Systèmes distribués : Rappel sur ordre et temps logique

Eric Cariou

Master Technologies de l'Internet 1ère année

Université de Pau et des Pays de l'Adour
Département Informatique

Eric.Cariou@univ-pau.fr

Temps dans un système distribué

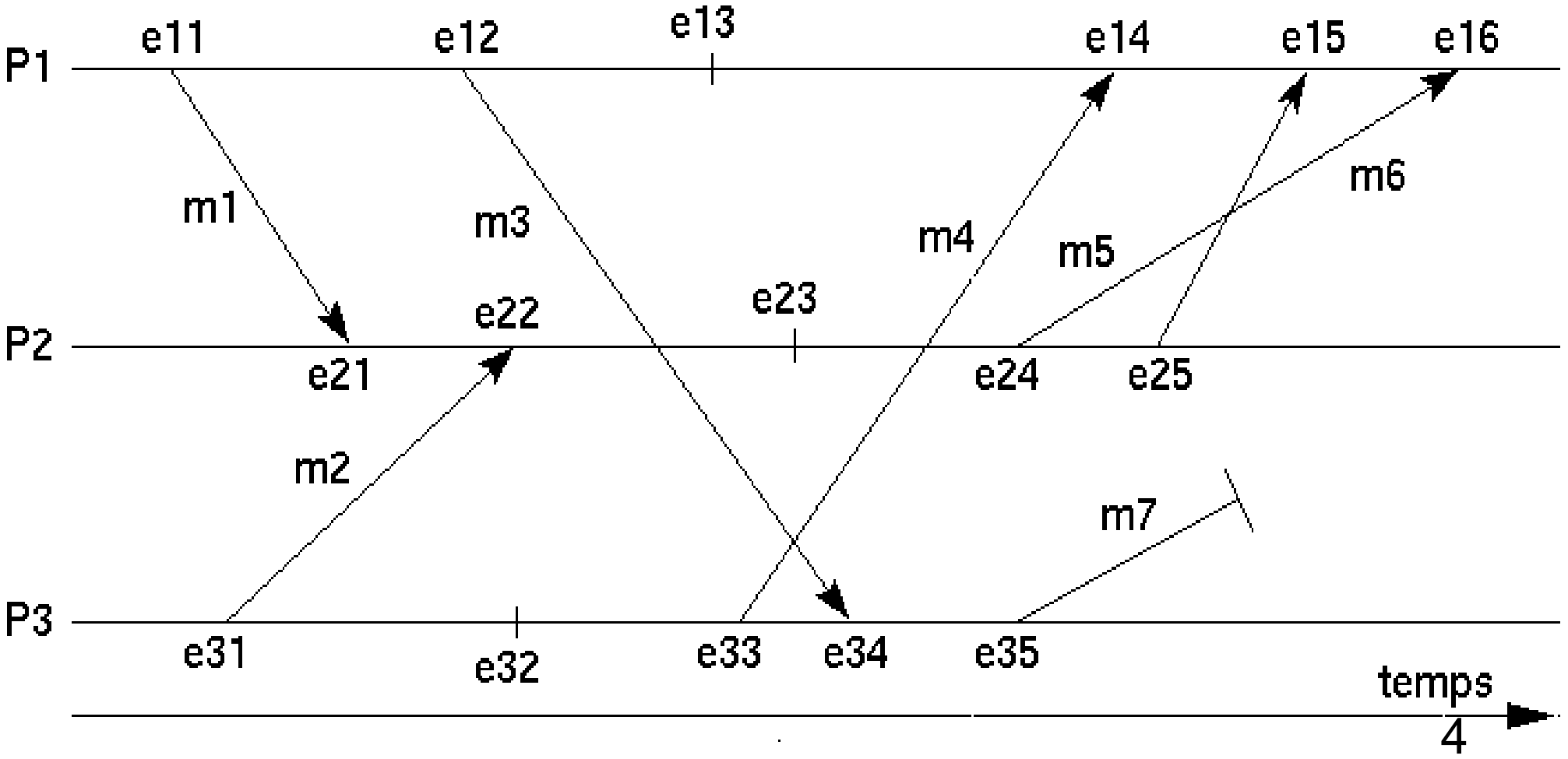
- Définir un temps global cohérent et « identique » (ou presque) pour tous les processus
  - Soit synchroniser au mieux les horloges physiques locales avec une horloge de référence ou entre elles
  - Soit créer un temps logique
- Temps logique
  - Temps qui n'est pas lié à un temps physique
  - But est de 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
    - Création horloge logique

