

# Systemes distribués

## Horloges logiques

M1 RID

# Plan

## ❑ **Partie 1 : Temps dans un système distribué**

- ❑ Temps logique
- ❑ Chronogramme
- ❑ Dépendance causale
- ❑ Parallélisme logique
- ❑ Délivrance

## ❑ **Partie 2 : Horloges logiques**

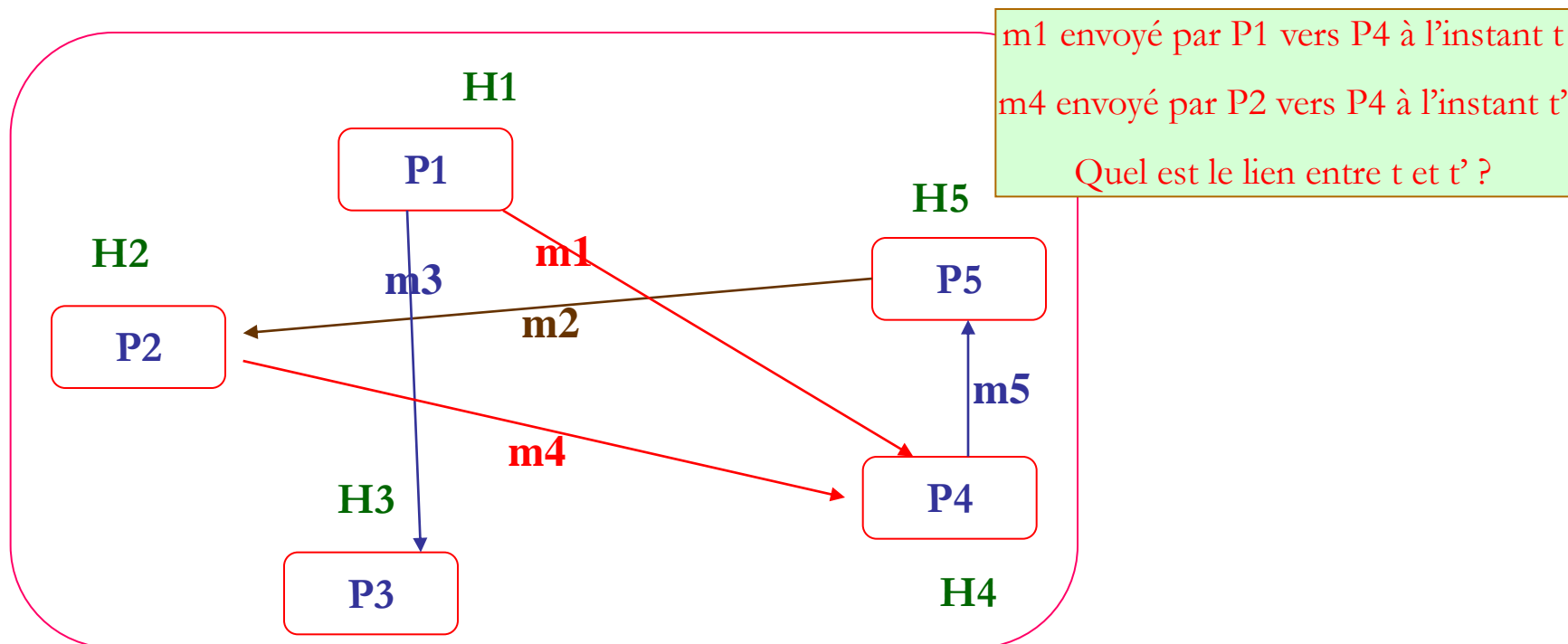
- ❑ Estampille (horloge de Lamport)
- ❑ Vectorielle (horloge de Mattern)
- ❑ Matricielle

# **Partie 1 :**

## **Temps dans un système distribué**

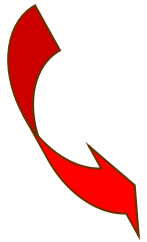
# Temps dans un système distribué

- ❑ **Objectif** : définir un **temps global** cohérent et « identique » (ou presque) pour tous les processus.
- ❑ Soit synchroniser au mieux les horloges physiques locales entre elles.
- ❑ Soit créer **un temps logique**.



# Temps logique

- ❑ Temps qui n'est pas lié à un temps physique.
- ❑ **But** : 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 crée un ordonnancement logique.



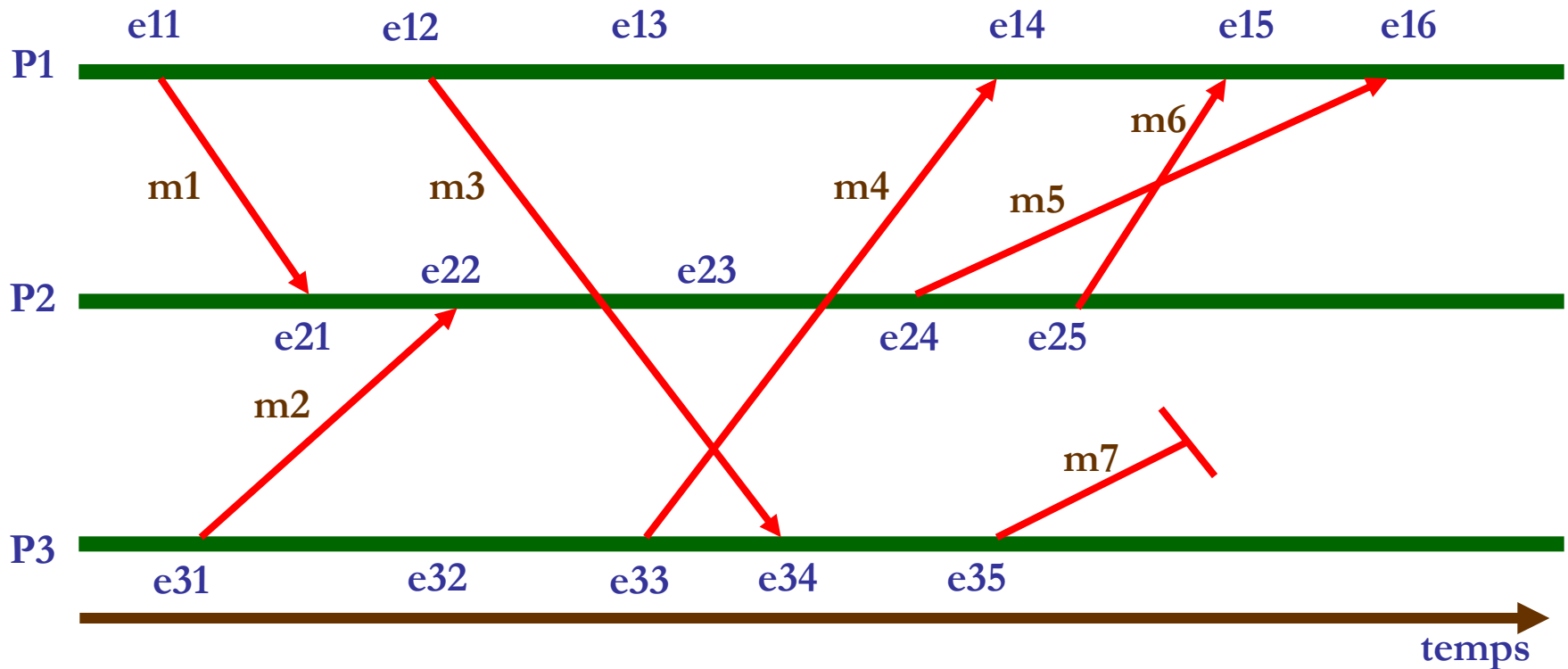
❑ **Création d'horloge logique.**

# 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 :
  - ❑ **Émission** d'un message à destination d'un autre processus.
  - ❑ **Réception** d'un message venant d'un autre processus.
  - ❑ **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 avec la possibilité de perte de message.
- ❑ **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.



