Module 8

Services de temps et de coordination

Michel Dagenais Polytechnique Montréal

Sommaire

1. Le temps
2. Les horloges logiques
3. Coordination
4. Conclusion

Sommaire

1. **Le temps**
2. Les horloges logiques
3. Coordination
4. Conclusion

- L'heure change d'un fuseau horaire `a l'autre, d'une saison `a l'autre et d'un pays `a 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.

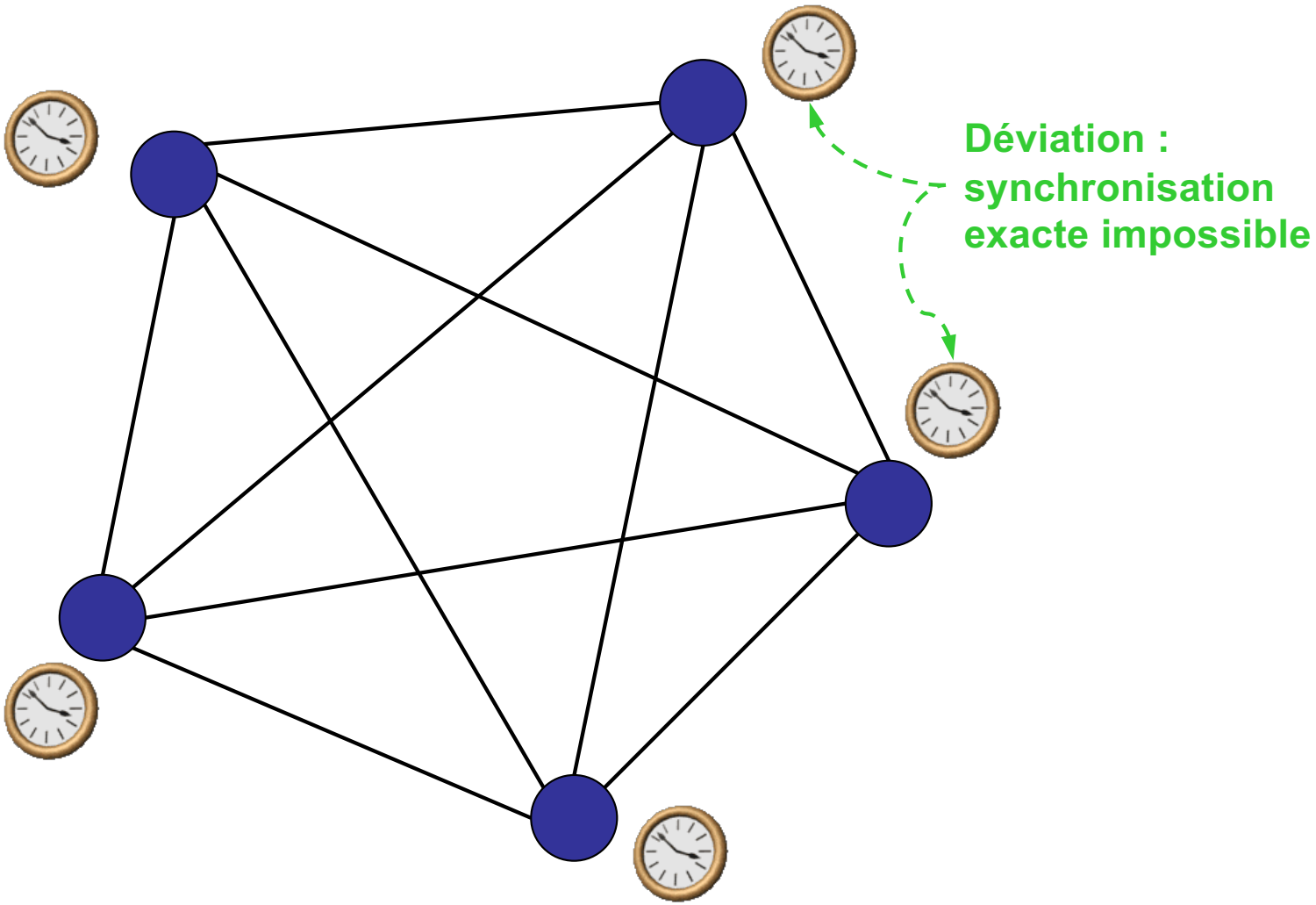
- Temps :
  - Le temps est très important dans les systèmes répartis ;
  - Il doit être mesuré avec précision.
    - On veut savoir avec précision quand un évènement est arrivé.

- Difficulté :
  - Il n'y a pas d'horloge globale pour mesurer le temps à travers le système réparti.

- 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

- **Horloge physique :**
  - Horloge d'ordinateur : cristal au quartz qui vibre à une fréquence précise, 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 + (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 de l'horloge (*clock skew*).

- **Horloge physique :**



Synchronisation

- Le temps de propagation 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 Positioning 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 horloges atomiques ou GPS 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 sur plusieurs heures.

Synchronisation à partir d'une source externe

- **International Atomic Time (IAT).**
- **Coordinated Universal Time (UTC) :**
  - Norme internationale basée sur le temps atomique mais ajustée en fonction du temps astronomique ;
  - Signaux UTC :
    - **Synchronisés et diffusés régulièrement à partir de stations terrestres ou satellites.**
      - » **Station radio terrestre WWV au USA ;**
      - » **Satellites GOES et GPS.**

- **Synchronisation des horloges physiques :**

  - **Synchronisation externe :**
    - horloge Ci synchronisée par rapport à une source de temps externe.

$$|S(t) - C_i(t)| < D, i = 1, 2, \ldots, N$$

Source de temps UTC

Limite de synchronisation, D>0

  - **Synchronisation interne :**
    - horloges Ci synchronisées entre elles avec un certain degré de précision.

$$|C_i(t) - C_j(t)| < D, i=1,2,\ldots,N$$

Pourquoi est-il déconseillé de remonter dans le temps? Comment se défaire d'une avance de 4 secondes en 8 secondes?