Chronogramme

- Chronogramme
  - 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

- Trois processus tous reliés entre-eux par des canaux
- Temps de propagation des messages quelconques et possibilité de perte de message



Chronogramme

- 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
- Règle de numérotation d'un événement
  - e_{xy} avec x le numéro du processus et y le numéro de l'événement pour le processus, dans l'ordre croissant

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 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 e34 $\rightarrow$ e35) et e11 $\rightarrow$ e13
  - Dépendance causale entre e12 et e32 ?
    - A priori non : absence de dépendance causale
    - Des événements non liés causalement se déroulent en parallèle
- Relation de parallélisme : ||
  - e || e' $\Leftrightarrow \neg((e \rightarrow e') \vee (e' \rightarrow e))$
  - Parallélisme logique : ne signifie pas que les 2 événements se déroulent simultanément mais qu'il peuvent se dérouler dans n'importe quel ordre

Horloges logiques

- Horloges logiques
  - Datation de chacun des événements du système
  - Avec respect des dépendances causales entre evts
- 3 familles d'horloge
  - Estampille (horloge de Lamport)
    - Une donnée par événement : information au niveau global
  - Vectorielle (horloge de Mattern)
    - Un vecteur par événement : information sur chacun des autres processus
  - Matricielle
    - Une matrice par événement : information sur les informations que connaisse chacun des processus des autres processus

