

Chapitre 2

Ordre et temps logique Eric Cariou (Eric.Cariou@univ-pau.fr)

1. Ordonnement des événements

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

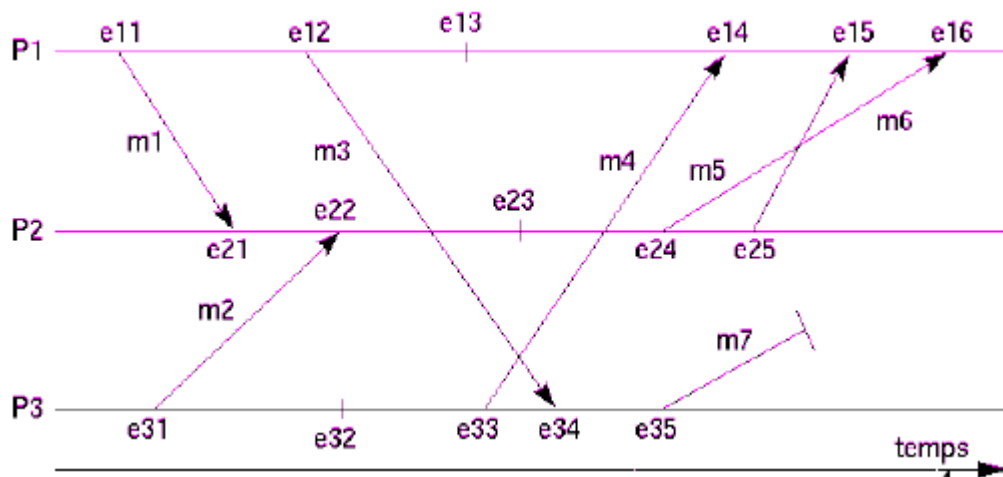
Temps logique

- Temps qui n'est pas lié à un temps physique
- But est de pouvoir préciser l'ordonnement 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 horloge logique

A. Chronogramme :

Décrit l'ordonnement temporel des événements des processus et des échanges de messages

Exemple : Trois processus tous reliés entre-eux par des canaux



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.

B. 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 $f \rightarrow e'$

C. Ordonnement des événements

Les dépendances causales définissent des ordres partiels pour des ensembles d'événements du système

Sur exemple précédent

Quelques dépendances causales autour de e_{12} :

- Localement : $e_{11} \rightarrow e_{12}$, $e_{12} \rightarrow e_{13}$
- Sur message : $e_{12} \rightarrow e_{34}$
- Par transitivité : $e_{12} \rightarrow e_{35}$ (car $e_{34} \rightarrow e_{35}$) et $e_{11} \rightarrow e_{13}$

Dépendance causale entre e_{12} et e_{32} ?

- A priori non : absence de dépendance causale
- Des événements non liés causalement se déroulent en parallèle

D. Relation de parallélisme : \parallel

- $e \parallel e' \Leftrightarrow \neg((e \rightarrow e') \vee (e' \rightarrow e))$
- Parallélisme logique : ne signifie pas que les 2 événements se déroulent simultanément mais qu'ils peuvent se dérouler dans n'importe quel ordre

2. Horloges logiques

Datation de chacun des événements du système avec respect des dépendances causales entre evts

3 familles d'horloge :

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

A. Horloge de Lamport

Une date (estampille) est associée à chaque événement :

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 $(nb = nb' \text{ et } s < s')$

Mais pas la réciproque :

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

C'est-à-dire : soit $e \rightarrow e'$, soit $e \parallel e'$

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 :