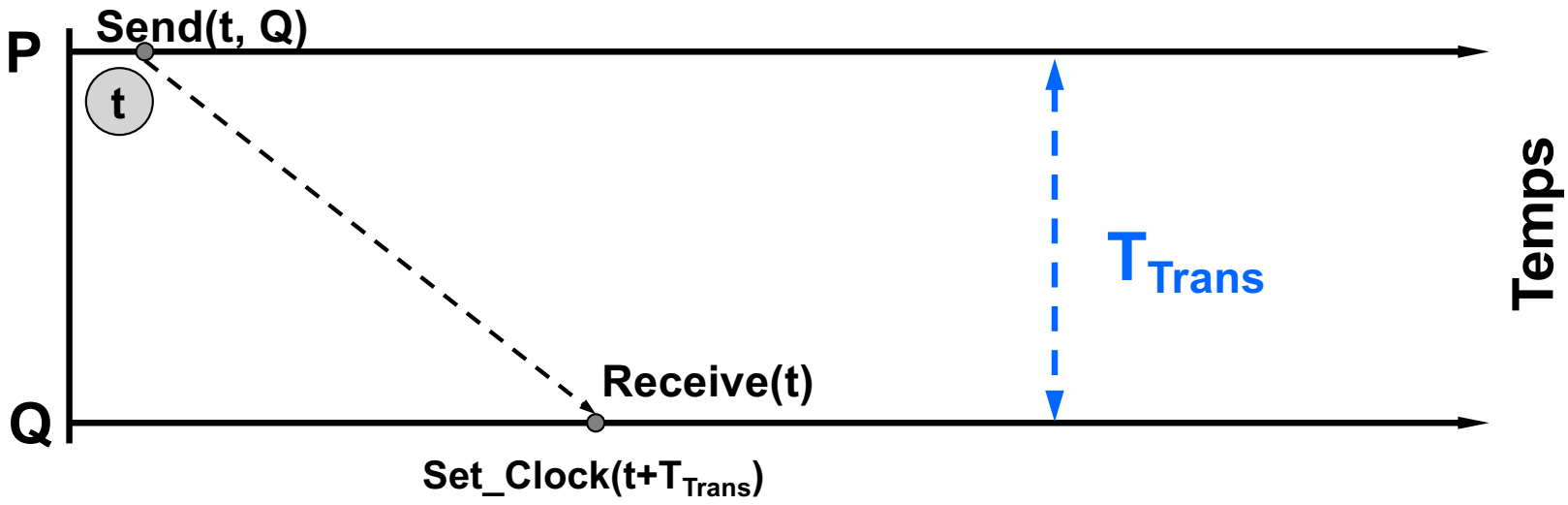
Plusieurs applications comme « make » se basent sur l'hypothèse que le temps avance toujours.

Soit S la bonne heure et S1 l'heure de l'ordinateur.

- Au moment présent, $S = S1 - 4$.
- Dans 8 secondes, on veut que $S+8=S1+c\times8$.
- Donc, $S1-4+8=S1+4=S1+c\times8$, donc $4=c\times8$, et $c=0.5$.
- Il faut donc faire avancer le temps à la moitié de sa vitesse normale pendant les 8 prochaines secondes.

Synchronisation des systèmes synchrones :

- **Cas le plus simple :**
  - Deux processus P et Q veulent synchroniser leurs horloges ;
  - Le système est synchrone



• Synchronisation des systèmes synchrones (suite)

– **Problème**

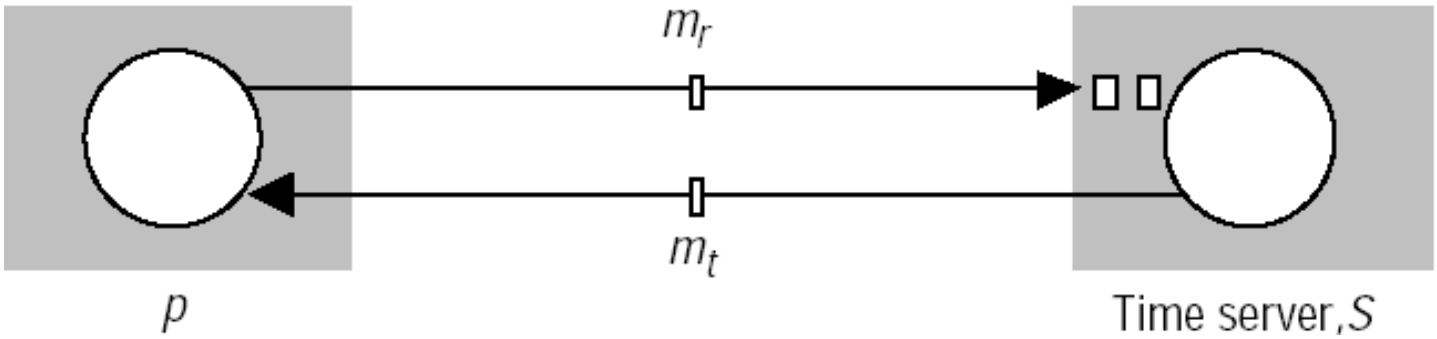
- T_{Trans} peut varier et il est inconnu.

– **Solution**

- Utiliser d'autres mesures (U = MAX – MIN).

