
P2
P3
0
0
0
e_1^1
e_1^2
e_1^3
e_2^1
e_2^2
e_2^3
e_2^4
e_3^1
e_3^2
e_3^3
e_3^4
m1
m2
temps

Questions

Ordre total ou partiel ?

- Que dire des évènements e_1^2, e_2^1 et e_3^2 ? Proposer une solution pour les ordonner.
- Dater l'ensemble des évènements du système (donner un ordre).

A quoi sert ces estampilles

Exemple : en cas d'erreur lors de l'évènement e_2^4 de la figure précédente, l'horloge de Lamport dit que toutes les "causes" de l'évènement e_2^4 ont une horloge < 5. La source de l'erreur sera à rechercher dans tous les événements ayant une telle valeur d'horloge.

Pourquoi ne pas utiliser les horloges physiques ?

Autrement dit, peut on utiliser les horloges physiques (date et heure) pour mettre en oeuvre la causalité ?

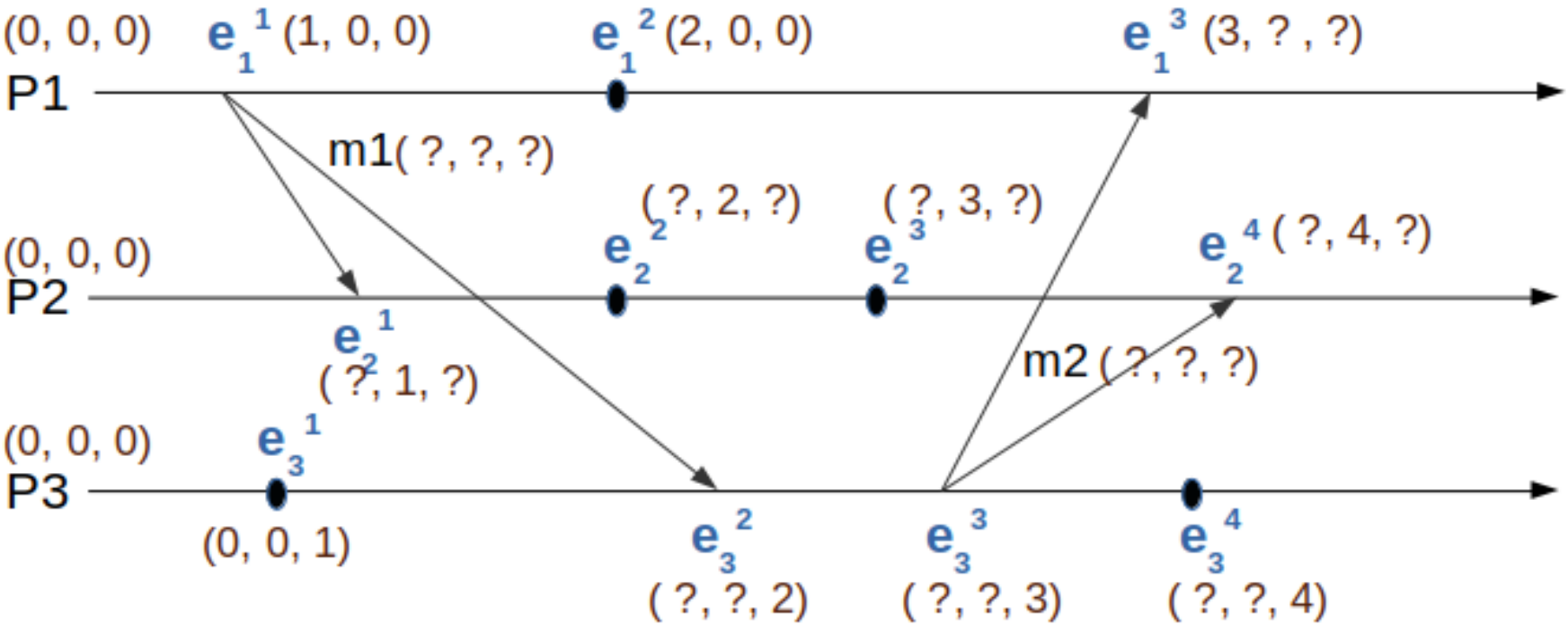
Introduction

- Appelée horloge vectorielle et Définie par C. Fidge et F. Mattern (1988).
- Pour capturer la causalité dans un système réparti, avec des avantages par rapport à l'horloge de Lamport.

Principe

- Soit $\Pi = \{P_1, ..., P_N\}$ l'ensemble des processus d'un système réparti S.
- Chaque P_i gère un vecteur HV_i de N entiers, initialisé à $(0, ...0)$
- Chaque évènement e se voit associé une horloge $HV(e)$ comme suit :
  - Si e est un évènement interne : $HV_i[i] = HV_i[i] + 1$; $HV(e) = HV_i$.
  - Si e est l'émission d'un message m : $HV_i[i] = HV_i[i] + 1$; $HV(e) = HV_i$; ajouter à m une estampille $EV(m) = HV_i$.
  - Si e est la réception d'un message $< m, EV(m) >$: $HV_i[i] = HV_i[i] + 1$; $\forall j \neq i$, $HV_i[j] = max(HV_i[j], EV(m)[j])$; $HV(e) = HV_i$.