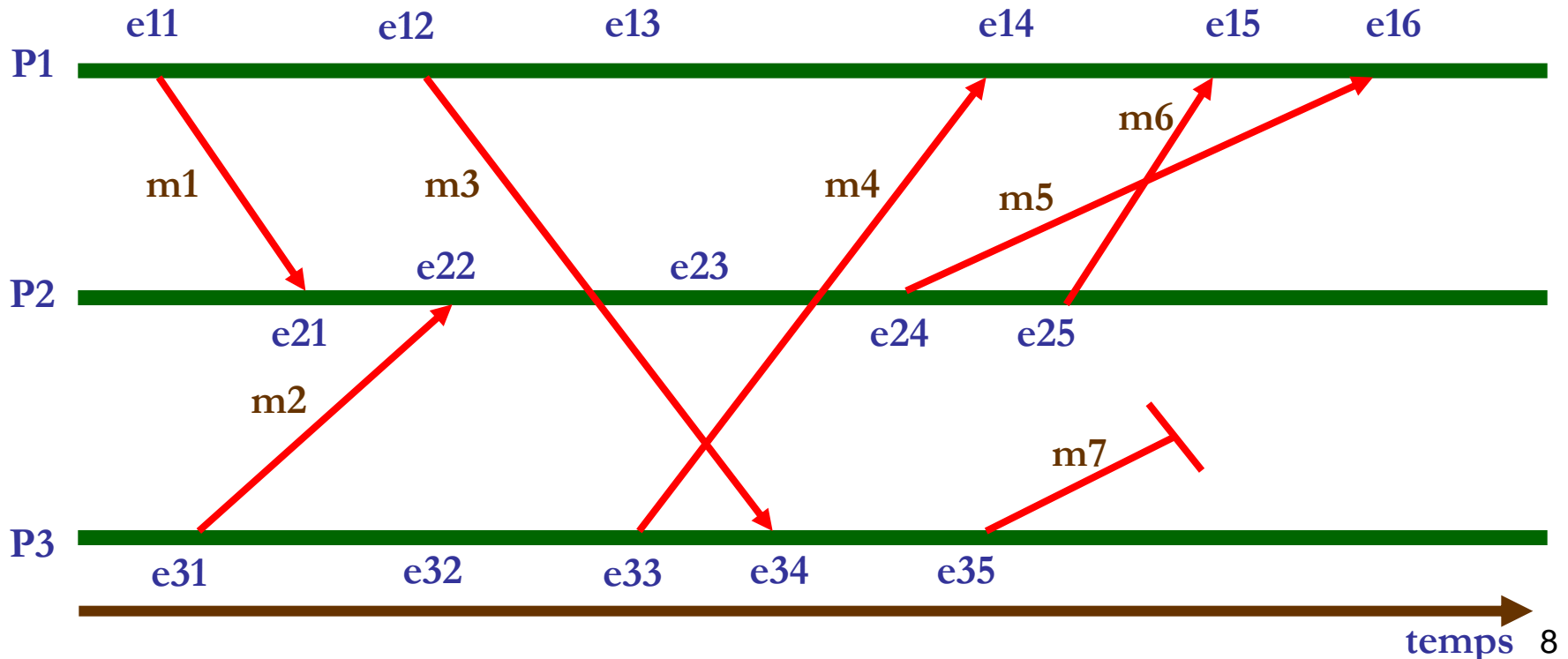
# Exemples d'événements

## □ 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.



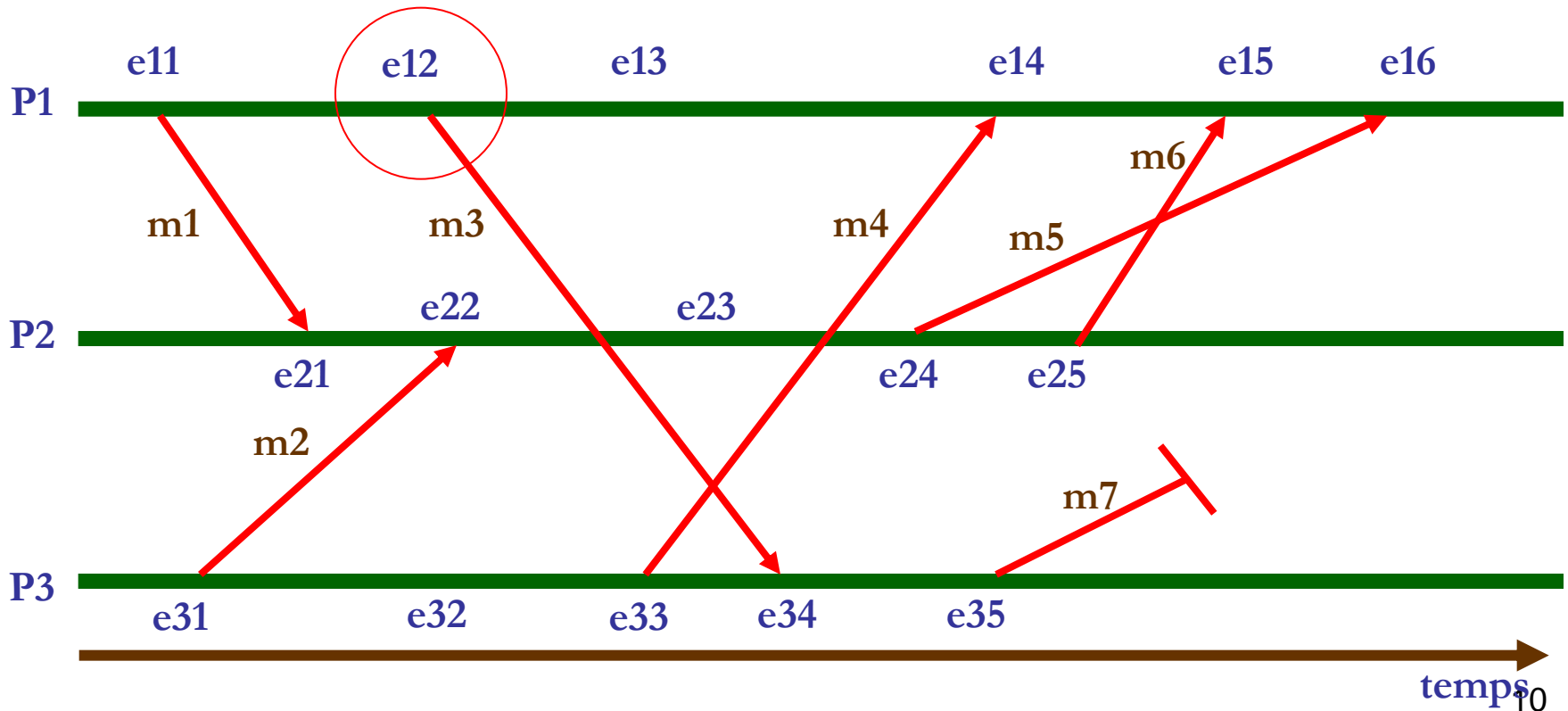


# Dépendance causale

- ❑ **Relation de dépendance causale** : Il y a une dépendance causale entre 2 événements si un événement doit avoir lieu avant l'autre.
- ❑ **Notation** :  $e \rightarrow e'$ .
  - ❑  $e$  doit se dérouler avant  $e'$ .
- ❑ Si  $e \rightarrow e'$ , alors une des trois conditions suivantes doit être vérifiée pour  $e$  et  $e'$ .
  - ❑ Si  $e$  et  $e'$  sont des événements d'un même processus,  $e$  précède localement  $e'$ .
  - ❑ Si  $e$  est l'émission d'un message,  $e'$  est la réception de ce message.
  - ❑ Il existe un événement  $f$  tel que  $e \rightarrow f$  et  $f \rightarrow e'$ .

# Dépendance causale

- ❑ Sur l'exemple précédent, quelques dépendances causales autour de e12.
- ❑ Localement :  $e11 \rightarrow e12$ ,  $e12 \rightarrow e13$ .
- ❑ Sur message :  $e12 \rightarrow e34$ .
- ❑ Par transitivité :  $e12 \rightarrow e35$  (car  $e12 \rightarrow e34$  et  $e34 \rightarrow e35$ ).
  - ❑  $e11 \rightarrow e13$  (car  $e11 \rightarrow e12$  et  $e12 \rightarrow e13$ ).