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 m_t + $Délai_{aller\text{-}retour}$/ 2
  - Incertitude = $Délai_{aller\text{-}retour}$/ 2
  - L'inconvénient est que le serveur peut tomber en panne : plus moyen de synchroniser les horloges!
    - Une solution serait de placer plusieurs serveurs de temps.

Synchronisation par l'algorithme de Berkeley

- Concept
  - Un coordonnateur (primaire) interroge les autres ordinateurs (secondaires) dont les horloges sont à synchroniser ;
  - Les secondaires envoient la valeur de leurs horloges au primaire ;
  - Le primaire estime leurs temps locaux en observant le délai d'un aller-retour ;
    - **Il calcule la moyenne des valeurs reçues et de sa propre valeur**.
  - Le primaire renvoi les ajustements nécessaires aux secondaires.

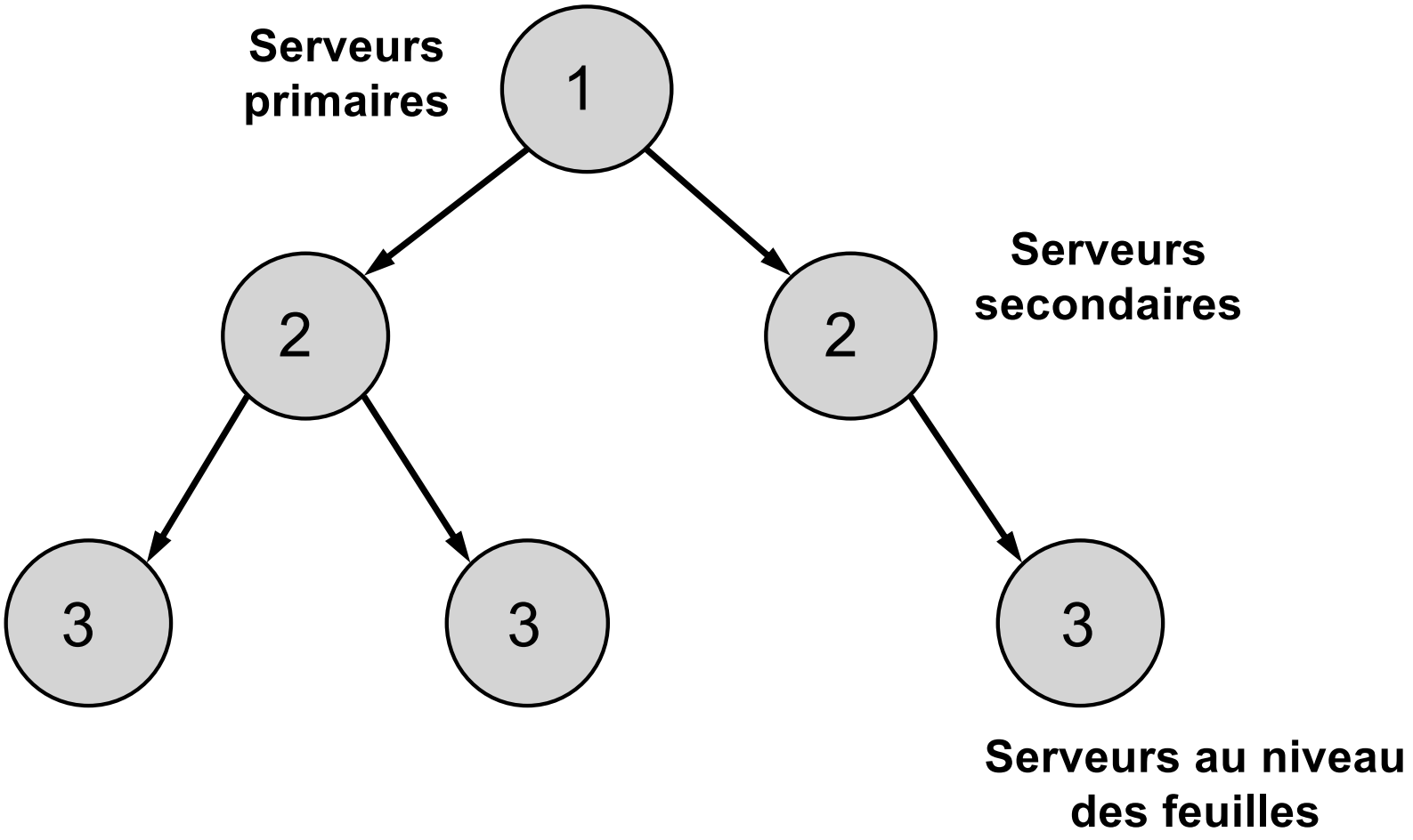
- Si le primaire tombe en panne
  - Un autre est élu comme primaire.

Network Time Protocol (NTP)

- La méthode de Christian et l'algorithme de Berkeley sont conçus pour être utilisés dans des intranets

- **Network Time Protocol :**
  - Norme de synchronisation d'horloges à travers Internet
  - Définie dans le RFC 958
  - Le service NTP est fourni par un réseau de serveurs localisés à travers Internet
    - **Serveurs primaires :** connectés directement à une source de temps UTC (radio ou satellite)
    - **Serveurs Secondaires :** synchronisés avec les serveurs primaires

Network Time Protocol (NTP)