Exemple


(0, 0, 0)
e1 1 (1, 0, 0)
e1 2 (2, 0, 0)
e1 3 (3, ? , ?)
P1
m1(?, ?, ?)
(?, 2, ?)
(?, 3, ?)
(0, 0, 0)
e2 2
e2 3
e2 4 (?, 4, ?)
P2
e2 1
(?, 1, ?)
m2 (?, ?, ?)
(0, 0, 0)
e3 1
P3
(0, 0, 1)
e3 2
(?, ?, 2)
e3 3
(?, ?, 3)
e3 4
(?, ?, 4)

Relation d'ordre

Propriétés

- $HV \leq HV' \equiv \forall i, HV[i] \leq HV'[i]$
- $HV < HV' \equiv HV \leq HV'$ et $HV \neq HV'$
- $HV \| HV' \equiv \neg(HV < HV')$ et $\neg(HV' < HV)$

Dépendance causale

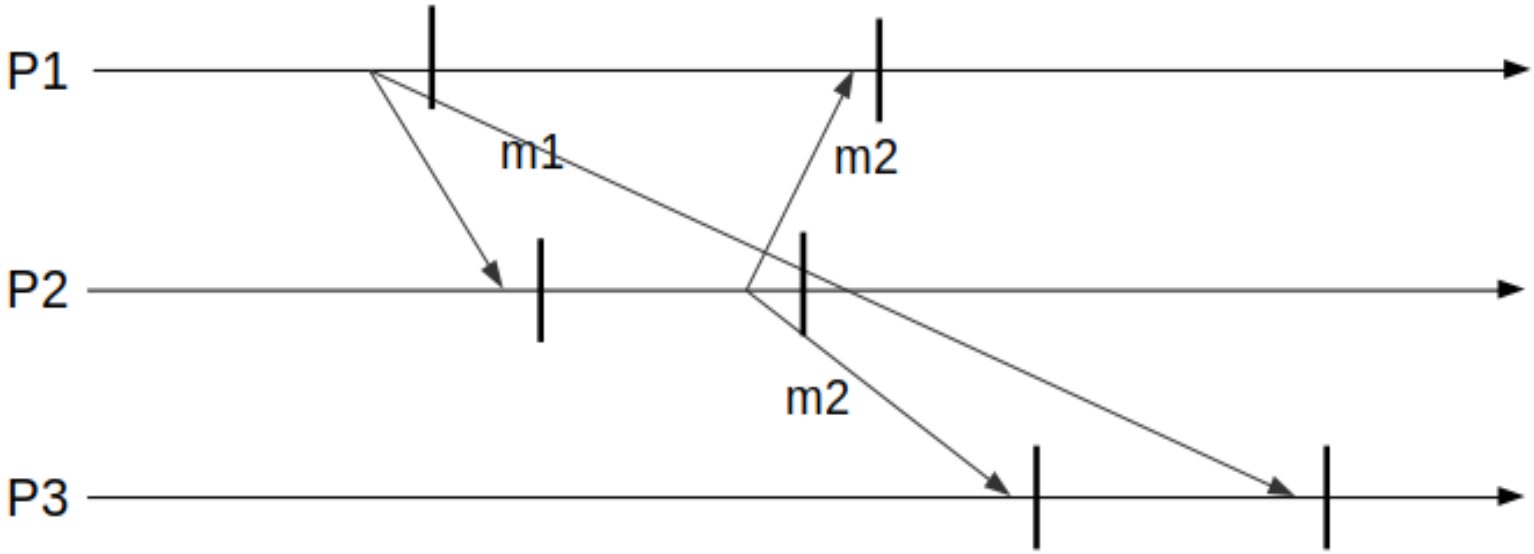
- $\forall e, e' : e \rightarrow e' \Leftrightarrow HV(e) < HV(e')$
- $e \neq e', e \| e' \Leftrightarrow HV(e) \parallel HV(e')$ (e et e' sont concurrents)

Exercice

- Ordonner les évènements e_1^2, e_2^1 et e_3^2 ?

Délivrance causale : diffusion (1/3)

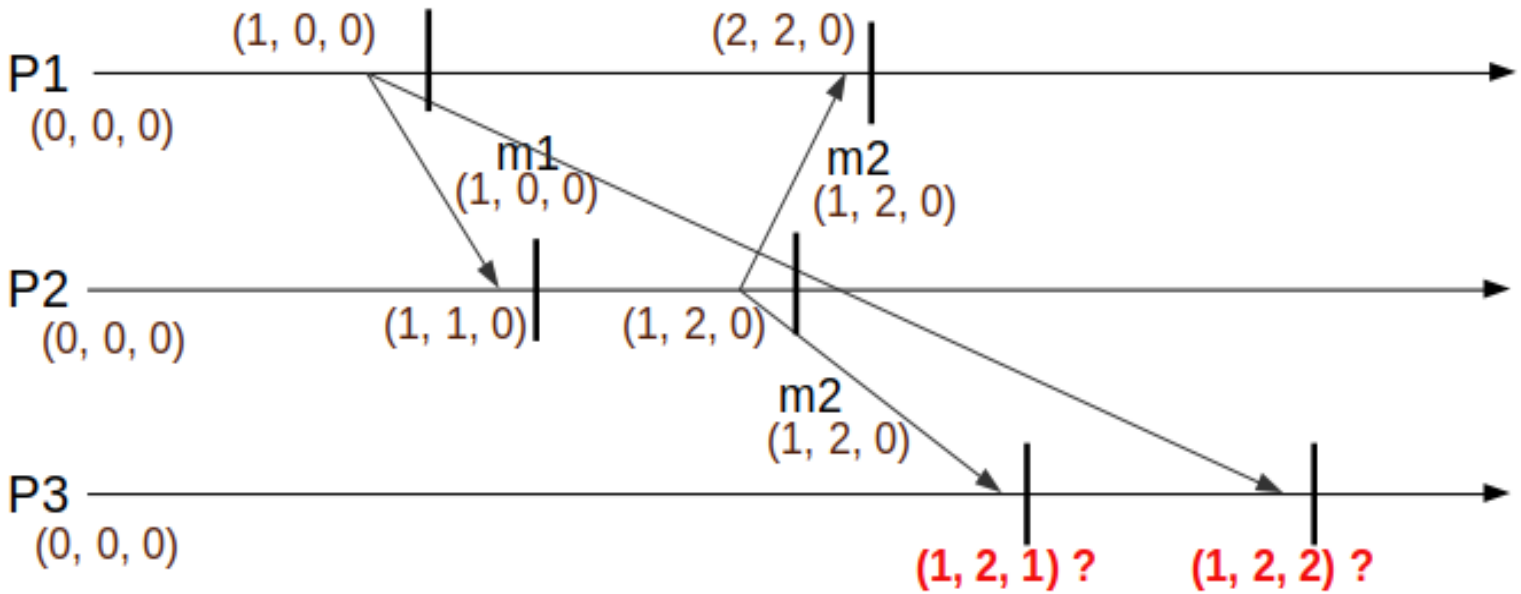
Retour à un exemple de la diffusion fiable



L'horloge vectorielle permet-elle de constater un désordre dans l'arrivée des messages ?

