Le temps en système distribué

Alain Cournier / Stéphane Devismes

Master Informatique

Université de Picardie

Plan du cours

Le temps dans un système distribué

Le Temps dans un système distribué

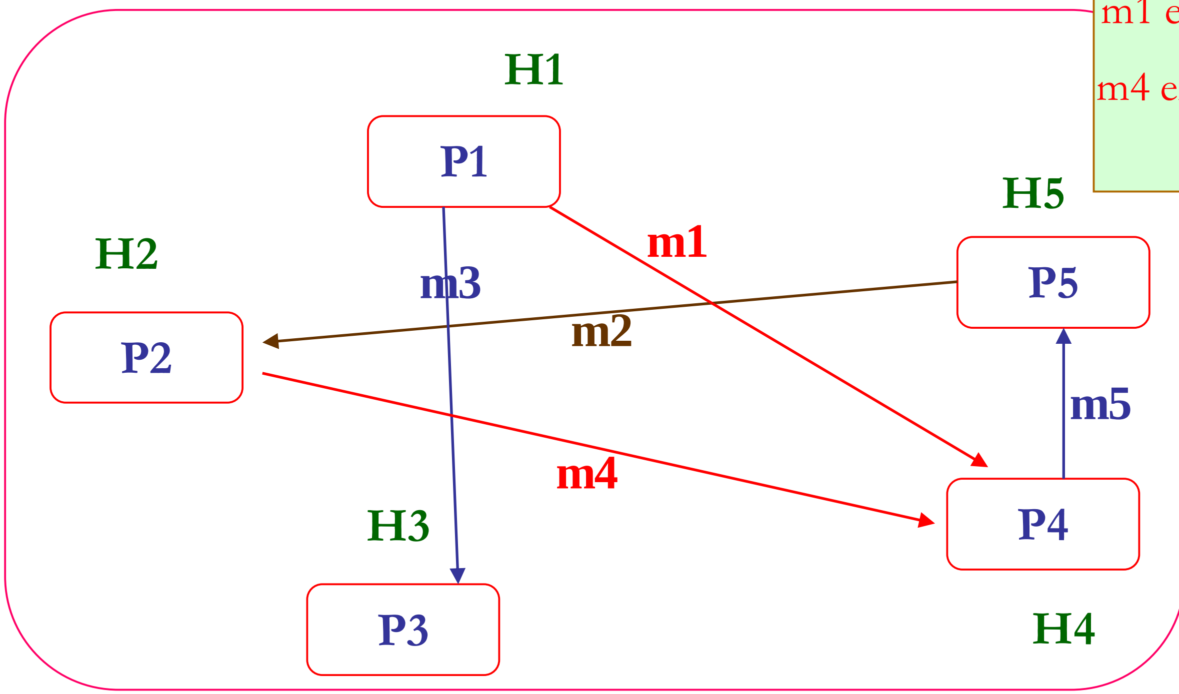
Objectif : définir un **temps global** cohérent et « identique » (ou presque) pour tous les processus.

- Créer **un temps logique.**
  - Temps qui n’est pas lié à un temps physique.

m1 envoyé par P1 vers P4 à l’instant t

m4 envoyé par P2 vers P4 à l’instant t’

Quel est le lien entre t et t’ ?



Temps logique

- **Exemple d'application :** pouvoir préciser l'ordonnancement de l'exécution des processus et de leur communication.
- En fonction des événements locaux des processus, des messages envoyés et reçus, on créé **un ordonnancement logique.**

- **Utiliser une horloge logique.**

Chronogramme

Définition : décrit l'ordonnancement temporel des événements des processus et des échanges de messages.

Chaque processus est représenté par une ligne.

Trois types d'événements signalés sur une ligne :

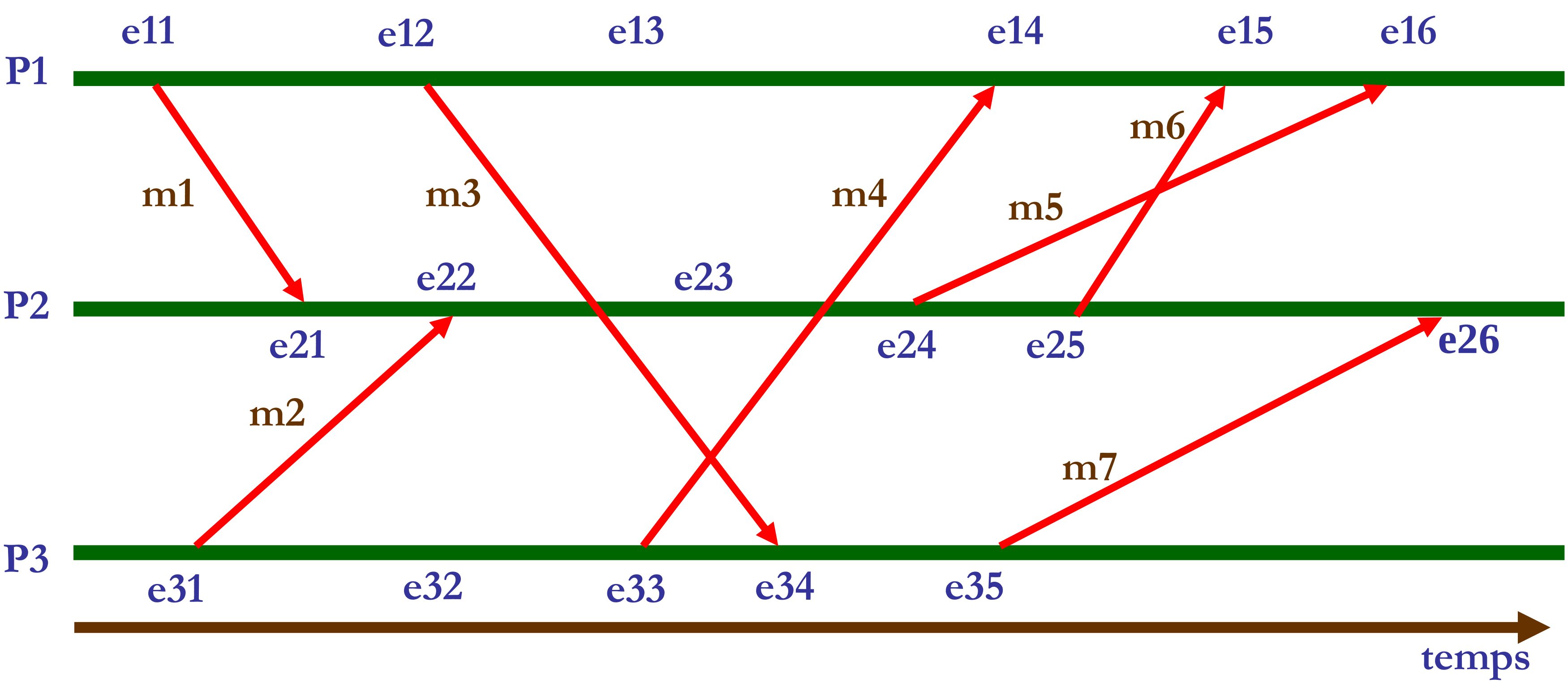
1. **Émission** d'un message à destination d'un autre processus.
2. **Réception** d'un message venant d'un autre processus.
3. **Événement interne** dans l'évolution du processus.

Les messages échangés doivent respecter la topologie de liaison des processus via les canaux.

Chronogramme

Exemple : trois processus tous reliés entre eux par des canaux

Règle de numérotation d'un événement : eXY avec X le numéro du processus et Y le numéro de l'événement pour le processus, dans l'ordre croissant.



Exemples d'événements

Processus P1 :

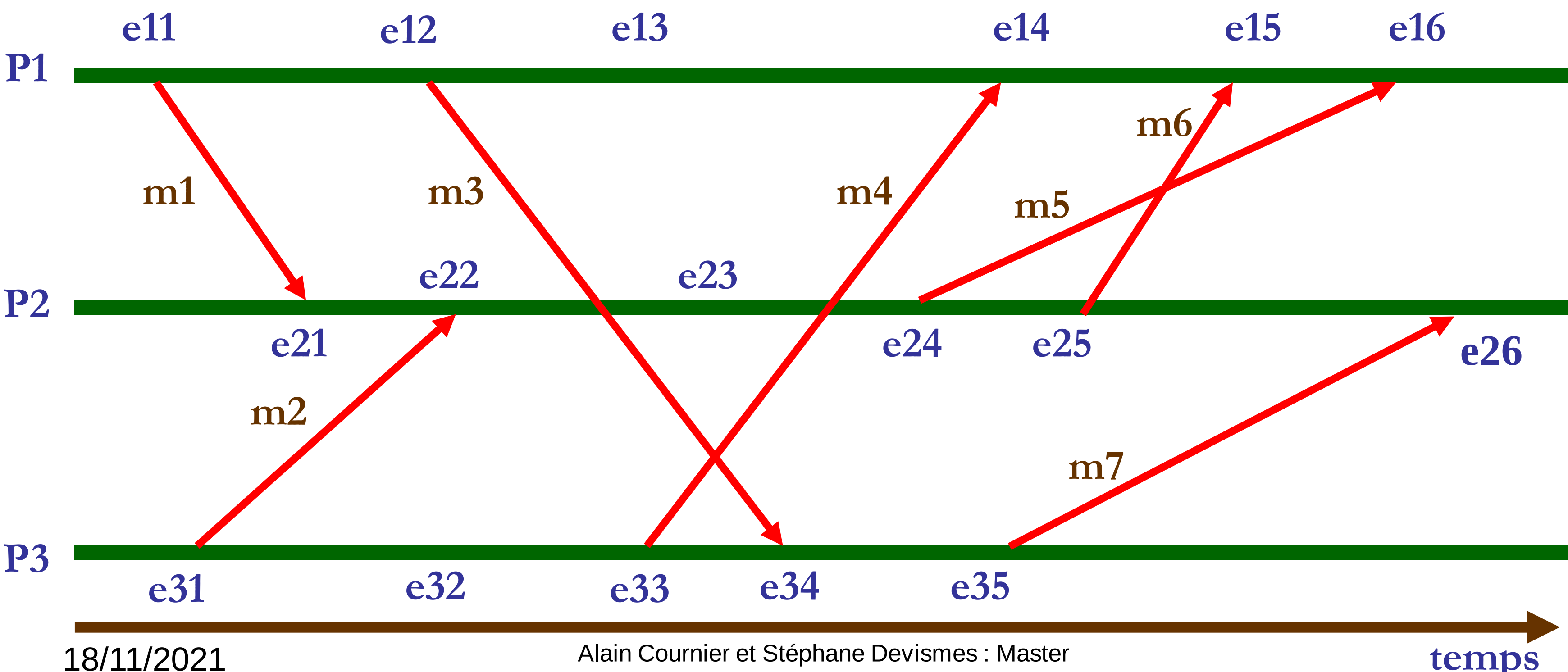
e11 : événement d'émission du message m1 à destination du processus P2.

e13 : événement interne au processus.

e14 : réception du message m4 venant du processus P3.

Processus P2 : message m5 envoyé avant m6 mais m6 reçu avant m5.

Processus P3 : le message m7 est perdu par le canal de communication.



Alain Cournier et Stéphane Devismes : Master Informatique, Université de Picardie

Dépendance causale

Relation de dépendance causale : Il y a une dépendance causale entre 2 événements si un événement **doit se produire avant l'autre.**

Notation : e → e'.

e doit se produire avant *e'*.

