ÉCOLE POLYTECHNIQUE MONTRÉAL

Module 6:

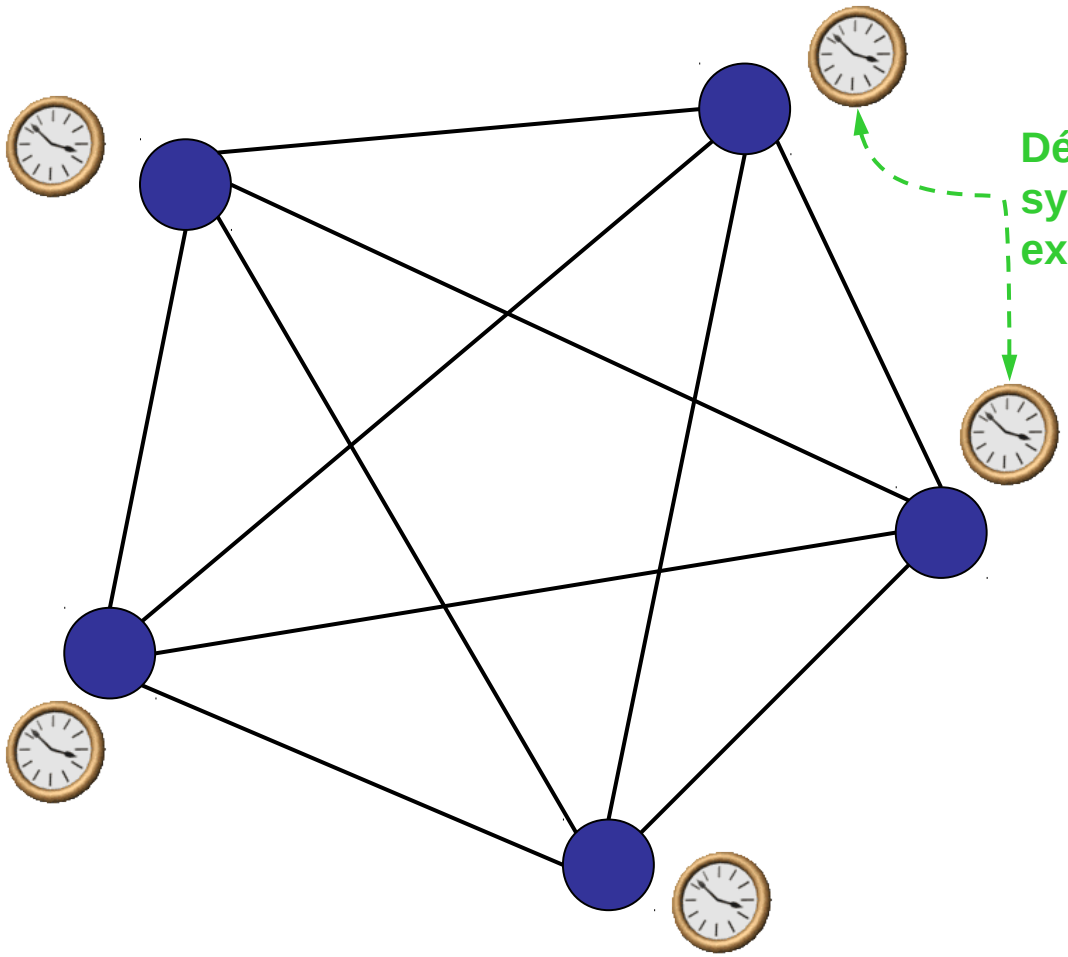
Service de temps

- **Temps**
  - L'heure change d'un fuseau horaire à l'autre, d'une saison à l'autre et d'un pays à l'autre en raison de l'heure avancée.
  - La durée des jours change à cause du frottement des marées et des courants du noyau terrestre.
  - Le temps universel (UTC): une seconde est 9 192 631 770 périodes du césium 133, nombre de secondes par jour ajustable, point de référence GMT (Greenwich Mean Time).
  - L'horloge des ordinateurs dérive avec le temps comme les montres.
  - En cas de retard, avancer d'un coup ou lentement.
  - En avance, ralentir pour laisser le temps nous ratrapper... remonter dans le temps est dangereux.