Délivrance causale : diffusion (2/3)

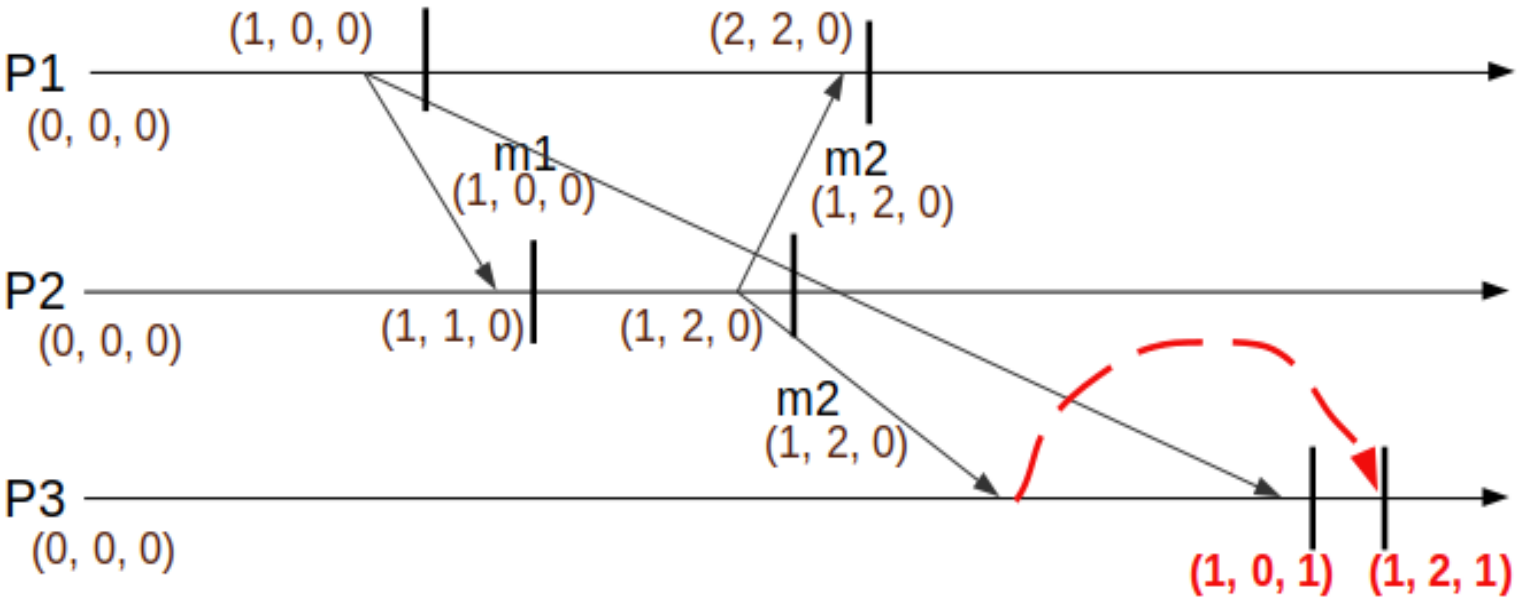
Si on date les évènements comme précédemment, on obtiendrait :



Mais quelle information P_3 a lors de la réception de $m1$?

Délivrance causale : diffusion (3/3)

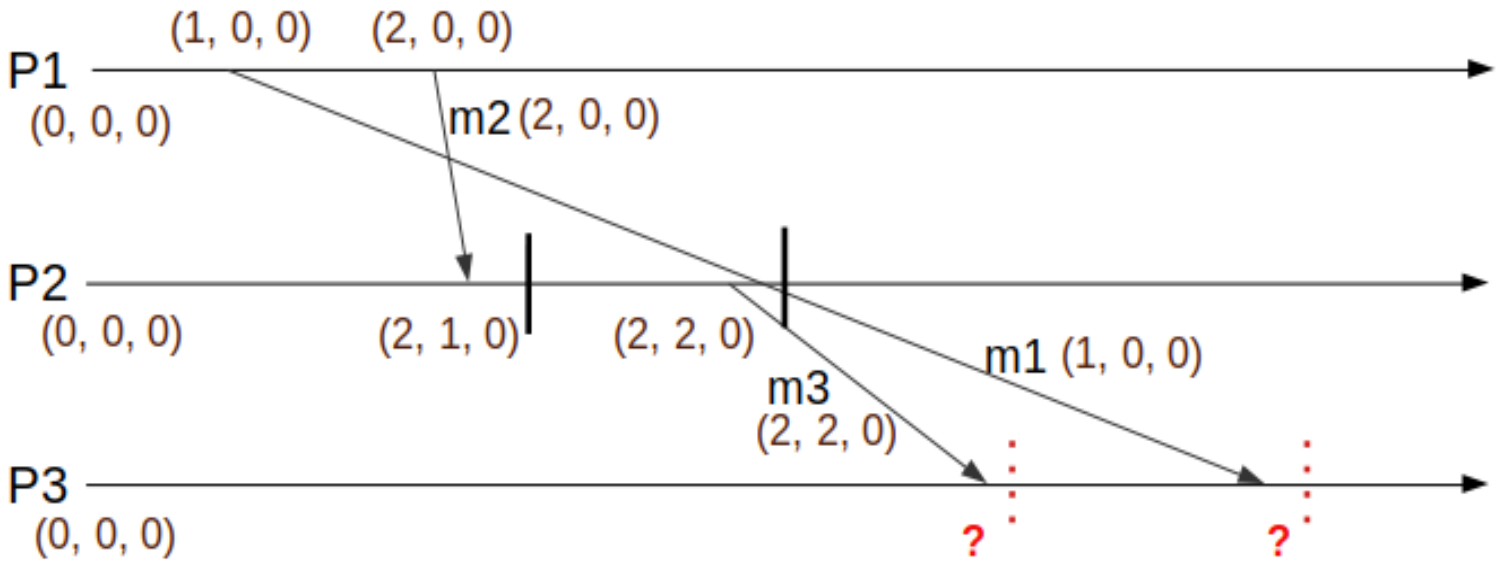
Correction :



Dans le cas de l'exemple de la diffusion fiable, l'horloge vectorielle permet de remettre dans l'ordre des messages arrivant dans le désordre.