Si e → e', alors une des trois conditions suivantes doit être vérifiée pour *e* et *e'*.

1. Si *e* et *e'* sont des événements d'un même processus, *e* précède localement *e'*.
2. Si *e* est l'émission d'un message, *e'* est la réception de ce même message.
3. Il existe une suite finie d'événements $f_1, f_2, \ldots, f_k$ tels que $e \rightarrow f_1$ et $f_k \rightarrow e'$ et pour tout indice i compris entre 1 et k-1 $f_i \rightarrow f_{i+1}$ (fermeture transitive).

Ordonnancement des événements :

Les dépendances causales définissent un **ordre partiel** pour des ensembles d'événements du système.

Dépendance causale

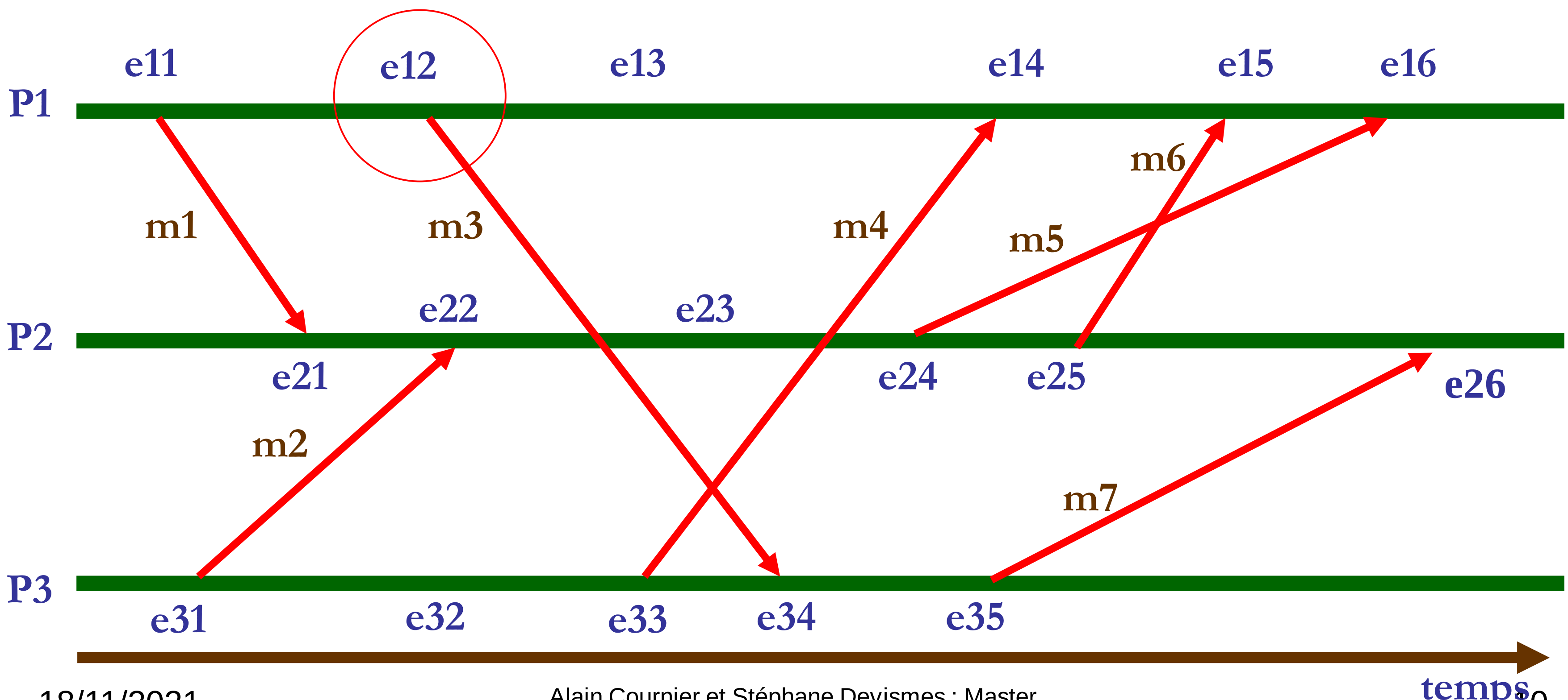
Sur l'exemple précédent, quelques dépendances causales autour de e12

Localement : e11 → e12, e12 → e13.

Sur message : e12 → e34.

Par transitivité : e12 → e26 (car e12 → e34 et e34 → e35 et e35 → e26).

- e11 → e13 (car e11 → e12 et e12 → e13).

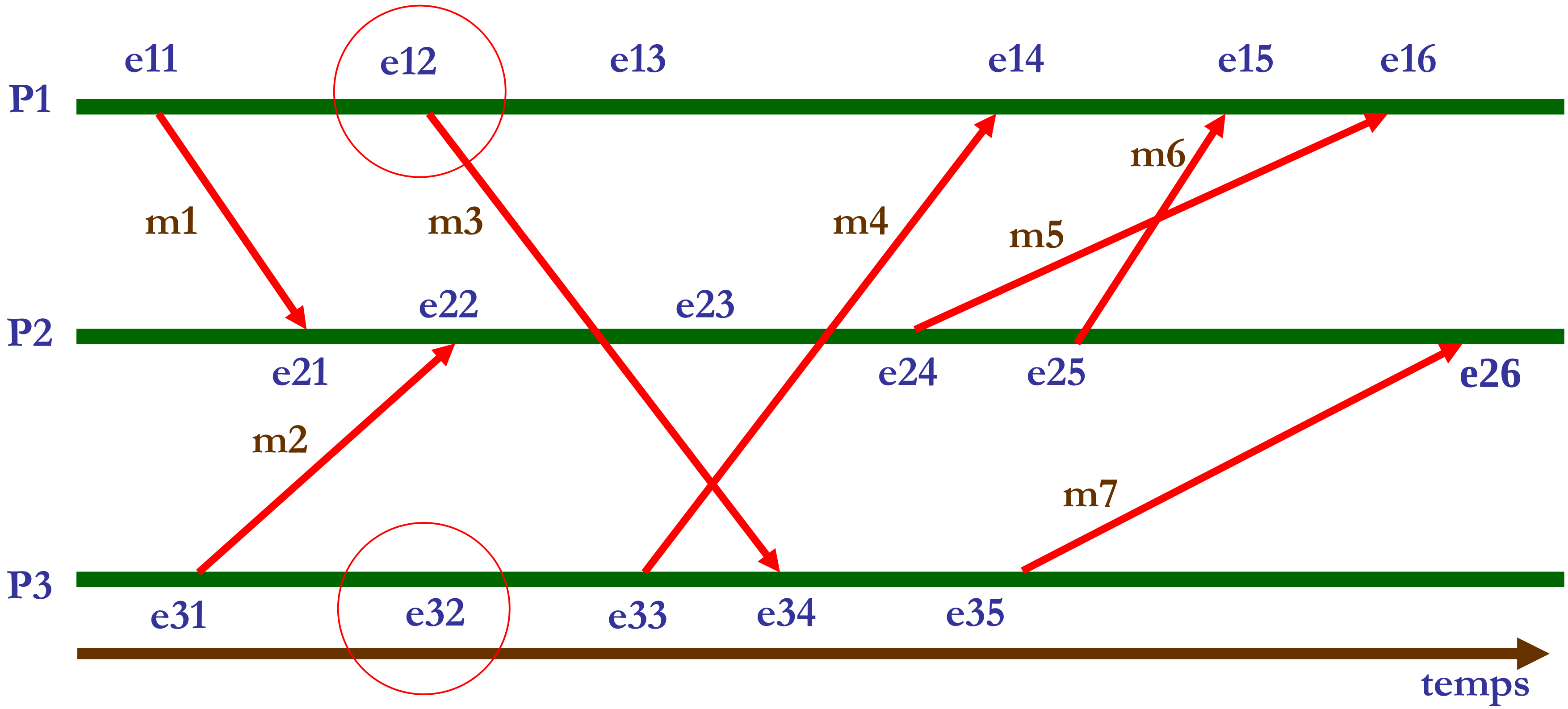


Dépendance causale

Question : dépendance causale entre e12 et e32 ?

A priori non : absence de dépendance causale.

➢ **Des événements non liés causalement se déroulent en parallèle.**



Parallélisme logique

e et e' sont en dépendance causale $\Leftrightarrow$ **$((e \rightarrow e')$ ou $(e' \rightarrow e))$.**

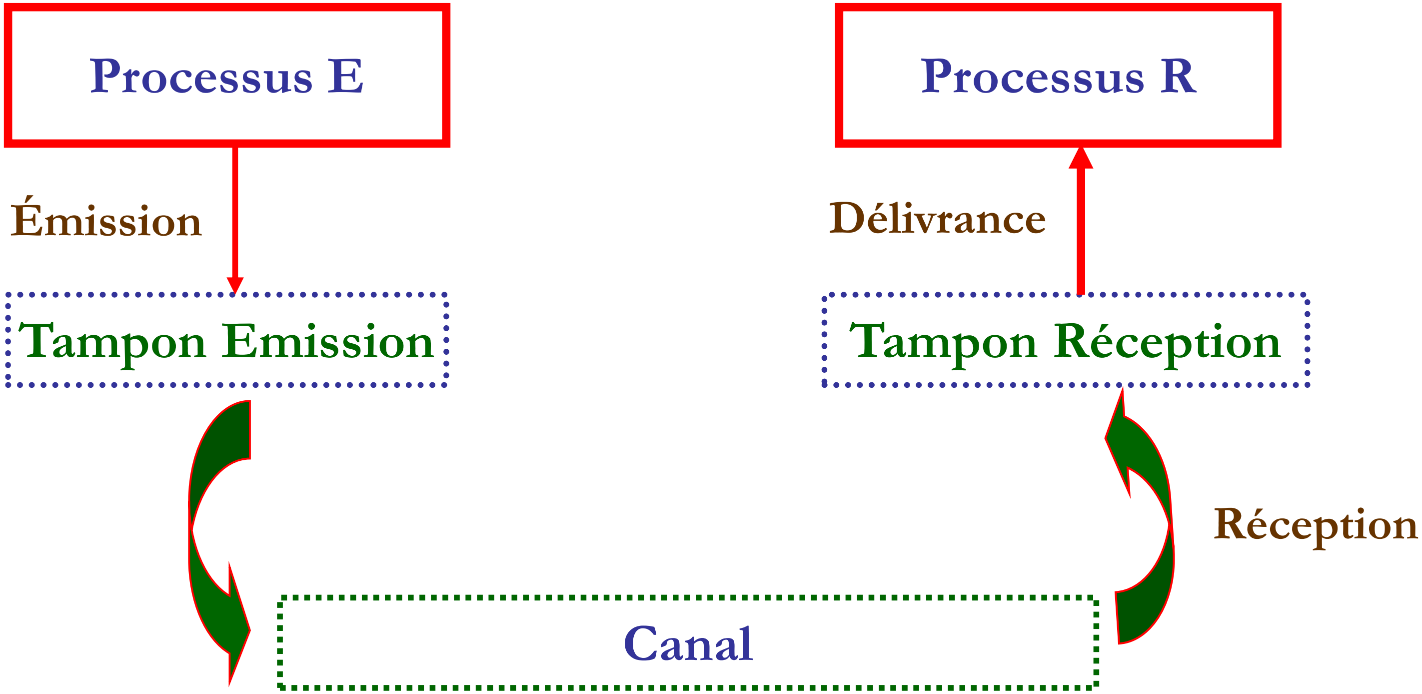
Relation de parallélisme : ||.

