

*Ordre, temps, état  
dans les systèmes répartis*

*Michel RIVEILL*

*Projet SIRAC  
INRIA Rhône-Alpes  
655, avenue de l'Europe  
38330 Montbonnot*

*D'après une première série de transparents réalisée par  
Sacha Krakowiak*

# *Etat global et ordre des événements*

---

## ■ *Systèmes répartis à évolution asynchrone*

- *pas de mémoire commune (communication par messages)*
- *pas d'horloge commune*

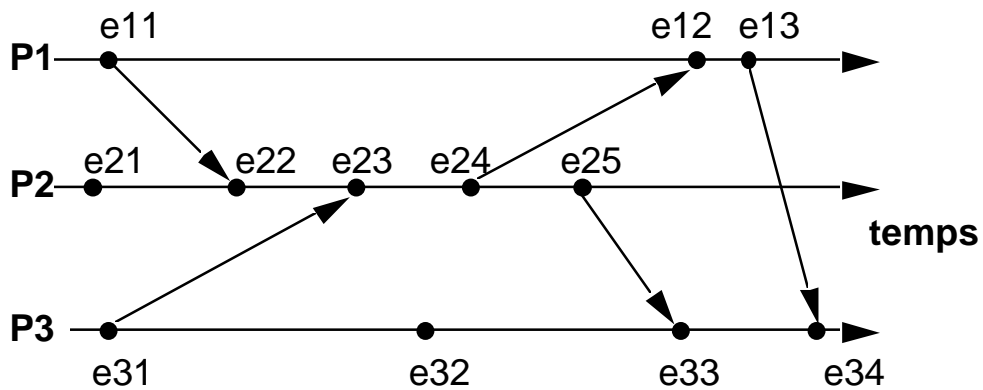
## ■ *Propriétés du système de communication*

- *temps de communication potentiellement longs par rapport aux temps de transition locaux*
- *temps de transmission variables*
- *(éventuellement) non-préservation de l'ordre des messages*

## ■ *Conséquences*

- *perception différente des mêmes événements depuis des sites distincts*
- *impossibilité de définir simplement un "état global instantané"*
- *conséquences sur*
  - *la synchronisation*
  - *l'observation et la mise au point*
  - *la tolérance aux fautes*

# Un modèle pour l'étude de la synchronisation dans un système réparti



*Système = ensemble de processus (1 par site) + canaux de communication*

*Processus = séquence d'événements locaux + mémoire locale + horloge locale*

*Événement = changement d'état du processus (événement interne), ou émission d'un message, ou réception d'un message*

*Problèmes fondamentaux :*

définir un état global

définir un temps global

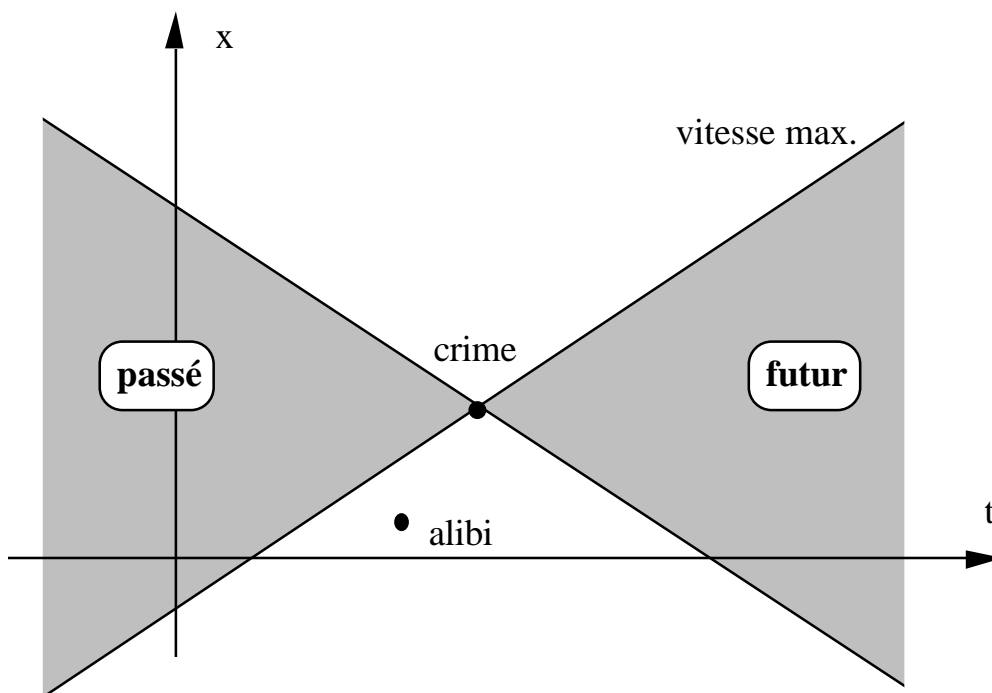
*déterminables localement par tout processus*

*temps global = temps virtuel (ordre sur les événements)*

*tout ordre doit être compatible avec la causalité*

# Illustration de la dépendance causale

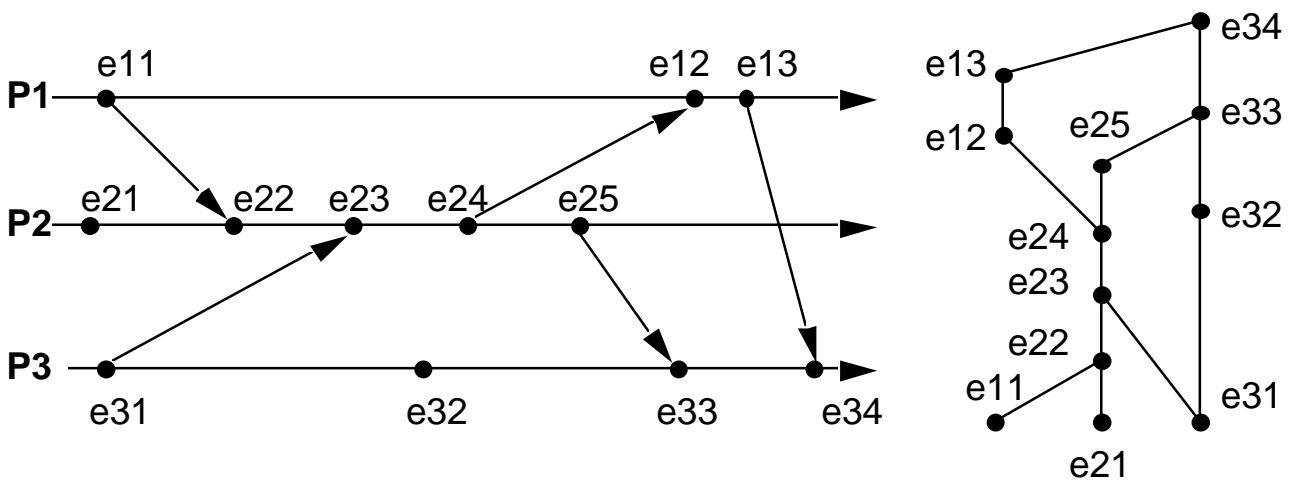
- *Relation dans l'espace-temps*
- *Exemple (à 1 dimension d'espace)*