Horloge de Lamport

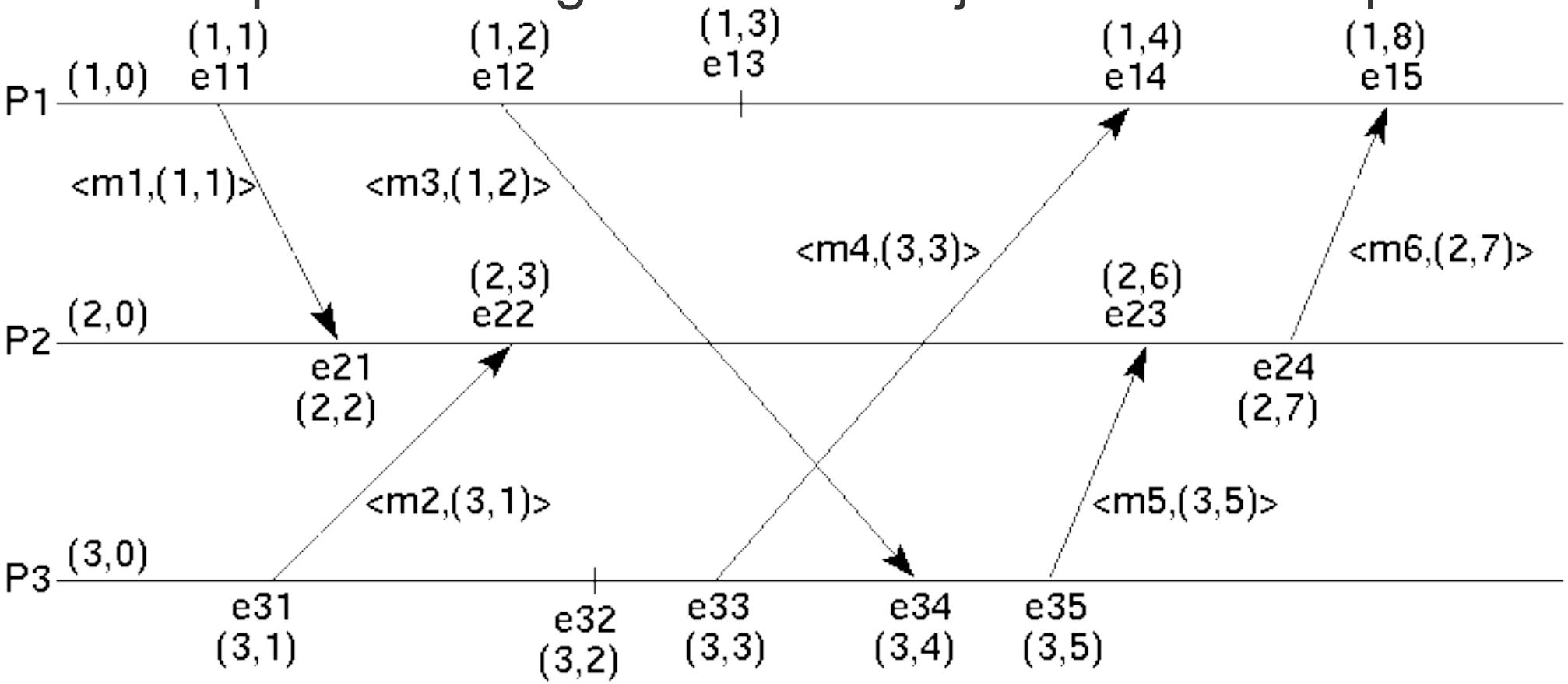
- Lamport, 1978
  - Une date (estampille) est associée à chaque évenement : couple (s, nb)
    - s : numéro du processus
    - nb : numéro d'événement
  - Respect dépendance causale
    - $e \rightarrow e' \Rightarrow H(e) < H(e')$
      - Avec H(s, nb) < H(s' , nb') si (nb < nb') ou (nb = nb' et s < s')
    - Mais pas la réciproque :
      - $H(e) < H(e') \Rightarrow \neg\ (e' \rightarrow e)$
        - C'est-à-dire : soit $e \rightarrow e'$, soit $e \parallel e'$
- Invariant sur les dates
  - Pour deux dates d'un même processus, les numéros de ces dates sont différentes
  - $\forall d,d' : d.s = d'.s \Rightarrow d.nb \neq d'.nb$
    - Il n'y pas deux événements locaux ayant le même numéro

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 incrémente l'horloge locale
    - L'événement est daté localement par Hi
  - Émission d'un message par Pi
    - On incrémente Hi de 1 puis on envoie le message avec (i, Hi) comme estampille
  - Réception d'un message *m* avec estampille (s, nb)
    - Hi = max(Hi, nb) +1 et marque l'événement de réception avec Hi
      - Hi est éventuellement recalée sur l'horloge de l'autre processus avant d'être incrémentée

Horloge de Lamport

- Exemple : chronogramme avec ajouts des estampilles



- Date de e23 : 6 car le message m5 reçu avait une valeur de 5 et l'horloge locale est seulement à 3
- Date de e34 : 4 car on incrémente l'horloge locale vu que sa valeur est supérieure à celle du message m3
- Pour e11, e12, e13 ... : incrémentation de +1 de l'horloge locale

Horloge de Lamport

- Ordonnancement global
  - Via Hi, on ordonne tous les événements du système entre eux
  - Ordre total obtenu est arbitraire
    - Si dépendance causale entre 2 evts : ordre respecte dépendance
    - Si indépendance causale entre : choix d'un ordre entre les 2
      - Pas de problème en pratique puisqu'ils sont indépendants
- Ordre total, noté e << e' : *e* s'est déroulé avant *e'*
  - Soit *e* événement de *Pi* et *e'* événement de *Pj* :
    e << e' $\Leftrightarrow$ (Hi(e) < Hj(e')) v (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
  - Ordre total global obtenu pour l'exemple
    - e11 << e31 << e12<< e21 << e32 << e13 << e22 << e33 << e14 << e34 << e35 << e23 << e24 << e15