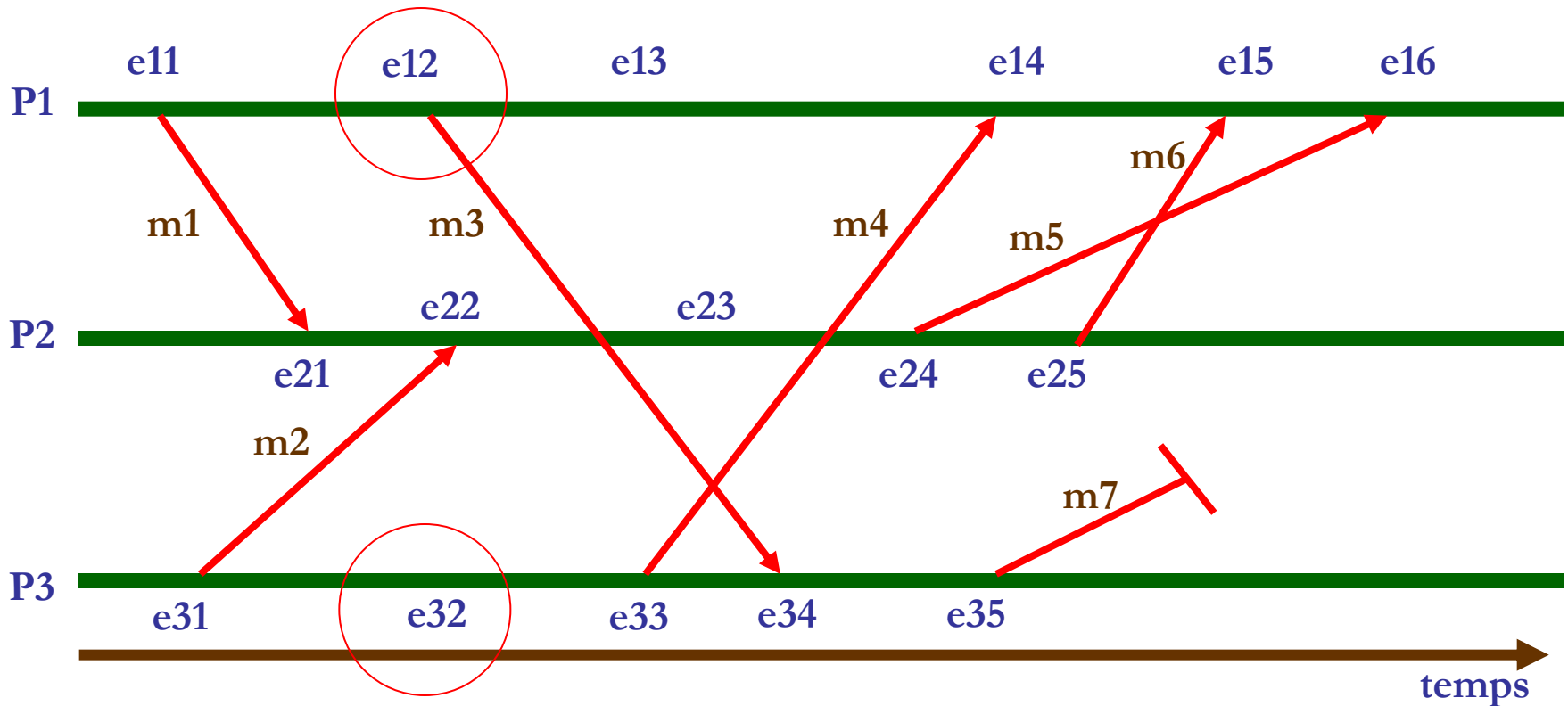


# Dépendance causale

❑ 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

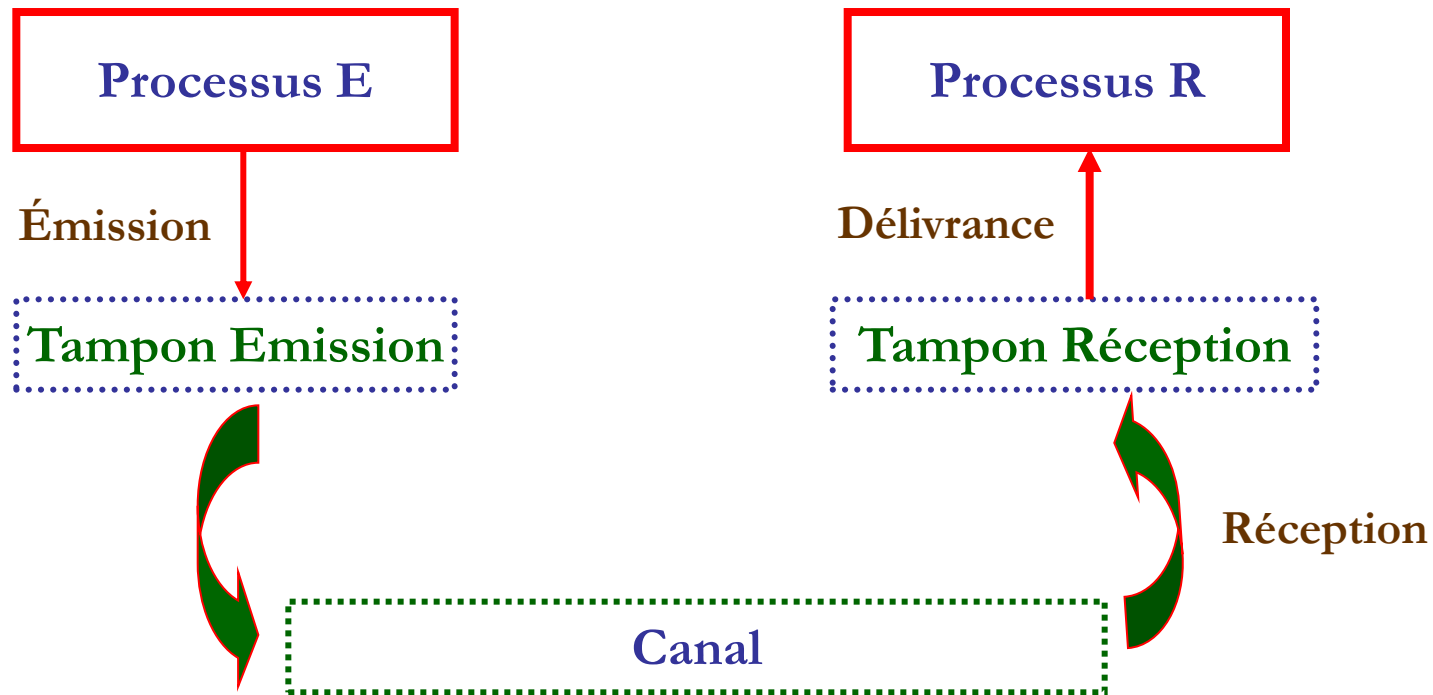
- ❑ e et e' sont en dépendance causale  $\Leftrightarrow ((e \rightarrow e') \text{ ou } (e' \rightarrow e))$ .
- ❑ Relation de parallélisme :  $||$ .
- ❑  $e || e' \Leftrightarrow \neg ((e \rightarrow e') \text{ ou } (e' \rightarrow e))$ .

Négation 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).

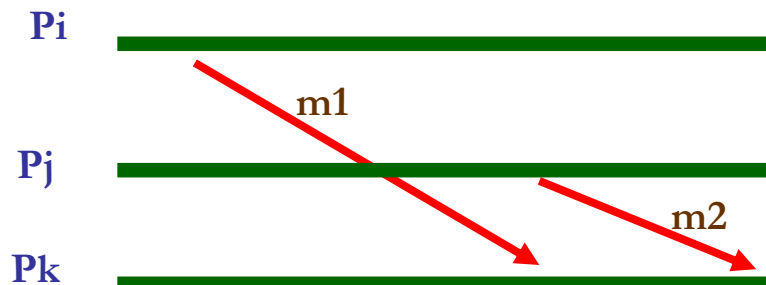
# Délivrance FIFO vs délivrance causale

- ❑ **Délivrance FIFO** : cette propriété assure que si deux messages sont envoyés successivement de  $P_i$  vers un même destinataire  $P_j$ , le premier sera délivré à  $P_j$  avant le second.



**Exemple :**  
communication  
avec les sockets

- ❑ **Délivrance causale** : cette propriété étend la précédente à des communications à destination d'un même processus en provenance de plusieurs autres.
- ❑ Elle assure que si l'envoi du message  $m1$  par  $P_i$  à destination de  $P_k$  précède (causalement) l'envoi du message  $m2$  par  $P_j$  à destination de  $P_k$ , le message  $m1$  sera délivré avant le message  $m2$  à  $P_k$ .



## Partie 2 :

# 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  $P_i$  possède une horloge locale logique  $H_i$ , initialisée à 0.

❑ Sert à dater les événements.

## ❑ Pour chaque événement local de $P_i$ .

❑  **$H_i = H_i + 1$**  : on incrémente l'horloge locale.

❑ L'événement est daté localement par  $H_i$ .

## ❑ Émission d'un message par $P_i$ .

❑ **On incrémente  $H_i$  de 1** puis on envoie le message avec  $(i, H_i)$  comme estampille.

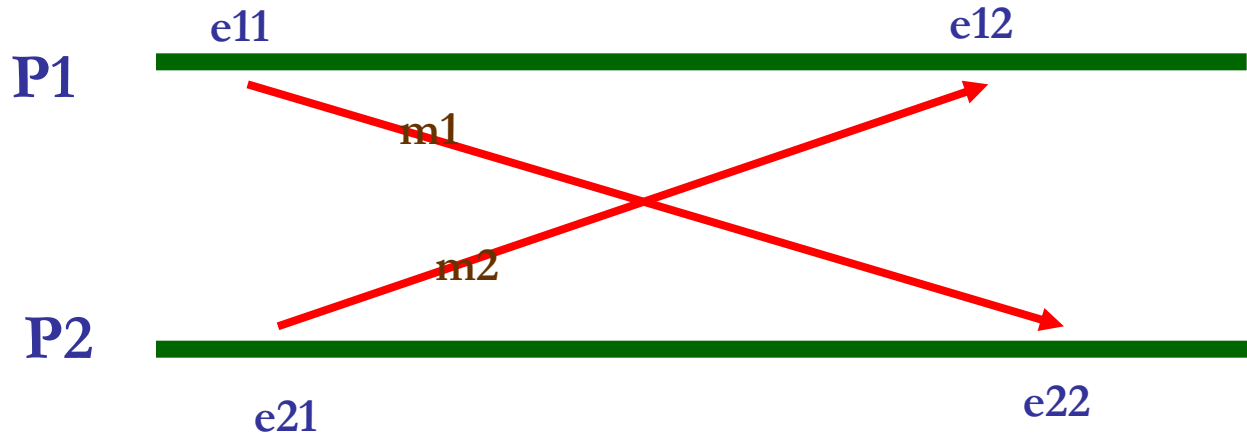
## ❑ Réception d'un message $m$ avec estampille $(i, nb)$

❑  **$H_j = \max(H_j, nb) + 1$**  et marque l'événement de réception avec  $H_j$ .

↑  
**Temps de l'horloge locale dans  $P_j$**

# Horloge de Lamport et dépendance causale

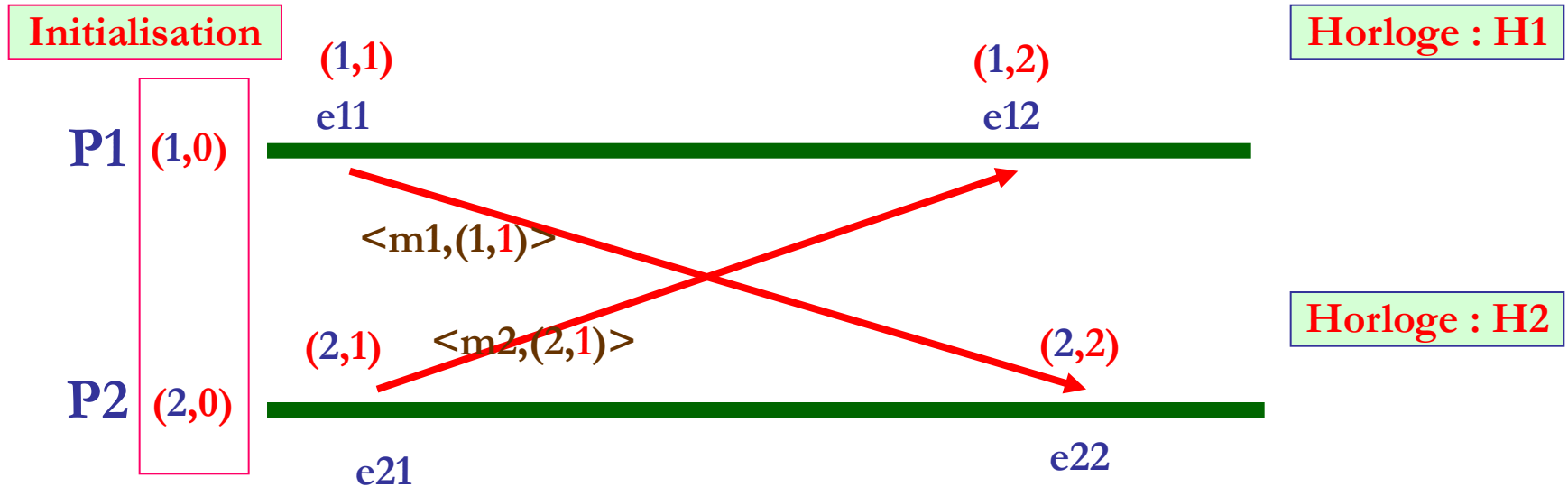
❑ Exemple 1 : Echange de messages entre 2 processus.



- ❑ Les dépendances causales :  $e11 \rightarrow e12$ ,  $e11 \rightarrow e22$ ,  $e21 \rightarrow e22$ ,  $e21 \rightarrow e12$ .
- ❑ Absence de dépendance causale entre e11 et e21.
  - ❑ **Parallélisme logique entre e11 et e21.**
- ❑ **L'horloge de Lamport respecte la dépendance causale :**
  - ❑  $(e \rightarrow e') \Rightarrow (H(e) < H(e'))$ .
  - ❑ Exemple :  $(e11 \rightarrow e12) \Rightarrow (H(e11) < H(e12))$ .

# 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  $(nb = nb' \text{ et } s < s')$ .
- ❑ Ordonnancement global :

(1,1)

(1,2)

(2,1)

(2,2)

- ❑ Résultat : e11, e21, e12, e22.
- ❑  $H(e11) < H(e21) < H(e12) < H(e22)$ .