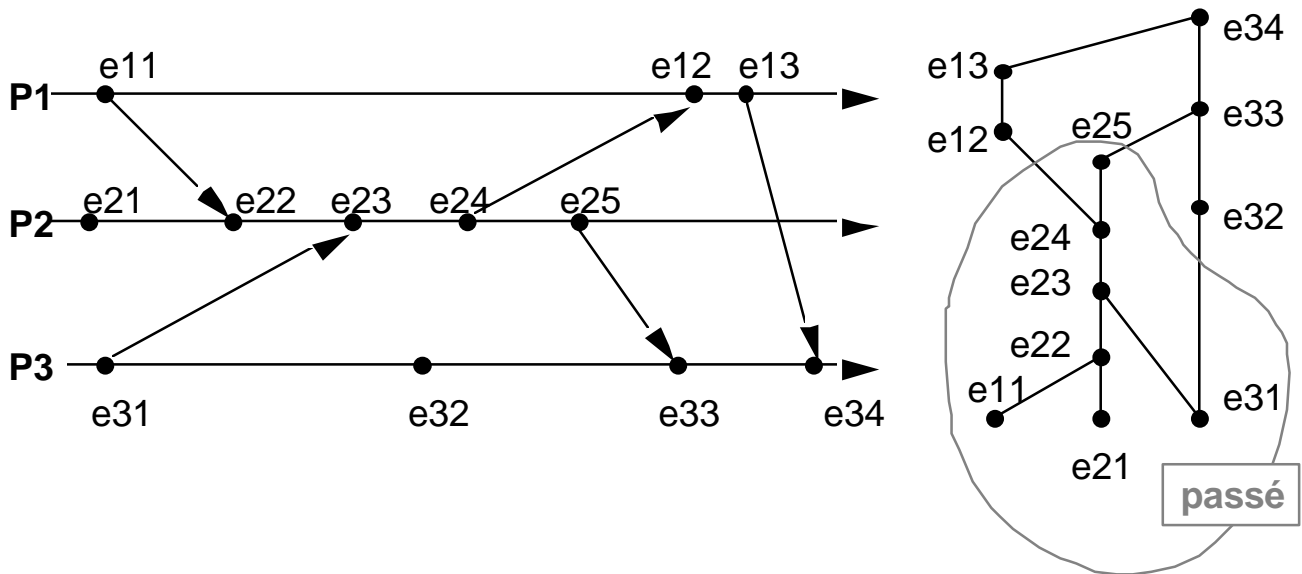
- *Dans un système réparti :*
  - *Interactions locales à un site (mémoire commune)*
  - *Interactions entre sites (messages)*

# Relation de dépendance causale

- **Ordre partiel (noté  $\hat{f}_i$ ) sur l'ensemble des événements du système, tel que :**
  - $a \hat{f}_i b \iff$  l'une des conditions suivantes est vraie :
    - $a$  et  $b$  sont des événements d'un même processus, et  $a$  précède localement  $b$
    - $a$  est l'émission d'un message  $m$  et  $b$  la réception de  $m$
    - il existe un événement  $c$  tel que  $a \hat{f}_i c$  et  $c \hat{f}_i b$
- **La dépendance causale est une dépendance potentielle (si  $a \hat{f}_i b$ ,  $a$  est susceptible d'influencer  $b$ )**
- **La relation de dépendance causale est invariante par toute transformation des temps locaux qui préserve l'ordre local et respecte la causalité pour les messages**



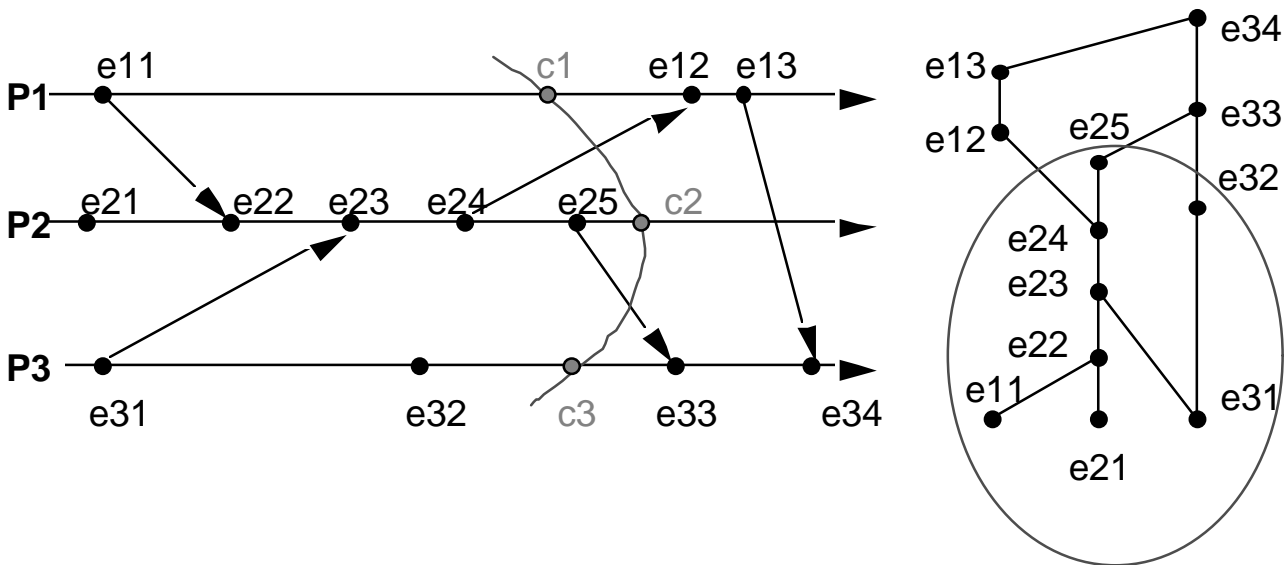
## Relation de dépendance causale (suite)



- **Passé (ou historique) d'un événement  $e$  (noté  $hist(e)$ )** :  $e$  " ensemble des événements  $e'$  tels que  $e' \text{ fi } e$ . Seul le passé strict de  $e$  peut influencer  $e$ .
- **Chaîne causale** :  $e_0, \dots, e_n$  tels que  $e_{i-1} \text{ fi } e_i, i=1, \dots, n$
- **Événements concurrents (causalement indépendants)**
  - $a \parallel b \Leftrightarrow \neg(a \rightarrow b) \text{ et } \neg(b \rightarrow a)$
  - aucun des 2 événements n'appartient au passé de l'autre
  - aucun des 2 événements ne peut influencer sur l'autre
- **Applications**
  - définition de la cohérence
  - tolérance aux fautes
  - observation et mise au point
  - mesure du parallélisme

# Etat global et coupures

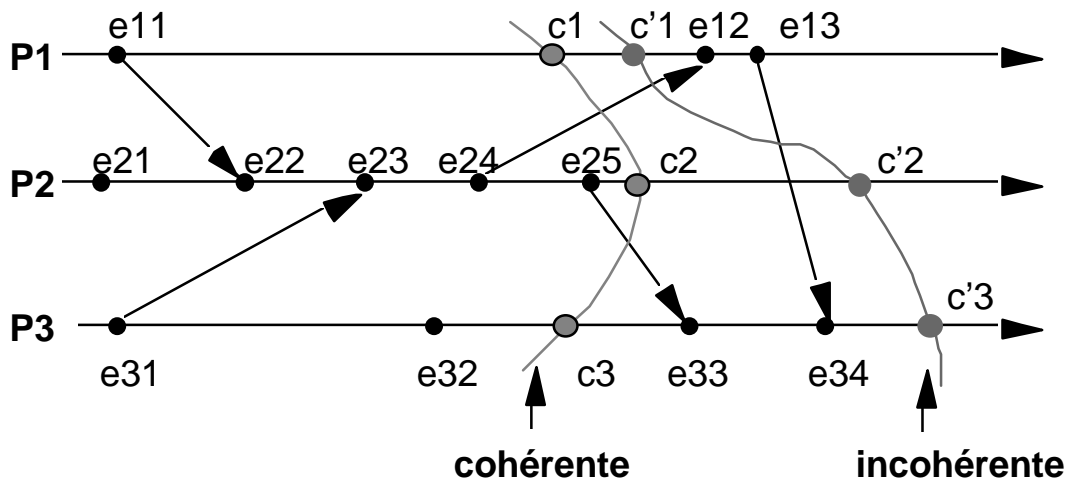
- *Calcul distribué = ensemble d'événements  $E$*
- *Coupure = sous-ensemble fini  $C$  de  $E$ , tel que*
  - *$a \in C$  et  $(b$  précède localement  $a) \Rightarrow (b \in C)$*



- *Coupure = “photographie” instantanée de l'état d'un système : ensemble d'événements (virtuels), un par processus ; permet de définir un passé et un futur (par rapport à la coupure)*
- *Relation d'ordre (partiel) sur les coupures, par inclusion des passés*
- *Etat associé à une coupure  $C$  = ensemble d'états locaux  $s_i$  (un par processus) tel que l'effet local de tout événement de  $C$  soit pris en compte dans  $s_i$*

