

# Plan

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

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

## ❑ **Partie 2 : Horloges de Lamport**

## ❑ **Partie 3 : Exclusion mutuelle basée sur les Horloges de Lamport**

# **Partie 1 :**

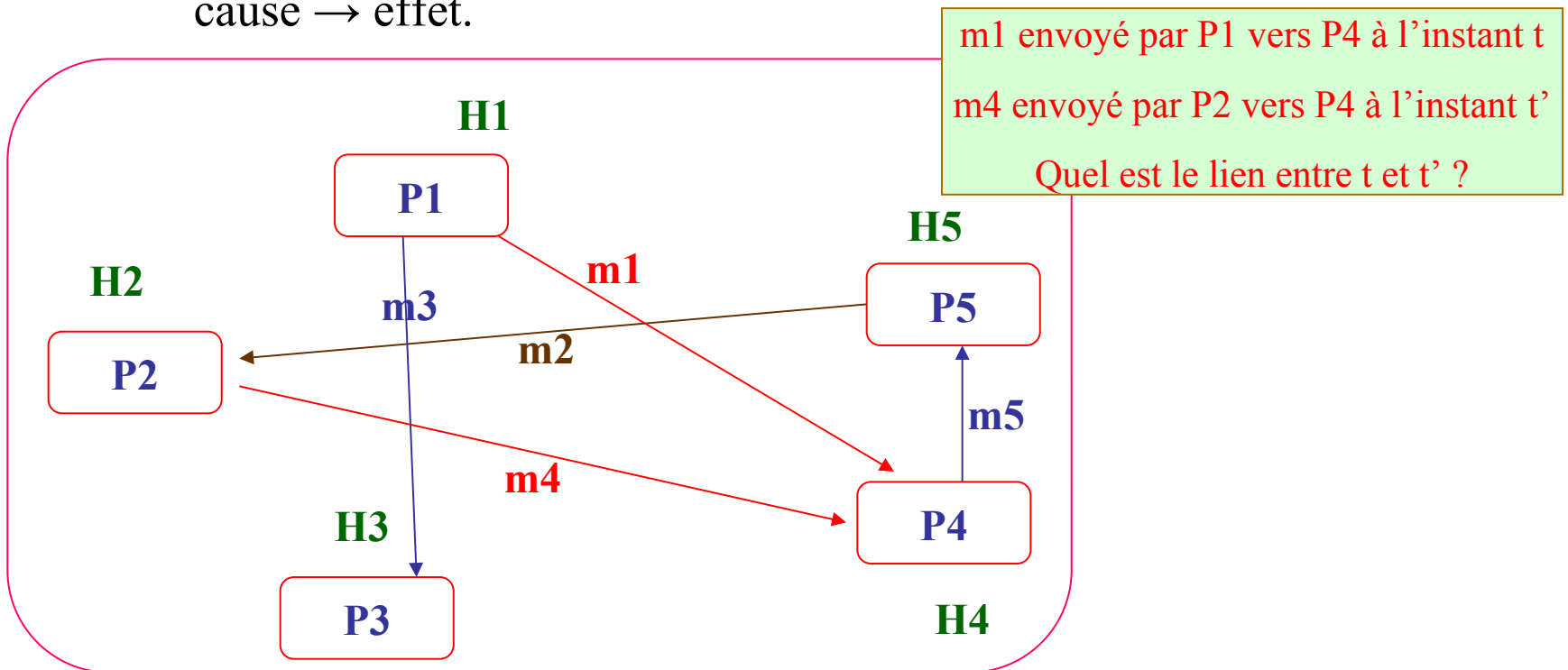
## **Temps dans un système distribué**

# Temps dans un système distribué

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

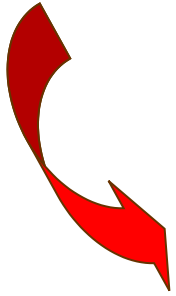
❑ Créer **un temps logique**.

❑ Temps qui n'est pas lié à un temps physique mais à une relation de cause → effet.



# Temps logique

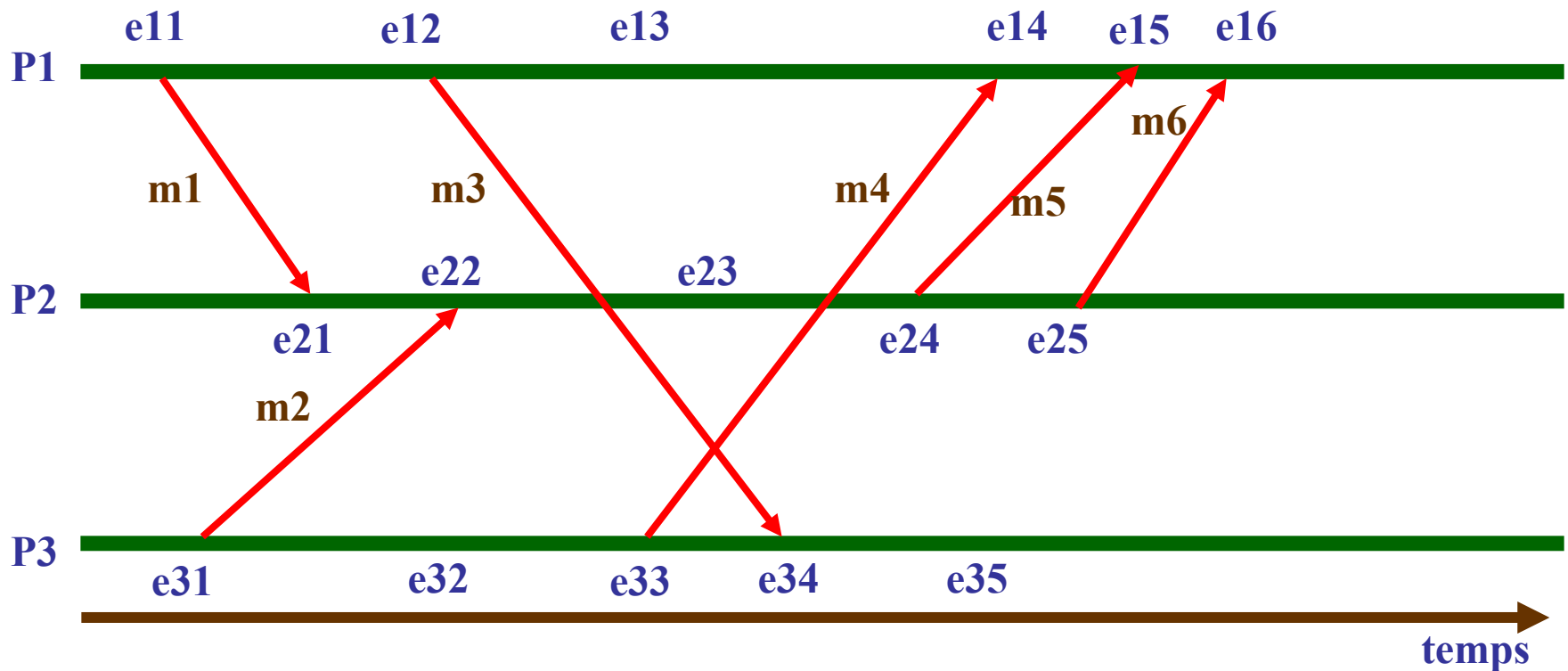
- ❑ **Exemple d'application** : pouvoir préciser l'ordonnancement de l'exécution des processus et de leur communication.
- ❑ En fonction des événements locaux des processus, des messages envoyés et reçus, on crée **un ordonnancement logique**.