Délivrance causale : communications point à point

Soit la situation suivante :



P_3 a t-il des informations lui permettant de retarder la délivrance de m_3 ?

Introduction

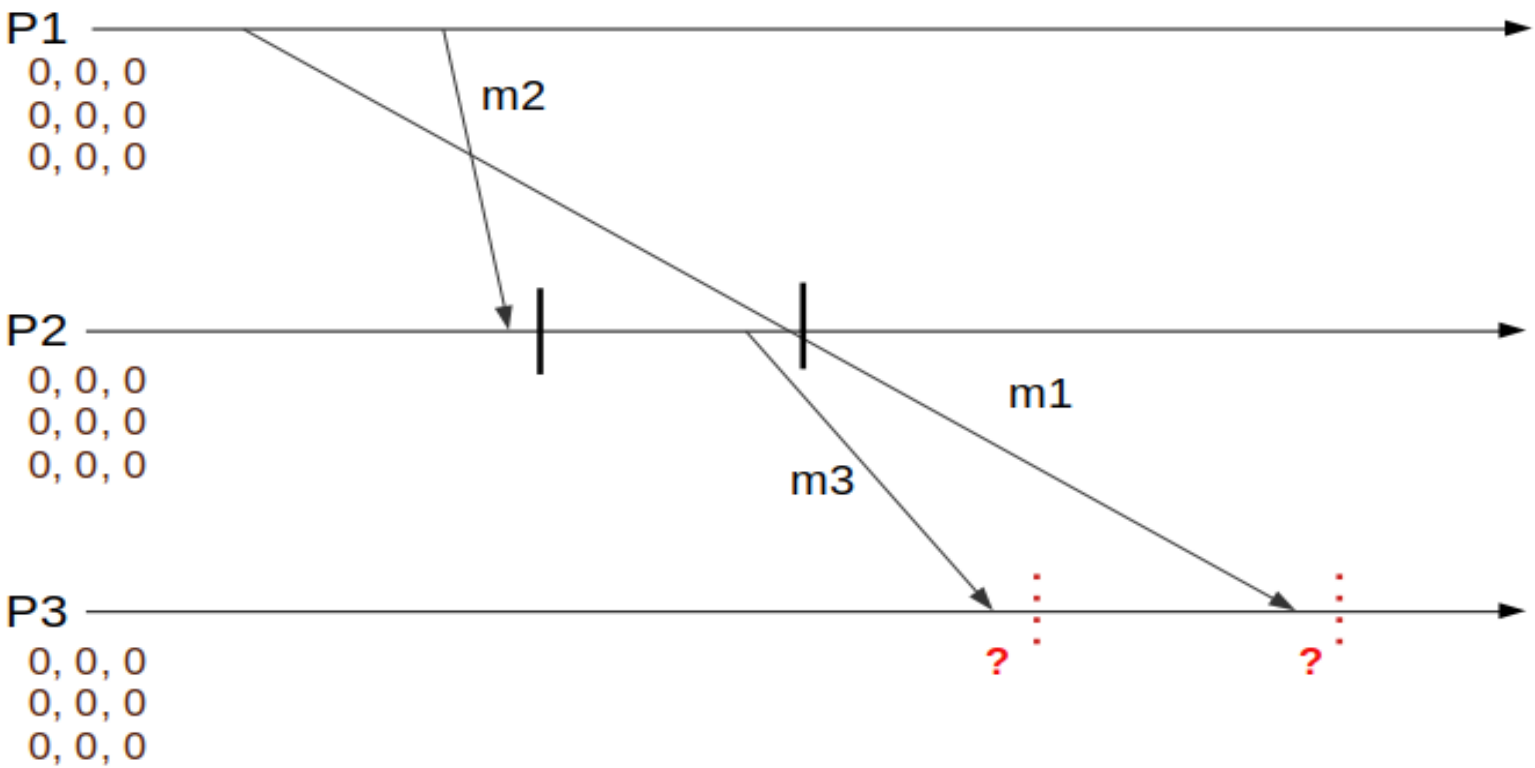
- Objectif : avoir un système de datation plus précis.
- Une horloge HM_i est maintenue par chaque P_i sous forme de matrice à $N * N$ (N étant le nombre total de processus).
- Chaque élément $HM_i[j, k]$ $(j \neq k)$ est égal au nombre de messages envoyés par P_j vers P_k et dont P_i a connaissance (envois causalement antérieur à l'instant présent).

Principe

Soit $\Pi = \{P_1, ..., P_N\}$ l'ensemble des processus d'un système réparti S. Chaque P_i gère une matrice HM_i de $N * N$ entiers, chaque élément initialisé à 0.
Chaque évènement e se voit associé une horloge $HM(e)$ comme suit :

- Si e est un évènement interne : $HM_i[i,i] = HM_i[i,i] + 1$; $HM(e) = HM_i$.
- Si e est l'émission d'un message m de P_i vers P_j : $HM_i[i,i] = HM_i[i,i] + 1$; $HM_i[i,j] = HM_i[i,j] + 1$; $HM(e) = HM_i$; ajouter à m une estampille $EM(m) = HM_i$.
- Si e est la réception d'un message $< m, EM(m) >$ de la part de P_j :
  - vérifier que m peut être délivré :
    - $EM[j,i] = HM_i[j,i] + 1$?
    - $\forall k \neq i,j : EM[k,i] \leq HM_i[k,i]$?
  - Si oui, mettre à jour l'horloge et délivrer m
    - $HM_i[i,i] = HM_i[i,i] + 1$
    - $HM_i[j,i] = HM_i[j,i] + 1$
    - $\forall k \neq i,j$ et $\forall l \neq i : HM_i[k,l] = max(HM_i[k,l], EM[k,l])$

Exemple : retour à la délivrance causale (communications point à point)



Introduction

Utilité multiple

- observation du comportement
- détection de propriétés (terminaison, prédicats, etc.)
- reprise après panne, etc.