# Coupures cohérentes

- *Cohérence = respect de la causalité (un message ne peut pas venir du futur)*



- *Coupure cohérente = coupure fermée par la relation de dépendance causale*  

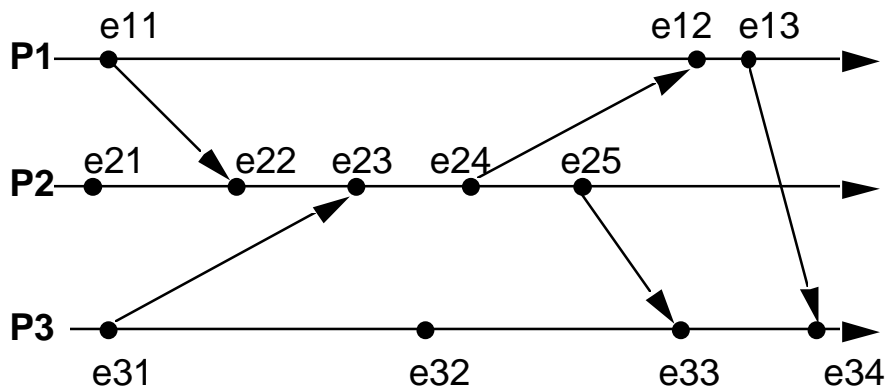
$$a \prec C \text{ et } (b \text{ fi } a) \vdash (b \prec C)$$
- *Etat global cohérent = état associé à une coupure cohérente*
- *Exemple de coupure cohérente : le passé d'un événement*
- *Les coupures cohérentes permettent de définir un ordre de précédence entre états globaux cohérents*

# Calculs et observations distribués

- **Observation d'un calcul réparti  $E$  = "linéarisation" de  $E$  (définition d'un ordre total, soit  $\ll$ , sur les événements de  $E$ )**
- **Exemple : vue de la séquence des événements de  $E$  par un observateur interne ou externe au système**
- **Observation valide = compatible avec la relation de précedence causale (pourrait être observée par un observateur externe réel)**

$$e, e' \in E : e \rightarrow e' \Rightarrow e \ll e'$$

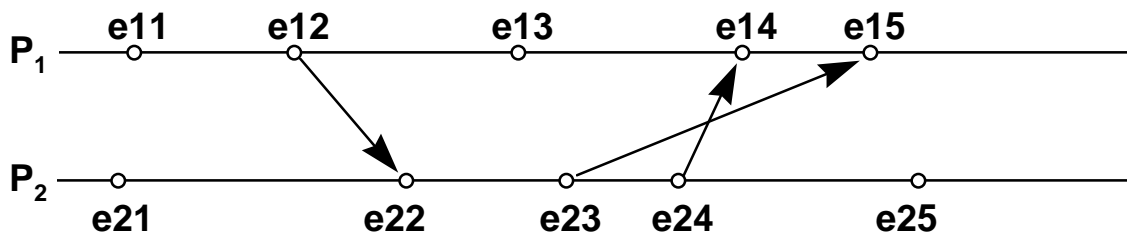
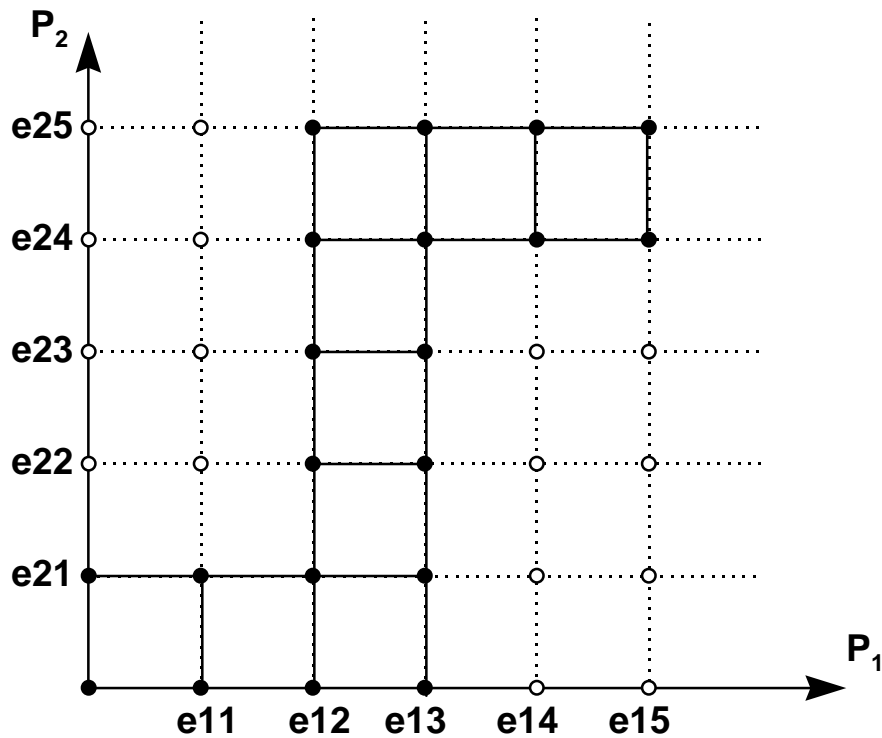
- **Exemple :**
  - $e11 e21 e31 e22 e23 e32 e24 e25 e12 e33 e13 e34$   
valide
  - $e11 e21 e31 e22 \underline{e32 e23} e24 e25 e12 e33 e13 e34$   
valide
  - $e11 e21 e31 e22 e23 e32 e24 e25 e12 e33 \underline{e34 e13}$   
invalide



# Observations et coupures

■ *L'ensemble des coupures cohérentes est un treillis*

- *tout couple  $C1, C2$  a un inf  $C1 \cap C2$  et un sup  $C1 \cup C2$*



■ *Observation valide :*

- *chemin dans le treillis, sans retour en arrière*

# Condition de validité d'une observation

---

- *On construit une observation au moyen d'un processus extérieur  $p_0$  (observateur) qui reçoit notification de tous les événements du système.*

*Hypothèses :*

- *chaque événement  $e$  est daté par une horloge logique. Soit  $H(e)$  cette date.*
- *on connaît une borne sup.  $\delta$  sur le temps de transmission des messages*

- *Une observation (construite au temps  $T$ ) est la suite des événements reçus par  $p_0$  au temps  $T-\delta$ , ordonnés par leurs dates.*

- *l'observation est valide ssi :*

$$e \text{ fi } e' \supset H(e) < H(e')$$

*(il n'y a pas de messages en transit)*

**N.B.**  $H(e) < H(e') \supset \neg (e' \text{ fi } e)$   
*donc, ou bien  $e \text{ fi } e'$ , ou bien  $e \parallel e'$*

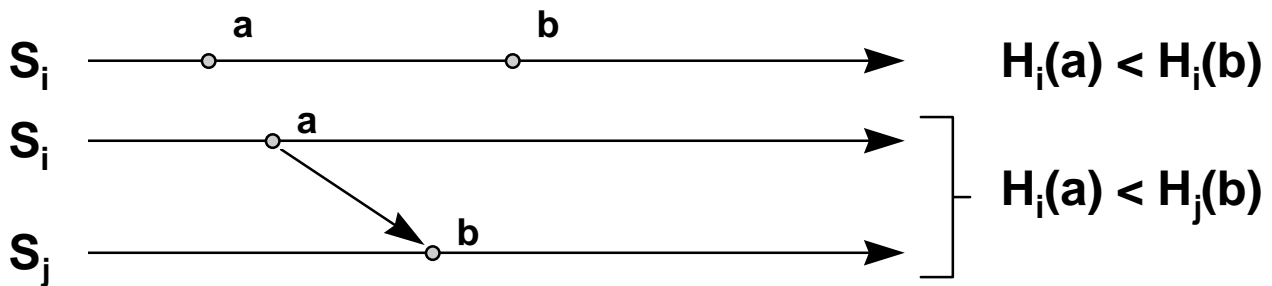
# Réalisation des horloges logiques (Lamport, 1978)

**Objectif : réaliser une datation des événements**

- assurant la condition de validité
- déterminable par consultation locale

■ Un compteur entier  $H_i$  est maintenu sur chaque site  $S_i$ .