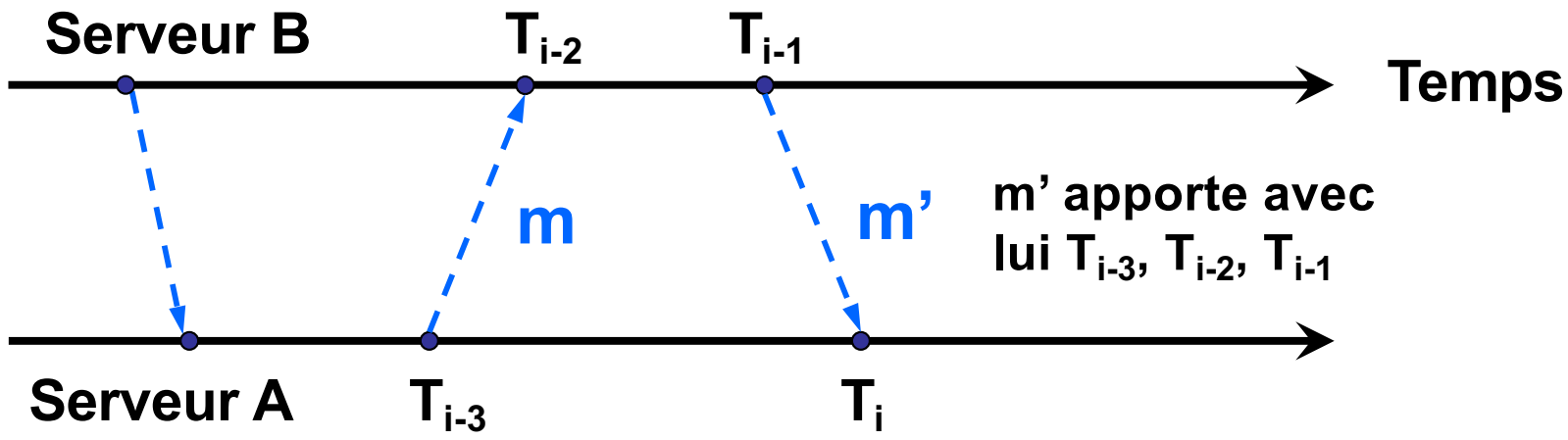
Quelles sont les considérations importantes lors du choix d'un serveur NTP?

- Il faut un serveur précis, et à une courte distance sur le réseau pour minimiser l'imprécision apportée par le délai de transmission.
- Par contre, il faut aussi faire attention à ne pas être trop vulnérable. Si le serveur peut ne plus être disponible ou être compromis, il est bon d'avoir des moyens de vérification et de compensation: horloge locale, autres serveurs. . .

- **Modes de synchronisation des serveurs :**
  - **Diffusion sélective - multicast**
    - Utilisé par 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 leurs horloges les valeurs reçues en considérant un faible délai.
    - Ce mode atteint des niveaux de précision relativement faible.
  - **Appel de procédure :**
    - Un serveur accepte les requêtes des autres ordinateurs ;
    - Renvoie une réponse contenant une estampille (valeur de l'horloge locale).
    - Convenable lorsque le niveau de précision requis doit être élevé.

Calcul de décalage

- Un processeur demande à l'autre le temps par envoi de messages.
- Les temps d'envoi et de réception sont notés pour en tenir compte dans le calcul de décalage.



- Soit le temps de transmission t de A vers B et t' de B vers A.
- $T_{i-2} = T_{i-3} + t + d$ et $T_i = T_{i-1} + t' - d$
- Soit $a = T_{i-2} - T_{i-3}$ et $b = T_{i-1} - T_i$
- $imprecision = t + t' = a - b$ avec $t > 0$ et $t' > 0$
- $ajustement = \frac{a+b}{2}$, *précision* = (a-b)/2
- $d = ajustement + \frac{t'-t}{2}$
- $\frac{a+b}{2} - \frac{a-b}{2} \leq decalage \leq \frac{a+b}{2} + \frac{a-b}{2}$
- $decalage = \frac{a+b}{2} \pm \frac{a-b}{2}$

Exercice 8.1

Un serveur A échange des messages de temps avec un serveur B. B reçoit 16:34:13.430 de A à 16:34:23.480, et A reçoit 16:34:25.700 à 16:34:15.725. Quelles sont la différence et l'imprécision?

- $a = 23.480 - 13.430 = 10.05$
- $b = 25.7 - 15.725 = 9.975$
- Le décalage est donc de:

$$\frac{a+b}{2} \pm \frac{a-b}{2} = \frac{10.05+9.975}{2} \pm \frac{10.05-9.975}{2}$$
$$= \frac{20.025}{2} \pm \frac{0.075}{2}$$
$$= 10.0125 \pm .0375$$
$$\approx 10.013 \pm .038$$

- Soit $a = T_{i-2} - T_{i-3}$ et $b = T_{i-1} - T_i$
- $imprecision = t + t' = a - b$ avec $t > 0$ et $t' > 0$
- Imprécision = t + t' = a – b = 10.05 – 9.975 = 0,075

Exercice 8.2

Un ordinateur A envoie un message à B à 11h11m11.150s pour obtenir le temps et reçoit une réponse à 11h11m11.600s, ces deux temps étant mesurés avec l'horloge de A. L'ordinateur B reçoit la requête de A à 11h11m05.200s et retourne sa réponse à A à 11h11m05.500s, ces deux temps étant mesurés avec l'horloge de B. Quel est l'ajustement à appliquer sur A? Quel est la précision associé?

a = 11h11m05.200s - 11h11m11.150s = -5.950s
b = 11h11m05.500s - 11h11m11.600s = -6.100s
Ajustement = (a+b)/2 = (-5.950s - 6.100s)/2 = -6.025s
Précision = (a-b)/2 = (-5.950s + 6.100s)/2 = 0.075s

L'ajustement à appliquer à A est de -6.025s et la précision est de 0.075s.