Horloge de Mattern

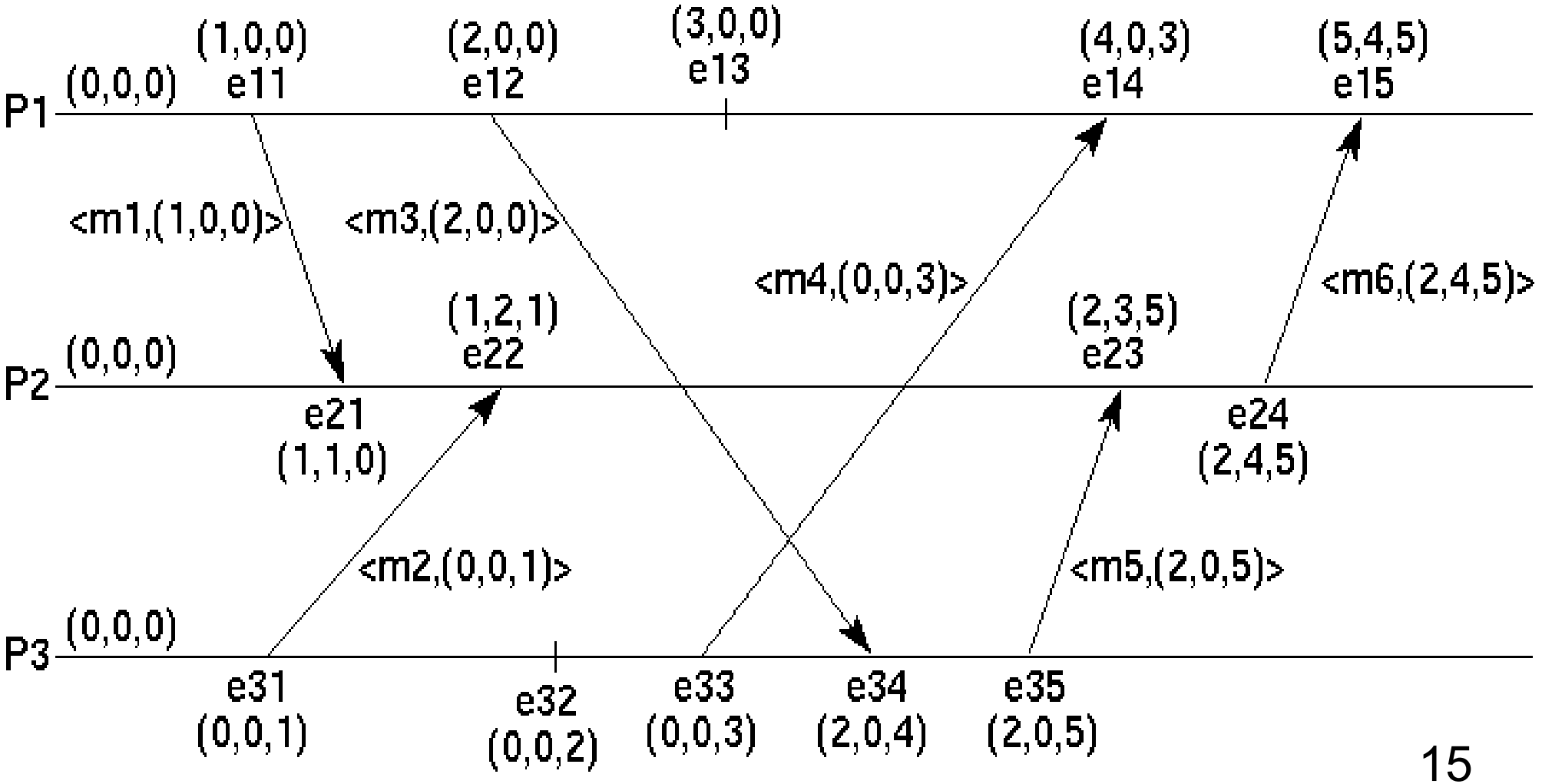
- Horloge de Mattern & Fidge, 1989-91
  - Horloge qui assure la réciproque de la dépendance causale
    - $H(e) < H(e') \Rightarrow e \rightarrow e'$
  - Permet également de savoir si 2 événements sont parallèles (non dépendants causalement)
  - Ne définit par contre pas un ordre total global
- Principe
  - Utilisation de vecteur V de taille égale au nombre de processus
    - Localement, chaque processus *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
    - 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*, on met à jour les cases j ≠ i de son vecteur local *Vi*
    - ∀j : Vi [j] = max (Vm [j], Vi [j])
      - Mémorise le nombre d'événements sur Pj qui sont sur Pj dépendants causalement par rapport à l'émission du message
      - La réception du message est donc aussi dépendante causalement de ces événements sur Pj