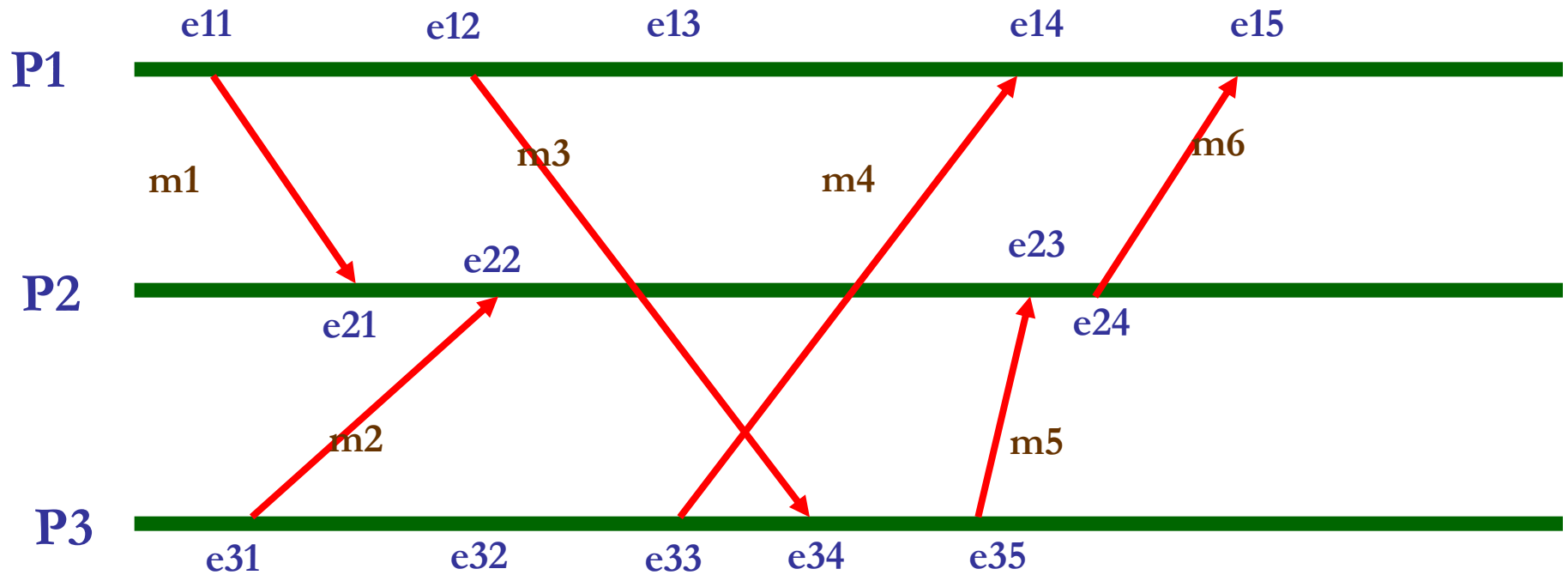
# Horloge de Lamport

- ❑ Ordonnancement global :  $e_{11}, e_{21}, e_{12}, e_{22}$ .
- ❑  $H(e_{11}) < H(e_{21}) < H(e_{12}) < H(e_{22})$ .
- ❑ L'horloge de Lamport respecte la dépendance causale :
  - ❑  $(e \rightarrow e') \Rightarrow (H(e) < H(e'))$ .
- ❑ Selon l'horloge :  $H(e_{11}) < H(e_{21})$ .
- ❑ Pourtant, il y a une absence de dépendance causale entre  $e_{11}$  et  $e_{21}$ .

- ❑ Pour résumé :
- ❑ L'horloge de Lamport respecte la dépendance causale :
  - ❑  $(e \rightarrow e') \Rightarrow (H(e) < H(e'))$ .
  - ❑ Mais pas la réciproque :
    - ❑  $(H(e) < H(e')) \not\Rightarrow (e \rightarrow e')$ .

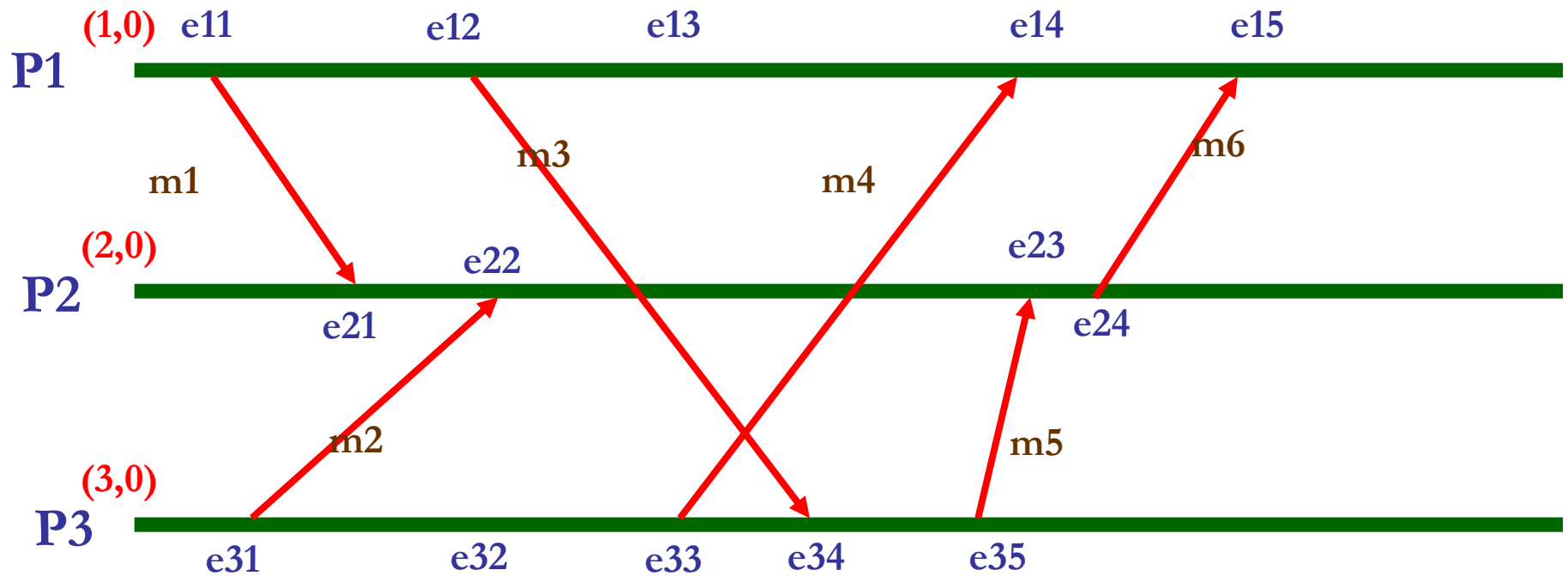
# Horloge de Lamport

□ Exemple 2 : chronogramme avec ajouts des estampilles.



# Horloge de Lamport

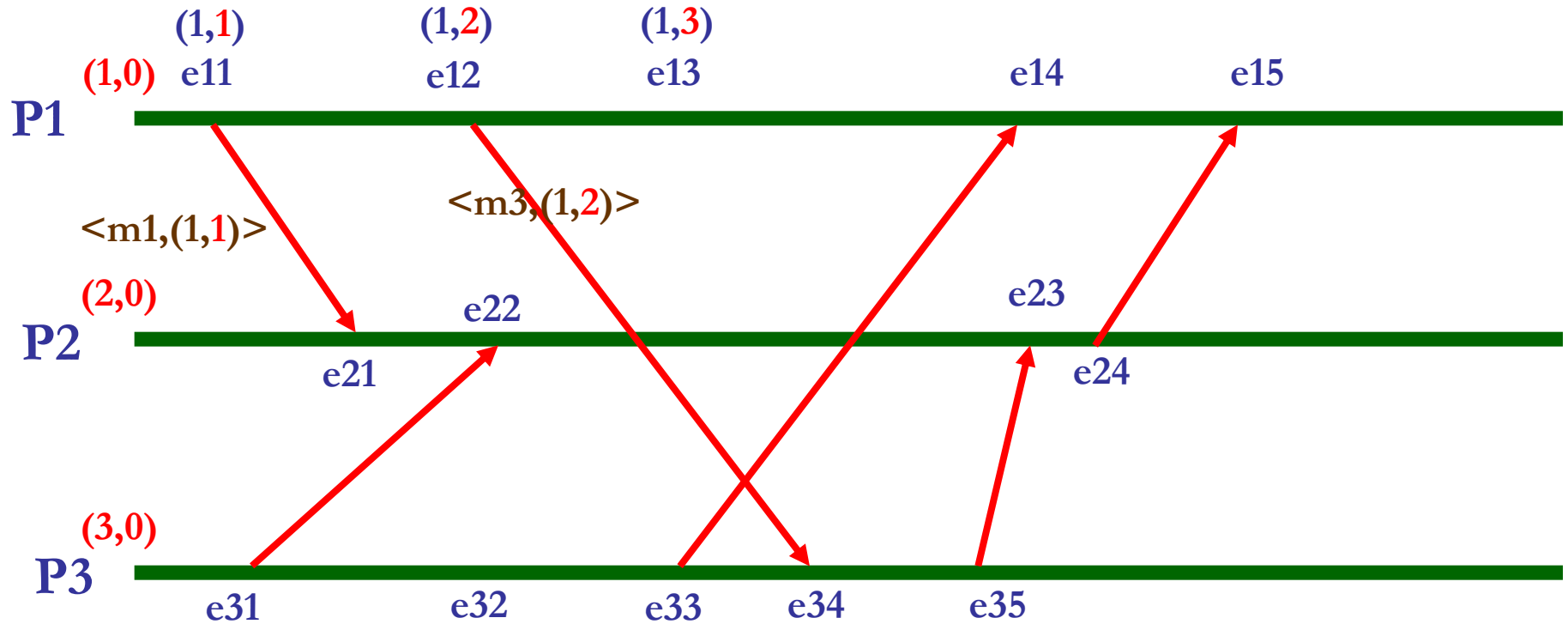
❑ Exemple 2 : chronogramme avec ajouts des estampilles.