Un client C envoie un message au serveur S pour lui demander l'heure exacte. Ce message est envoyé à 16h05m00.200s et la réponse est reçue à 16h05m00.662s, heure du client. La réponse indique que l'heure exacte du serveur, au moment où il a répondu, était de 16h05m06.222s.

- En utilisant **l'algorithme de Christian**, calculez le décalage à appliquer à l'heure du client et donnez l'intervalle d'incertitude sur cette valeur de décalage?

Solution

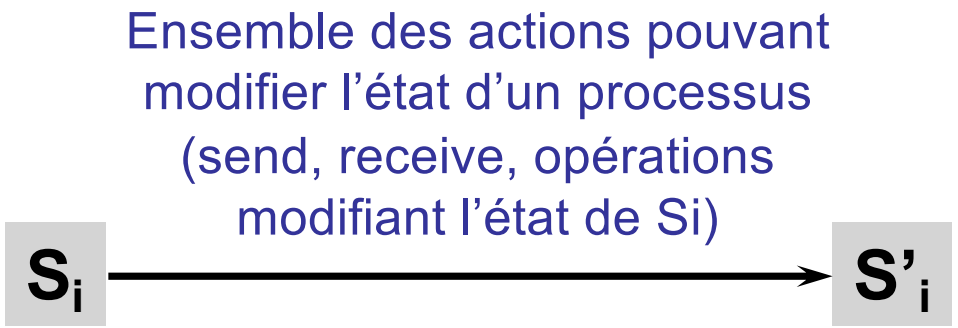
- On suppose que la moitié de l'intervalle, entre les deux temps sur le client, est pour que le message se rende au serveur et l'autre moitié est prise pour acheminer la réponse (délai aller-retour).
- L'incertitude est l'intervalle: 16h05m00.662s - 16h05m00.200s = 0.462s (délai aller-retour), soit +/- 0.231s pour le retour (Incertitude = $\text{Délai}_{\text{aller-retour}}/ 2$)
- La nouvelle heure devrait être augmentée du temps pour acheminer la réponse 16h05m06.222s+0.231s = 16h05m06.453s.
- Le décalage à appliquer est donc de 16h05m06.453s − 16h05m00.662s = 5.791s.

Services de temps et de coordination

1. Le temps
2. **Les horloges logiques**
3. Coordination
4. Conclusion

État des processus et systèmes répartis

- Collection de N processus P_i où i = 1, 2, 3, …, N
  - Les processus sont indépendants
  - Il n'y a pas de mémoire partagée entre les processus
- S_i : état du processus P_i
  - 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 P_i



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 : $\rightarrow_i$

$e \rightarrow_i e'$ si l'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 $\rightarrow_i$

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

Évènements dans un système réparti

- **Il est plus important de connaître si un évènement s'est produit avant ou après un autre plutôt que le moment précis où il s'est produit**
  - Notion d'ordre des évènements
  - On met en accord les processus sur l'ordre des événements, et non sur le temps exact de ces événements.

- Événement *e* s'est produit avant l'événement *f* si le temps d'occurrence de l'événement *e* est inférieur à celui de *f* ;

- Pour un seul processus P1 avec des évènements e1, e2, e3, … la relation $\rightarrow$ permet de définir un ordre total sur les évènements.

Évènements dans un système réparti

- Les systèmes répartis sont composés d'un ensemble de processus indépendants où il n'y a pas d'horloge globale ;
  - Le problème est que la relation $\rightarrow$ ne peut être définie d'une manière directe.

- La relation $\rightarrow$ peut être appliquée à des événements appartenant à différents processus ;

$$\textit{Send(msg1)} \rightarrow \textit{receive(msg1)}$$

La relation permet donc de réaliser
un ordonnancement partiel des événements

Évènements dans un système réparti

– La relation $\rightarrow$ se définie sur un ensemble d'événements d'un système distribué comme suit :

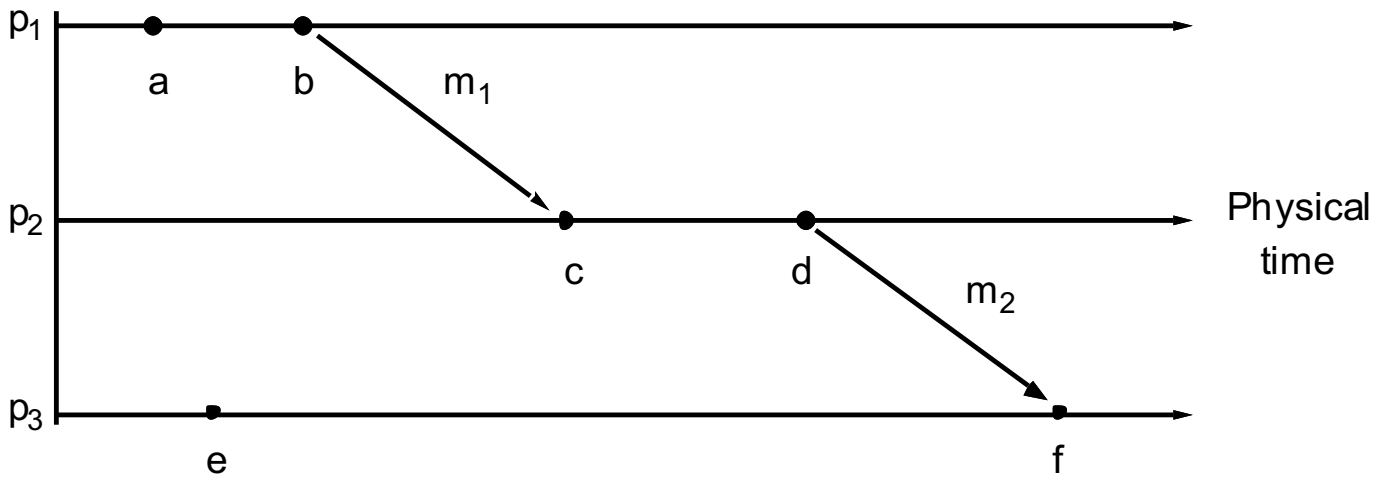
HB1 : Si $\exists$ processus P_i : $\boldsymbol{a} \rightarrow_i \boldsymbol{b}$, alors $\boldsymbol{a} \rightarrow \boldsymbol{b}$