- **Problématique**
  - Le temps est très important dans les systèmes répartis. On veut savoir avec précision quand un évènement est arrivé.
  - Il n'y a pas d'horloge globale pour mesurer le temps à travers le système réparti, étant donné le délai pour envoyer par réseau une lecture de temps.
  - Algorithmes de synchronisation servent à maintenir la cohérence des données réparties, éliminer les mises à jour redondantes, vérifier l'authenticité des requêtes envoyées au serveur, coordonner certaines opérations, tracer l'exécution du système...

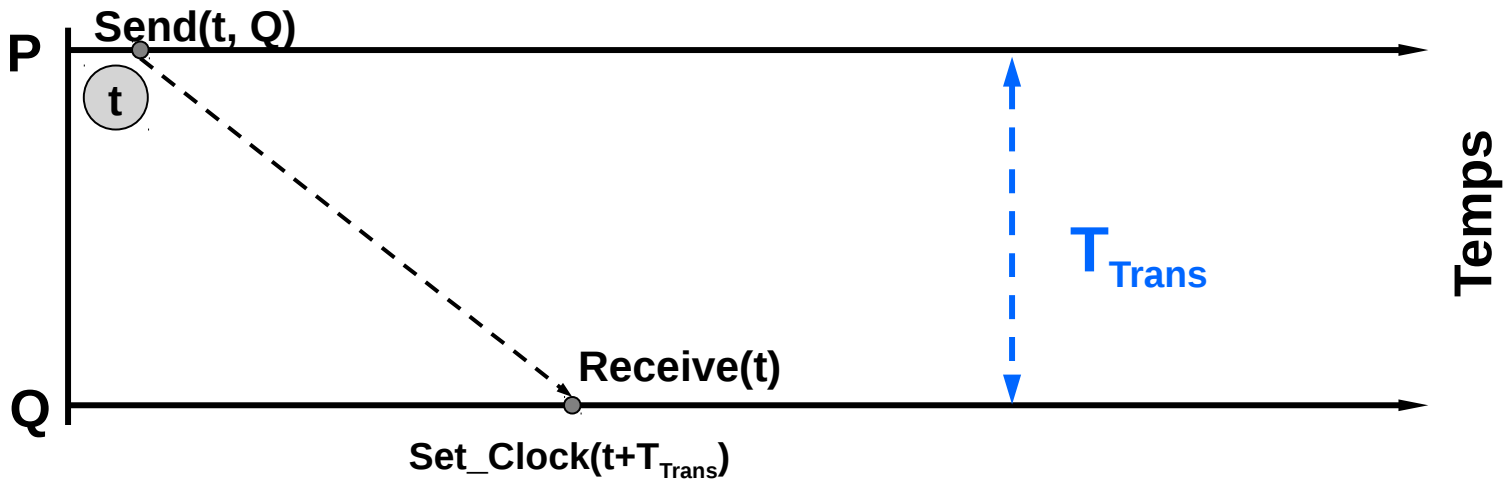
- **Horloge physique :**
  - Horloge d'ordinateur: cristal au quartz qui vibre à une fréquence précise (e.g. à plus ou moins 10^{-6}), avec une interruption après un certain nombre d'oscillations, ce qui génère un tic.
  - L'horloge est obtenue par un calcul d'un temps initial (heure lue du circuit RTC mémorisé avec une pile): temps lu au démarrage + nombre de tics depuis ce temps initial * fréquence des tics.
  - Les cristaux diffèrent d'un ordinateur à l'autre, ce qui cause une désynchronisation entre les systèmes multi-ordinateurs, appelée dérive des horloges (*clock skew*).

- **Horloge physique :**



- **Synchronisation**
  - Le temps de propagation de l'information est limité par la vitesse de la lumière.
  - Délais liés à la préemption et au traitement des interruptions pour recevoir un message de synchronisation.
  - Le GPS (Global Positionning System) donne l'heure juste à 1us près à travers le monde. La différence d'heure reçue entre les satellites est causée par la distance et donne la position par triangulation.
  - Plusieurs ordinateurs (20 à 30) sont connectés à des GPS, des antennes qui reçoivent le signal horaire, ou des horloges atomiques, et agissent comme serveurs de temps primaires.
  - On compte quelques milliers de serveurs de temps secondaires qui prennent l'heure des serveurs primaires via l'Internet.
  - La précision est typiquement de 30ms ou mieux 99% du temps.
  - Avec une horloge calibrable, il est possible d'obtenir une précision de 1ms (temps qu'un signal électrique franchisse environ 300 km) sur plusieurs heures.