■ Un événement  $e$  sur le site  $i$  est daté par  $H(e) = H_i$



■ Initialisation :  $H_i = 0$ , pour tout  $i$

■ A chaque événement local sur site  $i$  :

- $H_i := H_i + 1$
- $e$  est daté par  $H_i$

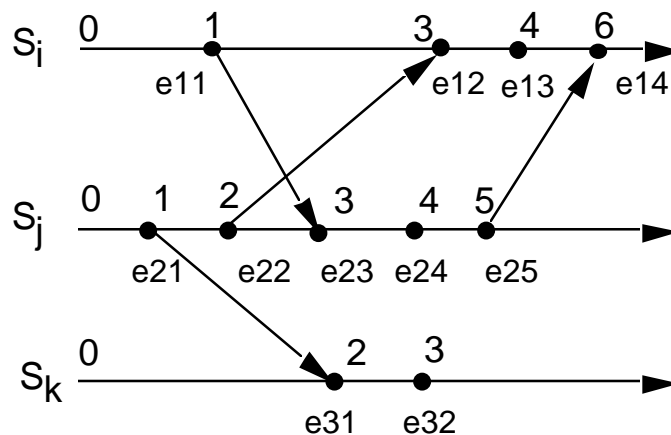
■ Chaque message est estampillé par la date de son émission :  $(m, E(m))$ , avec  $E(m) = H(\text{émission})$

■ A la réception d'un message  $(m, E(m))$  sur le site  $i$  :

- $H_i := \max(H, E(m)) + 1$

# Réalisation des horloges logiques (suite)

## ■ Exemple



## ■ Réalisation d'un ordre total ( $\ll$ )

Soit  $a$  sur  $S_i$ ,  $b$  sur  $S_j$

$$a \ll b \Leftrightarrow (H(a) < H(b)) \text{ ou } (H(a) = H(b) \text{ et } i < j)$$

L'ordre sur les sites est arbitraire

## ■ On dispose donc d'un ordre total sur les événements, détectable à partir d'informations locales uniquement

observation :  $e11 \ e21 \ e22 \ e31 \ e12 \ e23 \ e32$

$e13 \ e24 \ e25 \ e14$

## ■ Il ne doit pas y avoir de communications "cachées" (via d'autres canaux que ceux qui estampillent les messages)

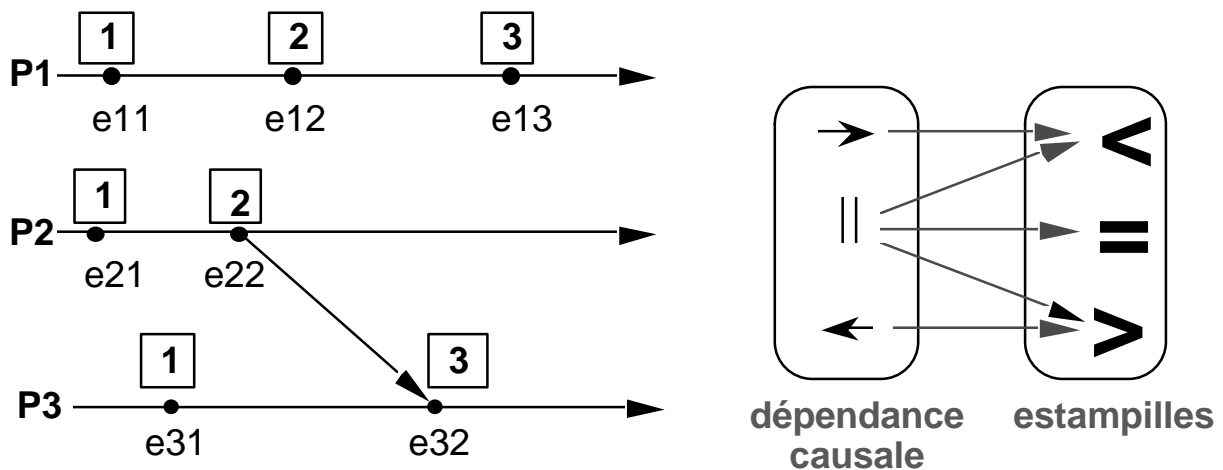
# Utilisation des estampilles

---

- **Algorithmes utilisant une “file d’attente virtuelle” répartie**
  - *exclusion mutuelle répartie*
  - *mise à jour de copies multiples*
  - *diffusion cohérente (ordre de réception uniforme)*
- **Détermination de l’accès “le plus récent”**
  - *gestion cohérente de caches multiples*
  - *fichiers répartis*
  - *mémoire virtuelle répartie*
- **Génération de noms uniques**
  - *désignation interne par noms universels*
  - *mécanismes d’authentification*
- **Synchronisation des horloges physiques**
  - *borne supérieure sur la dérive entre sites*

# Limitations de l'ordonnancement par estampilles

- *Les estampilles définissent un ordre total, mais la relation de dépendance causale est un ordre partiel*
- *Les estampilles “effacent” artificiellement la notion de dépendance causale*
- *Dans certains cas, l'ordre total est inutile*

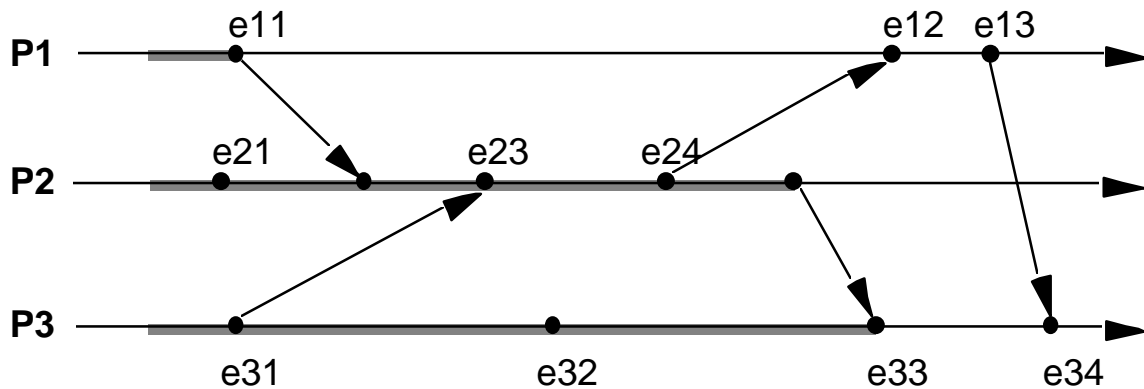


- *Les estampilles ne sont pas “denses”*  
*si  $H(e) < H(e')$ , on ne peut pas savoir s'il existe  $e''$  tel que  $e$  précède  $e''$  et/ou  $e''$  précède  $e'$*
- *Pour certaines applications (mise au point, mesure du parallélisme), on a besoin de caractériser l'indépendance causale. Propriété recherchée pour une horloge  $V$ :*  

$$e \text{ fi } e' \hat{=} V(e) < V(e')$$

# Une méthode de datation causale les historiques

- *Rappel : passé d'un événement  $e$*
- *$hist(e) = e \text{ " } \{ensemble des e' tels que e' \text{ fi } e\}$*



$$hist(e33) = \{e11 e21 e22 e23 e24 e25 e31 e32 e33\}$$

- *Idée : utiliser le passé de  $e$  pour la datation*
- *Le passé permet de déterminer la dépendance causale :*
  - *$hist(e)$  : ensemble des événements  $e'$  tels que  $e' \text{ fi } e$*

$$\begin{array}{ll}
 e \rightarrow e' \Leftrightarrow & e \in hist(e') \\
 e \parallel e' \Leftrightarrow & (e \notin hist(e')) \text{ et } (e' \notin hist(e))
 \end{array}$$

- *Inconvénient : taille de  $hist(e)$*
- *Remède : pour définir  $hist(e)$ , un événement par site suffit. C'est l'idée des horloges vectorielles*