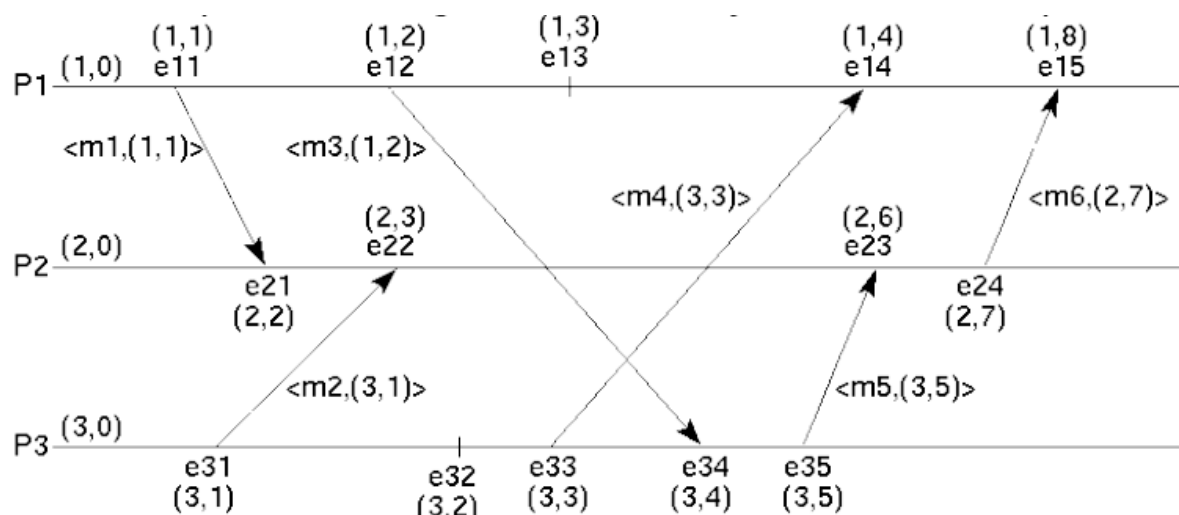
- $H_i = H_i + 1$: on incrémente l'horloge locale
- L'événement est daté localement par H_i

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 (s, nb) : $H_i = \max(H_i, nb) + 1$ et marque l'événement de réception avec H_i

H_i est éventuellement recalée sur l'horloge de l'autre processus avant d'être incrémentée

Exemple : chronogramme avec ajouts des estampilles



- Date de e_{23} : 6 car le message m_5 reçu avait une valeur de 5 et l'horloge locale est seulement à 3
- Date de e_{34} : 4 car on incrémente l'horloge locale vu que sa valeur est supérieure à celle du message m_3
- Pour $e_{11}, e_{12}, e_{13} \dots$: incrémentation de +1 de l'horloge locale

Ordonnancement global

Via H_i , 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 $e \ll e'$: e s'est déroulé avant e' :

- Soit e événement de P_i et e' événement de P_j :
 $e \ll e' \Leftrightarrow (H_i(e) < H_j(e')) \vee (H_i(e) = H_j(e') \text{ avec } i < j)$

Ordre total global obtenu pour l'exemple :

$e_{11} \ll e_{31} \ll e_{12} \ll e_{21} \ll e_{32} \ll e_{13} \ll e_{22} \ll e_{33} \ll e_{14} \ll e_{34} \ll e_{35} \ll e_{23} \ll e_{24} \ll e_{15}$

B. Horloge de 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

Fonctionnement de l'horloge

Initialisation : pour chaque processus P_i , $V_i = (0, \dots, 0)$

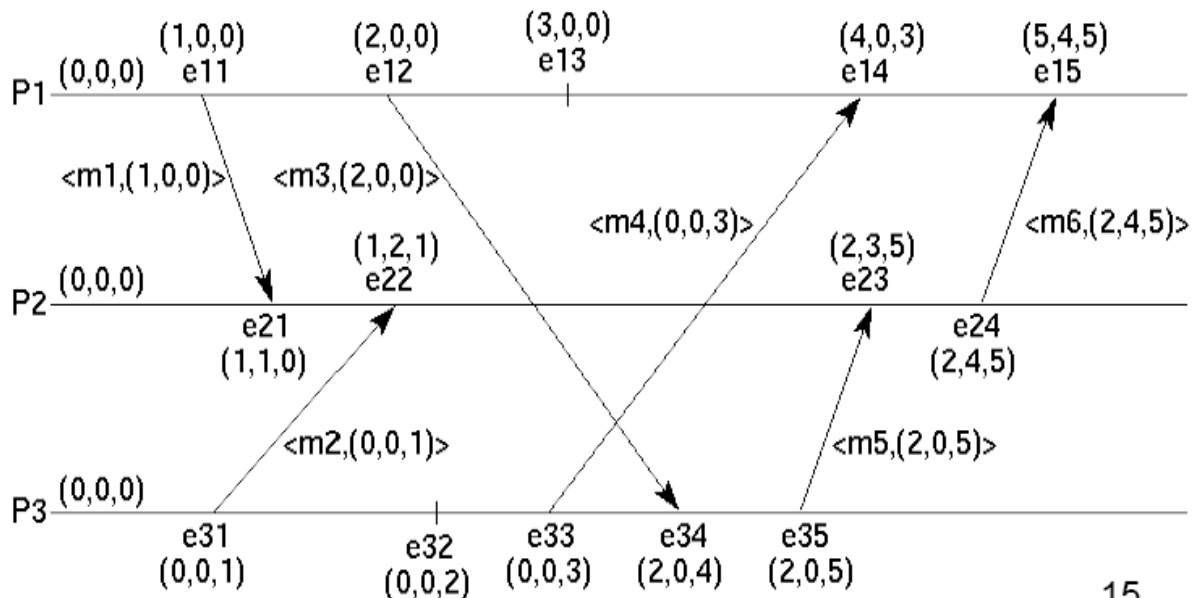
Pour un processus P_i , à chacun de ses événements (local, émission, réception) :

