Chapitre 5 :

Horloges logiques

Temps dans un système distribué

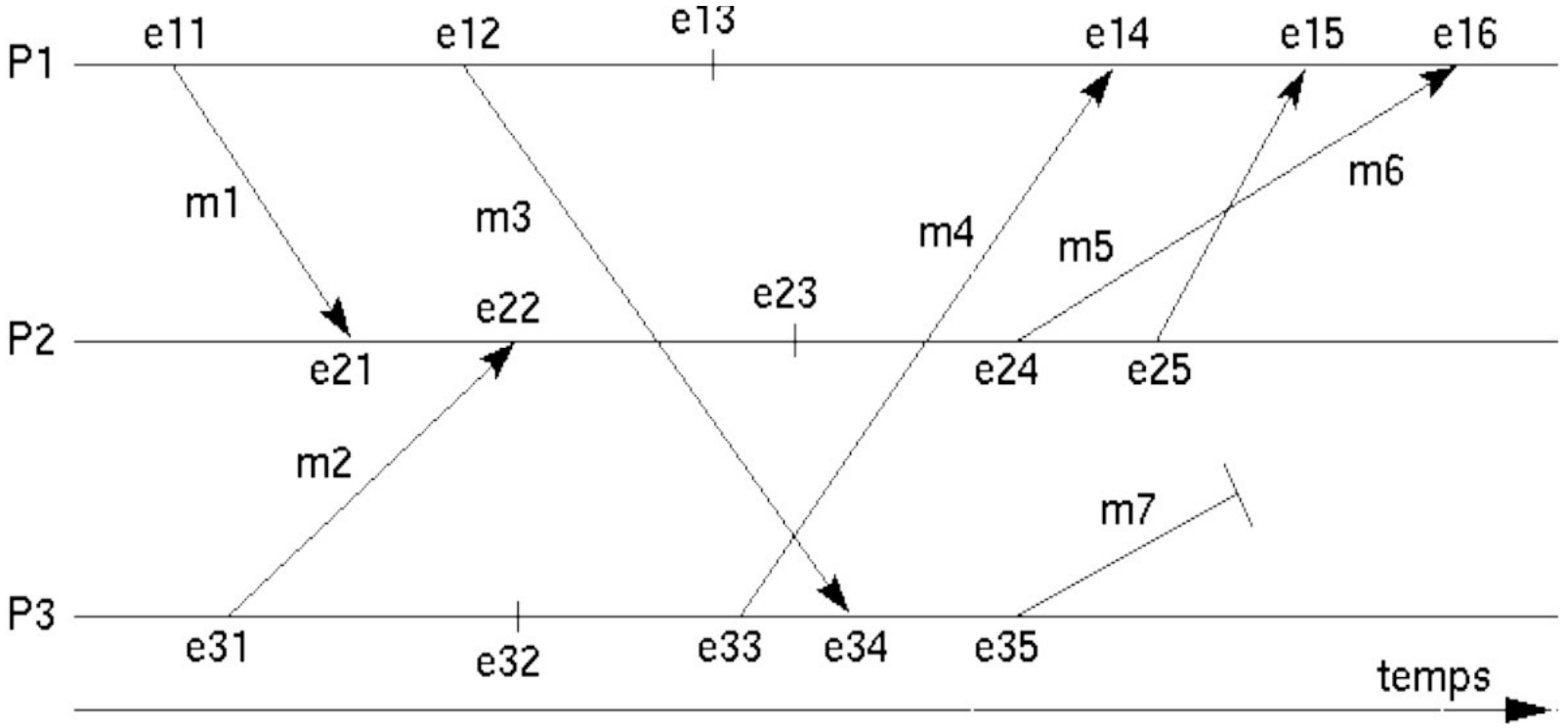
- Définir un temps global cohérent et identique (ou presque) pour tous les processus
  - Soit synchroniser au mieux les horloges physiques locales avec une horloge de référence ou entre elles
  - Soit créer un temps logique
    - Temps qui n'est pas lié à un temps physique
    - But est de pouvoir préciser l'ordonnancement de l'exécution des processus et de leur communication
    - En fonction des événements locaux des processus, des messages envoyés et reçus, on crée un ordonnancement logique
    - Trois familles d'horloge (par estampille, vectorielle et matricielle)