- ❑ **Utiliser une horloge logique.**

# 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.



# Chronogramme

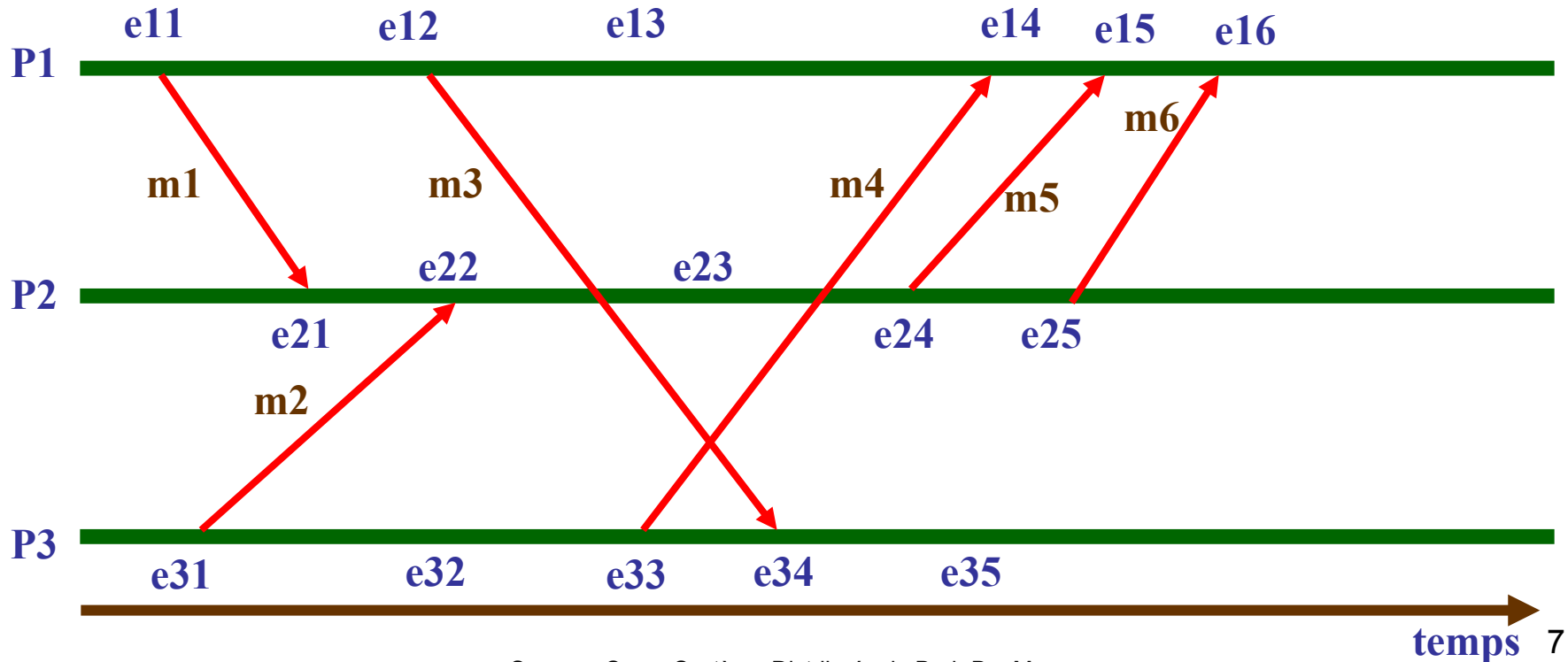
- ❑ **Définition** : décrit l'ordonnancement temporel des événements des processus et des échanges de messages.
- ❑ **Chaque processus est représenté par une ligne.**
- ❑ Trois types d'événements signalés sur une ligne :
  - ❑ **Émission** d'un message à destination d'un autre processus.
  - ❑ **Réception** d'un message venant d'un autre processus.
  - ❑ **Événement interne** dans l'évolution du processus.
- ❑ Les messages échangés doivent se faire sans perte, duplication ou ré-ordonnancement (les canaux de communication inter-processus sont des FIFO)