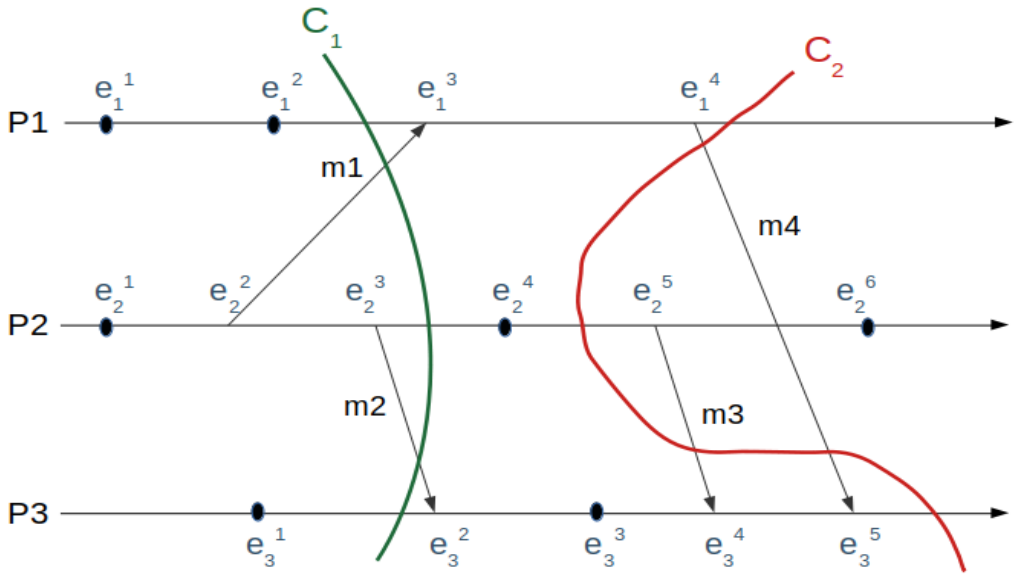
Comment capturer cet état ?

Via des "prises de photos"

- capture instantanée de l'ensemble du système (état local de chaque site et de chaque canal de communication) ?
  - Irréalisable, mais pour quelle raison ?
- capture des derniers évènements avant la photographie sur chaque site.
  - notion de coupure.
  - contrainte : aboutir à un état global cohérent (on parle de coupure cohérente).

Définition d'une coupure

Une coupure C est l'ensemble d'évènements tels que :
Si $e \in C$ et (e' précède **localement** e), alors $e' \in C$

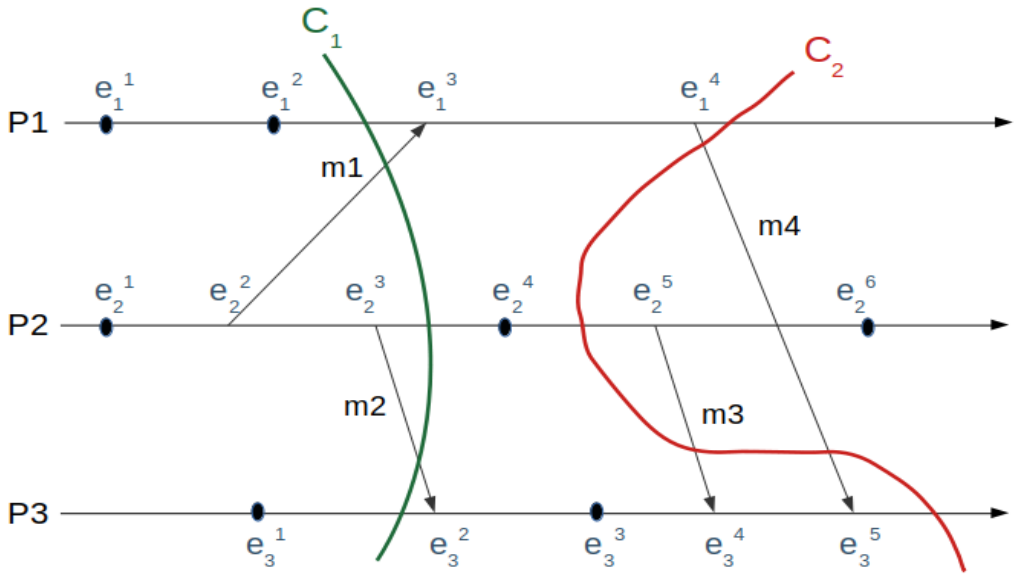


Exercice

- $C_1 = \{ ? \}$
- $C_2 = \{ ? \}$

Définition d'une coupure

Une coupure C est l'ensemble d'évènements tels que :
Si $e \in C$ et (e' précède **localement** e), alors $e' \in C$