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

- 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
- 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

Relation de dépendance causale

- Il y a une dépendance causale entre 2 événements si un événement doit avoir lieu avant l'autre
- Notation : e → e'
  - *e* doit se dérouler avant *e'*
- Si e → 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 → f et f → e'

Relation de dépendance causale

- Sur exemple précédent
  - Quelques dépendances causales autour de e12
    - Localement : e11 → e12, e12 → e13
    - Sur message : e12 → e34
    - Par transitivité : e12 → e35 (car e34 → e35), e11 → 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' ⇔ ¬((e → e') v (e' → 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

- Les dépendances causales définissent des ordres partiels pour des ensembles d'événements
- Buts d'une horloge logique (selon le type d'horloge)
  - Créer un ordre total global sur tous les événements de tous les processus
  - Déterminer si un événement a eu lieu avant un autre ou s'il n'y a pas de dépendances causales entre eux
  - Vérifier que des propriétés d'ordre sur l'arrivée de messages sont respectées
- Horloge logique
  - Fonction H(e) : associe une date à chaque événement
  - Respect des dépendances causales

Horloges logiques

- Datation de chacun des événements du système
  - Avec respect des dépendances causales entre événements
- Trois 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

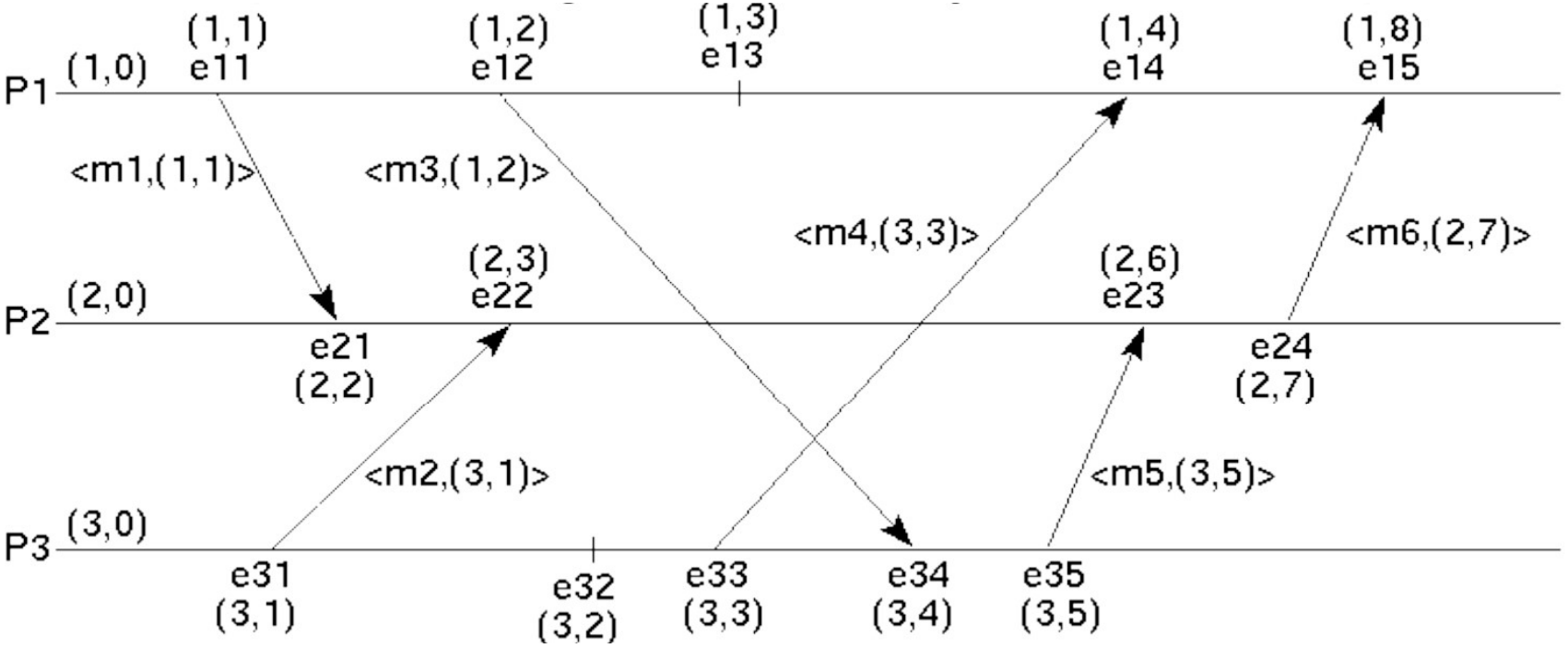