# 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.



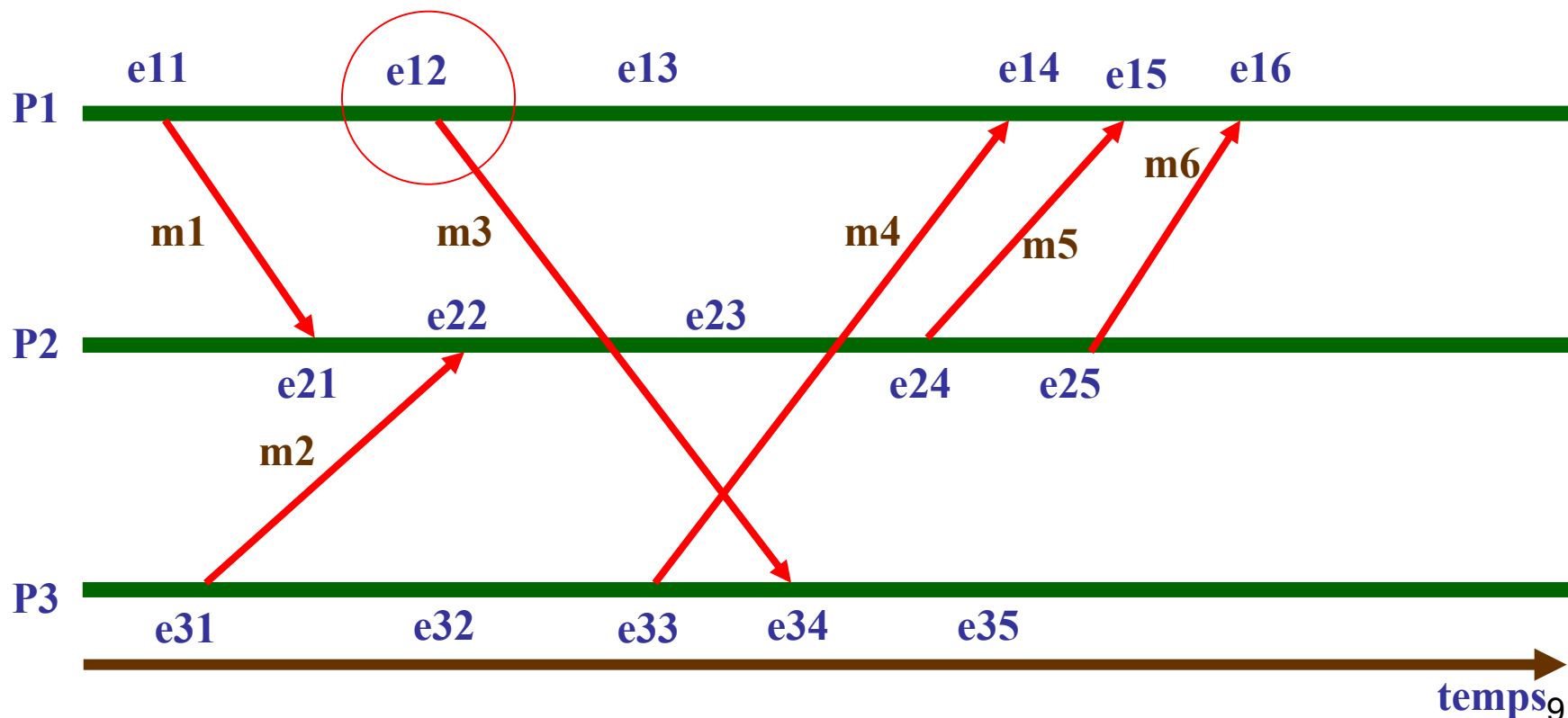
# Dépendance causale

- ❑ **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  $f \rightarrow e'$ .
- ❑ **Ordonnancement des événements** :
  - ❑ Les dépendances causales définissent des **ordres partiels** pour des ensembles d'événements du système.



# 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  $e12 \rightarrow e13$ ).

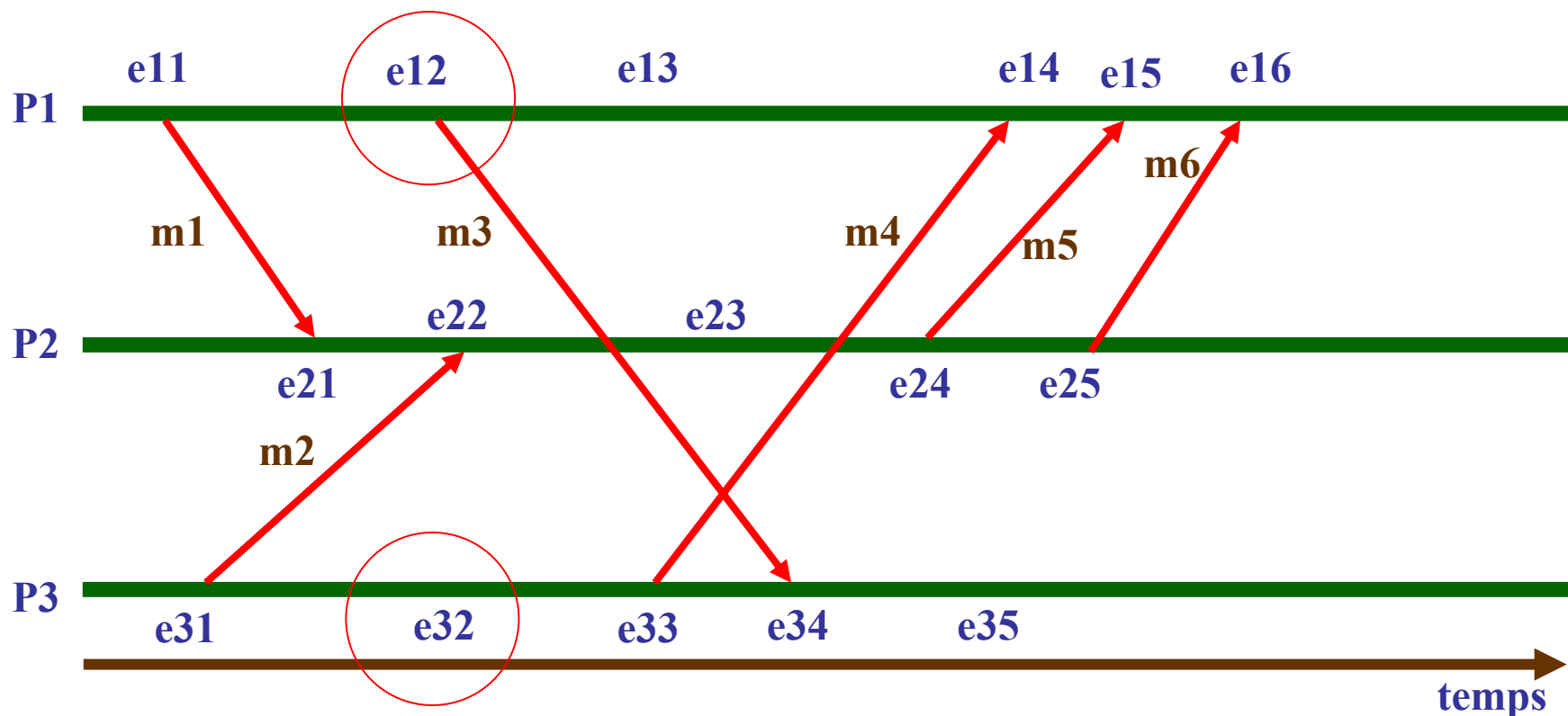


# Dépendance causale

❑ **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 :  $\parallel$ .
- ❑  $e \parallel e' \Leftrightarrow \neg ((e \rightarrow e') \text{ ou } (e' \rightarrow e))$ .

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.**

## **Partie 2 : Horloge de Lamport**

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

**But : Ordonner totalement les événements d'un système distribué, sans utiliser d'horloge physique**

# Horloge de Lamport

- ❑ Introduit en 1978 par **Leslie Lamport**.
  - ❑ <https://lamport.azurewebsites.net/pubs/time-clocks.pdf>
- ❑ 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 **nb**, initialisée à 0.

❑ Sert à dater les événements.

❑ Pour chaque événement local de  $P_i$ .

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

❑ L'événement est daté localement par nb.