- **Synchronisation des systèmes synchrones :**
  - **Cas le plus simple:**
    - Deux processus P et Q veulent synchroniser leurs horloges.
    - Le système est synchrone (délai fixe de transmission connu).

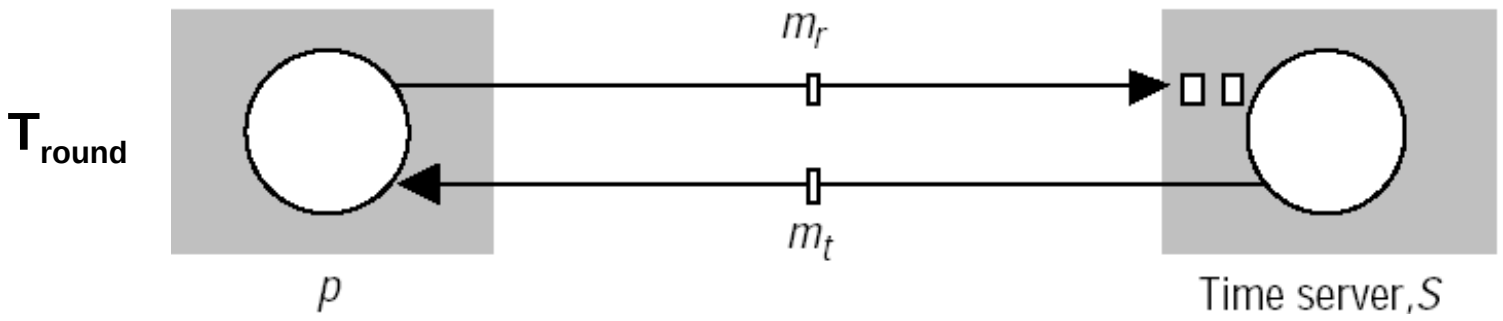


Synchronisation

- Si le temps de transmission peut varier dans un intervalle U, utiliser une valeur à l'intérieur de U = [MIN, MAX].

Scénario	Récepteur affecte à son horloge	Déviation maximale de l'horloge
1	t+MIN	U
2	t+MAX	U
3	t+(MAX+MIN)/2	U/2

- **Synchronisation par la méthode de Christian :**
  - Méthode basée sur la synchronisation externe ;
  - Elle utilise un serveur central de temps (*Time server*) qui est connecté à un périphérique recevant des signaux d'une source UTC ;



  - Processus P affecte à son horloge $t+T_{round}/ 2$

- **Synchronisation par l'algorithme de Berkeley**
  - Un coordonnateur (maître) interroge les autres ordinateurs (esclaves) dont les horloges sont à synchroniser.
  - Les esclaves envoient la valeur de leur horloge au maître.
  - Le maître estime leur temp local en observant le délai d'un aller-retour.
  - Il calcule la moyenne des valeurs reçues et de sa propre valeur.
  - Le maître renvoie les ajustements nécessaires aux esclaves.
  - Si le maître tombe en panne, un autre est élu comme maître.

Network Time Protocol (NTP)

- Défini en 1995 dans le RFC 958.
- Le service NTP est fourni par un réseau de serveurs localisés à travers l'Internet.
- Serveurs primaires connectés directement à une source de temps UTC.
- Serveurs secondaires synchronisés avec les serveurs primaires.