$e \mid\mid e' \Leftrightarrow \neg((e \rightarrow e')$ ou $(e' \rightarrow e))$.

Négation (logique) de la dépendance causale.

Parallélisme logique : ne signifie pas que les 2 événements se déroulent simultanément mais **qu'il peuvent se dérouler dans n'importe quel ordre**.

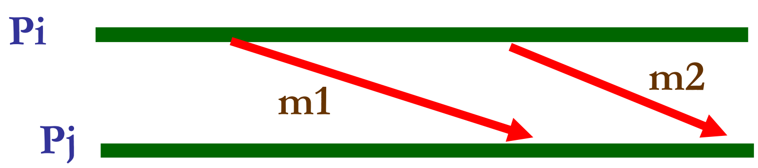
Délivrance



La délivrance d'un message : l'opération consistant à le rendre accessible aux applications clientes (ex: rôle du protocole TCP).

Délivrance FIFO et délivrance causale

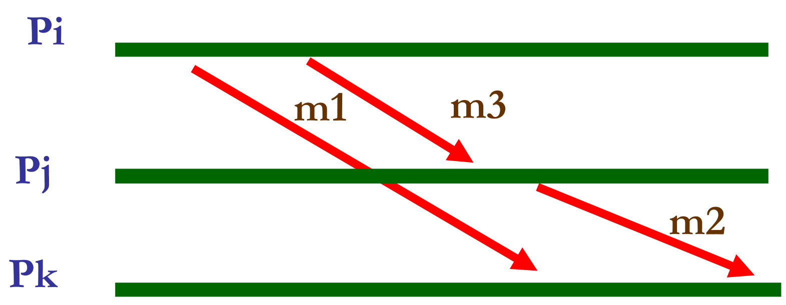
Délivrance FIFO : cette propriété assure que si deux messages sont envoyés successivement de Pi vers un même destinataire Pj, le premier sera délivré à Pj avant le second.



Exemple : communication avec les sockets

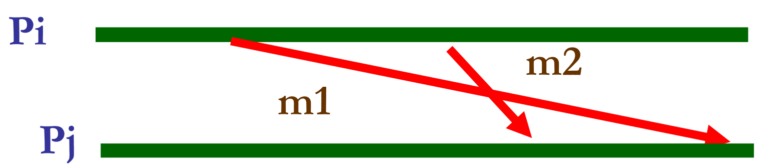
Délivrance causale : cette propriété étend la précédente à des communications à destination <u>d'un même processus</u> en <u>provenance de plusieurs autres</u>.

Elle assure que si l'envoi du message m1 par Pi à destination de Pk précède (causalement) l'envoi du message m2 par Pj à destination de Pk, le message m1 sera délivré avant le message m2 à Pk.

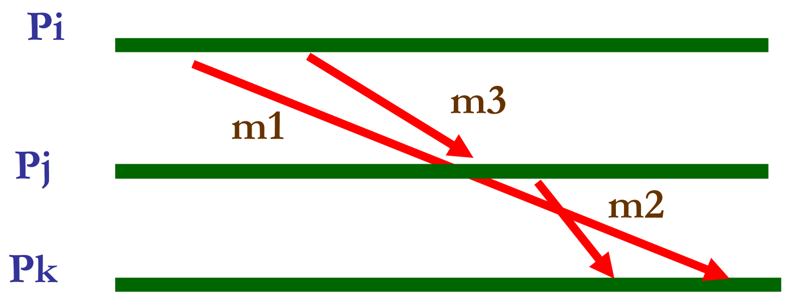


Délivrance FIFO et délivrance causale

Délivrance non FIFO :



Délivrance non causale :



Les horloges logiques

Horloges logiques

Principe : datation de chacun des événements du système avec respect des dépendances causales entre événements.

3 familles d'horloge :

- **Estampille (horloge de Lamport) :** une donnée par événement.
- **Vectorielle (horloge de Mattern) :** un vecteur par événement.
- **Matricielle :** une matrice par événement.

Horloge de Lamport

Horloge de Lamport

Introduit en 1978 par **Leslie Lamport.**

C'est le premier type d'horloge logique introduit en informatique.

Une date (estampille) est associée à chaque événement.

estampille représente un couple (i, nb).

- i : numéro du processus.
- nb : numéro d'événement.

Représente le temps de l'horloge logique.

Horloge de Lamport

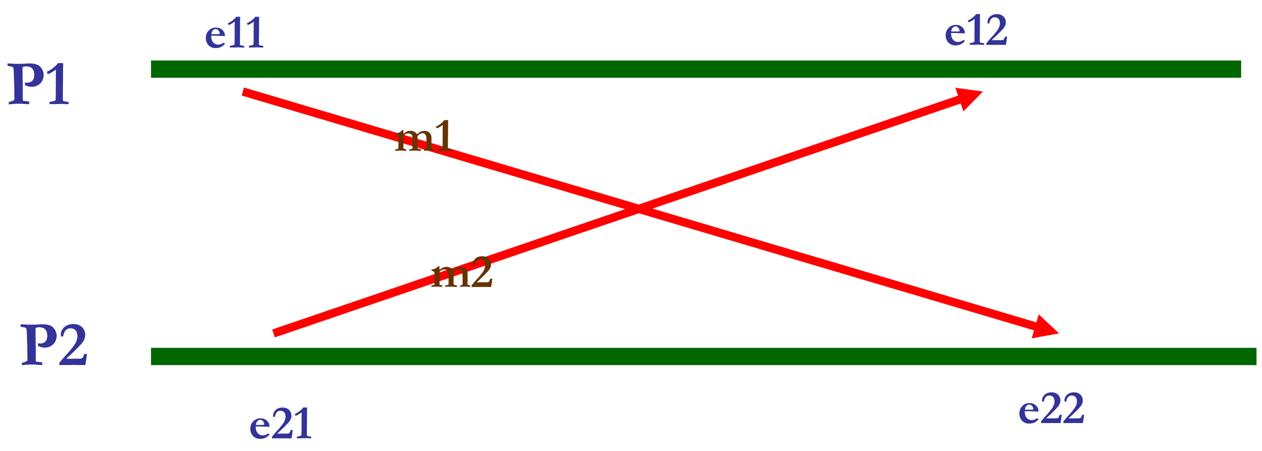
Création du temps logique :

- Localement, chaque processus Pi possède une horloge locale logique Hi, initialisée à 0.
  - Sert à dater les événements.
- **Pour chaque événement local de Pi.**
  - **Hi ← Hi + 1** : on augmente l'horloge locale.
  - L'événement est daté localement par Hi.
- **Émission d'un message par Pi.**
  - **On augmente Hi de 1** puis on envoie le message avec (i, Hi) comme estampille et dater l'événement d'émission par Hi.
- **Réception d'un message *m* par Pi avec estampille (j, nb)**
  - **Hi ← max(Hi, nb) +1** et dater l'événement de réception par Hi.

Temps de l'horloge locale dans Pi

Horloge de Lamport et dépendance causale

Exemple 1 : Echange de messages entre 2 processus.



Les dépendances causales : e11 → e12, e11 → e22, e21 → e22, e21 → e12.

Absence de dépendance causale entre e11 et e21.

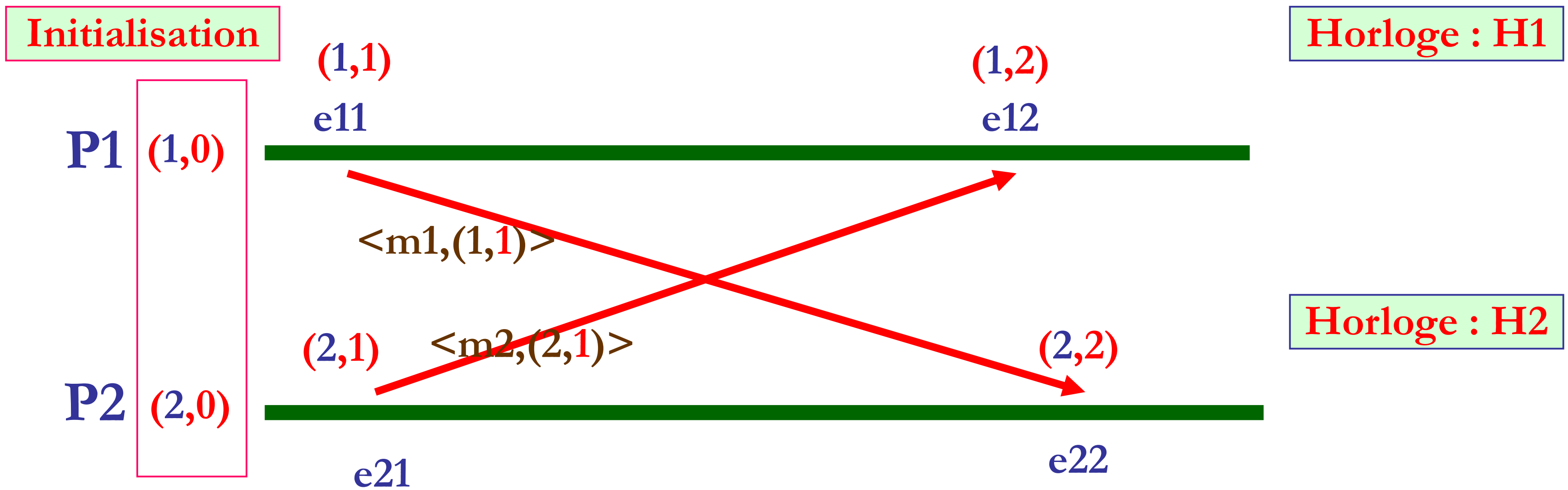
- **Parallélisme logique entre e11 et e21.**

L'horloge de Lamport respecte la dépendance causale :

- **(e → e') ⇒ (H(e) < H(e')).**
- **Exemple : (e11 → e12) ⇒ (H(e11) < H(e12)).**

Horloge de Lamport

Créer un temps logique à l'aide de l'horloge de Lamport.



Relation importante : H(s, nb) < H(s' , nb') **si (nb < nb')** ou **(nb = nb' et s < s')**.

Ordonnancement global (Ordre total) :

(1,1) (1,2)

(2,1) (2,2)

Résultat : e11, e21, e12, e22.

H(e11) < H(e21) < H(e12) < H(e22).

Horloge de Lamport

Ordonnancement global : e11, e21, e12, e22.

$H(e11) < H(e21) < H(e12) < H(e22)$.

L'horloge de Lamport respecte la dépendance causale :

$$(e \rightarrow e') \Rightarrow (H(e) < H(e')).$$

Selon l'horloge : **$H(e11) < H(e21)$.**

Pourtant, il y a une absence de dépendance causale entre e11 et e21.