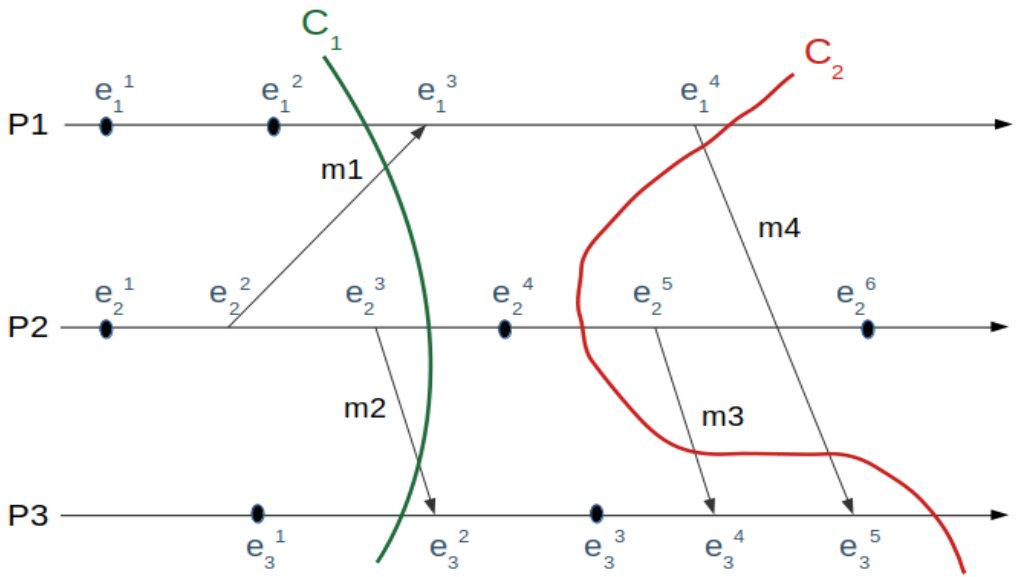


Exercice

- $C_1 = \{e_1^1, e_1^2, e_2^1, e_2^2, e_2^3, e_3^1\}$
- $C_2 = \{e_1^1, e_1^2, e_1^3, e_1^4, e_2^1, e_2^2, e_2^3, e_2^4, e_3^1, e_3^2, e_3^3, e_3^4, e_3^5\}$

Définition d'une coupure cohérente

Une coupure C est cohérente si la propriété suivante est vérifiée :
$e \in C$ et $(e' \rightarrow e) \Longrightarrow e' \in C$

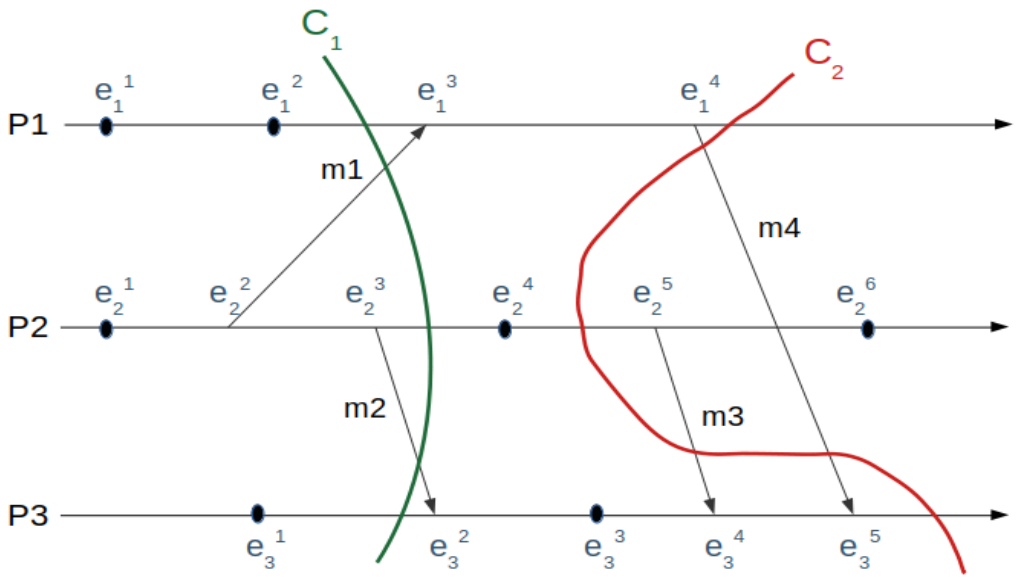


Exercice

- C_1 cohérente ou non ?
- C_2 cohérente ou non ?

Définition d'une coupure cohérente

Une coupure C est cohérente si la propriété suivante est vérifiée :
$e \in C$ et $(e' \rightarrow e) \Longrightarrow e' \in C$



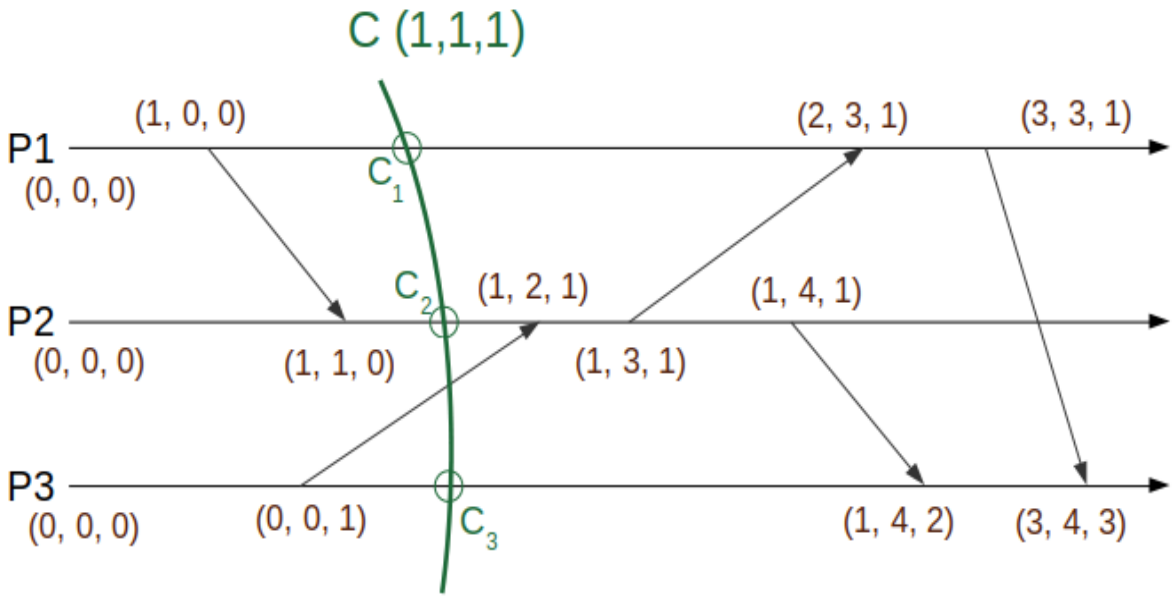
Exercice

- C_1 cohérente ? oui
- C_2 cohérente ? non : $e_2^5 \notin C_2$ alors que $e_3^4 \in C_2$ et $e_2^5 \rightarrow e_3^4$