❑ Émission d'un message par  $P_i$ .

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

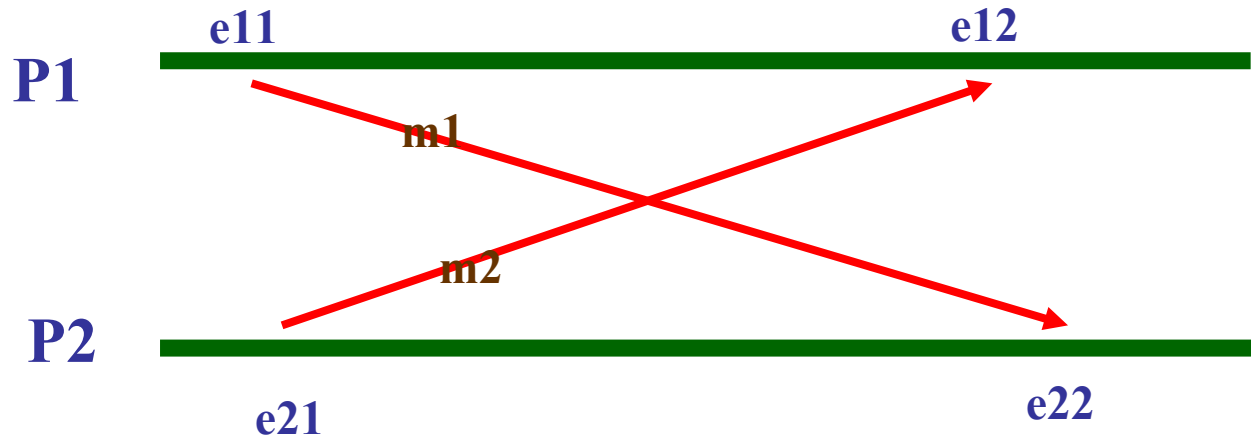
❑ Réception d'un message avec estampille (i, nb) par  $P_j$

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

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

# Horloge de Lamport et dépendance causale

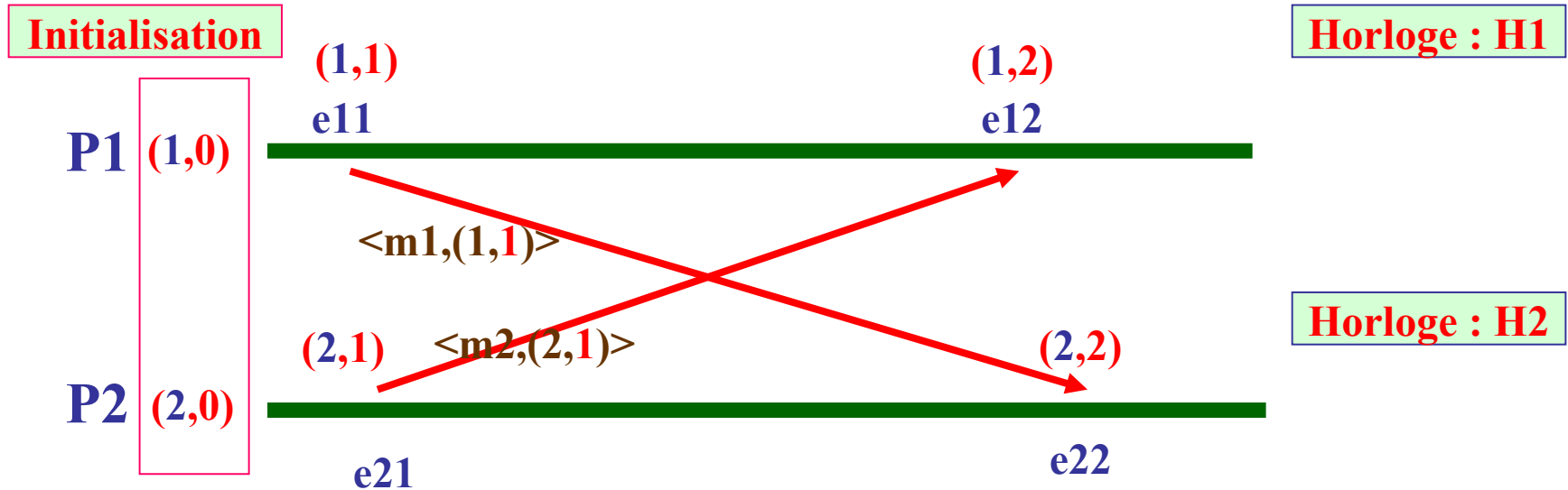
- ❑ **Exemple 1** : Echange de messages entre 2 processus.



- ❑ 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  $H(x)$  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 :  $H(i, nb) < H(j, nb')$  si  $(nb < nb')$  ou  $(nb = nb' \text{ et } i < j')$ .
- Ordonnancement global :

(1,1)

(1,2)

(2,1)

(2,2)