Horloge de Lamport

- Une date (estampille) est associée à chaque événement
  - H(e) = (x, y) « la date de l'événement e est le couple (x, y) »
    - x : numéro du processus
    - y : l'horloge logique locale du processus (valeur entière)
- Respect de la dépendance causale
  - $e \rightarrow e' \Rightarrow H(e) < H(e')$
    - Avec $H(e) < H(e')$ si $(H(e).y < H(e').y)$ ou $((H(e).y = H(e').y \text{ et } H(e').x < H(e').x))$
  - Mais pas la réciproque :
    - $H(e) < H(e') \Rightarrow \neg(e' \rightarrow e)$
      - C'est-à-dire : soit $e \rightarrow e'$, soit $e \parallel e'$
- Unicité des dates
  - Deux événements différents n'ont pas la même estampille
  - $\forall d, d' : d.x = d'.x \Rightarrow d.y \neq d'.y$
    - Il n'y pas deux événements locaux ayant le même numéro
- Deux dates sont toujours ordonnées

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 « la date : (i, Hi) »
  - Émission d'un message par Pi
    - On incrémente Hi de 1
    - L'événement est daté localement par Hi
    - Le message est envoyé avec (i, Hi) comme estampille
  - Réception d'un message *m* avec estampille (x, y)
    - Hi = max(Hi, y) +1 et on date l'événement de réception avec Hi
      - Hi est éventuellement réglée sur l'horloge de l'autre processus avant d'être incrémentée

Horloge de Lamport



- 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 la dépendance
    - Si indépendance causale entre 2 evts : 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' ⇔ (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