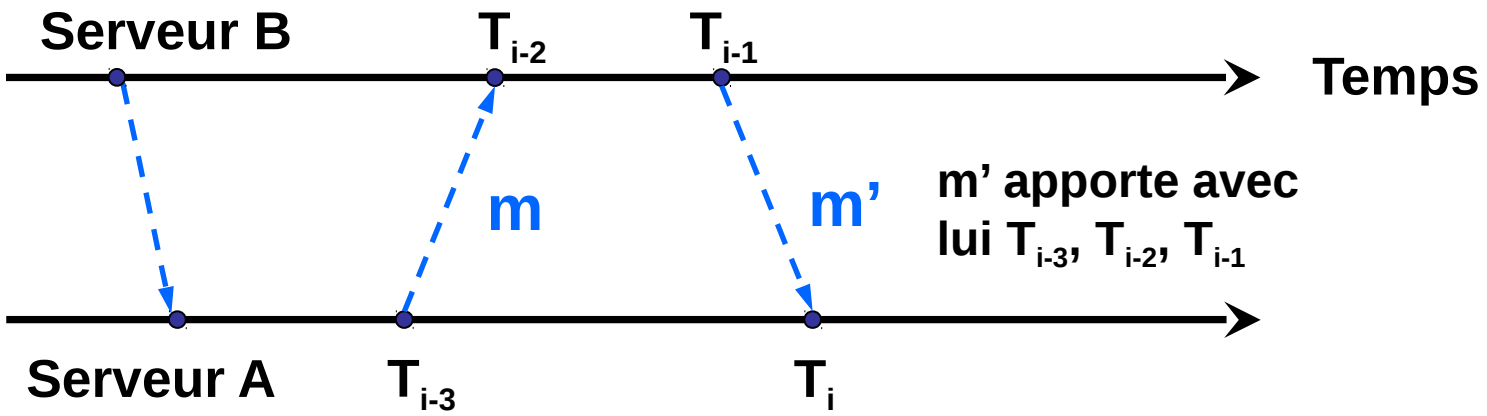
Modes de synchronisation des serveurs

- **Diffusion sélective - *multicast*:**
  - Utilisé dans des LANs à haut débit.
  - Un ou plusieurs serveurs diffusent le temps aux démons s'exécutant sur d'autres ordinateurs connectés au LAN.
  - Les démons qui reçoivent les messages affectent à leur horloge les valeurs reçues en considérant un faible délai.
- **Appel de procédure:**
  - Un serveur accepte les requêtes des autres ordinateurs ;
  - Renvoie une réponse contenant une estampille (valeur de l'horloge locale).
  - Offre une meilleure précision.

• Calcul du décalage

– **Mode symétrique :**

- Paire de processeurs opère en mode symétrique ;
- Les processeurs échangent des messages transportant des informations concernant le temps.



- **Calcul du décalage**
  - Soit le temps de transmission de t de A vers B et t' de B vers A.
  - T(i-2) = T(i-3) + t + d, T(i) = T(i-1) + t'- d.
  - a = T(i-2) - T(i-3), b = T(i-1) - T(i).
  - imprécision = t + t'= a - b, t > 0, t'> 0.
  - ajustement = (a + b) /2
  - d = ajustement + (t'- t)/2.
  - (a + b)/2 - (a - b)/2 <= o <= (a + b)/2 + (a - b)/2
  - On peut envoyer plusieurs messages et retenir ceux dont a et b sont les plus petits. Normalement seuls t et t' varient sur une courte prériode, en les minimisant on réduit l'imprécision.

- **Processus et systèmes répartis :**
  - **Collection de N processus Pi où i = 1, 2, 3, …, N.**
    - Les processus sont indépendants ;
    - Il n'y a pas de mémoire partagée entre les processus.
  - **Si: état du processus Pi.**
    - L'état = valeurs des variables, des ressources ou des objets locaux utilisés par le processus.
  - **La communication entre les processus ne se fait que par transmission de messages.**
  - **Processus Pi :** Ensemble des actions pouvant modifier l'état d'un processus (send, receive, opérations modifiant l'état de Si)

• Évènements dans un système réparti

- **Un évènement est l'occurrence d'une seule action.**

- **Un évènement modifie l'état d'un processus.**
  - L'ordre total unique des évènements est noté : $\rightarrow_i$