- $V_i[i] = V_i[i] + 1$
- Incrémenter le compteur local d'événement
- Si émission d'un message, alors V_i est envoyé avec le message

Pour un processus P_i , à la réception d'un message m contenant un vecteur V_m , on met à jour les cases $j \neq i$ de son vecteur local V_i : $\forall j : V_i[j] = \max(V_m[j], V_i[j])$

- Mémorise le nombre d'événements sur P_j qui sont sur P_j dépendants causalement par rapport à l'émission du message
- La réception du message est donc aussi dépendante causalement de ces événements sur P_j

Exemple : Même exemple que pour horloge de Lamport



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
- $V(e) \parallel V(e') \Rightarrow e \parallel e'$
S'il n'y a aucun ordre entre les 2 dates, les 2 événements sont indépendants causalement

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 (Horloge de Lamport respecte cela également)

$V(e_{32}) = (0,0,2)$ et $V(e_{13}) = (3, 0, 0)$

- On a ni $V(e_{32}) < V(e_{13})$ ni $V(e_{13}) < V(e_{32})$ donc $e_{32} \parallel e_{13}$
- L'horloge de Mattern respecte les indépendances causales
L'horloge de Lamport impose un ordre arbitraire entre les événements indépendants causalement

C. Horloge matricielle

n processus

Chaque processus i gère une matrice M_i de taille $n \times n$

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 $j \neq i$)
 - $M_i [j , k]$: nombre de messages que l'on sait que P_j a envoyé à P_k
 - $M_i [j , j]$: nombre d'événements que l'on connaît sur P_j (avec $j \neq k$)

Un processus P_i a une connaissance sur le nombre de messages qu'un processus P_j a envoyé à P_k

- 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

D. Conclusion

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 connaît 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

3. Les coupures

A. Définition coupure

Calcul distribué = ensemble E d'événements

Coupure C est un sous-ensemble fini de E tel que :

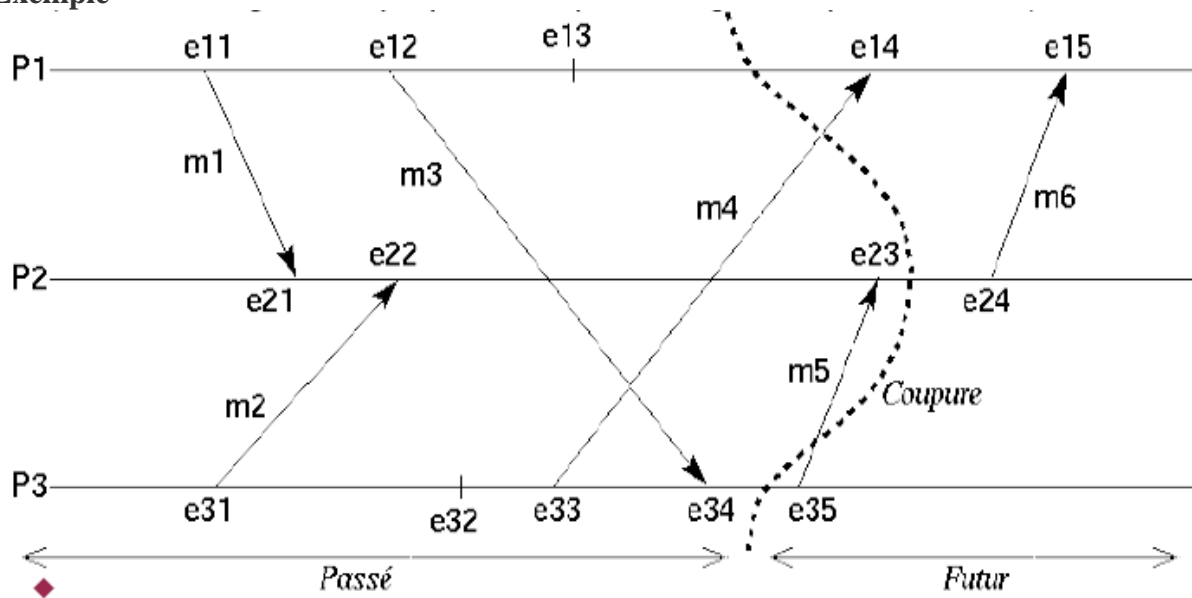
- Soit a et b deux événements du même processus : $a \in C$ et $b \rightarrow a \Rightarrow b \in C$
Si un événement d'un processus appartient à la coupure, alors tous les événements locaux le précédant y appartiennent également