# Horloges vectorielles (Fidge, Mattern 1988)

---

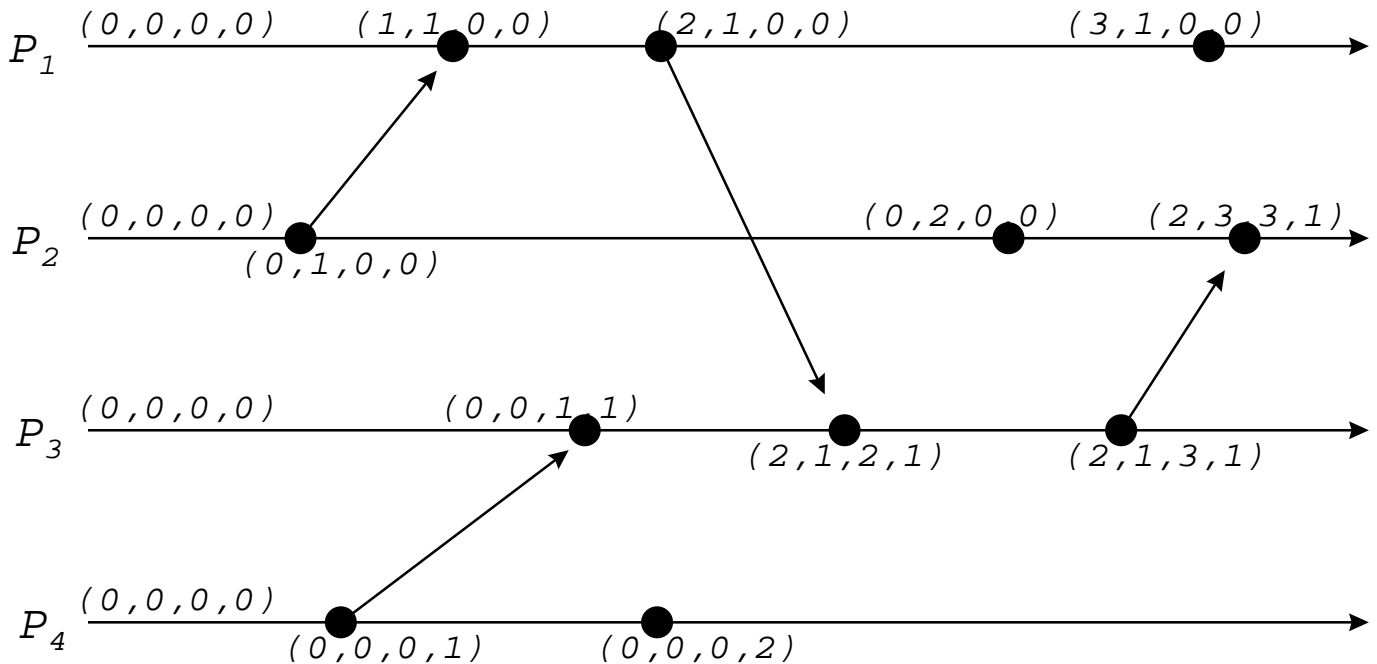
- **Définition :**  $hist_i(e) = (e' \in hist(e) \mid e' \in P_i)$   
 (projection de  $hist(e)$  sur  $P_i$ )  
 si  $hist(e33) = \{e11 \ e21 \ e22 \ e23 \ e24 \ e25 \ e31 \ e32 \ e33\}$   
 alors

  - $hist_1(e33) = \{e11\}$
  - $hist_2(e33) = \{e21 \ e22 \ e23 \ e24 \ e25\}$
  - $hist_3(e33) = \{e31 \ e32 \ e33\}$
  
- **Propriété :**  $e_{i,k} \in hist_i(e) \supset \exists j < k : e_{i,j} \in hist_i(e)$
  
- **Donc un seul entier suffit pour représenter  $hist_i(e)$**   
 $\Rightarrow$  le nombre d'événements de  $hist_i(e)$
  
- **Comme  $hist(e) = \bigcup hist_i(e)$ , il faut un vecteur  $V(e)$**   
**pour représenter  $hist(e)$**
  
- $\exists i \in n : V(e)[i] = k$ , tel que  $e_{i,k} \in hist_i(e)$   
 et  $e_{i,k+1} \notin hist_i(e)$
  
- $V(e)[i] =$  nombre d'événements de  $P_i$  "connus de  $e$ " (i. e. connus sur le site de  $e$  immédiatement après l'occurrence de  $e$ )

# Horloges vectorielles : réalisation

## ■ Définition

- On associe un vecteur  $V_i$  à chaque processus  $P_i$  (ou site  $S_i$ ),  $i = 1, \dots, n$
- Initialement,  $V_i = (0, \dots, 0)$
- A chaque événement local à  $P_i$   
 $V_i[i] := V_i[i] + 1$
- Chaque message  $m$  porte une estampille  $V_m$   
 $(V_m = V_i \text{ de l'émetteur})$
- A la réception de  $(m, V_m)$  par un processus  $P_i$  :  
 $V_i[i] := V_i[i] + 1$   
 $V_i[j] := \max(V_i[j], V_m[j])$  pour  $j = 1, \dots, n, j \neq i$



# Propriétés des horloges vectorielles

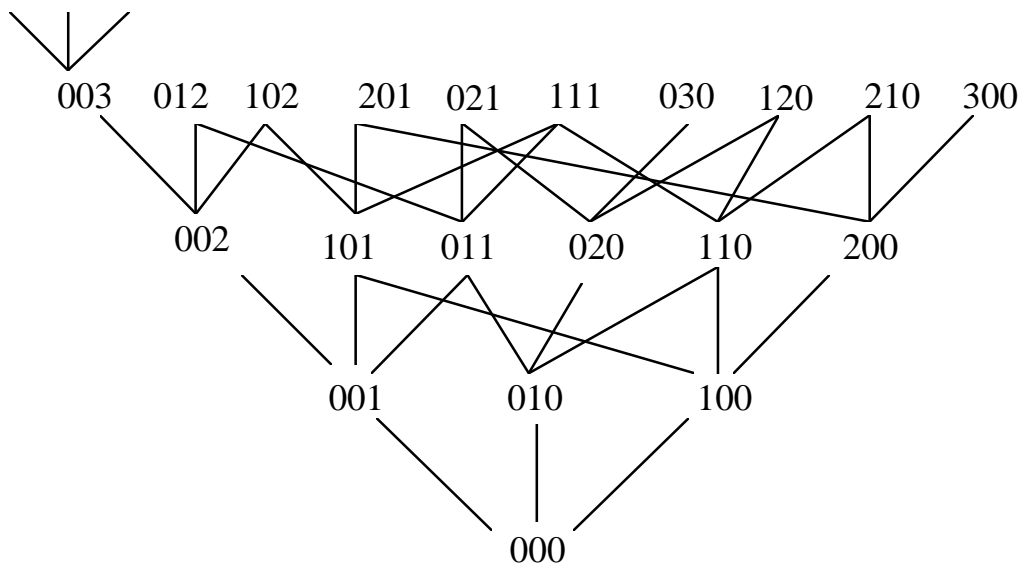
■ **Relation d'ordre (partiel) sur les horloges vectorielles :**

$V \in V'$  défini par  $\forall i : V[i] \in V'[i]$

$V < V'$  défini par  $V \in V'$  et  $V \neq V'$

$V \parallel V'$  défini par  $\neg (V < V')$  et  $\neg (V' < V)$

■ **La structure des horloges vectorielles est un treillis**



■ **Les horloges vectorielles sont “denses” :**

soit  $e_i \in P_i$ ,  $e_j \in P_j$ . Si  $V(e_i)[k] < V(e_j)[k]$ , pour  $k \neq j$ , alors il existe  $e_k$  tel que  $\neg (e_k \text{ fi } e_i)$  et  $(e_k \text{ fi } e_j)$

# Dépendance causale et horloges vectorielles

---

- **Les horloges vectorielles représentent exactement la dépendance causale**

$$\begin{aligned} \text{" } a, b : \quad a \text{ fi } b & \hat{=} \quad V(a) < V(b) \\ a \parallel b & \hat{=} \quad V(a) \parallel V(b) \end{aligned}$$

## Démonstration

- **Si  $a \text{ fi } b$ , alors par transitivité :**

$$\text{" } i : \{x \in P_i \mid x \text{ fi } b\} \hat{=} \{x \in P_i \mid x \text{ fi } a\}$$

(le passé de  $a$  est inclus dans le passé de  $b$ )

$$\text{Donc } \forall i : V_i[a] \leq V_i[b]$$

- **Supposons  $a \parallel b$ ,  $a \in P_k$ ,  $b \in P_l$**

soit  $\text{hist}_k(b) = \{x \in P_k \mid x \text{ fi } b\}$  (passé de  $b$  dans  $P_k$ )

soit  $c = \max(\text{hist}_k(b))$ , au sens de fi

Par construction,  $c \text{ fi } b$  (car  $c \in \text{hist}(b)$  et  $c \preceq b$ )