Coupures cohérentes et horloges vectorielles (1/2)

Date d'une coupure

La date d'une coupure $C = (c_1, c_2, ..., c_N)$ est définie par la valeur $HV(C) = sup(HV(c_1), HV(c_2), ..., HV(c_N))$ (calculée élément par élément)



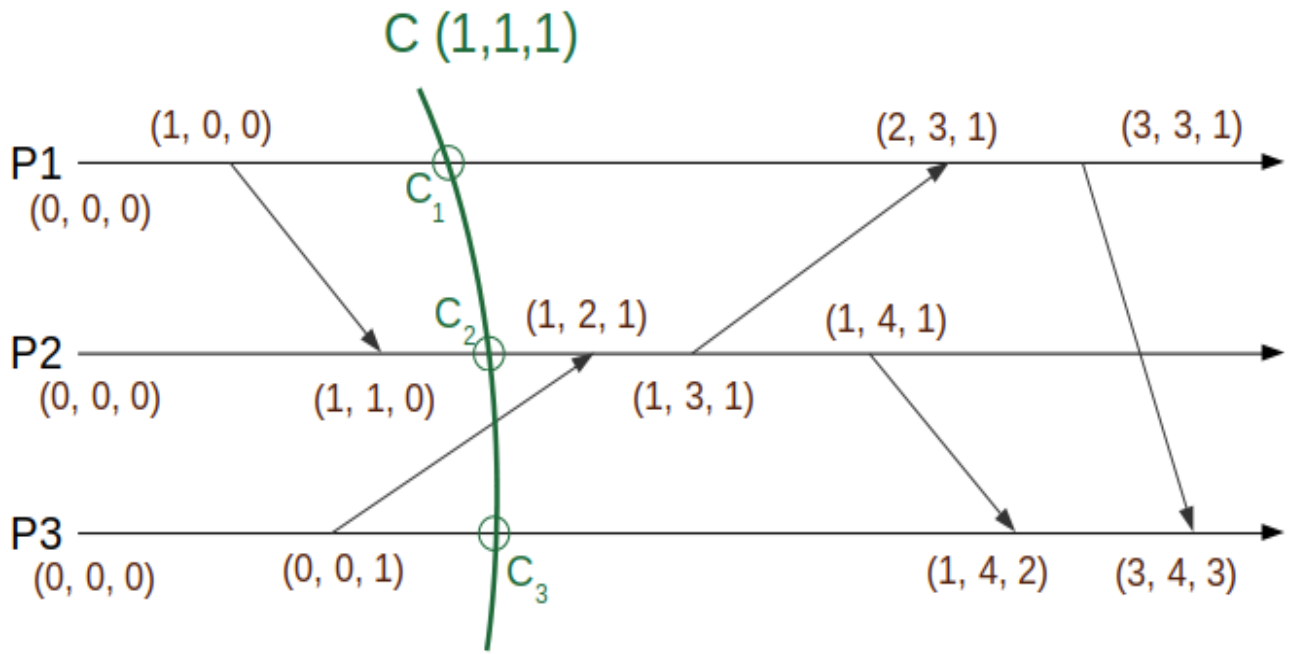
Condition de cohérence

C cohérente $\Leftrightarrow HV(C) = (HV(c_1)[1], HV(c_2)[2], ..., HV(c_N)[N])$.

Coupures cohérentes et horloges vectorielles 2/2

Exercice

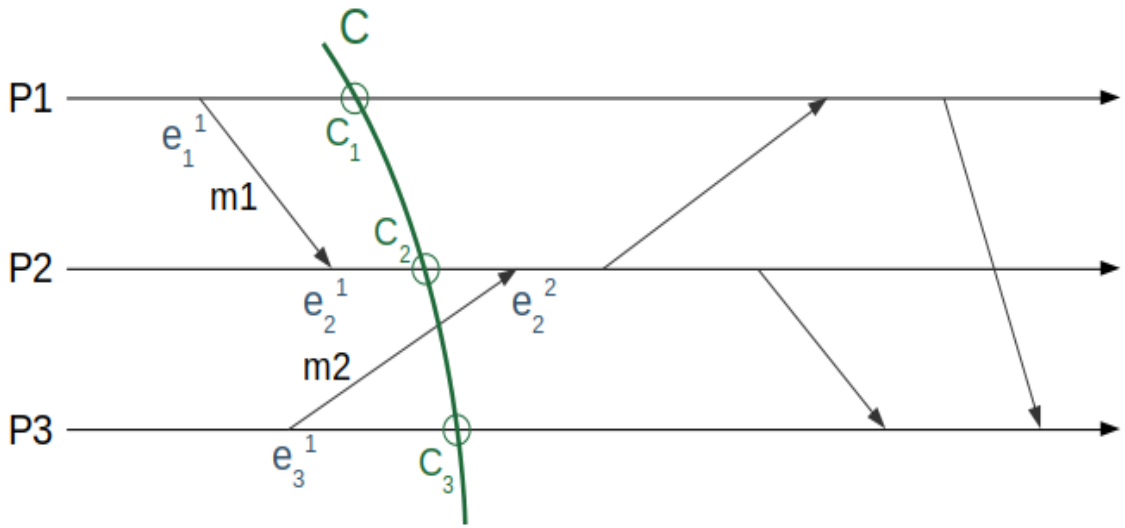
Donner une coupure non cohérente et utiliser les propriétés précédentes pour le justifier.