Horloge de Mattern

- Exemple : chronogramme d'application horloge de mattern
  - Même exemple que pour horloge de Lamport



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 $\neg(V < V') \wedge \neg(V' < V)$
- 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

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
    - Horloge de Lamport respecte cela également
  - 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
      - L'horloge de Lamport impose un ordre arbitraire entre les événements indépendants causalement

Horloge matricielle

- Horloge matricielle
  - *n* processus : matrice M de (n x n) pour dater chaque evt
  - Sur processus Pi
    - Ligne *i* : informations sur événements de Pi
      - Mi [i , i] : nombre d'événements réalisés par Pi
      - Mi [i , j] : nombre de messages envoyés par Pi à Pj (avec $j \neq i$)
    - Ligne *j* (avec $j \neq i$)
      - Mi [j , k] : nombre de messages que l'on sait que Pj a envoyé à Pk
      - Mi [j , j] : nombre d'événements que l'on connait sur Pj (avec $j \neq k$)
  - Un processus Pi a une connaissance sur le nombre de messages qu'un processus Pj a envoyé à Pk
    - 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

Horloges

- Informations données par les horloges
  - Estampille : connaissance global du nombre d'événements du système
  - Vectorielle: connaissance d'événements sur chacun des autres processus
  - Matricielles : connaissance de la connaissance d'événéments qu'un processus connait sur les autres
- Application, utilité
  - Estampille : ordonnancement global, gestion priorité
  - Vectorielle : validation de la cohérence d'un état global, propriétés sur de la diffusion
  - Matricielle : assurer la délivrance causale de messages entre plusieurs processus
    - Cf TD systèmes distribués L3 Informatique