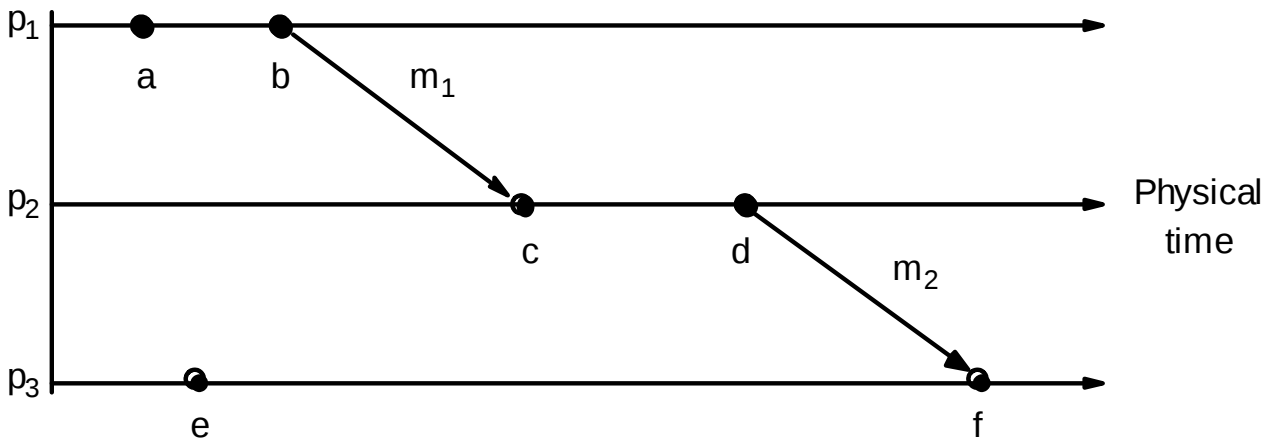
$e \rightarrow_i e'$ SSI événement e est survenu avant e' dans P_i

- **Historique de Pi est une série d'évènements survenus dans Pi et ordonnés par la relation $\rightarrow_i$**

$History(P_i) = h_i = < e^0_i, e^1_i, e^2_i, \ldots >$

- **Horloges logiques**
  - Souvent, seul l'ordre entre les événements importe.
  - Dans un même processus, les événements sont numérotés avec un compteur.
  - Lors d'un envoi de message, la correspondance est faite entre le décompte de l'envoyeur et celui du receveur (Te < Tr).
  - En l'absence de messages, on ne peut rien dire sur l'ordre réel entre deux événements dans des processus différents. On les considère concurrents.

- **Exemple:**



a → ***b*** : puisque ***a*** →$_1$ ***b*** (HB1)

b → ***c*** : b = send(m_1) et c = receive (m_1) (HB2)

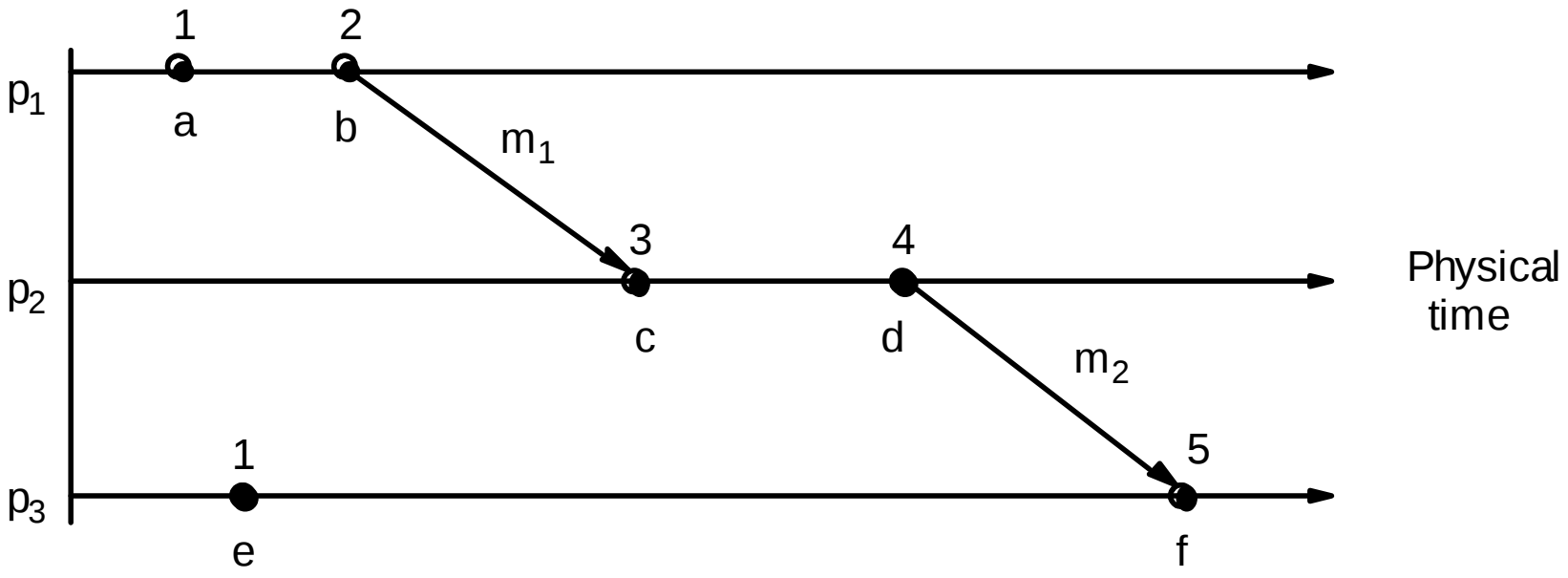
c → ***d*** : puisque ***c*** →$_2$ ***d*** (HB1)