B. 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 : $\forall i : \forall e \in C$ et e 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)

Exemple



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

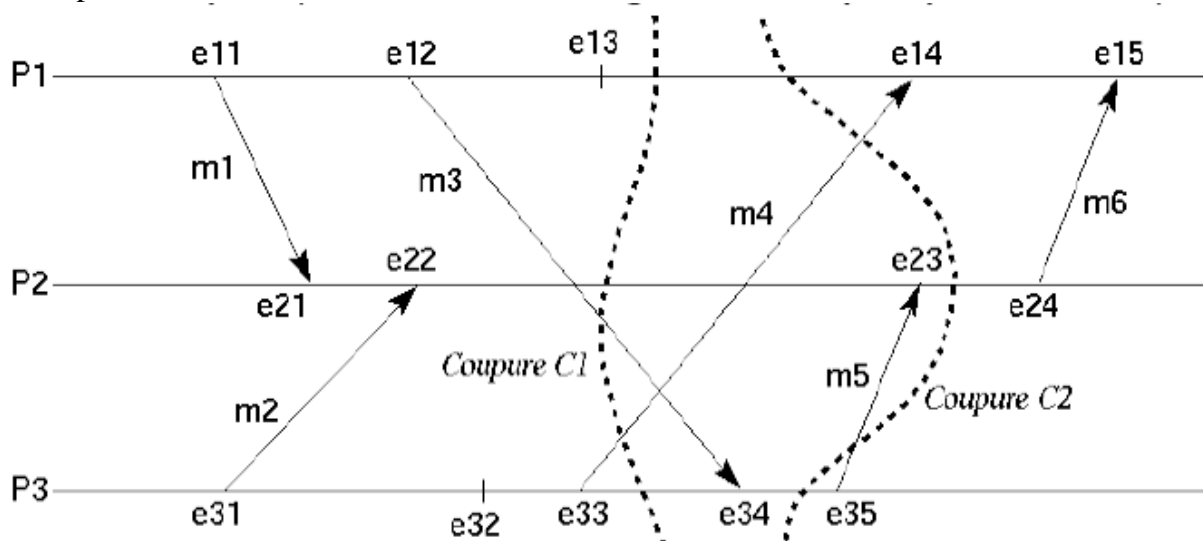
C. Coupure cohérente

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

Exemple



- Coupure C1 : cohérente
- Coupure C2 : non cohérente car $e23 \in C2$ et $e35 \rightarrow e23$ 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

D. 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 $V(C) = (V(e_1)[1] , \dots , V(e_i)[i] , \dots , V(e_N) [N])$
 Pour un processus P_i , 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 P_j ($i \neq j$) s'est déroulé après un événement e_i' de P_i 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 des coupures de l'exemple

Coupure $C1$: état = (e13, e22, e33)

- $V(e13) = (3,0,0)$, $V(e22) = (1,2,1)$, $V(e33) = (0,0,3)$
- $V(C) = (\max(\mathbf{3},1,0), \max(0,\mathbf{2},0), \max(0,1,\mathbf{3})) = (3,2,3)$
- 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)

- $V(e13) = (3,0,0)$, $V(e23) = (2,3,5)$, $V(e34) = (2,0,4)$
- $V(C) = (\max(\mathbf{3},2,2), \max(0,\mathbf{3},0), \max(0,\mathbf{5},4))$
- 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)