Alors  $c \text{ fi } a$  (sinon on aurait  $a \text{ fi } b$ ),

donc  $\text{hist}_k(a) \hat{=} \text{hist}_k(c) = \text{hist}_k(b)$ ,

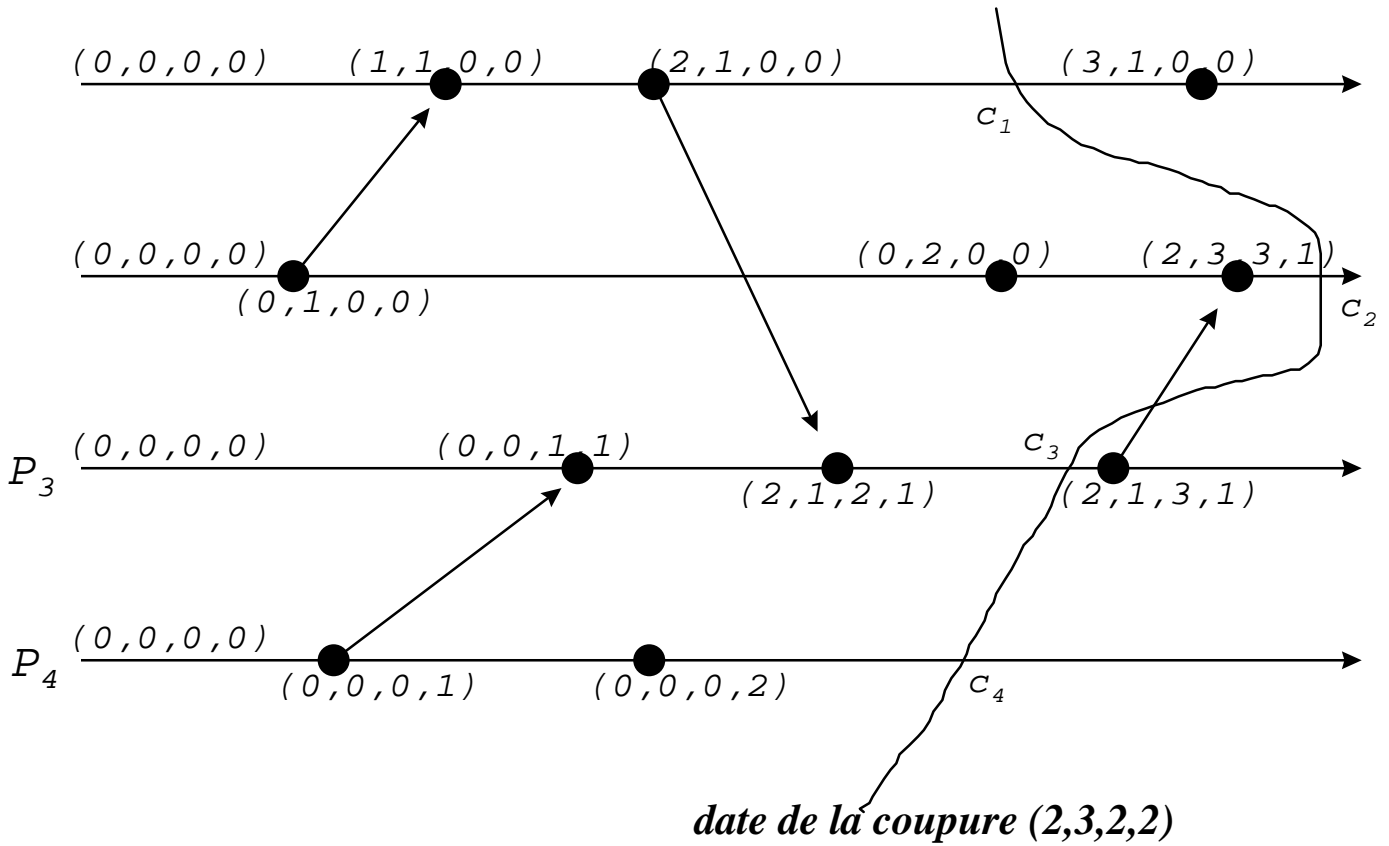
d'où  $Vb[k] < Va[k]$

Par un raisonnement symétrique :  $Va[l] < Vb[l]$

**Donc  $Va \parallel Vb$**

# Horloges vectorielles et coupures

Soit une coupure définie par  $c_1, \dots, c_n$ .



On définit la date de la coupure par

$$\begin{aligned}
 Vc &= \sup (V(c_1), \dots, V(c_n)) \\
 &= \sup ((2,1,0,0), (2,3,3,1), (2,1,2,1), (0,0,0,2)) \\
 &= (2,3,3,2)
 \end{aligned}$$

# Horloges vectorielles et coupures cohérentes

**La coupure est cohérente ssi :**

$$Vc = (V(c_1)[1], \dots, V(c_n)[n])$$

Si la coupure est cohérente : résulte directement du fait que  $V_i[i] \nmid V_j[i]$

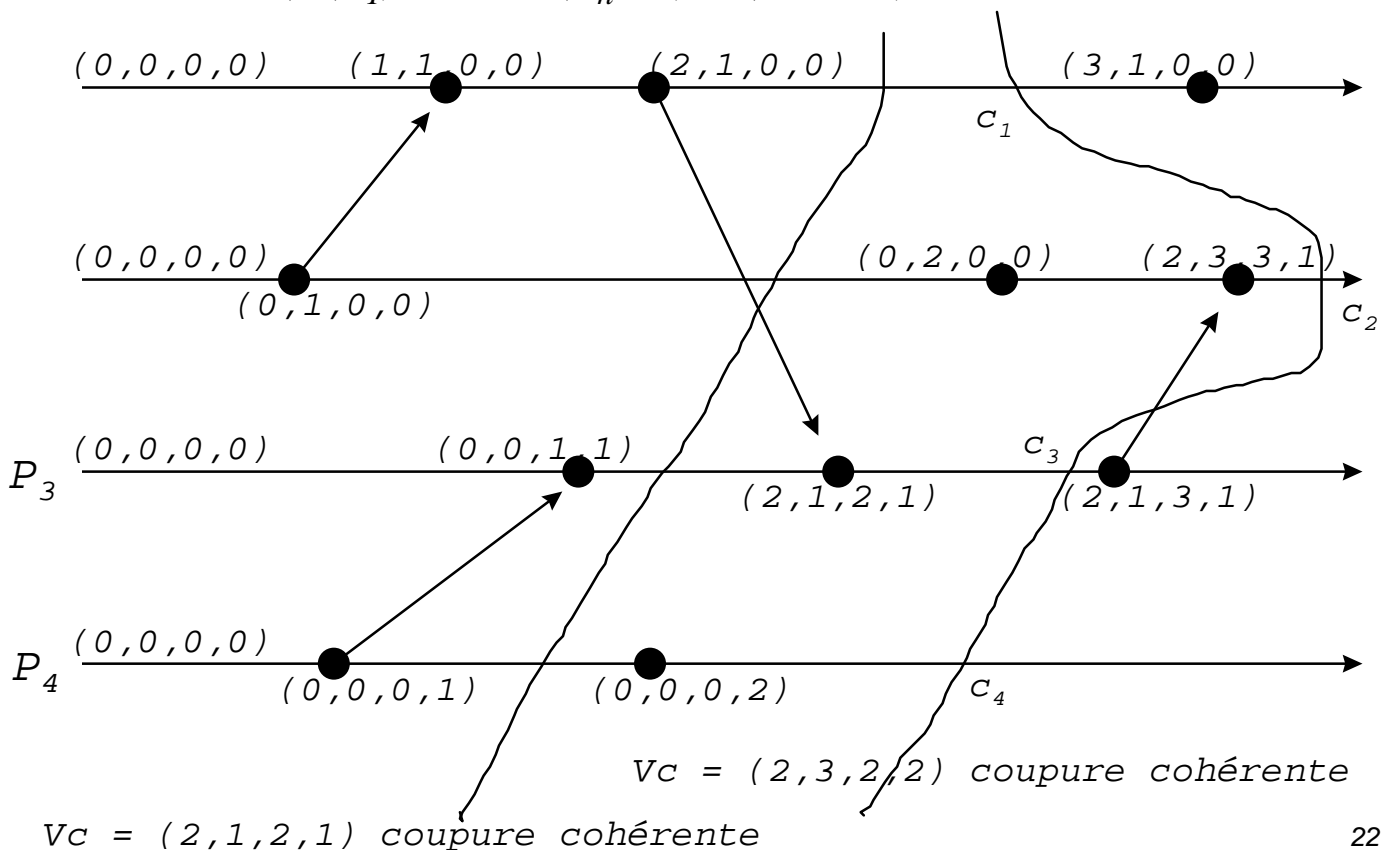
Si la coupure n'est pas cohérente, un message de  $P_i$  a été reçu par  $P_j$  avant la coupure et envoyé après. Si  $W$  est son estampille :  $V(c_i)[i] < W[i] \notin V(c_j)[i]$ .