d → ***f*** : d = send(m_2) et f = receive (m_2) (HB2)

a → ***f*** (HB3)

a ↛ ***e***, ***e*** ↛ ***a*** ⇒ événements concurrents!

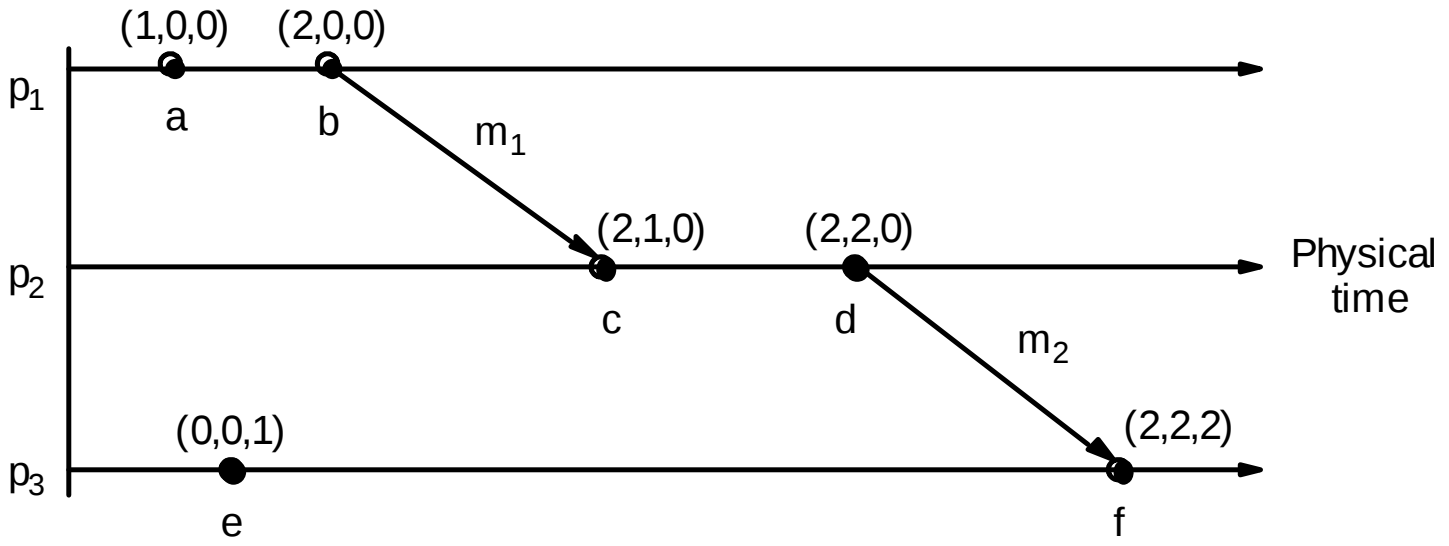
- **Exemple d'horloges logiques:**



Vecteurs de compteurs d'événements

- Vecteur de N entiers initialisés à 0 dans chacun des N processus.
- A chaque événement dans le processus i, Vi[i] est incrémenté.
- A chaque message envoyé par le processus i, Vi est joint.
- Lorsque le processus i reçoit un vecteur dans un message, il prend pour chaque entrée de son vecteur le maximum entre l'entrée présente et celle reçue (fusion des vecteurs).
- Chaque entrée dans le vecteur du processus i indique le dernier événement dans un autre processus qui peut avoir influencé le processus i (lien de causalité).