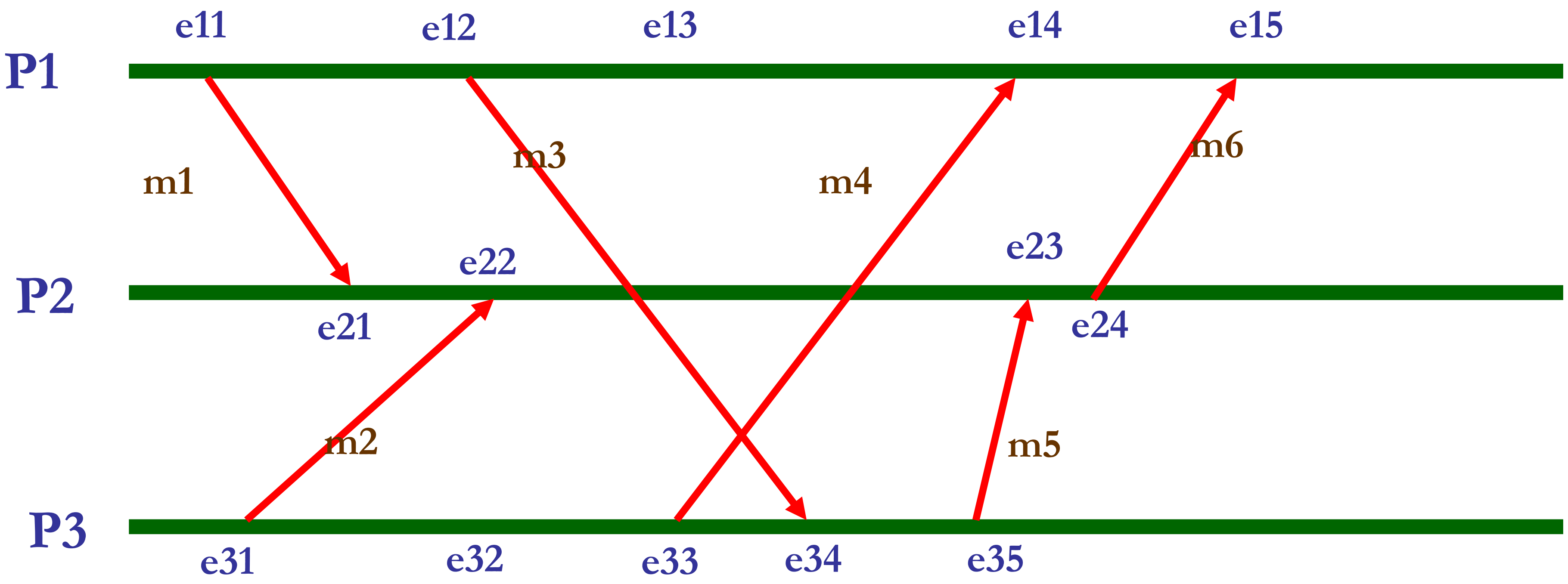
Pour résumé :

L'horloge de Lamport respecte la dépendance causale :

- $(e \rightarrow e') \Rightarrow (H(e) < H(e'))$.
- Mais pas la réciproque :
  - $(H(e) < H(e')) \not\Rightarrow (e \rightarrow e')$.

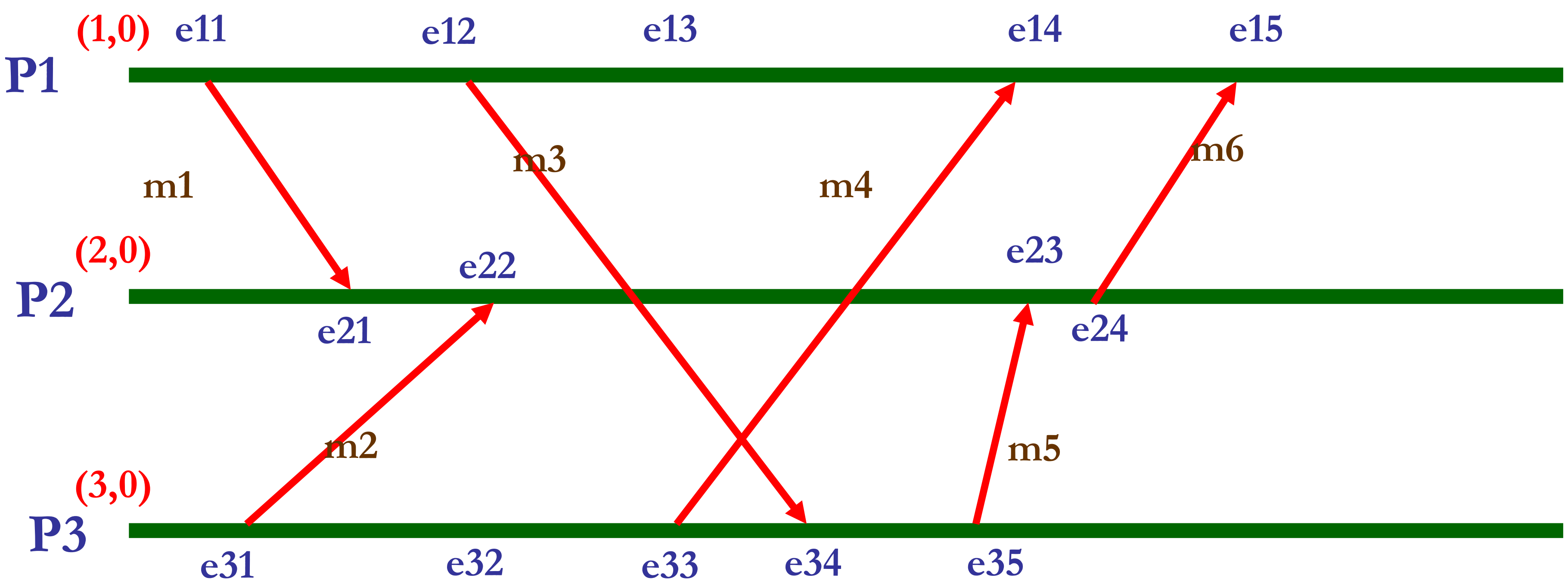
Horloge de Lamport

Exemple 2 : chronogramme avec ajouts des estampilles.



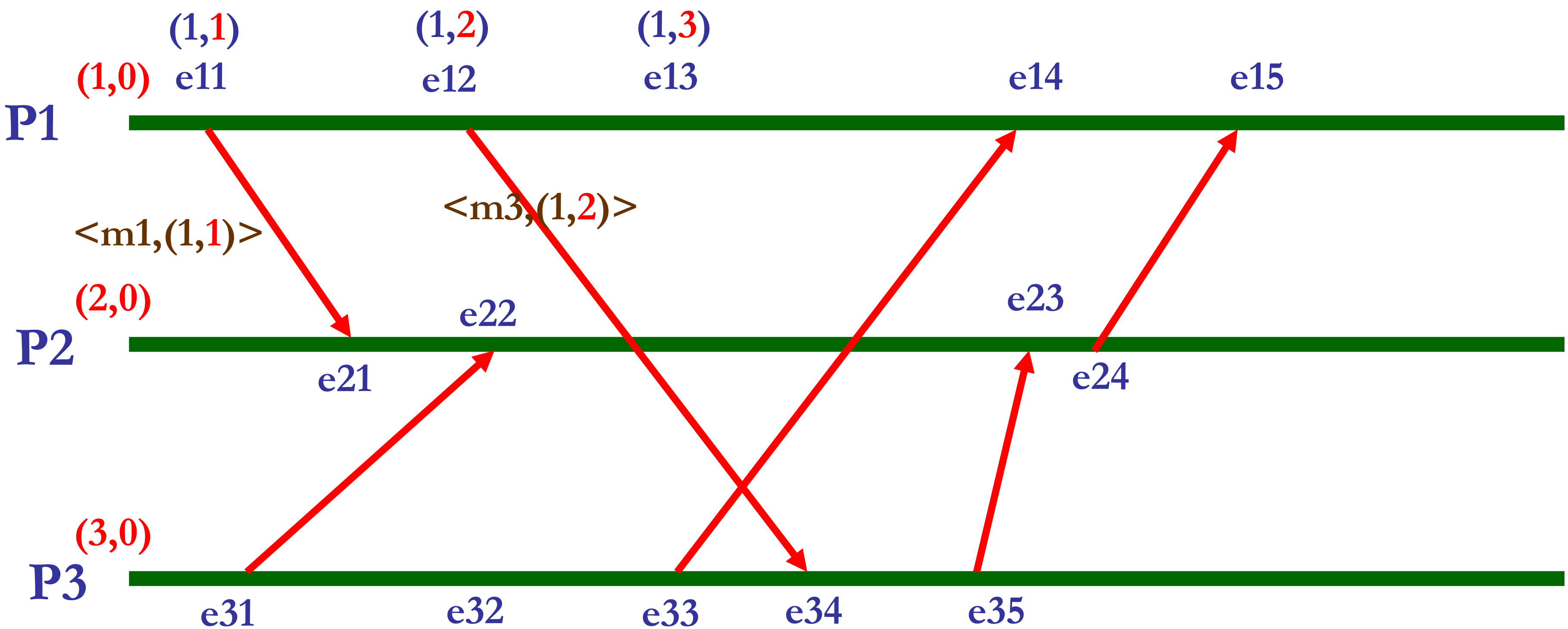
Horloge de Lamport

Exemple 2 : chronogramme avec ajouts des estampilles.



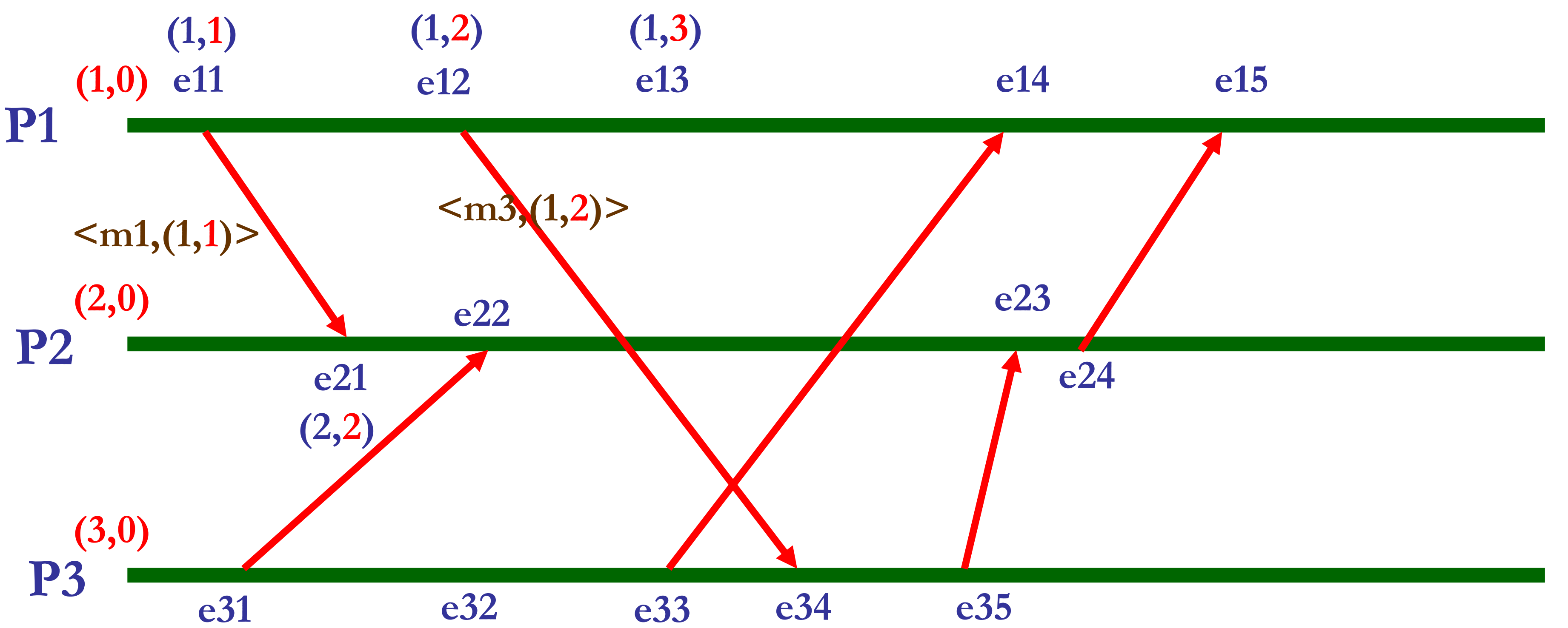
Horloge de Lamport

Exemple 2 : chronogramme avec ajouts des estampilles.