HB2 : $\forall$ m, send(m) $\rightarrow$ receive(m)

HB3 : Si a, b et c sont tels que $\boldsymbol{a} \rightarrow \boldsymbol{b}$ et $\boldsymbol{b} \rightarrow \boldsymbol{c}$, alors $\boldsymbol{a} \rightarrow \boldsymbol{c}$

Deux événements *a* et *b* sont simultanés si $a \not\rightarrow b$ et $b \not\rightarrow a$

- **Exemple de relations entre évènements**



$a \rightarrow b$: puisque $a \rightarrow_1 b$ (HB1)

$b \rightarrow c$: b = send(m_1) et c = receive (m_1) (HB2)

$c \rightarrow d$: puisque $c \rightarrow_2 d$ (HB1)

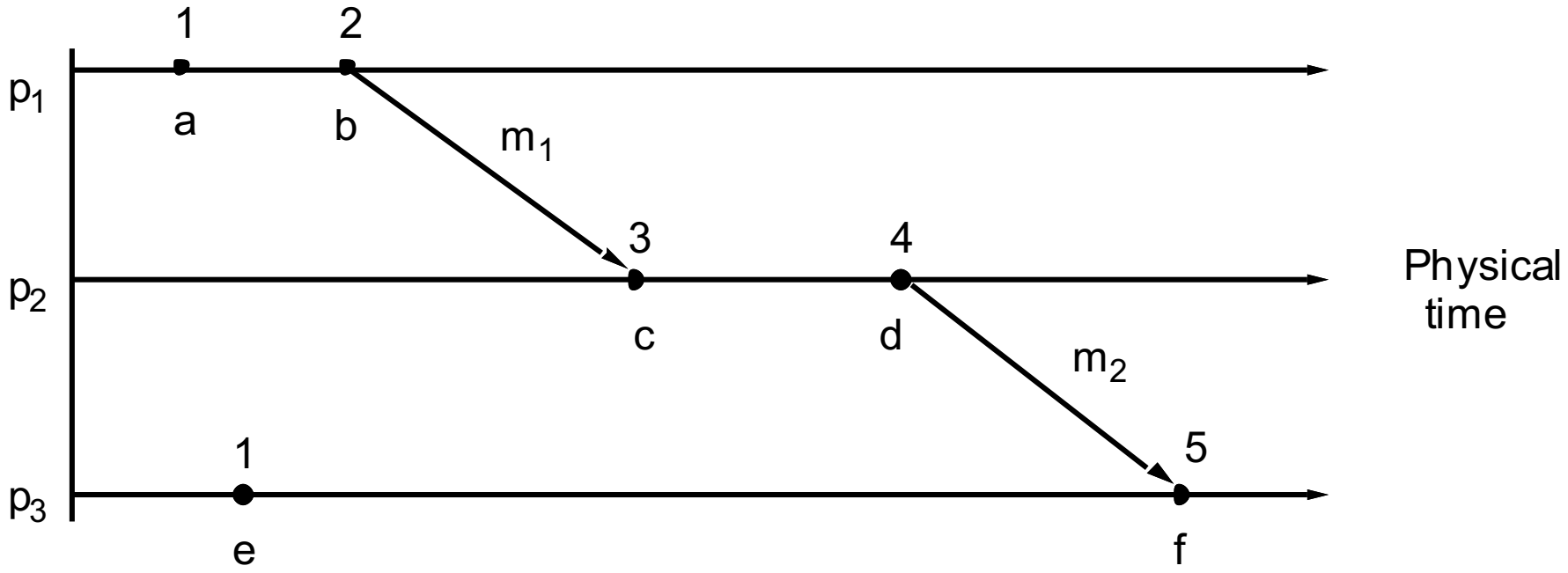
$d \rightarrow f$: d = send(m_2) et f = receive (m_2) (HB2)

$a \rightarrow f$ (HB3)

$a \not\rightarrow e$, $e \not\rightarrow a$ $\Rightarrow$ événements concurrents!