- **Exemple:**



Règles utilisées pour déterminer l'ordre global :

V = V' ssi V[j] = V'[j], ∀ j = 1, …, N

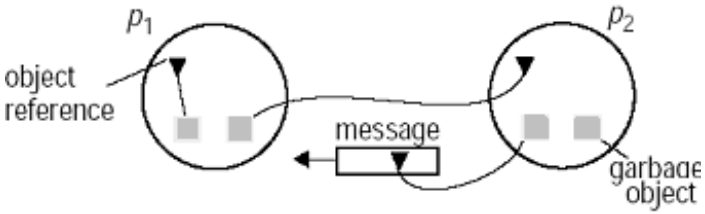
V ≤ V' ssi V[j] ≤ V'[j], ∀ j = 1, …, N

V < V' ssi V ≤ V' et V ≠ V'

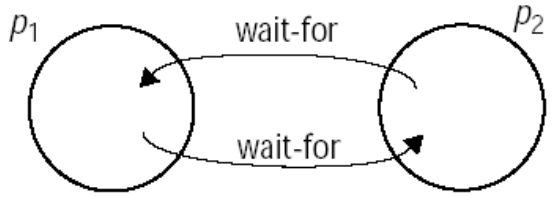
e → e' ⇔ V(e) < V(e')

• État global d'un système réparti:

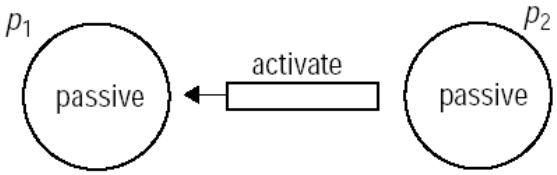
- Pour vérifier si une des propriétés particulières est vérifiée ou non durant son exécution. Pour la mise au point, vérifier les valeurs simultanées de variables dans plusieurs processus...
- **Quelques exemples d'applications :**



Ramasse-miettes (garbage collection): Un objet doit être supprimé s'il n'est plus rejoignable.



Interblocage réparti : Il y a un cycle dans le graphe d'attente de plusieurs processus.



Détecter la terminaison d'un algorithme réparti

- **État global d'un système réparti:**
  - L'état global est important mais difficile à obtenir sans tout figer.
  - Une propriété est stable si une fois atteinte elle demeure vraie. Ceci est vrai pour un verrou mortel et le rend ainsi plus facile à détecter.
  - Un système est sécuritaire pour une propriété si cette propriété reste vérifiée indépendamment de l'ordre dans lequel les événements non reliés causalement surviennent.
  - Un système est vivace si une certaine propriété devient éventuellement vraie, indépendamment de l'ordre dans lequel les événements non reliés causalement surviennent.

État global d'un système réparti

- Un système ξ est composé de N processus Pi où i = 1, 2, 3, …, N;
- L'historique d'un processus est:

$$History(P_i) = h_i = < e^0_i, e^1_i, e^2_i, \dots >$$

- L'historique fini est:

$$h^k_i = < e^0_i, e^1_i, e^2_i, \dots, e^k_i >$$

- L'historique global du système est:

$$H = H_1 \cup H_2 \cup \dots \cup H_N$$

- L'état global du système ξ est:

$$S = (S_1, S_2, \dots, S_N)$$

Seul certains états sont significatifs!