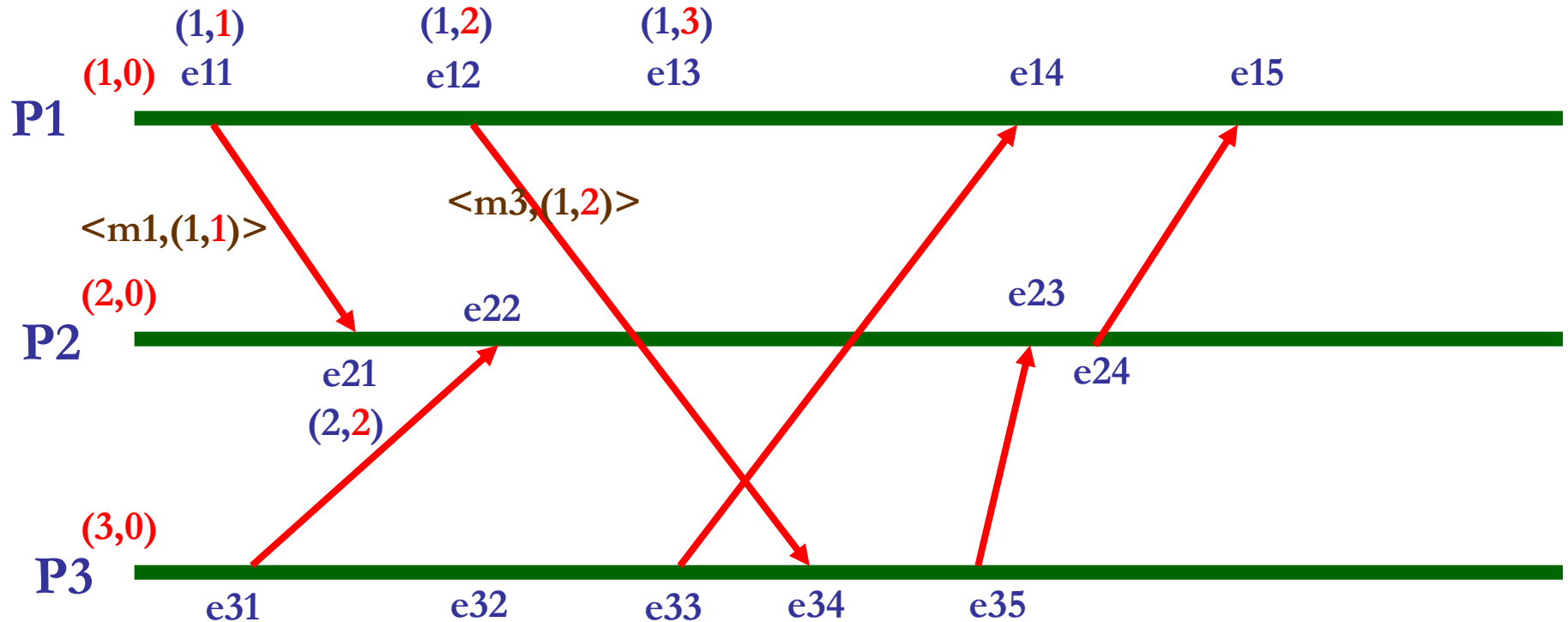
# Horloge de Lamport

□ Exemple 2 : chronogramme avec ajouts des estampilles.



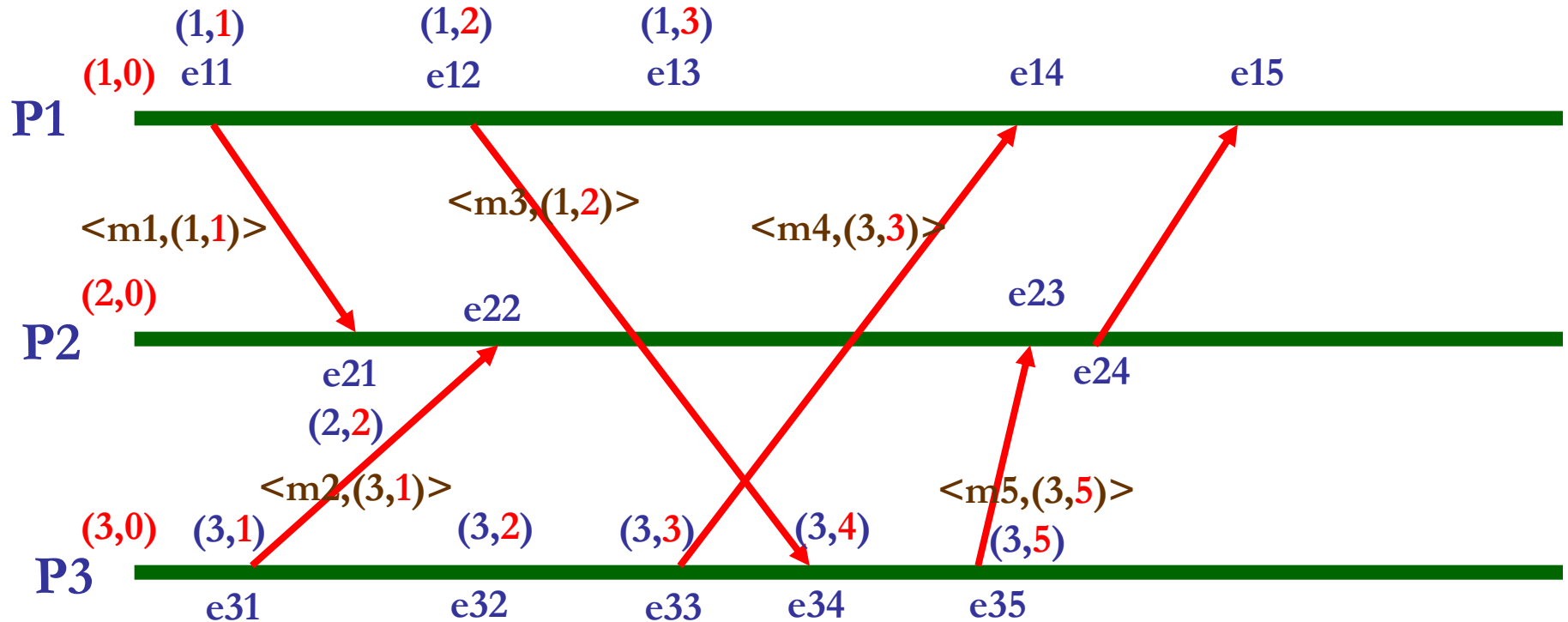
# Horloge de Lamport

❑ Exemple 2 : chronogramme avec ajouts des estampilles.



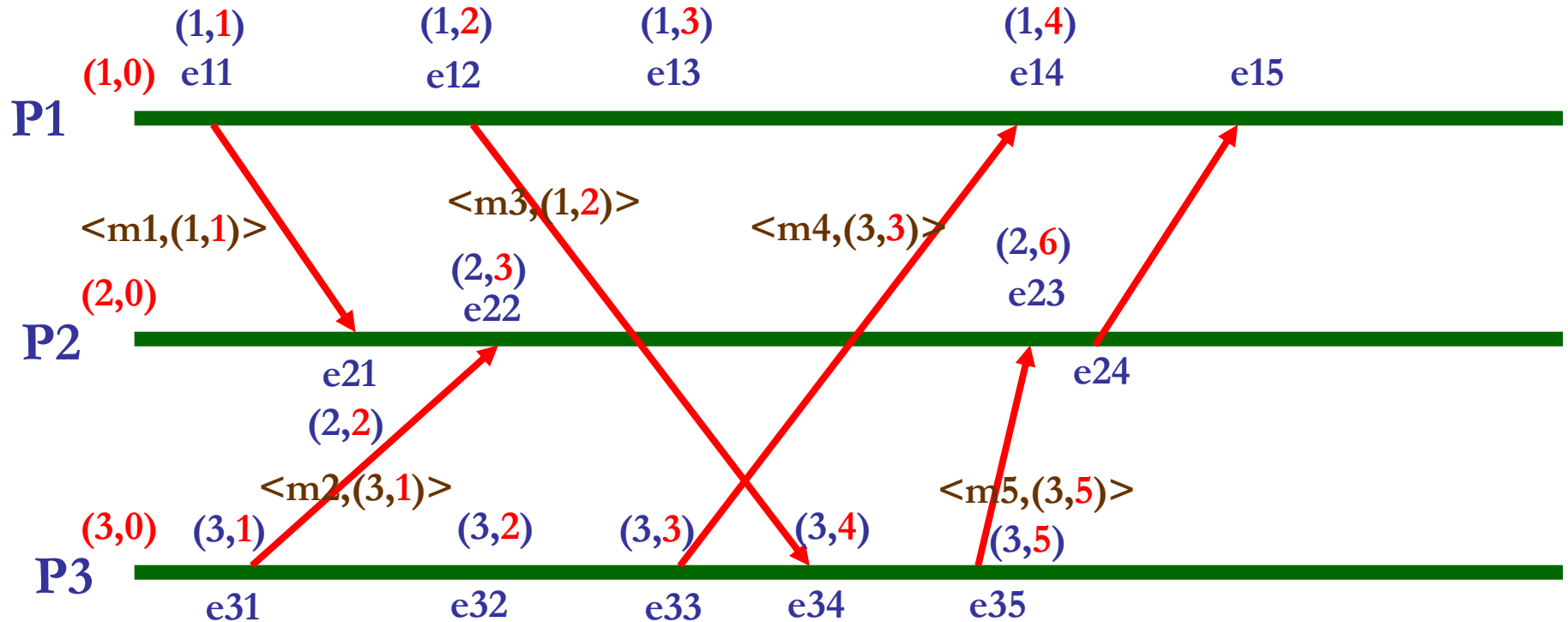
# Horloge de Lamport

□ Exemple 2 : chronogramme avec ajouts des estampilles.



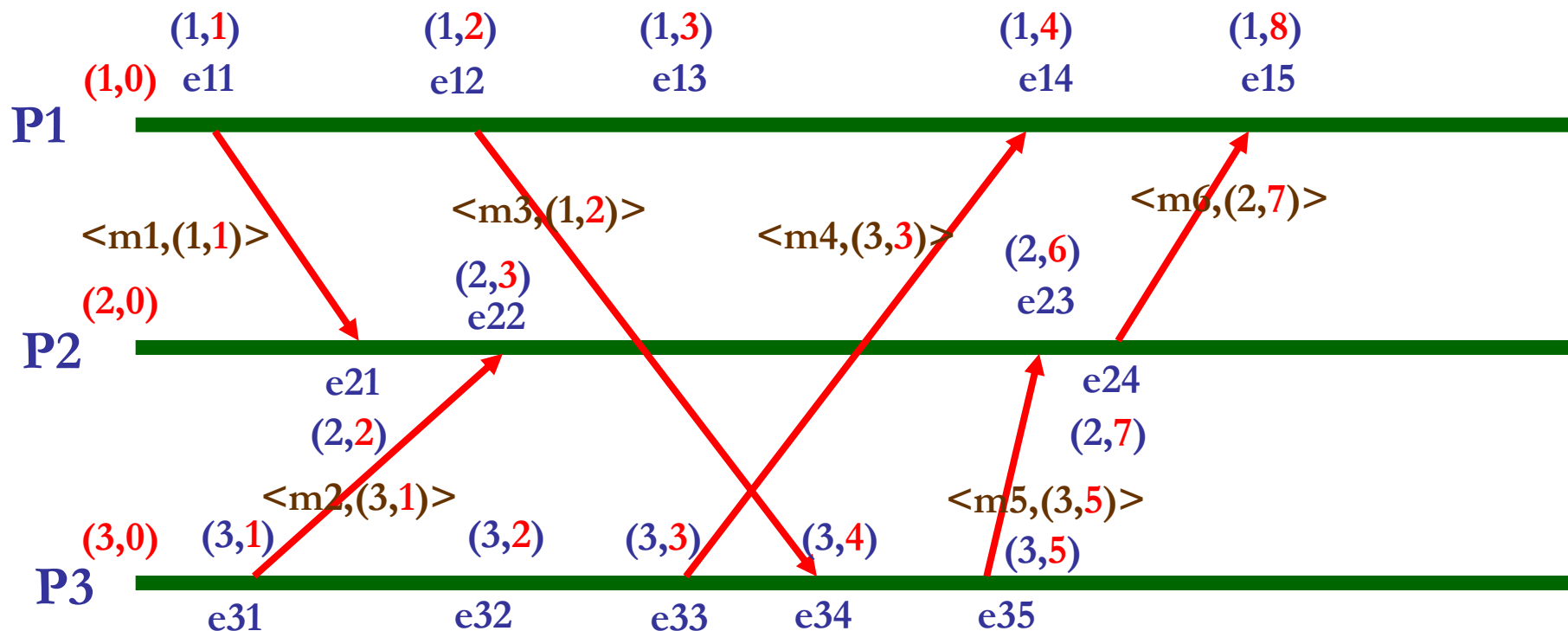
# Horloge de Lamport

❑ Exemple 2 : chronogramme avec ajouts des estampilles.