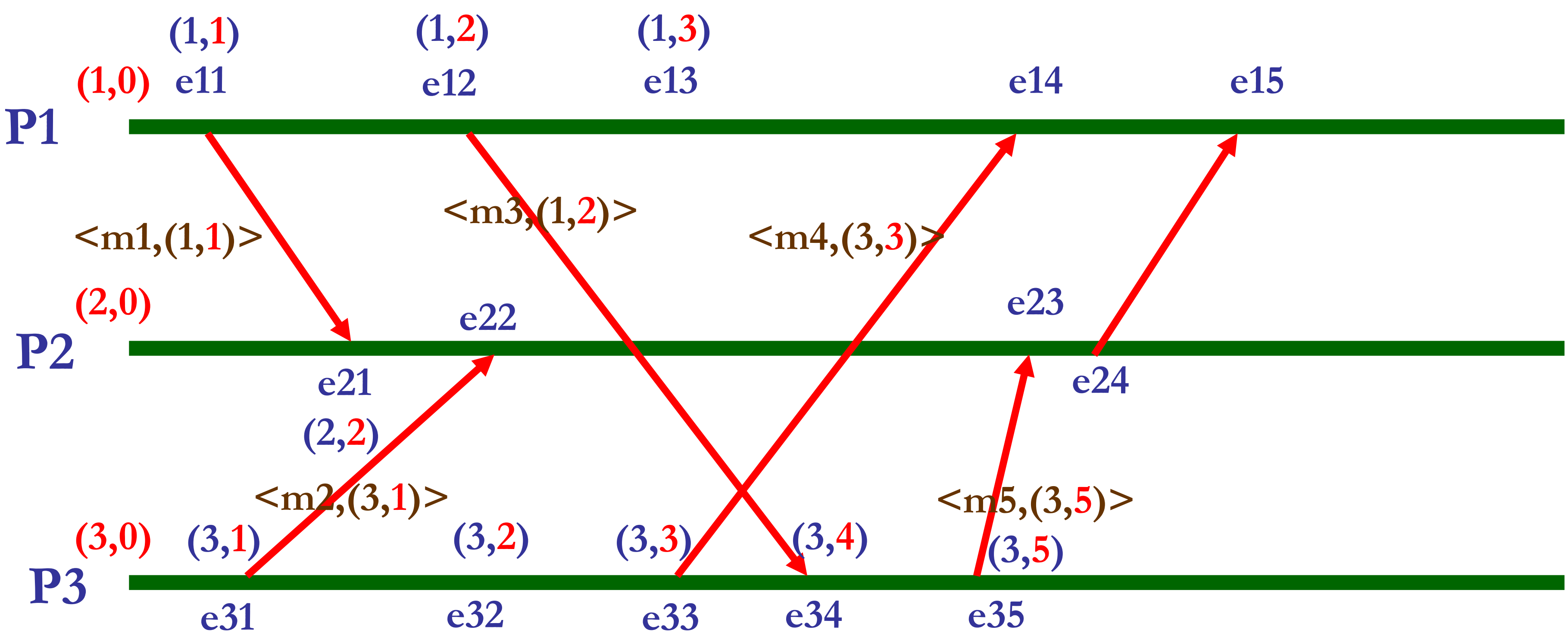
Horloge de Lamport

Exemple 2 : chronogramme avec ajouts des estampilles.



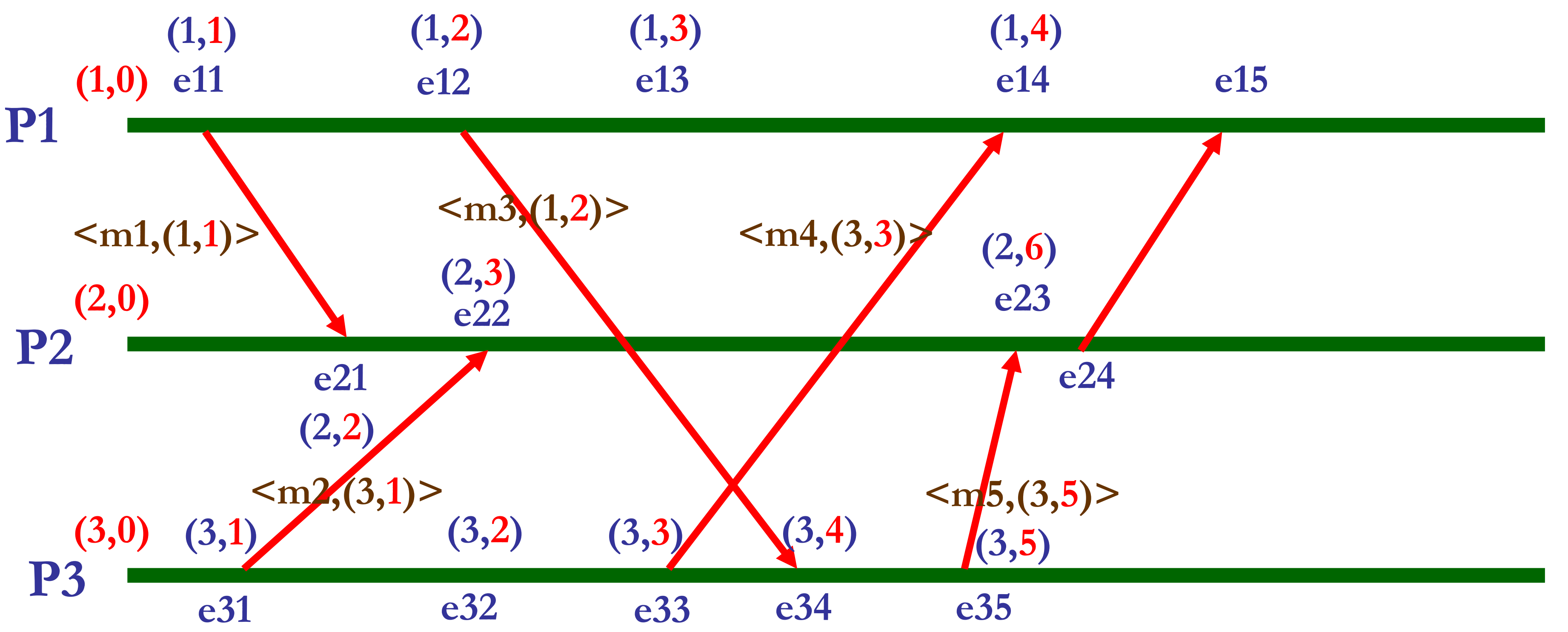
Horloge de Lamport

Exemple 2 : chronogramme avec ajouts des estampilles.



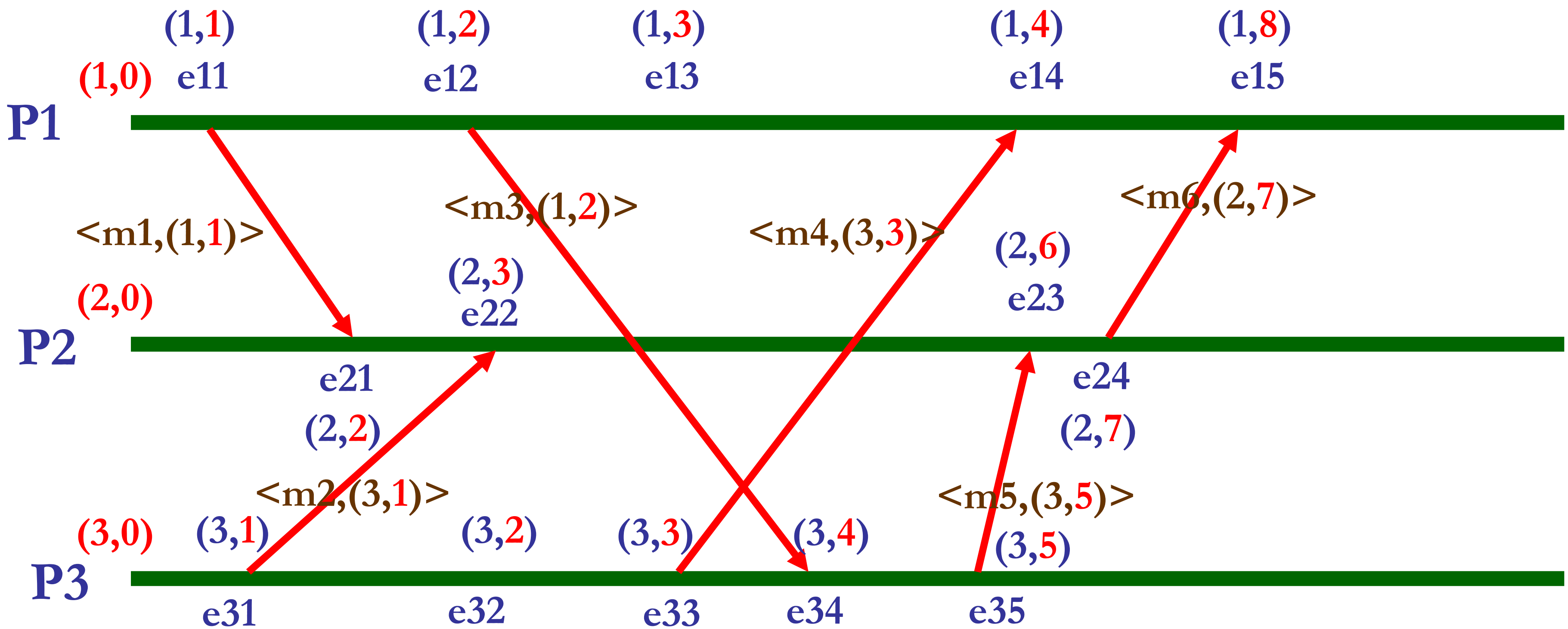
Horloge de Lamport

Exemple 2 : chronogramme avec ajouts des estampilles.



Horloge de Lamport

Exemple 2 : chronogramme avec ajouts des estampilles.



Pour e11, e12, e13 … : augmentation de 1 de l'horloge locale.

Date de e23 : 6 car le message m5 reçu avait une valeur de 5 et l'horloge locale vaut 3 (max (3,5)+1)=6.

Date de e34 : 4 car on incrémente l'horloge locale car sa valeur est supérieure à celle du message m3 (max (3,2)+1)=4.

Horloge de Lamport

Ordonnancement global :

- Via Hi, on ordonne tous les événements du système entre eux.

Ordre total, noté e << e' : *e* s'est déroulé avant *e'*.