Définir l'état d'un système réparti

L'état d'un système réparti (suite à une coupure) est composé de :

- l'ensemble des états de chaque site
- l'ensemble des états de chaque canal de communication



Exemple

- État de P_1 : e_1^1, état de P_2 : e_2^1, état de P_3 : e_3^1, état du canal $P_3 \rightarrow P_2$: $m2$

Enregistrer l'état d'un système : Algorithme centralisé de Chandy-Lamport (1985) (1/3)

Hypothèses

- Canaux de communication FIFO.
- Un seul processus initiateur démarre la construction d'une coupure cohérente

Principe

- Idée globale : les processus s'informent mutuellement de la construction d'une coupure en émettant un message "marqueur" vers l'ensemble des voisins. Les "marqueurs" représentent donc une demande de prise de photo et un moyen de "vider" les canaux et d'enregistrer leur état.
- Concrètement : le processus initiateur enregistre sont état local et envoie un message "marqueur" à l'ensemble de ses voisins. La réception d'un message "marqueur" par un processus qui n'a pas encore enregistré son état local provoque cet enregistrement et celui d'éventuels messages arrivants et l'émission d'un message "marqueur" à l'ensemble de ses voisins.

Enregistrer l'état d'un système : Algorithme centralisé de Chandy-Lamport (1985) (2/3)

Algorithme 1 : Algorithme Chandy -Lamport pour un site P_i (partie 1)

Initialisation :

photoLocalePrise : initialisée à *faux* ;

initiateur : initialisée à *faux* sauf pour l'unique processus initiateur;

Procédure construire_coupure() :

si *initiateur* **alors**

    Enregistrer_etat(P_i);

    $photoPrise = vrai$;

    pour $P_k \in Voisins$ **faire**

        début enregistrement des messages émis par P_k ; // $\neq <$ *MARQUEUR* $>$

        envoyer un message $<$ *MARQUEUR* $>$ à P_k;

Enregistrer l'état d'un système : Algorithme centralisé de Chandy-Lamport (1985) (2/3)

Algorithme 2 : Algorithme Chandy -Lamport pour un site P_i (partie 2)

Lors de la réception d'un message $< Marqueur >$ **depuis** P_j **:**

si $\neg photoPrise$ **alors**

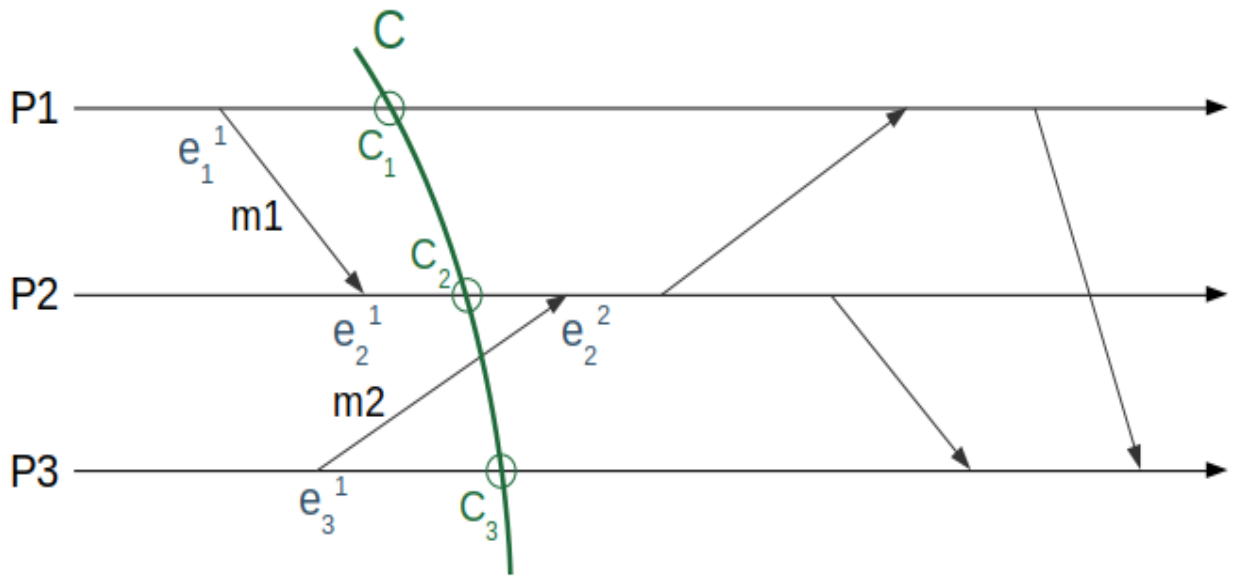
- Enregistrer_etat(P_i);
- $photoPrise = vrai$;
- **pour** $P_k \in Voisins, P_k \neq P_j$ **faire**
  - début enregistrement des messages émis par P_k ; // $\neq < MARQUEUR >$
- **pour** $P_k \in Voisins$ **faire**
  - envoyer un message $< MARQUEUR >$ à P_k;

sinon

- terminer l'enregistrement des messages émis par P_j;

Exercice

- Objectif : Tracer ce qui se produit en appliquant l'algorithme de Chandy-Lamport.
- Qui démarre ? La réponse est dans le figure.



Pour terminer ce chapitre

Causalité et horloges

- La causalité permet de tracer des liens entre évènements pour analyser une application distribuée (pas d'horloge globale, exécutions différentes, etc.)
- Plusieurs types d'horloges (plus précise = plus complexe).

Diffusion répartie

- Plusieurs algorithmes existants (vous avez un aperçu de quelques uns).
- Mise en évidence de quelques propriétés importantes : complexité (en nombre de messages, possibilité de pannes, hypothèses sur les canaux de communication, etc.

Calculer l'état global d'un système réparti

- Un aperçu d'un algorithme centralisé, mais des solutions distribuées existent.
- L'objectif dans ce chapitre étant de vous familiariser avec les fondamentaux.