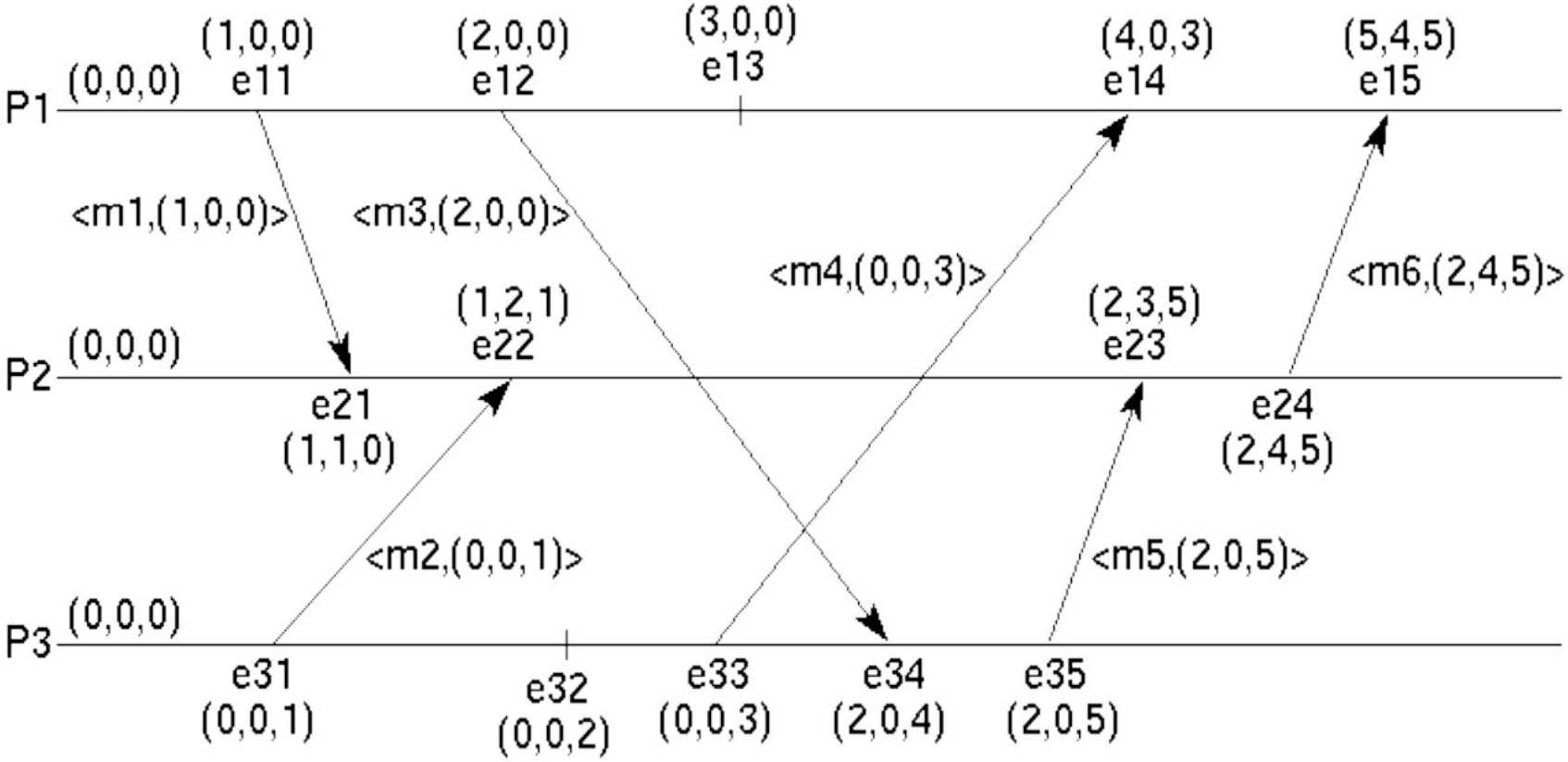
- Horloge de Mattern & Fidge, 1989-91
  - Horloge qui assure la réciproque de la dépendance causale
    - V(e) < V(e') ⇒ e → e‘
  - Permet également de savoir si 2 événements sont parallèles (indépendants causalement)
  - Ne définit par contre pas un ordre total global
- Principe
  - Utilisation de vecteur 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 une valeur 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 → e14
    - V(e14) < V(e15) donc e13 → e15
  - V(e35) = (2,0,5) et V(e23) = (2,3,5)
    - V(e35) < v(e23) donc e35 → 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

Limites

- Limites de l'horloge de Lamport
  - Elle respecte les dépendances causales mais avec *e* et *e′* tel que H(e) < H(e') on ne peut rien dire sur la dépendance entre *e* et *e‘*
    - Dépendance causale directe ou transitive entre *e* et *e′* ? Ou aucune dépendance causale ?
    - Exemple : H(e32) = 2 et H(e13) = 3 mais on a pas e32 → e13
- Limite de l'horloge de Mattern
  - Ne permet pas de définir un ordre global total
  - En cas de nombreux processus, la taille du vecteur peut-être importante et donc des données à transmettre relativement importante

Horloge matricielle

- Horloge matricielle
  - *n* processus : matrice M de (n x n) pour dater chaque événement
  - 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 ≠ i)
    - Ligne *j* (avec j ≠ i)
      - Mi [j , k] : nombre de messages que l'on sait que Pj a envoyé à Pk (j ≠ k)
      - Mi [j , j] : nombre d'événements que l'on connait sur Pj
- 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
    - On peut déterminer si on ne devait pas recevoir un message avant

Horloge matricielle

- 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'événements 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