# Horloge de Lamport

❑ Exemple 2 : chronogramme avec ajouts des estampilles.



❑ Pour  $e_{11}$ ,  $e_{12}$ ,  $e_{13}$  ... : incrémentation de +1 de l'horloge locale.

❑ **Date de  $e_{23}$  :** 6 car le message  $m_5$  reçu avait une valeur de 5 et l'horloge locale est seulement à 3 ( $\max(3,5)+1$ ).

❑ **Date de  $e_{34}$  :** 4 car on incrémente l'horloge locale vu que sa valeur est supérieure à celle du message  $m_3$  ( $\max(3,2)+1$ ).

# Horloge de Lamport

- ❑ **Ordonnancement global :**

- ❑ Via  $H_i$ , on ordonne tous les événements du système entre eux.

- ❑ **Ordre total, noté  $e \ll e'$  :  $e$  s'est déroulé avant  $e'$ .**

- ❑ Soit  $e$  événement de  $P_i$  et  $e'$  événement de  $P_j$  :

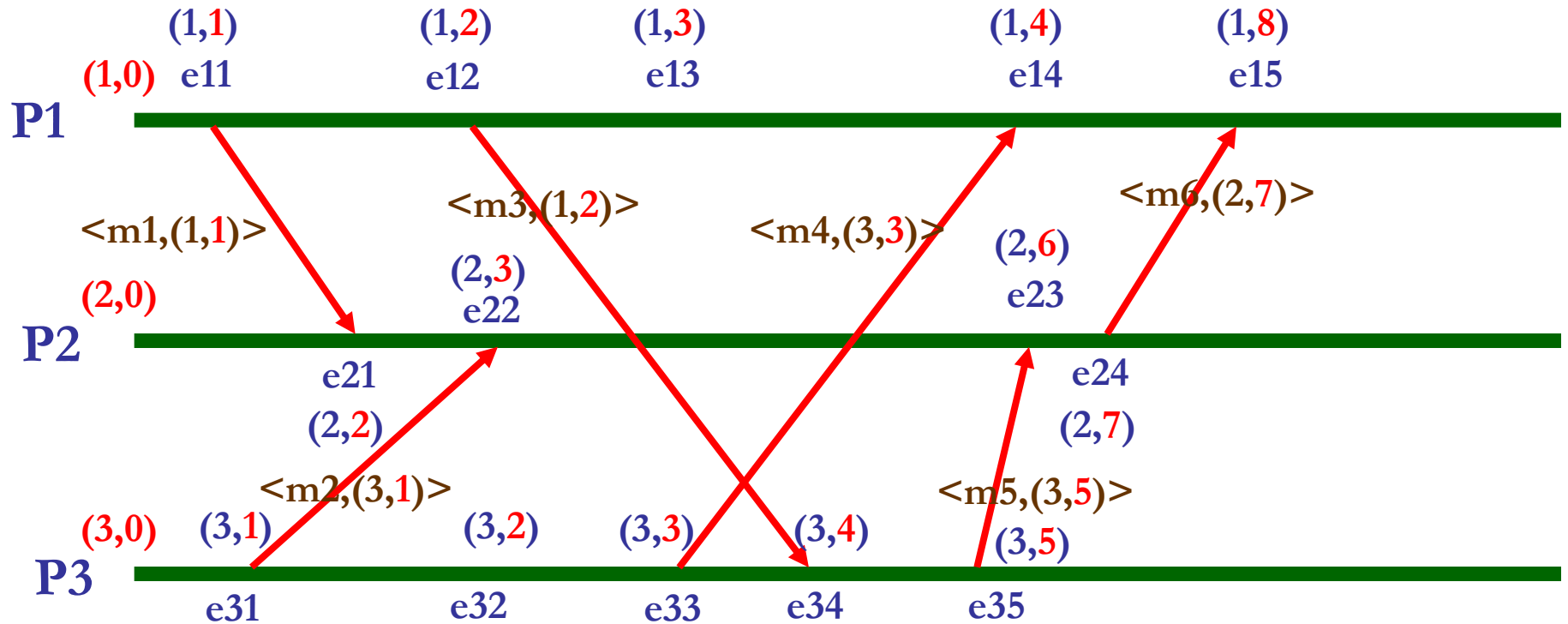
- ❑  $e \ll e' \Leftrightarrow (H_i(e) < H_j(e'))$  **ou**  $(H_i(e) = H_j(e'))$  avec  $i < j$ ).

- ❑ Localement (si  $i = j$ ),  $H_i$  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)

$e11 \ll e31 \ll e12 \ll e21 \ll e32 \ll e13 \ll e22 \ll e33 \ll e14 \ll e34 \ll e35 \ll e23 \ll e24 \ll e15.$

# Utilité de l'horloge de Lamport

**Faire l'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 :
  - ❑  $H(e) < H(e') \Rightarrow (e \rightarrow e')$ .
- ❑ Permet également de savoir si 2 événements sont parallèles (non dépendants causalement).
- ❑ Ne définit par contre pas un ordre total global.
- ❑ **Principe :**
- ❑ Utilisation de vecteur  $V$  de taille égale au nombre de processus.
  - ❑ Localement, chaque processus  $P_i$  a un vecteur  $V_i$ .
  - ❑ Un message est envoyé avec un vecteur de date.
- ❑ Pour chaque processus  $P_i$ , chaque case  $V_i[j]$  du vecteur contiendra des valeurs de l'horloge du processus  $P_j$ .

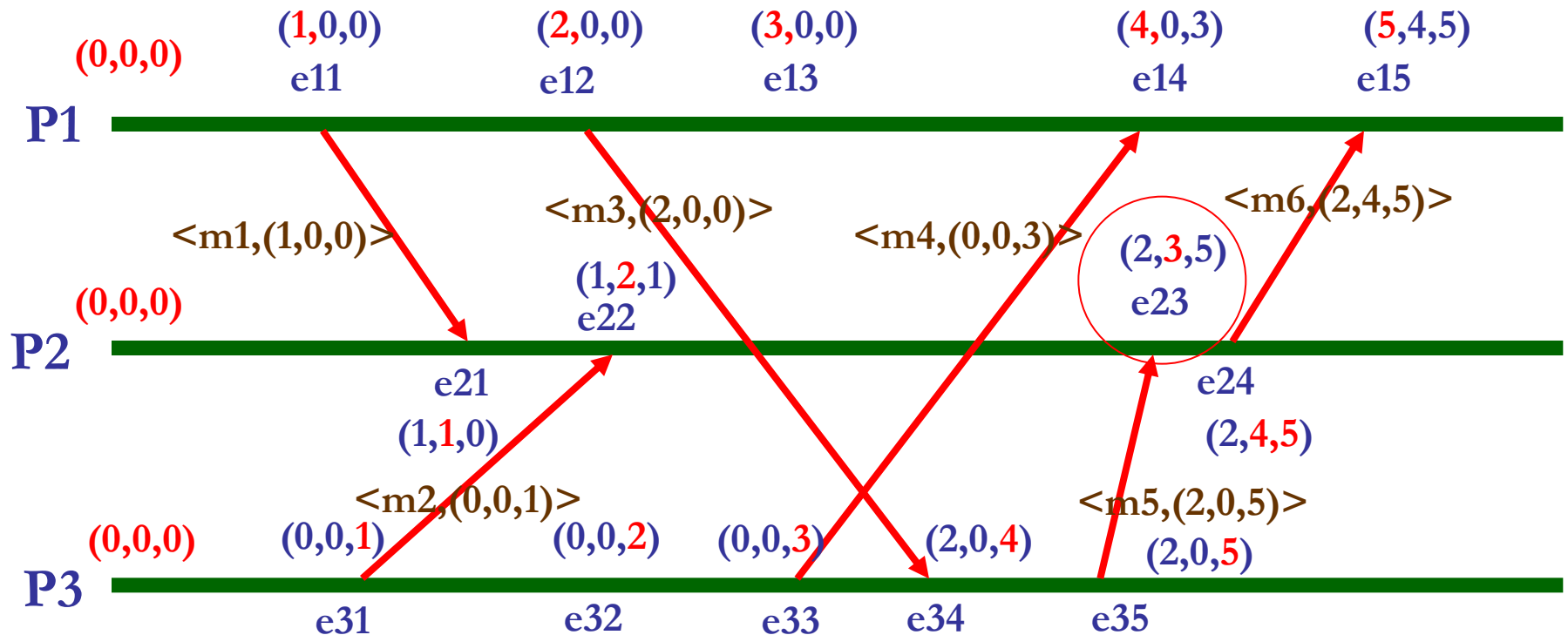
# Horloge de Mattern

## □ Fonctionnement de l'horloge :

- Initialisation : pour chaque processus  $P_i$ ,  $\mathbf{V}_i = (0, \dots, 0)$ .
- Pour un processus  $P_i$ , à chacun de ses événements (local, émission, réception) :  $\mathbf{V}_i[i] = \mathbf{V}_i[i] + 1$ .
  - Incrémentation du compteur local d'événement (garder les autres compteurs si événement local ou émission).
  - Si émission d'un message, alors  $\mathbf{V}_i$  est envoyé avec le message.
- Pour un processus  $P_i$ , à la réception d'un message  $m$  contenant un vecteur  $\mathbf{V}_m$  provenant du processus  $P_j$ , on met à jour les cases  $k \neq i$  de son vecteur local  $\mathbf{V}_i$ .
  - $\forall k \neq i : \mathbf{V}_i[k] = \max(\mathbf{V}_m[k], \mathbf{V}_i[k])$ .
    - Mémorise le nombre d'événements sur  $P_j$  qui doivent se dérouler avant la réception du message sur  $P_i$ .
    - La réception de ce message sur  $P_i$  dépend causalement de tous ces événements sur  $P_j$ .