Soit *e* événement de *Pi* et *e'* événement de *Pj* :

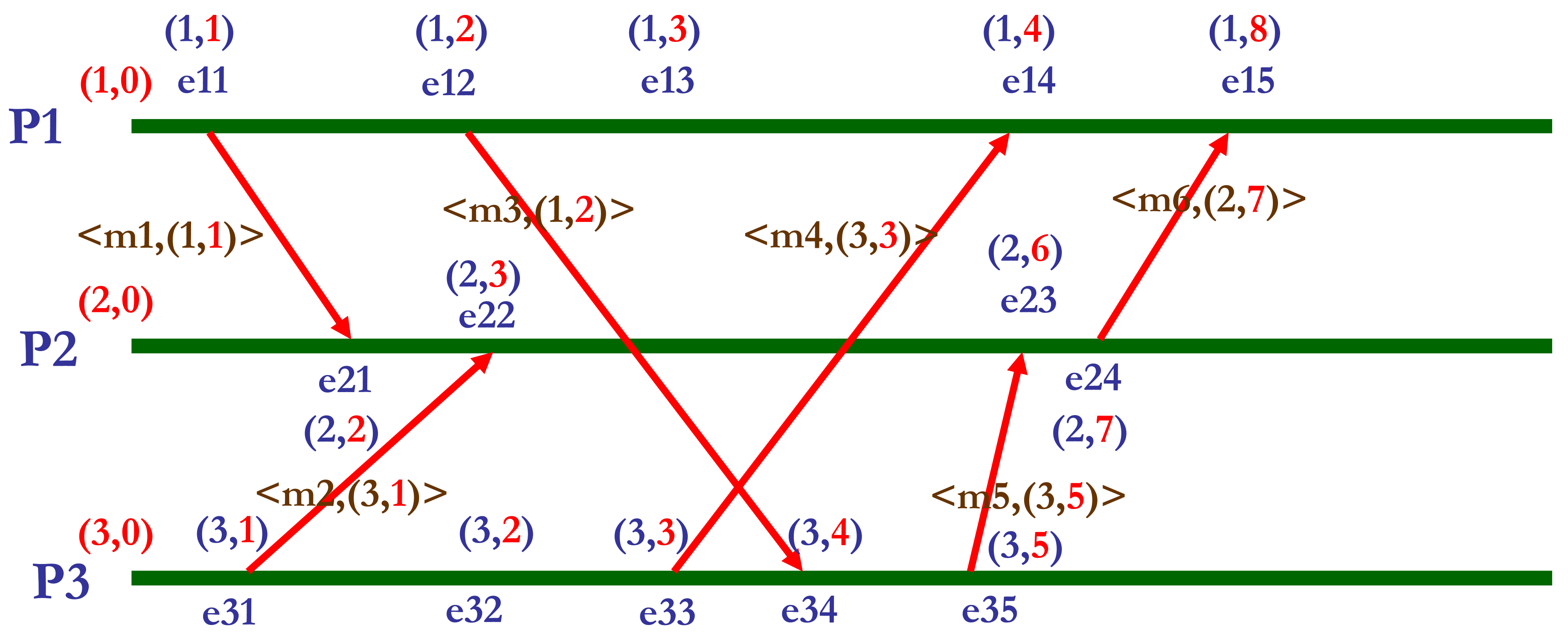
- **e << e' ⇔ (Hi(e) < Hj(e')) ou (Hi(e) = Hj(e') avec i < j).**

Localement (si i = j), Hi donne l'ordre des événements du processus.

Les 2 horloges de 2 processus différents permettent de déterminer l'ordonnancement des événements des 2 processus.

- Si égalité de la valeur de l'horloge, le numéro du processus est utilisé pour les ordonner.

Horloge de Lamport



Ordre total global obtenu pour l'exemple :

(1,1) (1,2) (1,3) (1,4) (1,8)

(2,2) (2,3) (2,6) (2,7)

(3,1) (3,2) (3,3) (3,4) (3,5)

e11 << e31 << e12<< e21 << e32 << e13 << e22 << e33 << e14 << e34 << e35 << e23 << e24 << e15.

Utilité de l'horloge de Lamport

Faire un ordonnancement global des événements dans un système distribué.

Horloge de Mattern

Horloge de Mattern

Introduit indépendamment en 1988 par *Colin Fidge* et *Friedemann Mattern*.

Horloge qui assure la réciproque de la dépendance causale :

- **H(e) < H(e') ⇒ (e → e').**

Permet également de savoir si 2 événements sont parallèles (non dépendants causalement).

Ne définit par contre pas un ordre total global (C’est normal).

Principe :

Utilisation de vecteur V de taille égale au nombre de processus.

- Localement, chaque processus *Pi* a un vecteur *Vi*.
- Un message est envoyé avec un vecteur de date.

Pour chaque processus *Pi*, chaque case *Vi[j]* du vecteur contiendra des valeurs de l’horloge du processus *Pj*.

Horloge de Mattern

Fonctionnement de l'horloge :

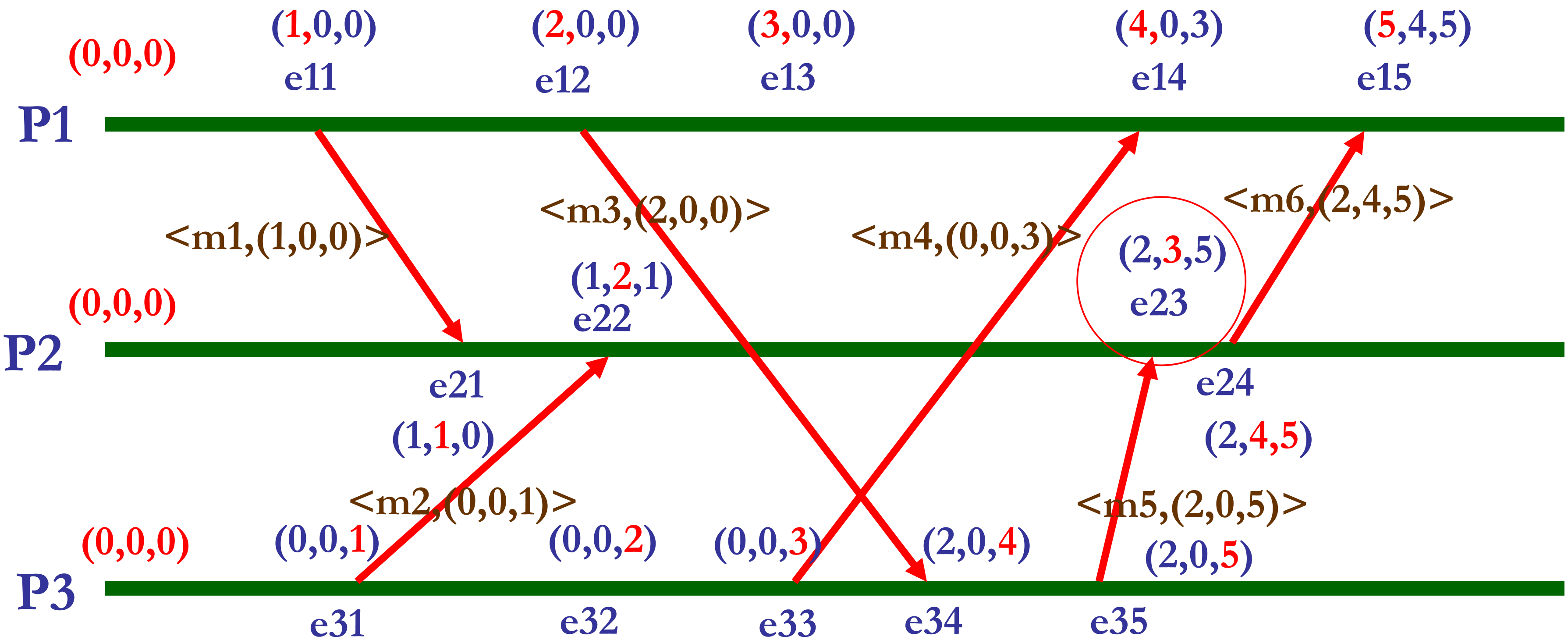
- Initialisation : pour chaque processus *Pi*, **Vi =(0, … , 0).**
- Pour un processus *Pi*, à chacun de ses événements (local, émission, réception) **: Vi [i] = Vi [i] + 1.**
  - Incrémentation du compteur local d'événement (garder les autres compteurs si événement local ou émission).
  - Si émission d'un message, alors *Vi* est envoyé avec le message.
- Pour un processus *Pi*, à la réception d'un message *m* contenant un vecteur *Vm* provenant du processus *Pj*, on met à jour les cases k ≠ i de son vecteur local *Vi*.
  - **∀ k ≠ i : Vi [k] = max (Vm [k], Vi [k]).**

    Mémorise le nombre d'événements sur Pj qui doivent se dérouler avant la réception du message sur Pi.

    La réception de ce message sur Pi dépend causalement de tous ces événements sur Pj.

Horloge de Mattern

Même exemple que pour horloge de Lamport :



V(e23) = (2,3,5) : (2 relatif à P1, 3 à P2, 5 à P3).

3 = 2+1 (Incrémentation du compteur local). <u>3ème</u> évent dans P2.

2 =max (2,0). 2 évents dans P1 (e11 et e12) qui sont en dépendance causale par rapport à l'évent e23.

5 =max (5,3). 5 évents dans P3 (e31, e32, e33, e34 et e35) qui sont en dépendance causale par rapport à l'évent e23.

Horloge de Mattern

Relation d'ordre partiel sur les dates (<) :

- $V \leq V'$ défini par $\forall$ i : $V[\,i\,] \leq V'[\,i\,]$.
- **$V < V'$ défini par $V \leq V'$ et $\exists j$ tel que $V\,[\,j\,] < V'\,[\,j\,]$.**
- $V \,||\, V'$ défini par $\neg\,((\,V < V'\,)$ ou $(\,V' < V\,))$.

Dépendance et indépendance causales :

Horloge de Mattern assure les propriétés suivantes, avec e et e' deux événements et $V(e)$ et $V(e')$ leurs datations.

- **$V(e) < V(e') \Rightarrow e \rightarrow e'$.**
  - Si deux dates sont ordonnées, on a forcément dépendance causale entre les événements datés.
- **$V(e) \,||\, V(e') \Rightarrow e \,||\, e'$.**
  - Si il n'y a aucun ordre entre les 2 dates, les 2 événements sont indépendants causalement (les 2 événements sont en parallèle).

Horloge de Mattern

Retour sur l'exemple :

V(e13) = (3,0,0), V(e14) = (4,0,3), V(e15) = (5,4,5).

V(e13) < V(e14) donc e13 $\rightarrow$ e14.

V(e14) < V(e15) donc e13 $\rightarrow$ e15.

V(e35) = (2,0,5) et V(e23) = (2,3,5).

V(e35) < v(e23) donc e35 $\rightarrow$ e23.

L'horloge de Mattern respecte les dépendances causales des événements ainsi que la réciproque.

Horloge de Lamport respecte uniquement les dépendances causales.

V(e32) = (0,0,2) et V(e13) = (3, 0, 0).

On a ni V(e32) < V(e13) ni V(e13) < V(e32) donc e32 || e13.

L'horloge de Mattern respecte les indépendances causales.

État Global

État global : état du système à un instant donné.