Exemple d'horloges logiques

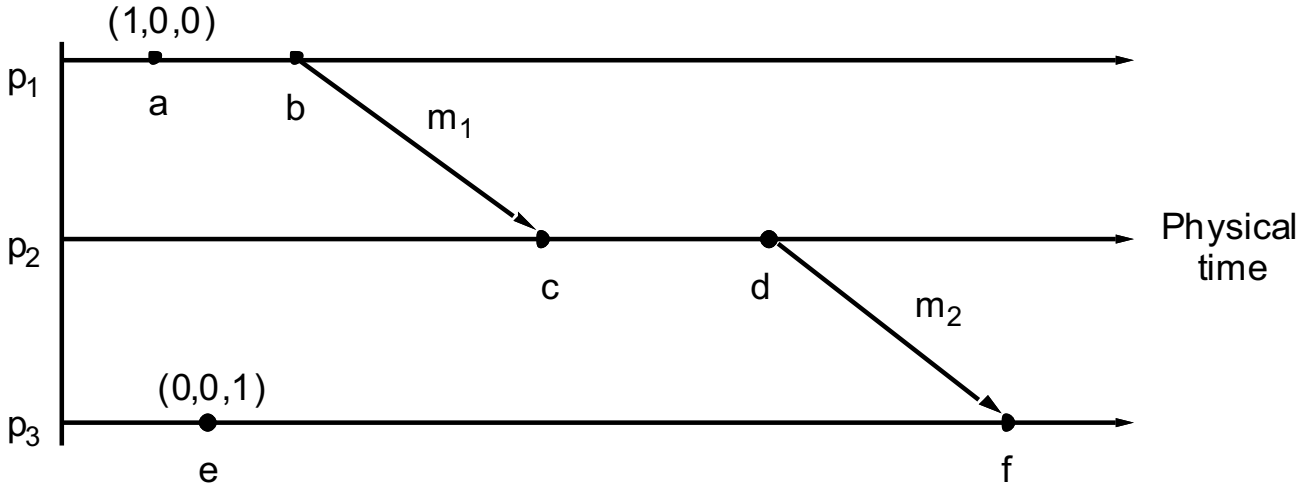
- L'horloge logique est incrémentée de 1 par rapport à la valeur précédente à chaque relation (séquence ou réception de message).
- Si un évènement est postérieur à un autre, sa valeur d'horloge logique est plus grande, l'inverse n'est pas nécessairement vrai (évènements concurrents).



Vecteur de compteurs des évènements

- Vecteur de N entiers initialisés à 0 dans chacun des N processus.
- A chaque évènement dans le processus i, Vi[i] est incrémenté.
- A chaque message envoyé par le processus i, Vi est inclus.
- 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).

- **Vecteurs d'horloges logiques**
  - Complétez les vecteurs manquants.



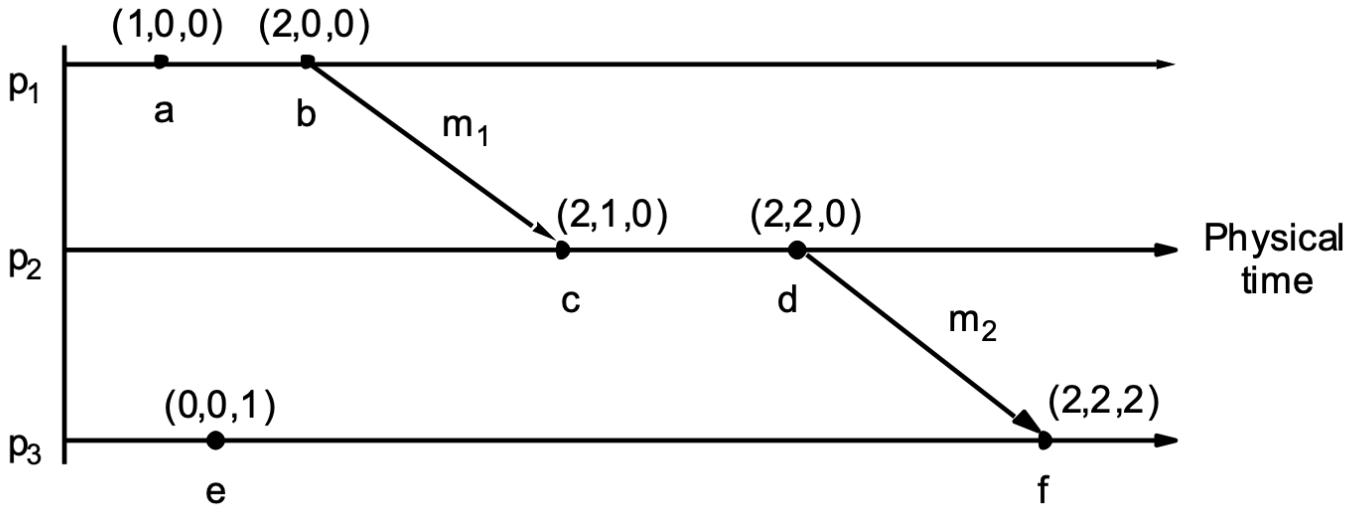
V = V' ssi V[j] = V'[j], ∀ j = 1, …, N

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

V < V' ssi V ≤ V' et V ≠ V'

} **e → e' ⇔ V(e) < V(e')**

- **Vecteurs d'horloges logiques (solution)**



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

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

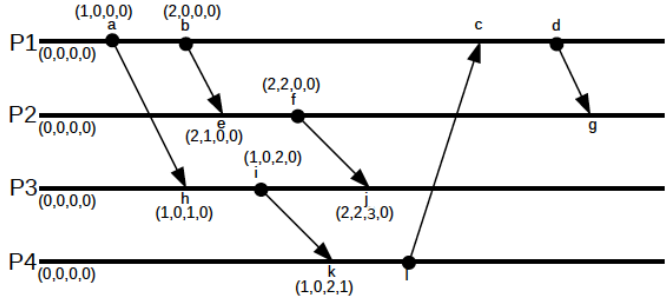
V < V' ssi V ≤ V' et V ≠ V'