# Horloge de Mattern

□ Même exemple que pour horloge de Lamport :



□  $V(e_{23}) = (2,3,5)$  : (2 relatif à P1, 3 à P2, 5 à P3).

□  $3 = 2+1$  (Incrémentation du compteur local). 3ème événement dans P2.

□  $2 = \max(2,0)$ . 2 événements dans P1 (e11 et e12) qui sont en dépendance causale par rapport à l'événement e23.

□  $5 = \max(5,3)$ . 5 événements dans P3 (e31, e32, e33, e34 et e35) qui sont en dépendance causale par rapport à l'événement 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  $V[j] < V'[j]$ .
  - ❑  $V \parallel V'$  défini par  $\neg ((V < V') \text{ ou } (V' < V))$ .
- ❑ **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.
  - ❑  $V(e) \parallel V(e') \Rightarrow e \parallel e'$ .
    - ❑ 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 :**
- ❑  $V(e_{13}) = (3,0,0)$ ,  $V(e_{14}) = (4,0,3)$ ,  $V(e_{15}) = (5,4,5)$ .
  - ❑  $V(e_{13}) < V(e_{14})$  donc  $e_{13} \rightarrow e_{14}$ .
  - ❑  $V(e_{14}) < V(e_{15})$  donc  $e_{14} \rightarrow e_{15}$ .
- ❑  $V(e_{35}) = (2,0,5)$  et  $V(e_{23}) = (2,3,5)$ .
  - ❑  $V(e_{35}) < v(e_{23})$  donc  $e_{35} \rightarrow e_{23}$ .
- ❑ **L'horloge de Mattern respecte les dépendances causales des événements ainsi que la réciproque.**
  - ❑ Horloge de Lamport respecte uniquement les dépendances causales.

- ❑  $V(e_{32}) = (0,0,2)$  et  $V(e_{13}) = (3, 0, 0)$ .
  - ❑ On a ni  $V(e_{32}) < V(e_{13})$  ni  $V(e_{13}) < V(e_{32})$  donc  $e_{32} || e_{13}$ .
- ❑ **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 :

$$((a \in C) \text{ et } (b \rightarrow a)) \Rightarrow (b \in C).$$

□ Si un événement d'un processus appartient à la coupure, alors tous les événements locaux le précédant y appartiennent également.



# Coupure

## □ Etat associé à une coupure :

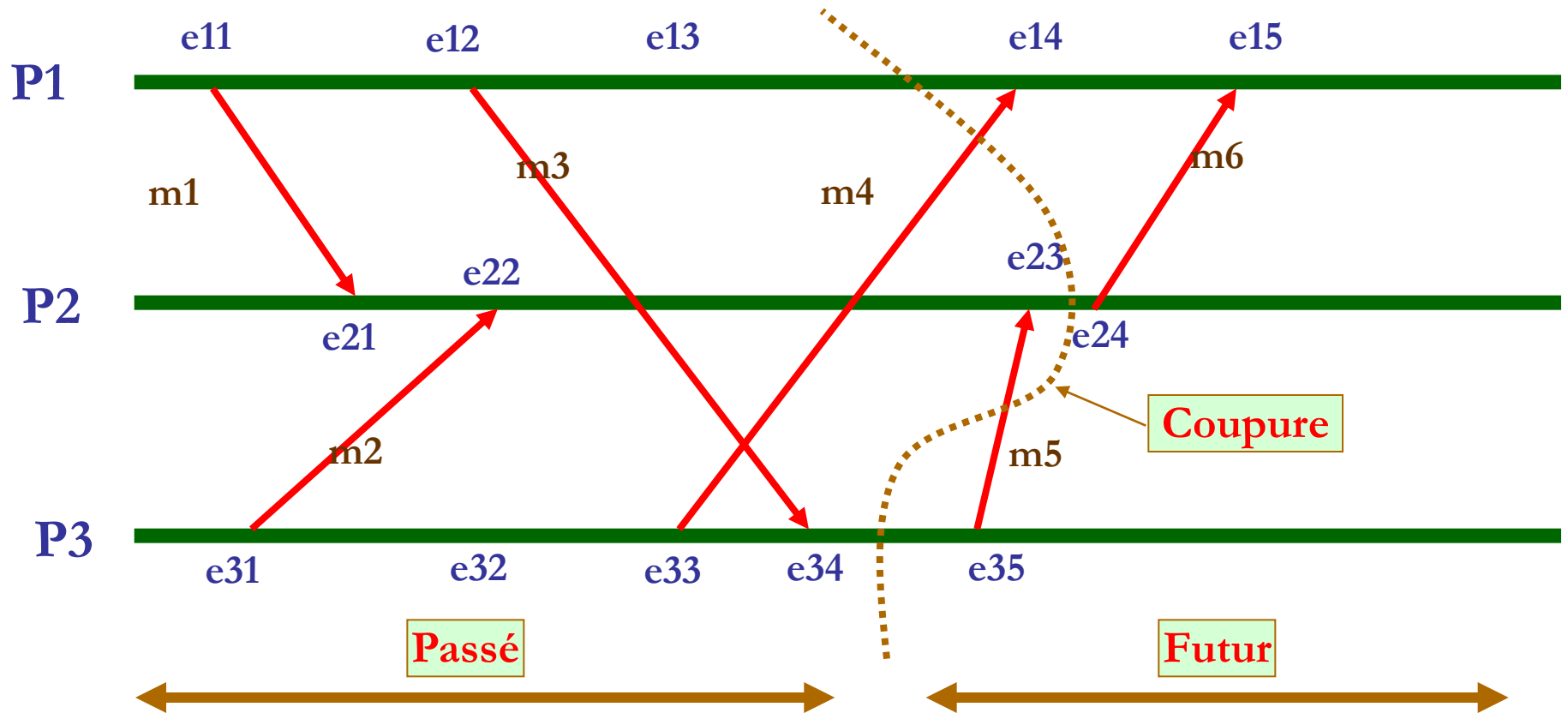
- Si le système est composé de  $N$  processus, l'état associé à une coupure est défini au niveau d'un ensemble de  $N$  événements  $(e_1, e_2, \dots e_i, \dots e_N)$ , avec  $e_i$  événement du processus  $P_i$  tel que :

$$\square \forall i : \forall e \in C \text{ et } e \text{ événement du processus } P_i \Rightarrow e \rightarrow e_i.$$

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

# Coupure

□ Exemple de coupure :



□ Coupure = ensemble  $\{e_{11}, e_{12}, e_{13}, e_{21}, e_{22}, e_{23}, e_{31}, e_{32}, e_{33}, e_{34}\}$ .

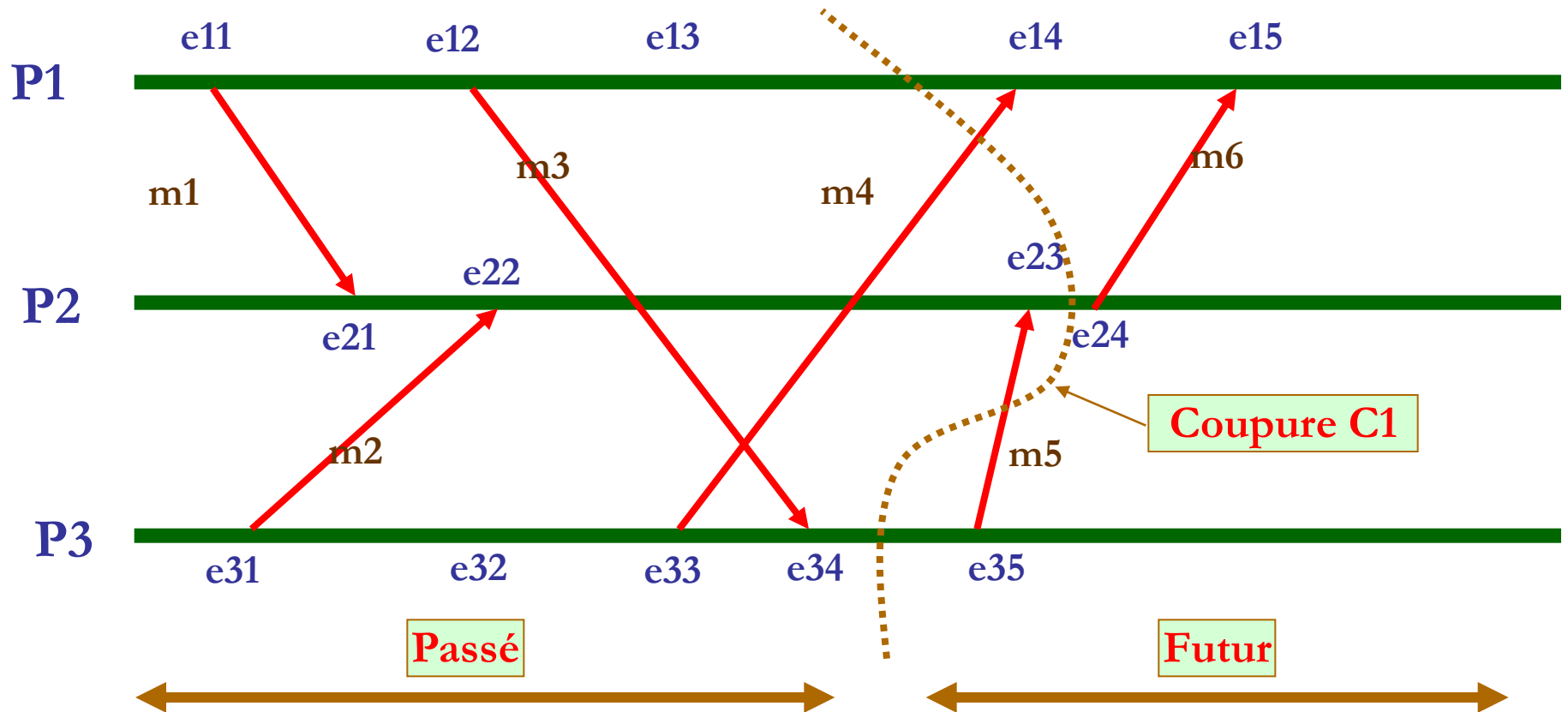
□ État défini par la coupure =  $(e_{13}, e_{23}, e_{34})$ .