$$\text{Donc : } Vc > (V(c_1)[1], \dots, V(c_n)[n])$$

Dans l'exemple :

$$Vc = (2, 3, 2, 2)$$

$$(V(c_1)[1], \dots, V(c_n)[n]) = (2, 3, 2, 2)$$



# *Applications pratiques*

---

## ■ *Coupures cohérentes*

- *Définition d'un état global cohérent*
  - enregistrement périodique pour reprise
  - observation et mise au point réparties
  - détection de propriétés stables
- *Mesure du degré de concurrence d'un système*
  - Le nombre de coupures cohérentes est une bonne mesure du degré de concurrence (nombre de sous-ensembles d'événements 2 à 2 concurrents)

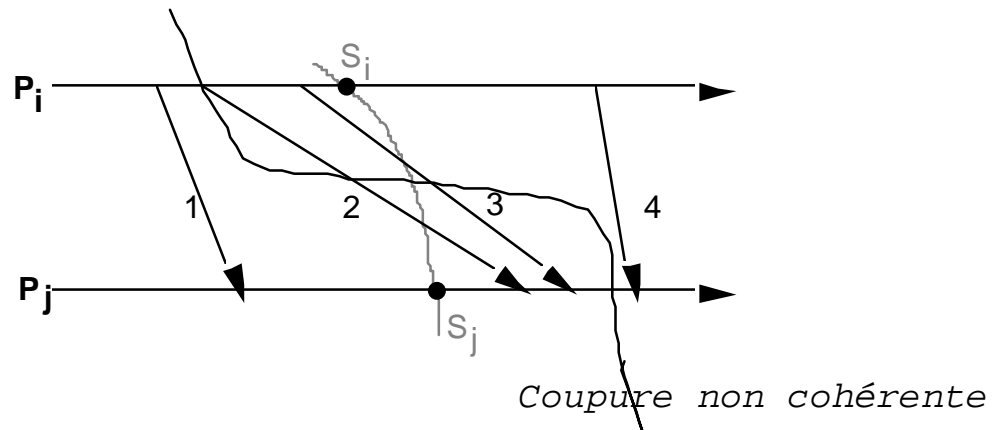
## ■ *Horloges vectorielles*

- *Datation*
- *Diffusion fiable causale*
- *Simulation répartie*
- *Calcul d'état global*

# Etat global cohérent

## ■ Etat du système

- $S = \{S_1, S_2, \dots, S_i, S_j, \dots, S_n\}$  cohérent ssi il correspond à une coupure cohérente :
  - "  $i, j$  : aucun message envoyé par  $P_i$  après  $S_i$  n'est reçu par  $P_j$  avant  $S_j$



## ■ Etat des canaux

Etat global du canal  $\langle i, j \rangle$  pour l'état global  $S$  :  
Tous les messages envoyés par  $P_i$  avant  $S_i$  et non encore reçus par  $P_j$  dans l'état  $S_j$

Exemple : état  $\langle i, j \rangle = (2, 3)$

**Hypothèse** : l'ordre des messages est préservé

# Détermination d'un état cohérent (Chandy - Lamport 1985)

---

- **Hypothèse : respect de l'ordre des messages (canaux FIFO)**
- **Principe**
  - Utiliser un message "marqueur" pour séparer "avant" et "après" enregistrement pour chaque processus
- **Algorithme (à exécuter par tout processus  $P_j$ )**
  - Première réception du marqueur (depuis  $P_j$ )
    - 1) Enregistrer état ( $P_j$ )
    - 2) Enregistrer état (canal  $\langle j, i \rangle$ ) comme vide
    - 3) Diffuser le marqueur à tous ses voisins (processus vers lesquels il a un canal)
 {1-2-3} doit être atomique
  - Réceptions suivantes du marqueur (depuis  $P_k$ )
 

Enregistrer état (canal  $\langle k, i \rangle$ ) comme : tous les messages reçus de  $P_k$  entre l'enregistrement de état ( $P_j$ ) et la réception du marqueur de  $P_k$
- **Nécessité de détecter la terminaison (enregistrement de tous les états) et de regrouper l'ensemble des données enregistrées**

---

## *Détermination d'un état cohérent*

*(suite)*

### ■ *Hypothèses*

- *le réseau de communication est connexe*
- *un processus "élu" particulier lance l'enregistrement*

### ■ *Propriétés*

- *tout processus enregistre son état (diffusion du marqueur par "inondation")*
- *tout processus enregistre l'état des canaux qui aboutissent à lui*
- *l'algorithme se termine*

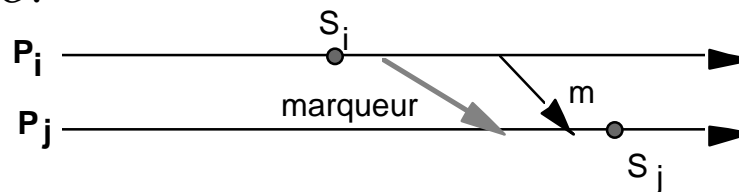
### ■ *Cohérence de l'état*

- *l'état global enregistré correspond à une coupure cohérente*
- *les états enregistrés des canaux sont corrects pour la coupure concernée*

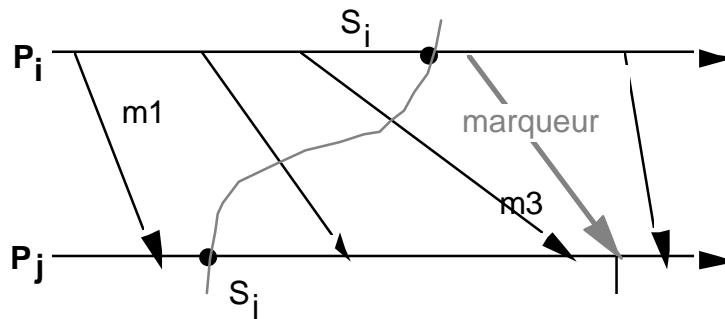
## Détermination d'un état cohérent

■ *L'état global enregistré correspond à une coupure cohérente*

- *Si l'état était incohérent, il existerait  $P_i$ ,  $P_j$  et un message  $m$  tels que  $P_i$  ait émis  $m$  après l'enregistrement de son état et que  $P_j$  l'ait reçu avant. Impossible à cause de l'atomicité et de la propriété FIFO.*



■ *Les états enregistrés des canaux sont corrects pour la coupure concernée*

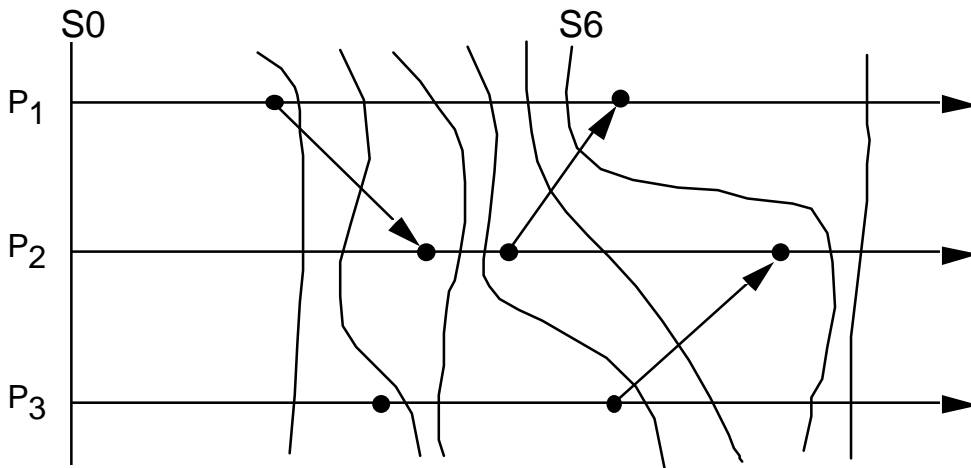


*Etat (canal  $\langle i, j \rangle$ ) enregistré par  $P_j$  à la réception du marqueur : messages reçus de  $P_i$  depuis l'enregistrement de son propre état (ici :  $m_2$ ,  $m_3$ ). Correct d'après la propriété FIFO.*