État Global

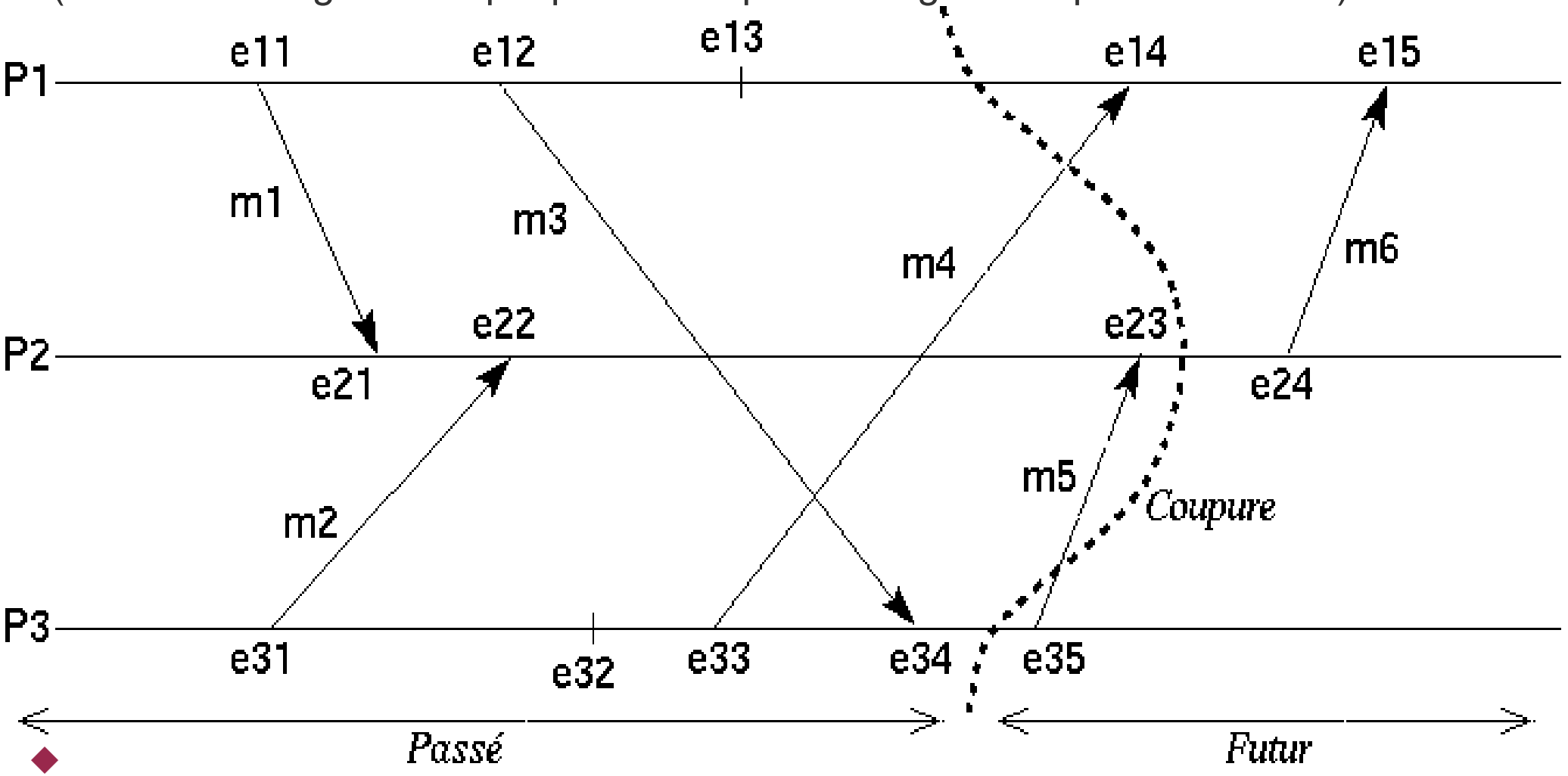
- État global
  - État du système à un instant donné
  - Buts de la recherche d'états globaux
    - Trouver des états cohérents à partir desquels on peut reprendre un calcul distribué en cas de plantage du système
    - Détection de propriétés stables, du respect d'invariants
    - Faciliter le debugging et la mise au point d'applications distribuées
  - 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 coupure
  - 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
- 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 $(e_1, e_2, \ldots e_i, \ldots e_N)$, avec e_i événement du processus Pi tel que
    - $\forall i : \forall e \in C$ et *e* événement du processus Pi $\Rightarrow e \rightarrow e_i$
    - L'état est défini à la frontière de la coupure : l'événement le plus récent pour chaque processus

Coupure

- Exemple de coupure
  (même chronogramme que pour exemples horloges Lamport et Mattern)