- **Cliché cohérent de l'état global:**

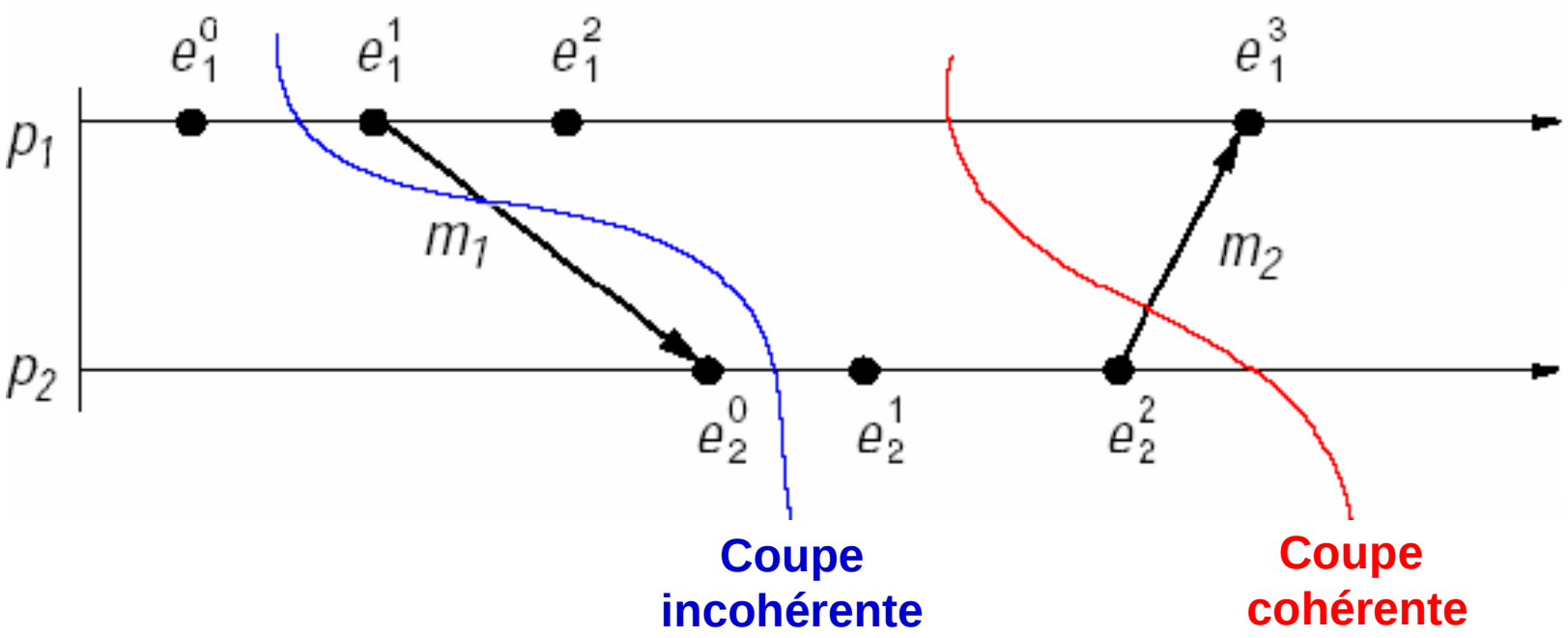
  - Une coupe (*cut*) est un sous-ensemble de l'état global:

$$C = h_1^{C_1} \cup h_2^{C_2} ¿ \ldots h_N^{C_N}$$

  - Une coupe *C* est cohérente si pour chaque événement qu'elle inclut, tous les événements qui sont survenus avant la coupe sont inclus :

∀ événement **e** ∈ ***C***, ***f*** → ***e*** ⇒ ***f*** ∈ ***C***

- **Sous-ensemble de l'état global d'un SR (suite) :**
  - **Un état global cohérent correspond à une coupe cohérente.**



- **Obtention d'un cliché cohérent**
  - Un processus mémorise son état, envoie un message demandant la mémorisation aux autres processus connectés en sortie, et enregistre tous les messages reçus d'autres processus qui n'ont pas encore mémorisé leur état.
  - Un processus qui reçoit l'ordre de mémorisation fait de même.
  - A la fin, l'état global est l'état de chaque processus plus les messages mémorisés (déjà envoyés avant que le processus d'origine n'ait mémorisé son état mais reçus après que le processus courant ait mémorisé son état).

- **Calcul d'un cliché cohérent possible**
  - Il est possible d'enregistrer les événements (changements d'état) de tous les processus. Chaque processus envoie une trace à un moniteur central avec pour chaque événement un vecteur de compteurs d'événements.
  - Un état global est défini par un numéro de dernier événement pour chaque processus impliqué. Cet état est cohérent si pour chaque dernier événement les entrées des vecteurs de compteurs d'événements sont inférieures ou égales au dernier événement inclus pour le processus correspondant.
  - Dans un système synchrone, un état est cohérent si les différences entre les temps des événements sont inférieures à l'imprécision des horloges.

- **Propriétés d'un cliché cohérent possible**
  - Construire un treillis d'états globaux possibles (cohérents).
  - Traverser le treillis un niveau à la fois.
  - Si un des états globaux possibles rejoint a une propriété, cette propriété est possible.
  - Si on arrête à chaque état global possible pour lequel la propriété est vraie et qu'on ne puisse ainsi arriver à l'état final, cette propriété est nécessairement vraie.