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



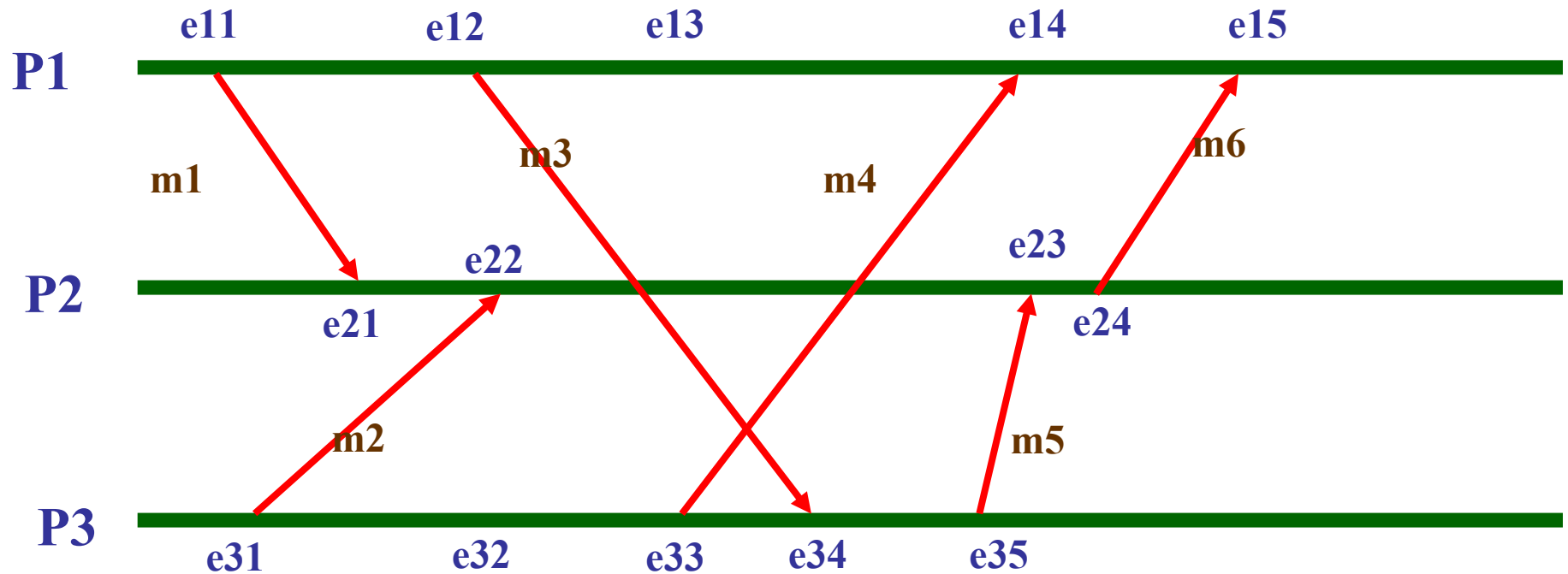
# Horloge de Lamport

- ❑ 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 :
    - ❑  **$(H(e) < H(e')) \not\Rightarrow (e \rightarrow e')$** .

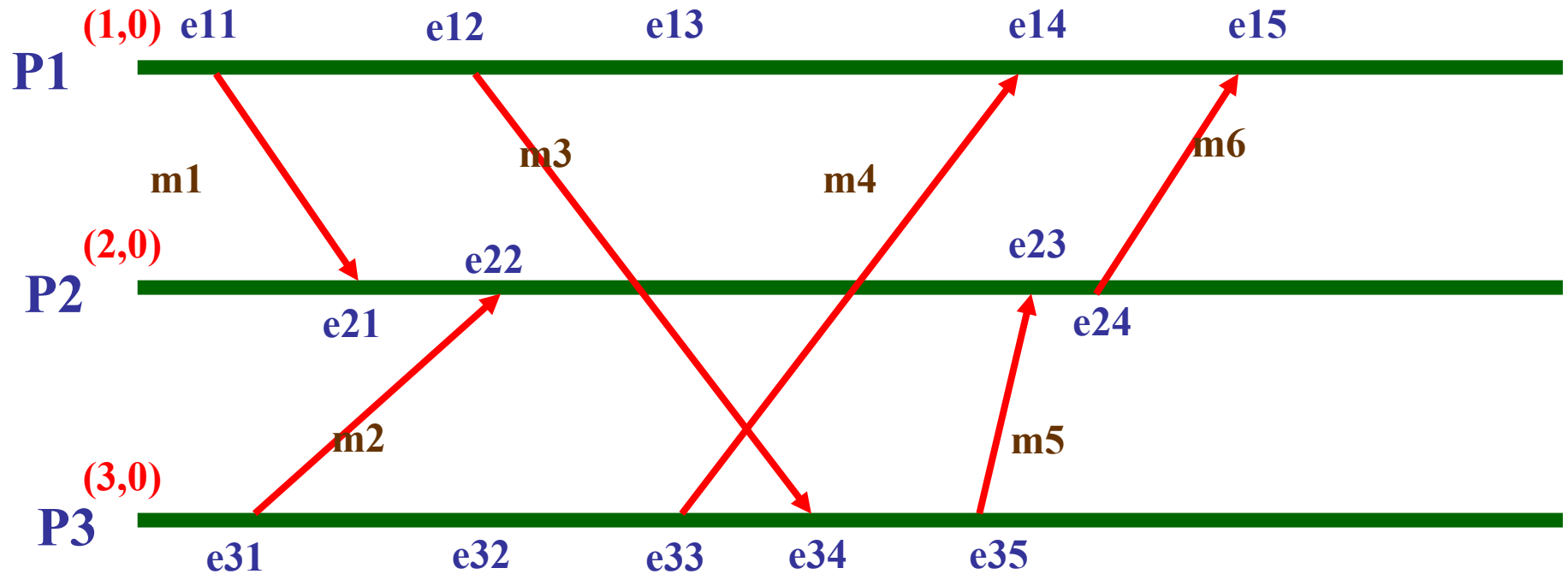
# Horloge de Lamport

□ Exemple 2 : chronogramme avec ajouts des estampilles.