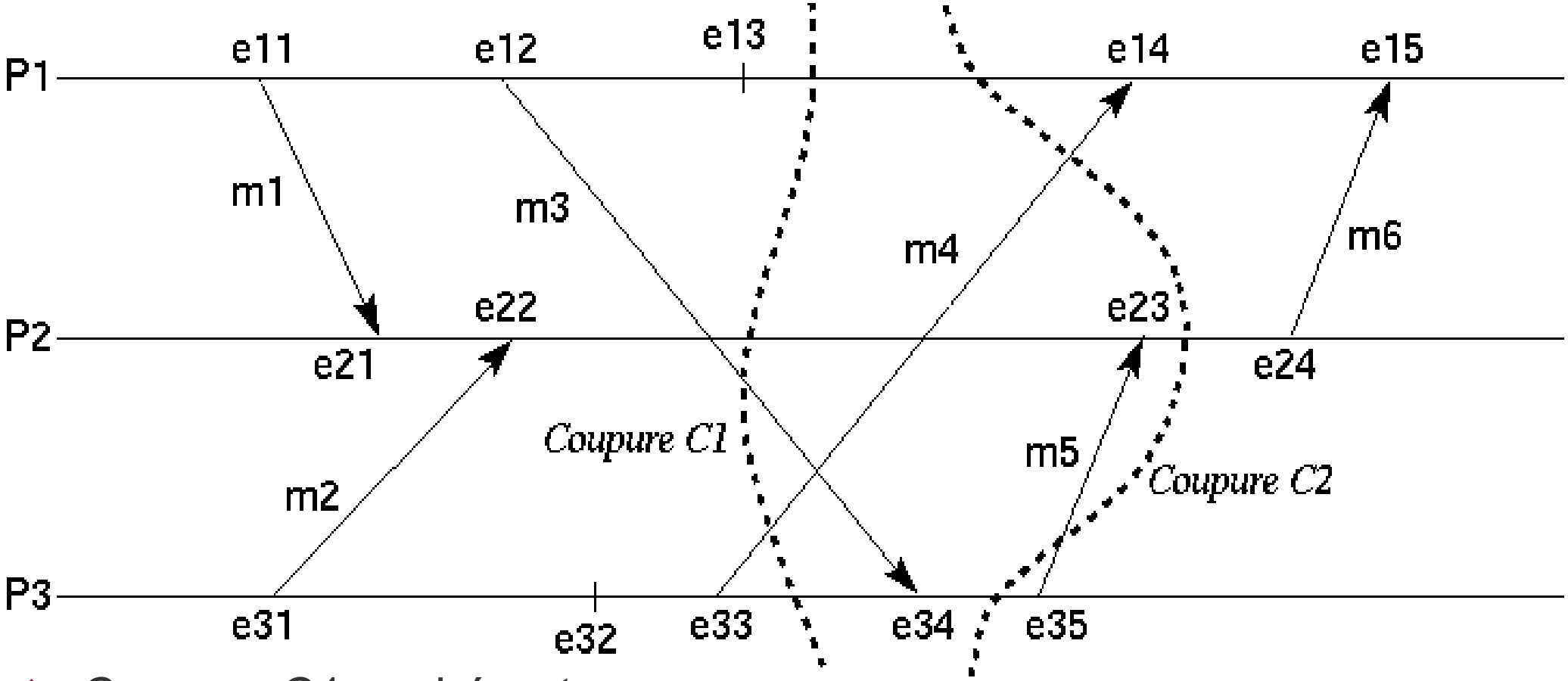
- Coupure = ensemble { e11, e12, e13, e21, e22, e23, e31, e32, e33, e34 }
- État défini par la coupure = (e13, e23, e34)

Coupure/état cohérent

- 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 :
      $a \in C$ 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 cohérente

- Exemple (même chronogramme que précédent)



- Coupure C1 : cohérente
- Coupure C2 : non cohérente car e35 $\rightarrow$ e23 mais e35 $\notin$ C2
  - La réception de *m5* est dans la coupure mais pas son émission
  - *m5* vient du futur par rapport à la coupure

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 Pi, $V(e_i)$ la datation de e_i et V(C) la datation de la coupure
    - $V(C) = \max (V(e_1), \ldots , V(e_N))$:
      $\forall i : V(C)[i] = \max (V(e_1)[i] , \ldots , V(e_N)[i])$
    - 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 $V(C) = (V(e_1)[1], \ldots , V(e_i)[i], \ldots , V(e_N) [N])$
  - Pour un processus Pi, si l'événement e_i est le plus récent c'est lui qui a la date la plus récente pour C : sinon un événement e_j d'un processus Pj ($i \neq j$) s'est déroulé après un événement e_i' de Pi avec e_i' plus récent que e_i
    - $e_i \rightarrow e_i'$ et $e_i' \rightarrow e_j$ avec $e_i \in C$, $e_j \in C$ et $e_i' \notin C$

Datation Coupure

- Datation des coupures de l'exemple
  - Coupure C1 : 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 C2 : 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] ≠ 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)