- Buts de la recherche d'états globaux :
  - Trouver des états cohérents à partir desquels on peut reprendre un calcul distribué en cas de plantage du système.
- **Défini à partir de coupures.**

Coupure : photographie à un instant donné de l'état du système.

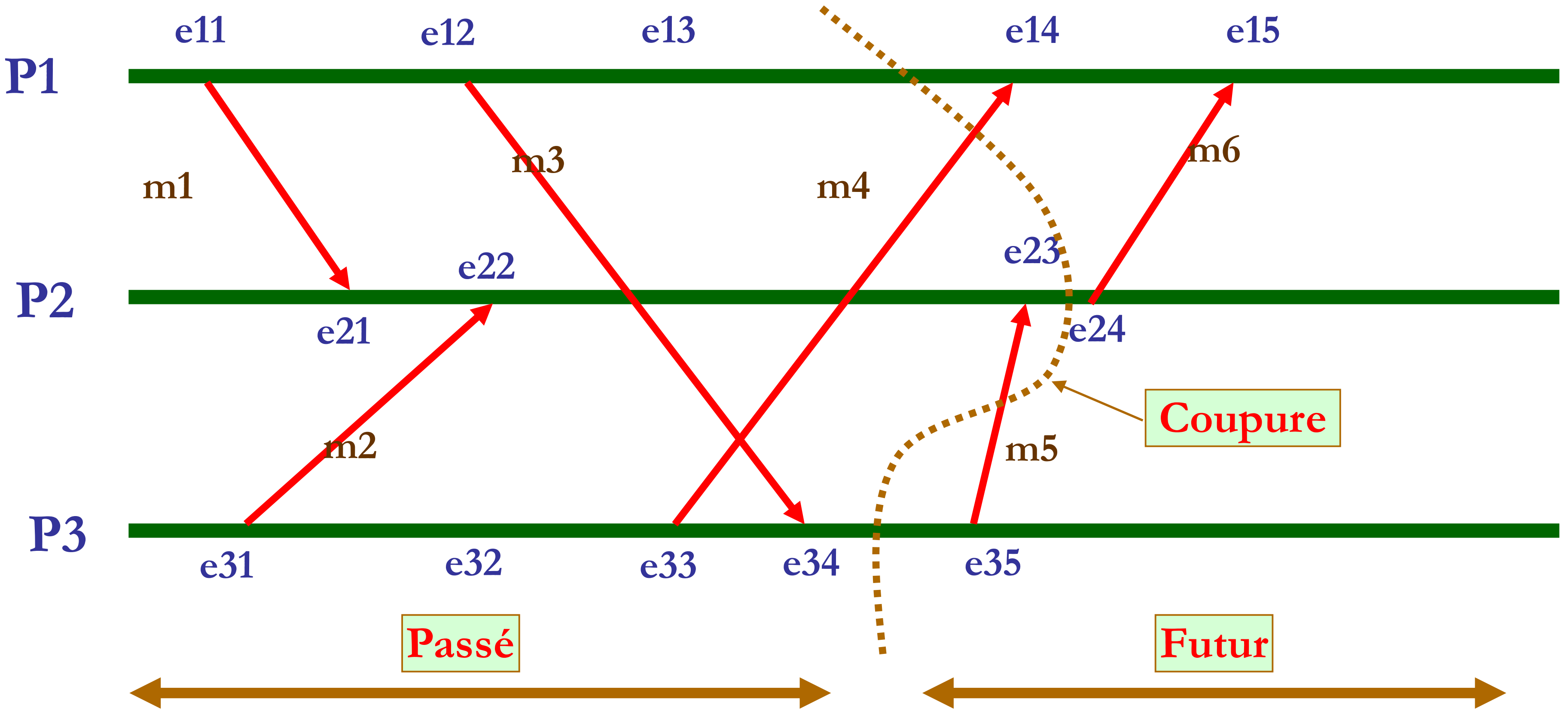
- Définit les événements appartenant au passé et au futur par rapport à l'instant de la coupure.

Coupure

Définition :

- Calcul distribué = ensemble E d'événements.
- Coupure C est un sous-ensemble fini de E tel que :
  - Soit a et b deux événements du même processus :
  - $((a \in C)$ et $(b \rightarrow a)) \Rightarrow (b \in C)$.
  - Si un événement d'un processus appartient à la coupure, alors tous les événements locaux le précédant y appartiennent également.

Exemple de coupure



Coupure = ensemble {e11, e12, e13, e21, e22, e23, e31, e32, e33, e34}.

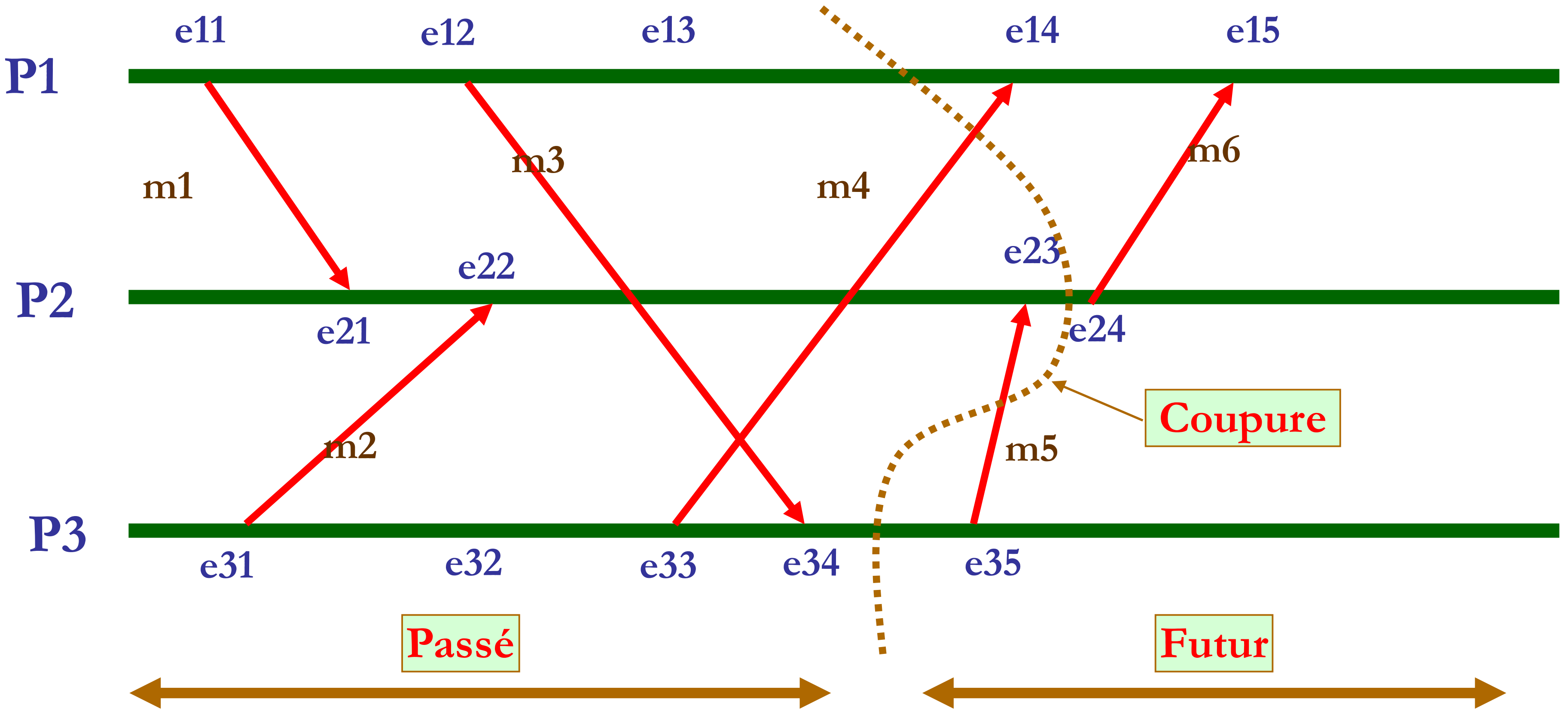
Etat associé à une coupure

Si le système est composé de N processus, l'état associé à une coupure est défini au niveau d'un ensemble de N événements **(e1, e2, ... ei, ... eN),** avec ei événement du processus Pi tel que :

- ***∀ i , ∀ e ∈ C :***
  - ***(e* événement du processus Pi) ⇒ (e → ei).**

L'état est défini à la frontière de la coupure : l'événement le plus récent pour chaque processus.

Exemple de coupure



État de la coupure = (e13, e23, e34).

Coupure cohérente

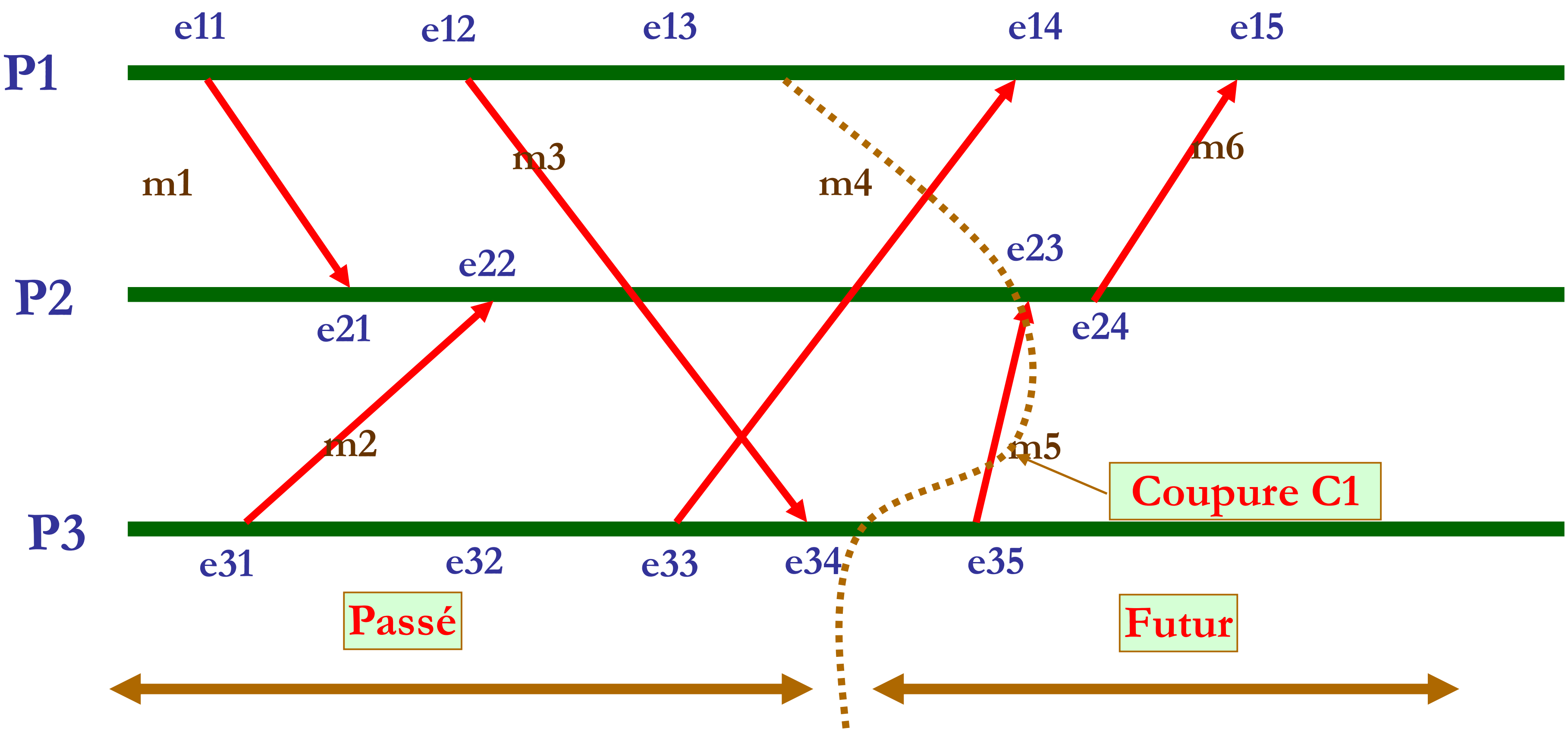
Coupure cohérente : coupure qui respecte les dépendances causales des événements du système et pas seulement les dépendances causales locales à chaque processus.