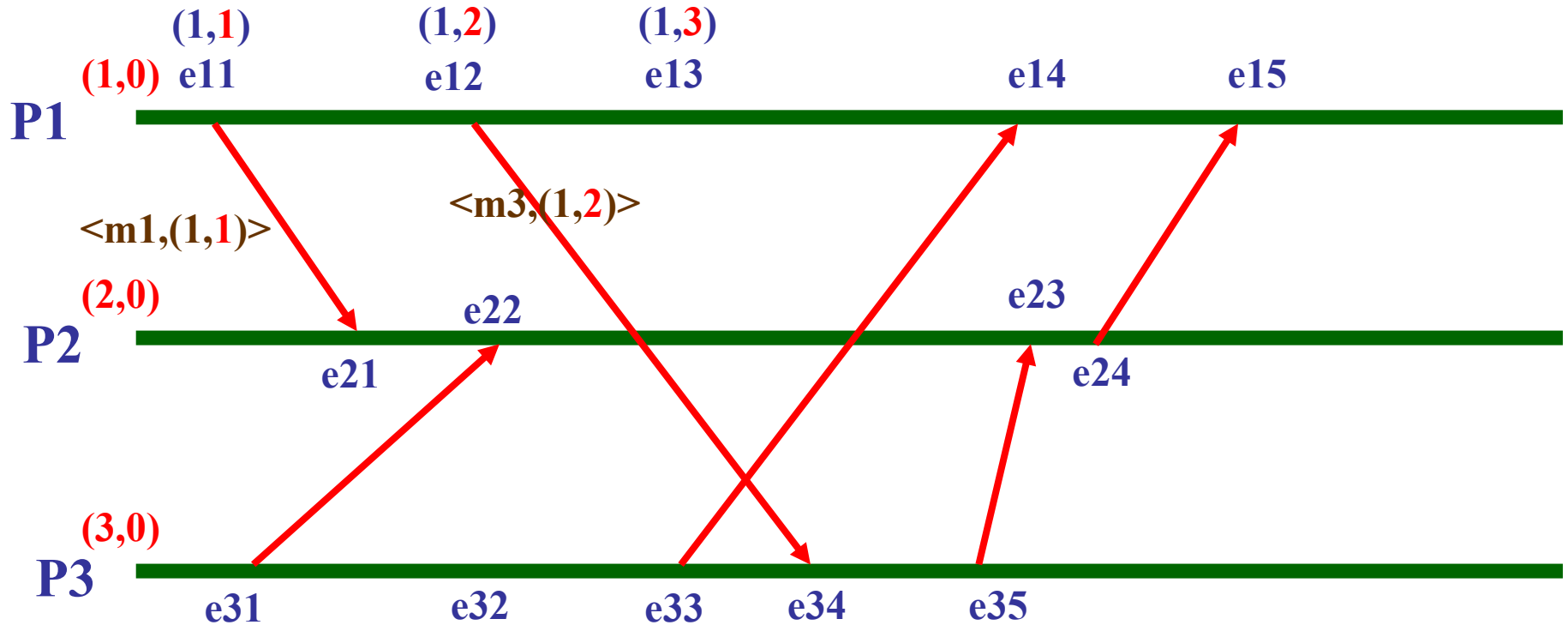
# Horloge de Lamport

❑ Exemple 2 : chronogramme avec ajouts des estampilles.



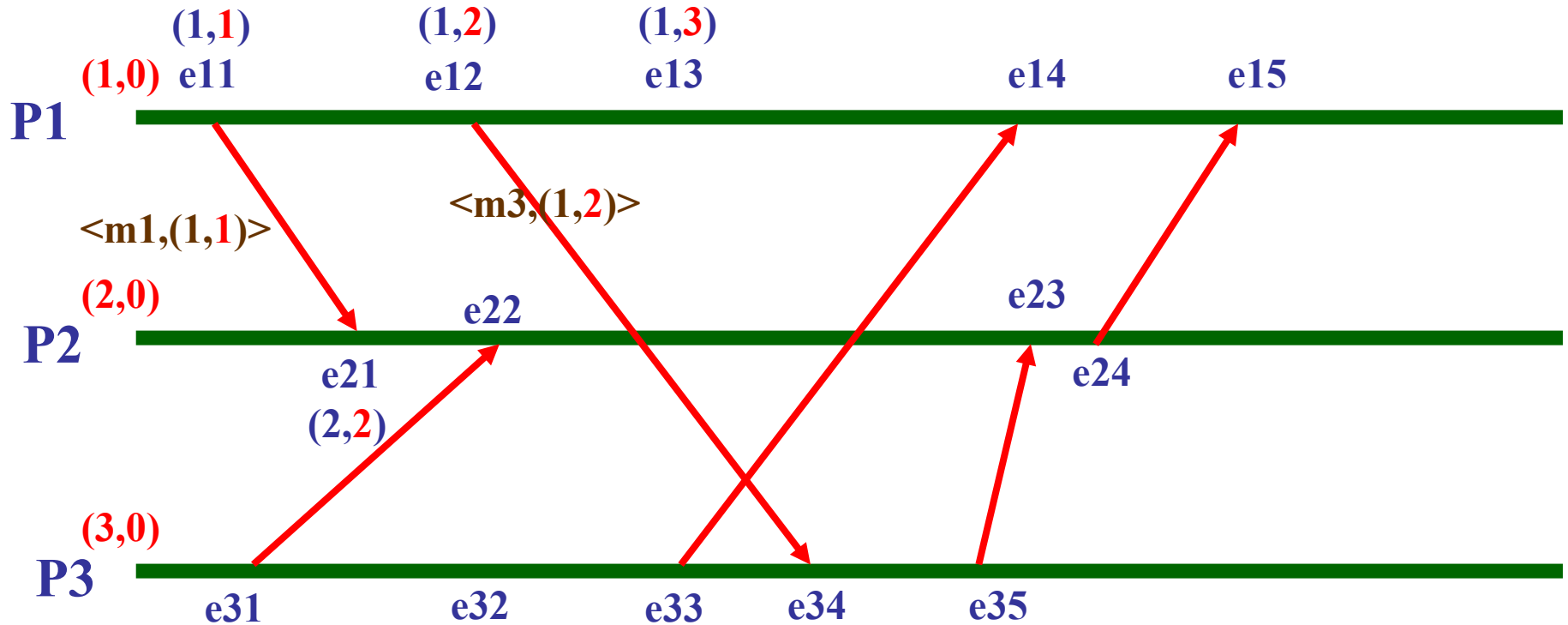
# Horloge de Lamport

❑ Exemple 2 : chronogramme avec ajouts des estampilles.