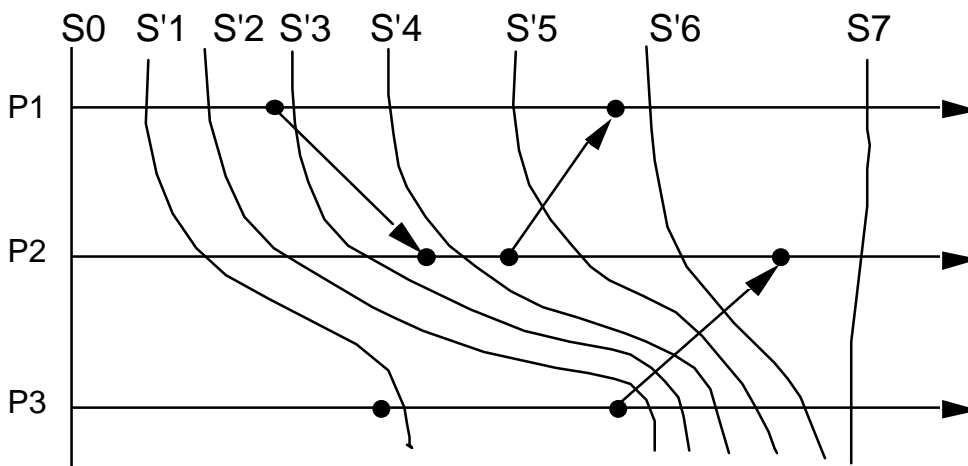
# Interprétation de l'état global

## Etats accessibles et états réalisés

- *Séquence réalisable = suite d'états cohérents ; on ajoute 1 événement à la fois*



- $Seq = (S0, S1, S2, S3, S4, S5, S6, S7)$ 
  - *S7 accessible depuis S0 par Seq*



- $Seq' = (S0, S'1, S'2, S'3, S'4, S'5, S'6, S7)$ 
  - *S7 accessible depuis S0 par Seq'*

# Utilisation de l'état global

## Détermination de propriétés stables

---

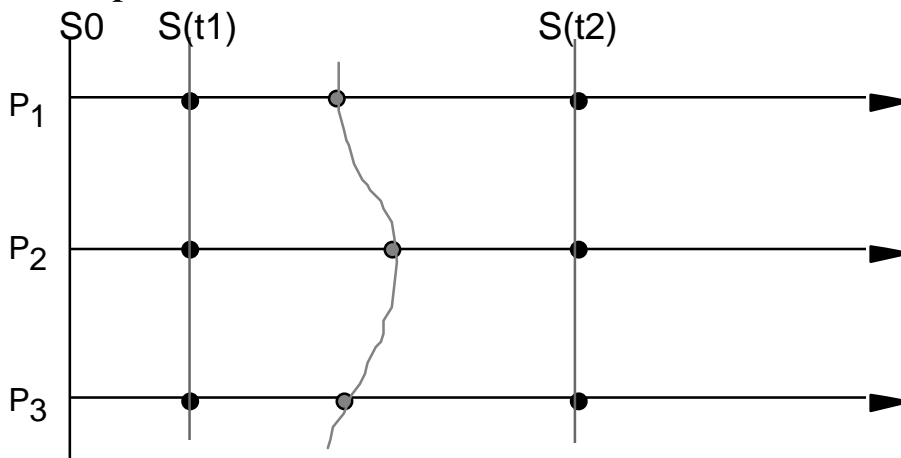
### ■ Propriétés stables

- $P$  vrai dans un état  $S \vdash P$  vrai dans tout état accessible depuis  $S$
- On ne considère que des états cohérents
- Exemples : Le calcul est terminé

*Le système est en interblocage*

*Au moins  $n$  messages ont été émis*

- Un algorithme de calcul d'état global peut servir à détecter des propriétés stables, même si l'état enregistré n'a pas été réellement traversé.



- *L'algorithme de Chandy-Lamport lancé à  $t1$  et se terminant à  $t2$  enregistre un état global  $S^*$  tel que :*

*$S^*$  accessible depuis  $S(t1)$  et  $S(t2)$  accessible depuis  $S^*$*

*Si  $P$  est une propriété stable :*

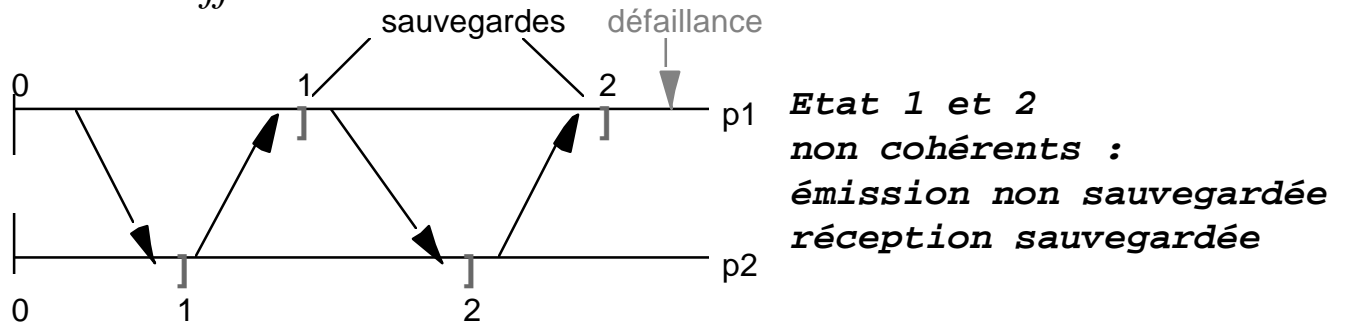
*$P(S^*) \vdash P(S(t2))$  et  $\neg P(S^*) \vdash \neg P(S(t1))$*

*La vérification de  $P(S^*)$  est donc significative*

# Sauvegarde et reprise d'un état global

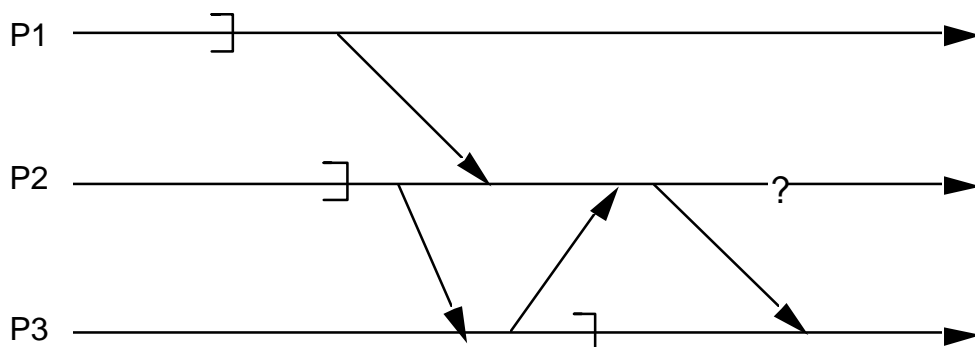
## ■ Conséquence de l'incohérence

- effet domino



## ■ Un algorithme de sauvegarde-reprise [Koo & Toueg 87]

- évite l'effet domino
- nécessite au plus 2 sauvegardes courantes par processus
- $P_i$  peut faire une sauvegarde au temps  $t_i$  si pour tout  $j$ ,  $P_j$  a fait une sauvegarde qui couvre l'envoi de tous les messages reçus par  $P_i$  de  $P_j$  avant l'instant  $t_i$
- Tolérance aux défaillances pendant la sauvegarde : protocole à deux phases (sauvegarde faite par tous ou aucun)



## Références récentes

---

### *Livres ou chapitres de livres*

- *Ö. Babaoglu, K. Marzullo, Consistent Global States of Distributed Systems: Fundamental Concepts and Mechanisms, in Distributed Systems (S. Mullender, ed.), Addison-Wesley, 1993*
- *M. Raynal, La communication et le temps dans les réseaux et les systèmes répartis, Eyrolles, 1991*

### *Revue, actes de congrès, rapports techniques*

- *F. Mattern, On the Relativistic Structure of Logical Time in Distributed Systems, in Datation et Contrôle des Exécutions Réparties, Bigre, n°78, édité par l'IRISA, mars 1992*
- *M. Raynal, About Logical Clocks for Distributed Systems, ACM Operating Systems Review, 26, 1, Jan. 1992, pp. 41-48*
- *M. Raynal, A. Schiper, S. Toueg, The causal ordering abstraction and a simple way to implement it, Information Processing Letters, 39, 6, Sept. 1991, pp. 343-350*
- *R. Schwarz, F. Mattern, Detecting Causal Relationships in Distributed Computations: in Search of the Holy Grail, Tech. Rept. SFB124-15/92, Dept. of Computer Science, Univ. of Kaiserslautern, Dec. 1992*