- Soit *a* et *b* deux événements du système :
  - $((a \in C) \text{ et } (b \rightarrow a)) \Rightarrow (b \in C)$.
- **Coupure cohérente :** aucun message ne vient du futur.

État cohérent : État associé à une coupure cohérente.

- Permet par exemple une reprise sur faute.

Exemple de coupure non cohérente



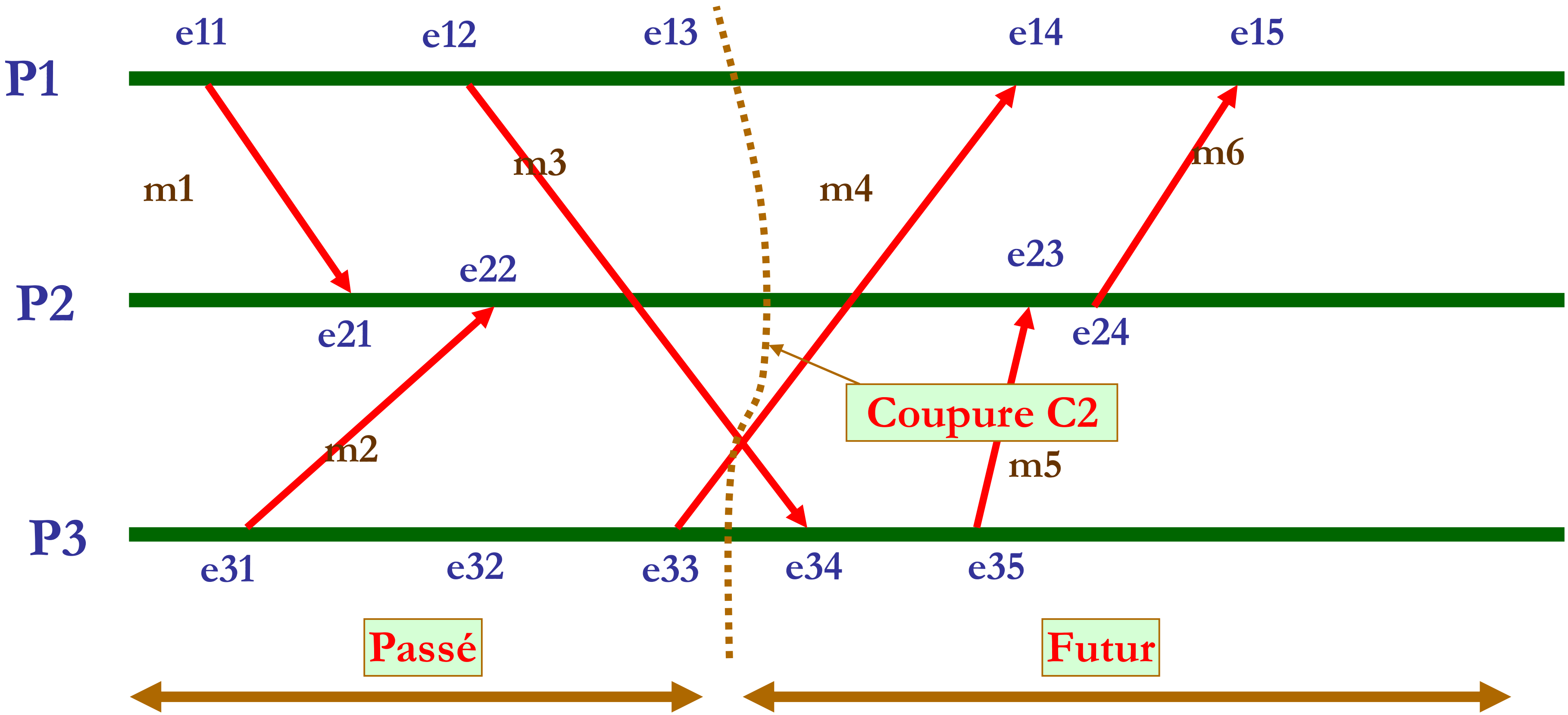
Coupure C1 : non cohérente car : e23 ∈ C1 et e35 → e23 mais e35 ∉ C1.

La réception de *m5* est dans la coupure mais pas son émission.

m5 vient du futur par rapport à la coupure.

Exemple de coupure cohérente

Coupure C2 : cohérente.



Datation coupure

Horloge de Mattern permet de dater la coupure.

Soit N processus, C la coupure, ei l'événement le plus récent pour le processus Pi, V(ei) la datation de ei et V(C) la datation de la coupure.

- V(C) = max (V(e1), ... , V(eN)) :
  - ∀ i : V(C)[i] = max (V(e1)[i] , ... , V(eN)[i]).
- Pour chaque valeur du vecteur, on prend le maximum des valeurs de tous les vecteurs des N événements pour le même indice.

Permet également de déterminer si la coupure est cohérente.

- Cohérent si **V(C) = (V(e1)[1], ... , V(ei)[i], ... , V(eN) [N]).**

Datation coupure

Datation des coupures de l'exemple :

Coupure C2 : état = (e13, e22, e33).

- V(e13) = (3, 0, 0).
- V(e22) = (1, 2, 1).
- V(e33) = (0, 0, 3).
- V(C) = (max(**3**,1,0), max(0,**2**,0), max(0,1,**3**)) = (3,2,3).
- Coupure cohérente car V(C)[1] = V(e13)[1], V(C)[2] = V(e22)[2], V(C)[3] = V(e33)[3].

Coupure C1 : état = (e13, e23, e34).

- V(e13) = (3,0,0), V(e23) = (2,3,5), V(e34) = (2,0,4).
- V(C) = (max(**3**,2,2), max(0,**3**,0), max (0,**5**,4)).
- Non cohérent car V(C)[3] $\neq$ V(e34)[3].
- D'après la date de e23, e23 doit se dérouler après 5 événements de P3 or e34 n'est que le quatrième événement de P3. Un événement de P3 dont e23 dépend causalement n'est donc pas dans la coupure (il s'agit de e35 se déroulant dans le futur).

Utilité de l'horloge de Mattern

Validation de la cohérence d'un état global.

Savoir si 2 événements sont en parallèle ou en dépendance causale.

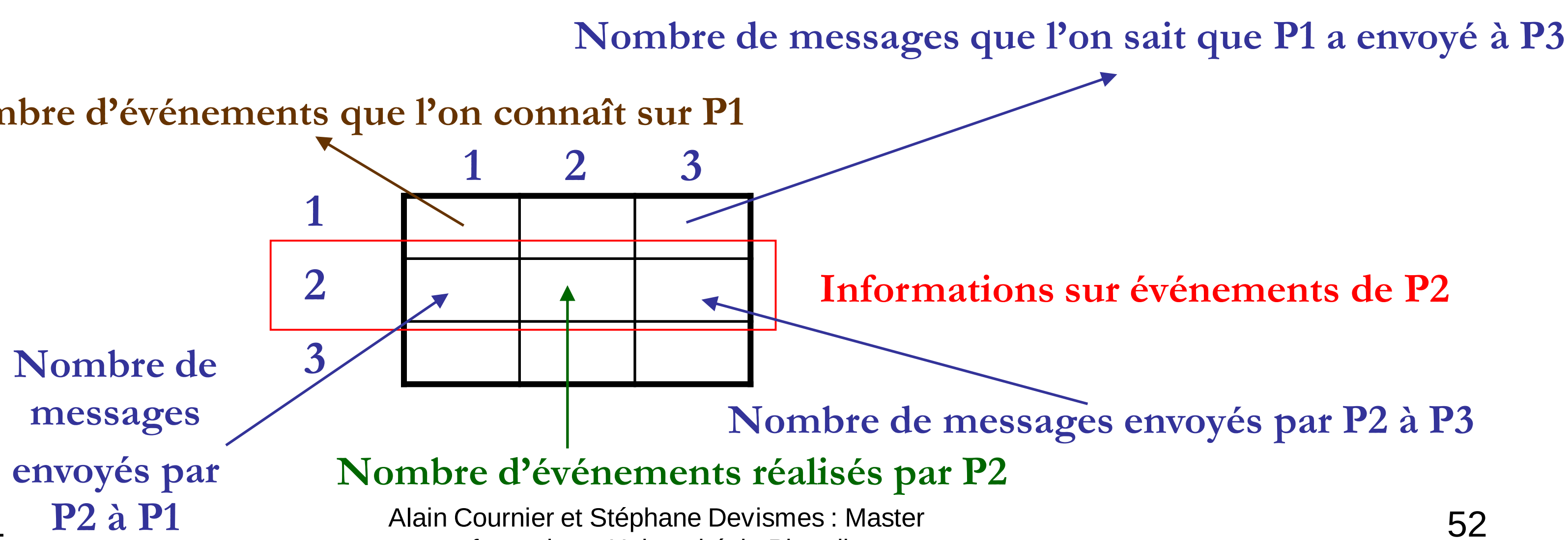
Horloge matricielle

Horloge matricielle

n processus : matrice ***M de (n x n)*** pour dater chaque événement.

Sur processus Pi :

- <u>Ligne *i* : informations sur événements de Pi :</u>
  - Mi [i , i] : nombre d'événements réalisés par Pi.
  - Mi [i , j] : nombre de messages envoyés par Pi à Pj (avec j ≠ i).
- <u>Ligne *j* (avec j ≠ i) :</u>
  - Mi [j , j] : nombre d'événements que l'on connaît sur Pj.
  - Mi [j , k] : nombre de messages que l'on sait que Pj a envoyé à Pk (avec j ≠ k).
- **Avec 3 processus :**



Exemple d'application

Considérons un système contenant 3 processus. Tous les processus possèdent des horloges logiques matricielles.

Supposons que l'horloge matricielle HM3 du processus 3 est HM3 =

6	2	2
1	5	1
1	2	7

A quoi correspond l'élément de l'horloge matricielle HM3 [3, 1], pour le processus 3 ?

Même question pour les éléments HM3 [1, 3], HM3 [2, 3] pour le processus 3.

Le processus 3 reçoit le message m en provenance du processus 1. L'estampille du message m est EMm =

8	2	3
2	9	2
1	1	3

Que peut déduire le processus 3 par rapport aux éléments EMm [1, 3], EMm [2, 3] ?

Le processus 3 peut-il délivré le message m (délivrance causale) ? Justifier votre réponse.

Utilité de l'horloge matricielle

Assurer la délivrance causale de messages entre plusieurs processus.