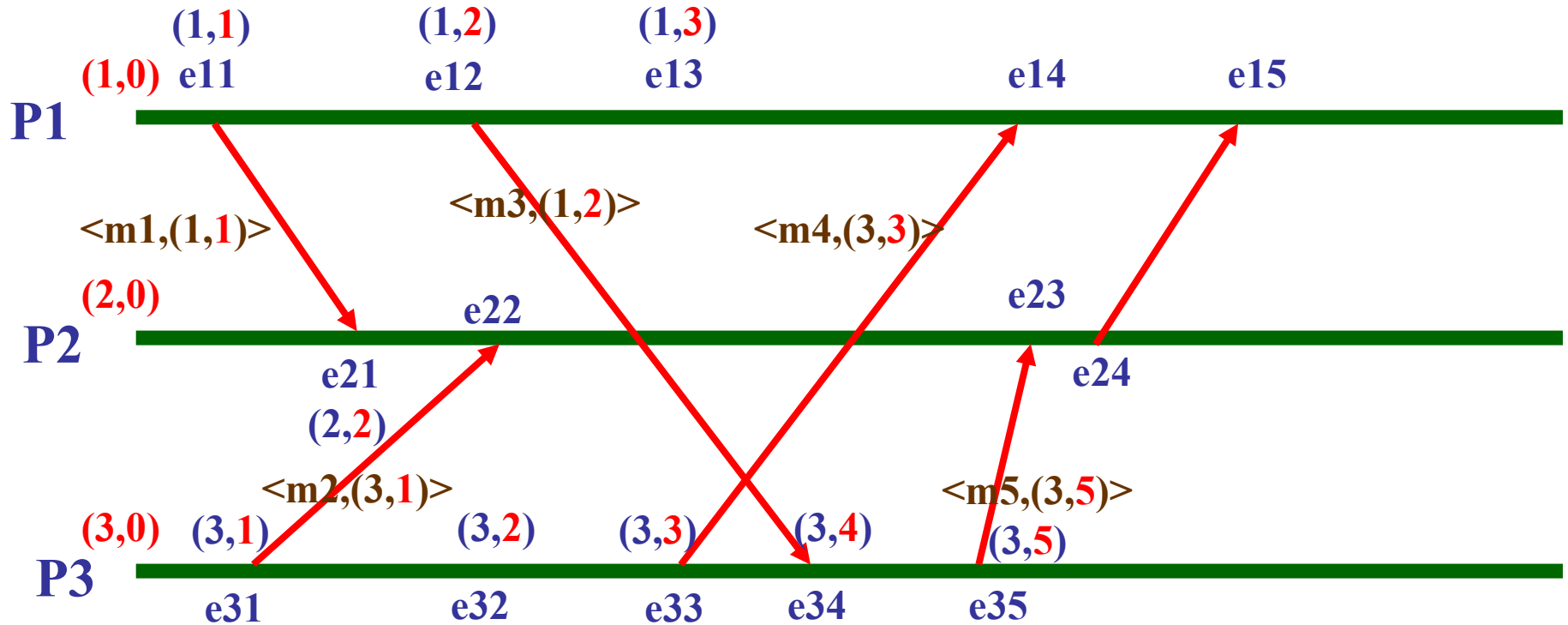
# Horloge de Lamport

❑ Exemple 2 : chronogramme avec ajouts des estampilles.



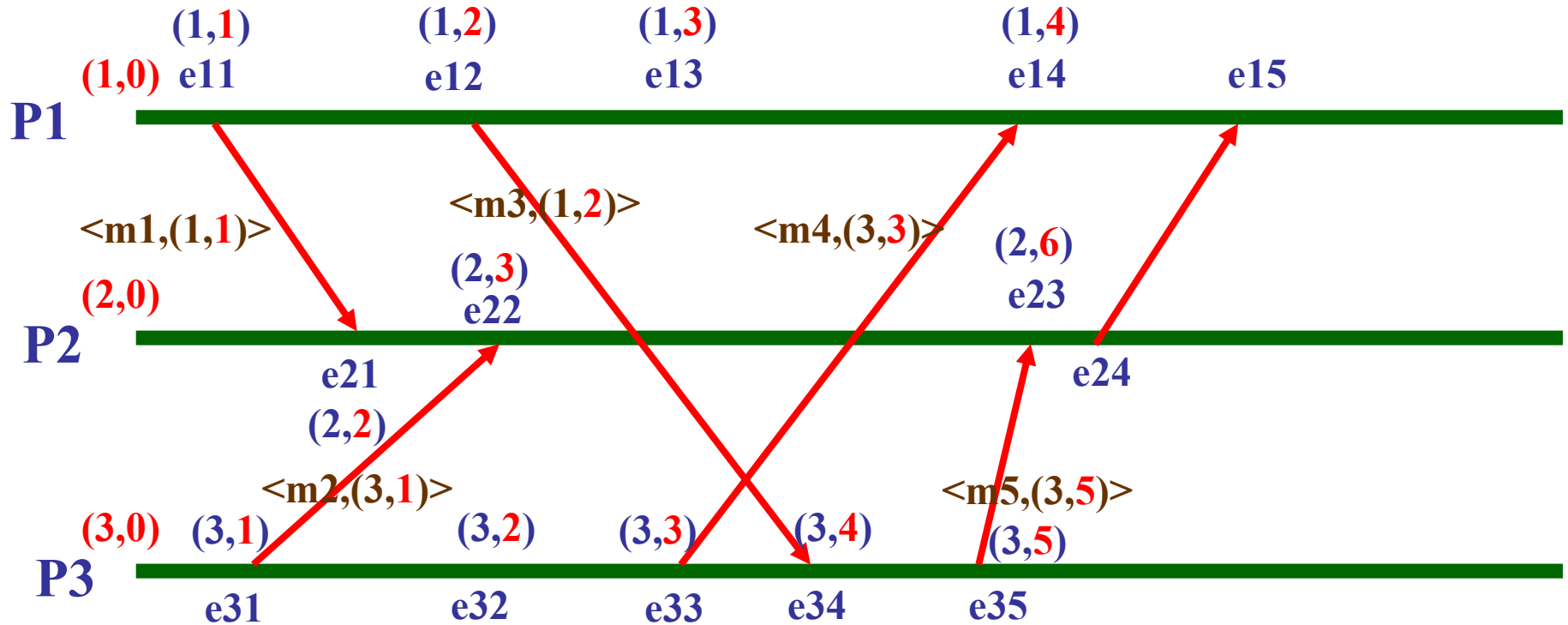
# Horloge de Lamport

❑ Exemple 2 : chronogramme avec ajouts des estampilles.