# 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.

# Coupure

□ Exemple de coupure :



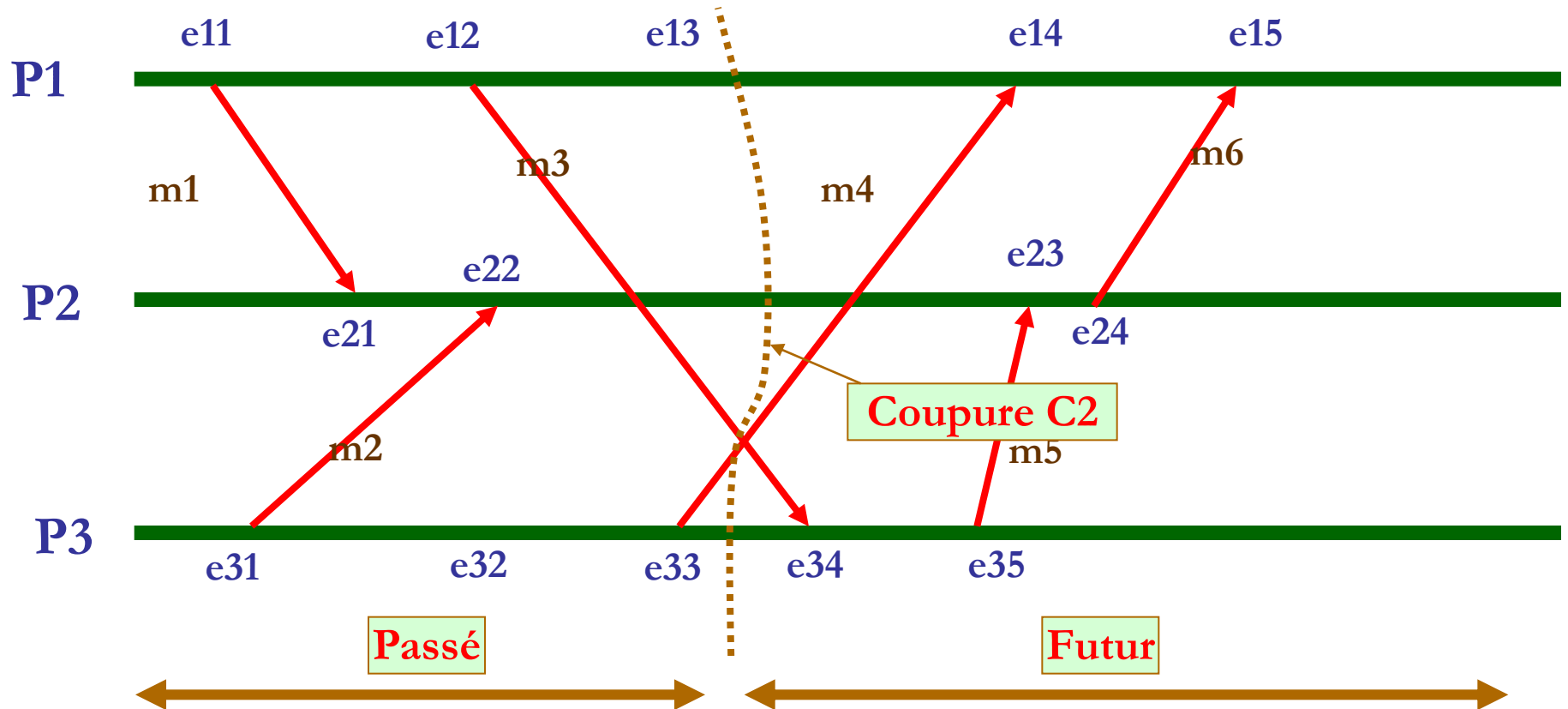
□ **Coupure C1 : non cohérente** car :  $e23 \in C1$  et  $e35 \rightarrow e23$  mais  $e35 \notin C1$ .

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

□  $m5$  vient du futur par rapport à la coupure.

# Coupure

❑ Coupure C2 : cohérente.



# Datation coupure

- ❑ **Horloge de Mattern permet de dater la coupure.**
- ❑ Soit  $N$  processus,  $C$  la coupure,  $e_i$  l'événement le plus récent pour le processus  $P_i$ ,  $V(e_i)$  la datation de  $e_i$  et  $V(C)$  la datation de la coupure.
  - ❑  $V(C) = \max ( V(e_1), \dots, V(e_N) ) :$ 
    - ❑  $\forall i : V(C)[i] = \max ( V(e_1)[i], \dots, V(e_N)[i] ) .$
  - ❑ 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  $V(C) = ( V(e_1)[1], \dots, V(e_i)[i], \dots, V(e_N)[N] ) .$

# 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.**

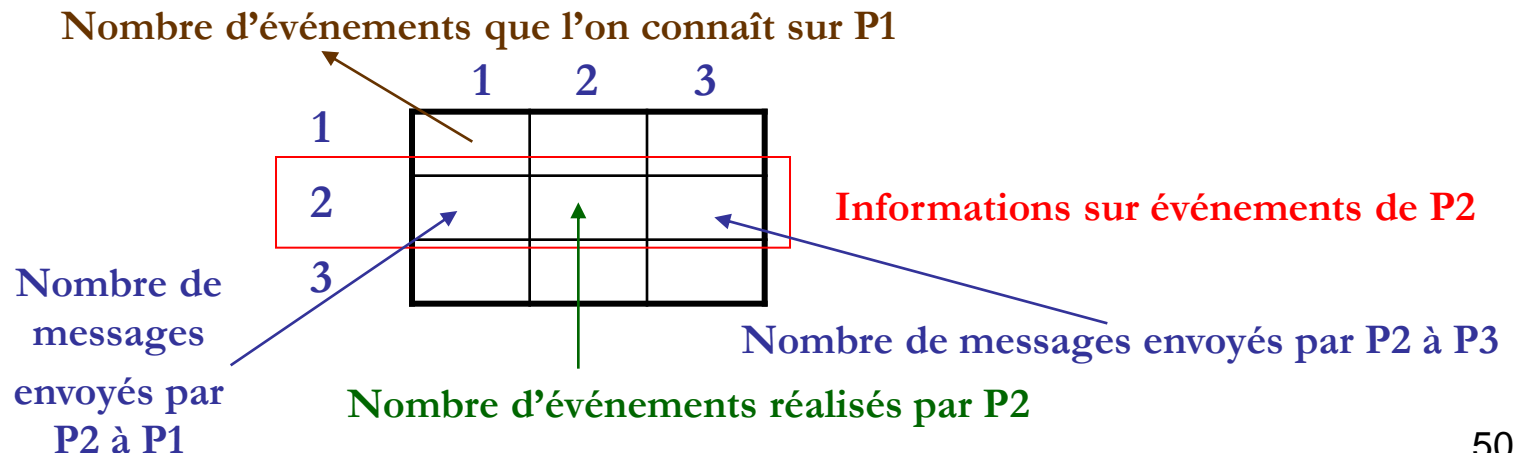
Permet également de savoir si 2 événements sont parallèles ou en dépendance causale.



# Horloge matricielle

# Horloge matricielle

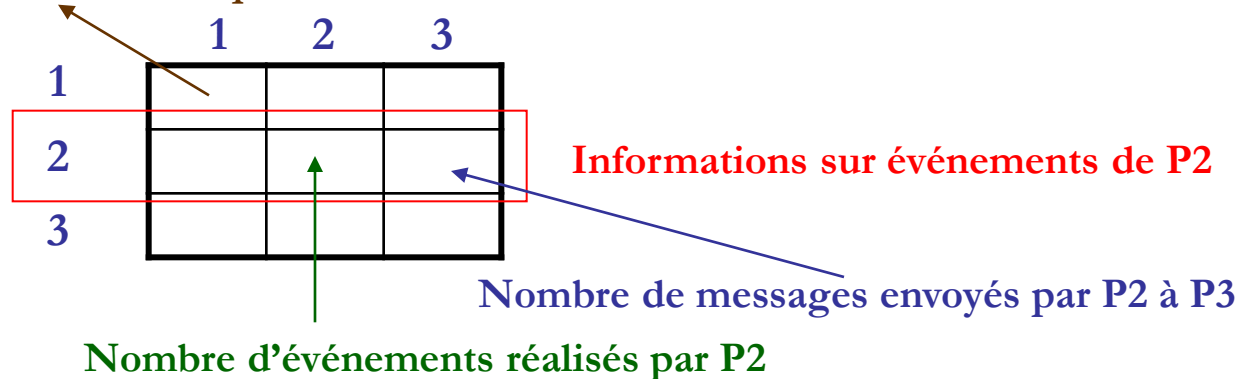
- ❑  $n$  processus : matrice  $M$  de  $(n \times n)$  pour dater chaque événement.
- ❑ **Sur processus  $P_i$  :**
  - ❑ Ligne  $i$  : informations sur événements de  $P_i$  :
    - ❑  $M_i[i, i]$  : nombre d'événements réalisés par  $P_i$ .
    - ❑  $M_i[i, j]$  : nombre de messages envoyés par  $P_i$  à  $P_j$  (avec  $j \neq i$ ).
  - ❑ Ligne  $j$  (avec  $j \neq i$ ) :
    - ❑  $M_i[j, j]$  : nombre d'événements que l'on connaît sur  $P_j$ .
    - ❑  $M_i[j, k]$  : nombre de messages que l'on sait que  $P_j$  a envoyé à  $P_k$  (avec  $j \neq k$ ).
- ❑ **Avec 3 processus :**



# Horloge matricielle

- Un processus  $P_i$  a une connaissance sur :
  - Le nombre de messages qu'un processus  $P_j$  a envoyé à  $P_k$ .
  - Le nombre d'événements sur  $P_j$ .

Nombre d'événements que l'on connaît sur P1



- Quand on reçoit un message d'un autre processus, on compare l'horloge d'émission avec l'horloge locale.
  - Peut déterminer si on ne devait pas recevoir un message avant.
  - Exemple à voir en TD.

# Utilité de l'horloge matricielle

**Assurer la délivrance causale de messages entre plusieurs processus.**