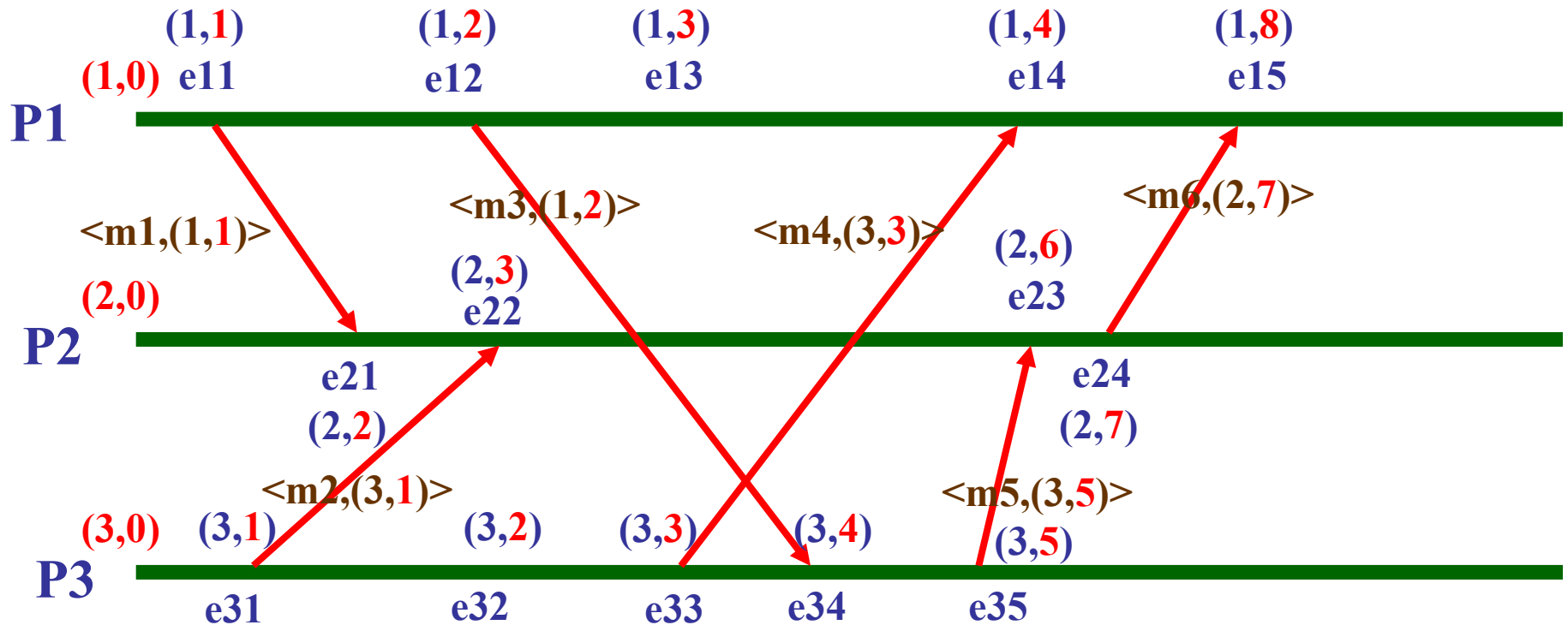
# Horloge de Lamport

❑ Exemple 2 : chronogramme avec ajouts des estampilles.



# Horloge de Lamport

❑ Exemple 2 : chronogramme avec ajouts des estampilles.



- ❑ Pour  $e_{11}, e_{12}, e_{13} \dots$  : incrémentation de +1 de l'horloge locale.
- ❑ Date de  $e_{23}$  : 6 car le message  $m5$  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  $m3$  ( $\max(3,2)+1$ ).



# 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  $P_j$  :

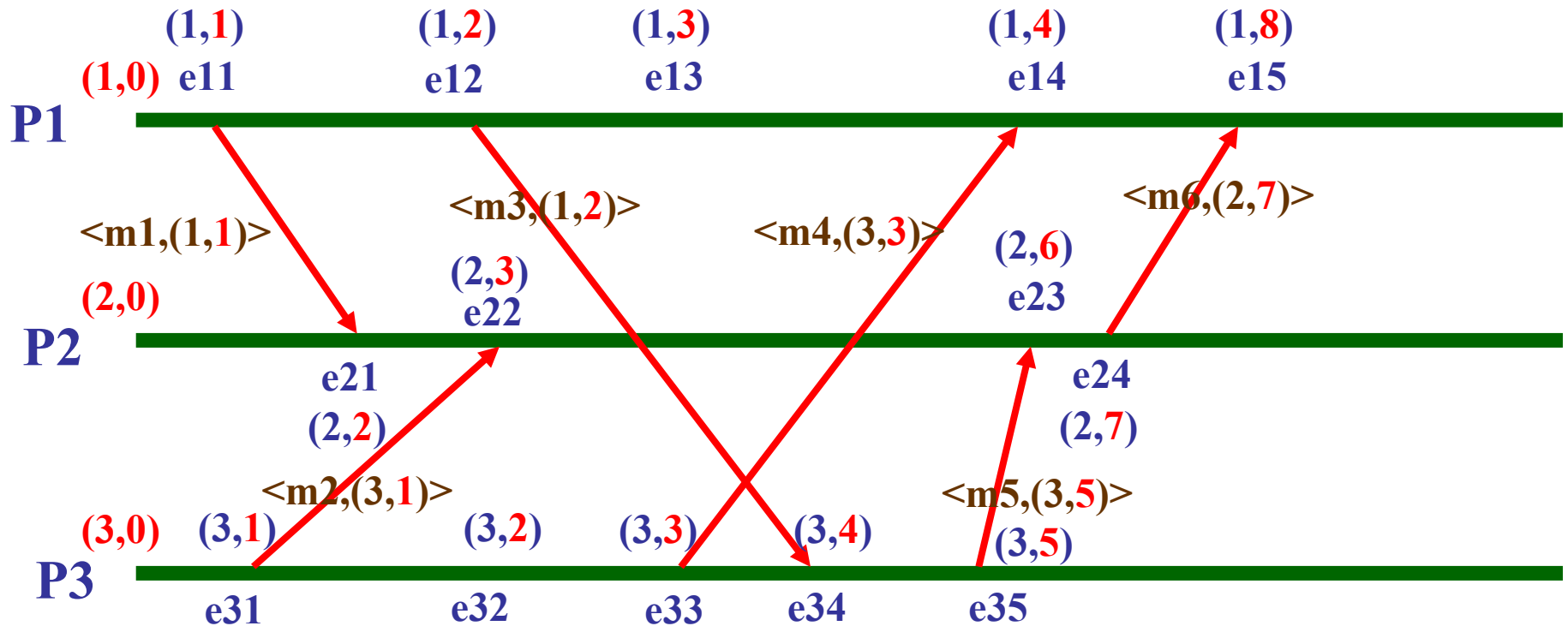
- ❑  $e \ll e' \Leftrightarrow (H_i(e) < H_j(e'))$  ou  $(H_i(e) = H_j(e') \text{ avec } i < j)$ .

- ❑ 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)

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

## **Partie 3 : Exclusion mutuelle basée sur les Horloge de Lamport**

# Algorithme

Chacune des 5 actions suivantes est considérée comme un évènement, On suppose que les files d'attente de requêtes contiennent le message  $(T_0:P_0, Request)$  indiquant que le processus  $P_0$  dispose de la ressource à  $T_0$ , plus petit que tous les autres

- 1 Pour demander une ressource exclusive  $P_i$  envoie le message  $(T_m:P_i, Request)$  à tous les autres processus et mets ce messages dans sa liste d'attente de requêtes.  $T_m$  est égal à la valeur de l'horloge de Lamport du message
- 2 Lorsqu'un processus  $P_j$  reçoit le message  $(T_m:P_i, Request)$  il le met dans sa file d'attente de requêtes et acquitte le message à  $P_i$
- 3 Pour libérer une ressource, le processus  $P_i$  retire tous les messages  $(T_m:P_i Requests)$  de sa file d'attente de requête et envoie un message  $(T_m:P_i, Release)$  à tous les processus.
- 4 Lorsqu'un processus  $P_j$  reçoit un message de  $P_i Release$ , il enlève tous message  $(T_m:P_i Request)$  de sa file d'attente
- 5 Le processus  $P_i$  reçoit un accès exclusif à la ressource lorsque les deux conditions suivantes sont satisfaites :
  - ❑ Il y a un message  $(T_m:P_i Request)$  dans la file d'attente des requêtes de  $P_i$  qui est ordonné avant tous les autres requêtes selon relation de causalité , c'est à dire si  $T_m$  est le plus petit de tous les messages de requête.
  - ❑  $P_i$  a reçu un message de tous les autres processus qui a un temps logique supérieur à  $T_m$ .