} **e → e' ⇔ V(e) < V(e')**

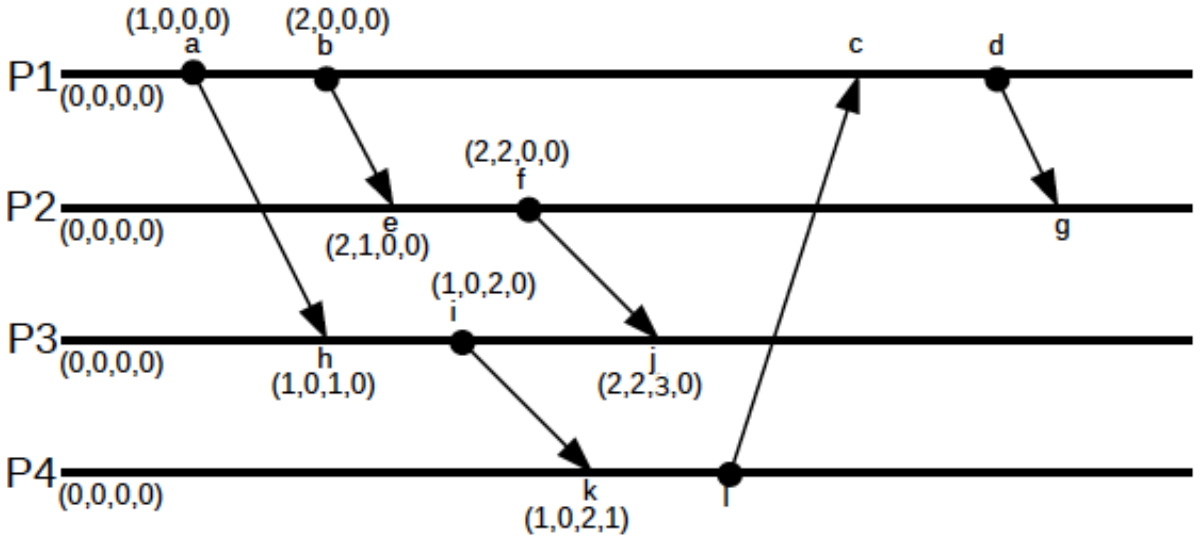
Exercice 8.5

Quatre processus, P1, P2, P3 et P4, démarrent a peu près en même temps et s'envoient des messages. Chaque processus maintient un vecteur de 4 compteurs d'événements, avec une entrée correspondant à chaque processus. Chaque processus, au moment d'envoyer un message, incrémente son compteur d'événements dans son vecteur et joint le vecteur au message. Chaque processus, lorsqu'il reçoit un message, incrémente son compteur d'événements dans son vecteur et fusionne son vecteur avec celui reçu dans le message. Les vecteurs sont donnés pour les points a, b, e, f, h, i, j et k.

i. Que peut-on dire de la relation temporelle entre les points a et j, . l'aide de ces vecteurs de compteurs d'événements?
ii. De la relation entre les points b et k?
iii. Donnez finalement les vecteurs de compteurs d'événements pour les points c et g.

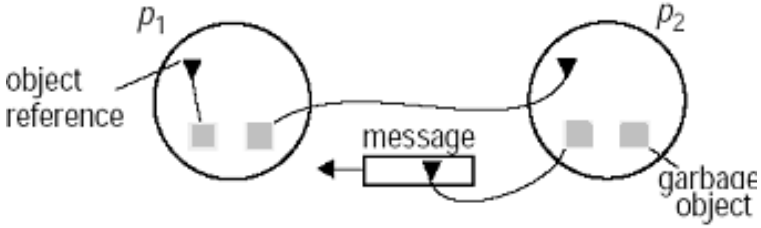


i. Le point j est postérieur au point a car toutes les entrées de son vecteur de compteurs d'événements sont supérieures, (2,2,3,0) vs (1,0,0,0).
ii. Par contre, les points b et k sont potentiellement concurrents car certaines valeurs dans les vecteurs sont supérieures à celles correspondantes dans l'autre vecteur, pour chacun des deux points, (2,0,0,0) vs (1,0,2,1).
iii. Pour le point c, le vecteur est (3,0,2,2). Pour le point g, le vecteur est (4,3,2,2).

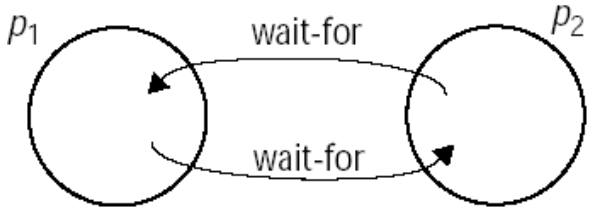


État global d'un système réparti

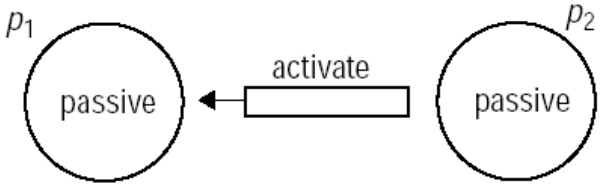
- **Pourquoi déterminer l'état global?**
  - Pour vérifier si une des propriétés particulières est vérifiée ou non durant son exécution!
- **Quelques exemples d'applications**



Garbage collection: Un objet doit être supprimé s'il n'y a aucune référence qui lui réfère dans le SD



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 (e.g. interblocage).
- Un système est sécuritaire pour une propriété si cette propriété reste vérifiée indépendamment de l'ordre dans lequel les évènements non reliés causalement surviennent (e.g. jamais d'interblocage).
- Un système est vivace pour une certaine propriété si elle devient éventuellement vraie, indépendamment de l'ordre dans lequel les évènements non reliés causalement surviennent (e.g. une demande de verrou doit être satisfaite).

- **État global d'un SR (suite) :**
  - Un système ξ est composé de N processus Pi où i = 1, 2, 3, …, N ;
  - L'historique d'un processus est

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

  - L'historique fini est

$$h_i^k = < e^0_i, e^1_i, e^2_i, \ldots, e^k_i >$$

  - L'historique globale du système est

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

  - L'état global du système ξ est

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

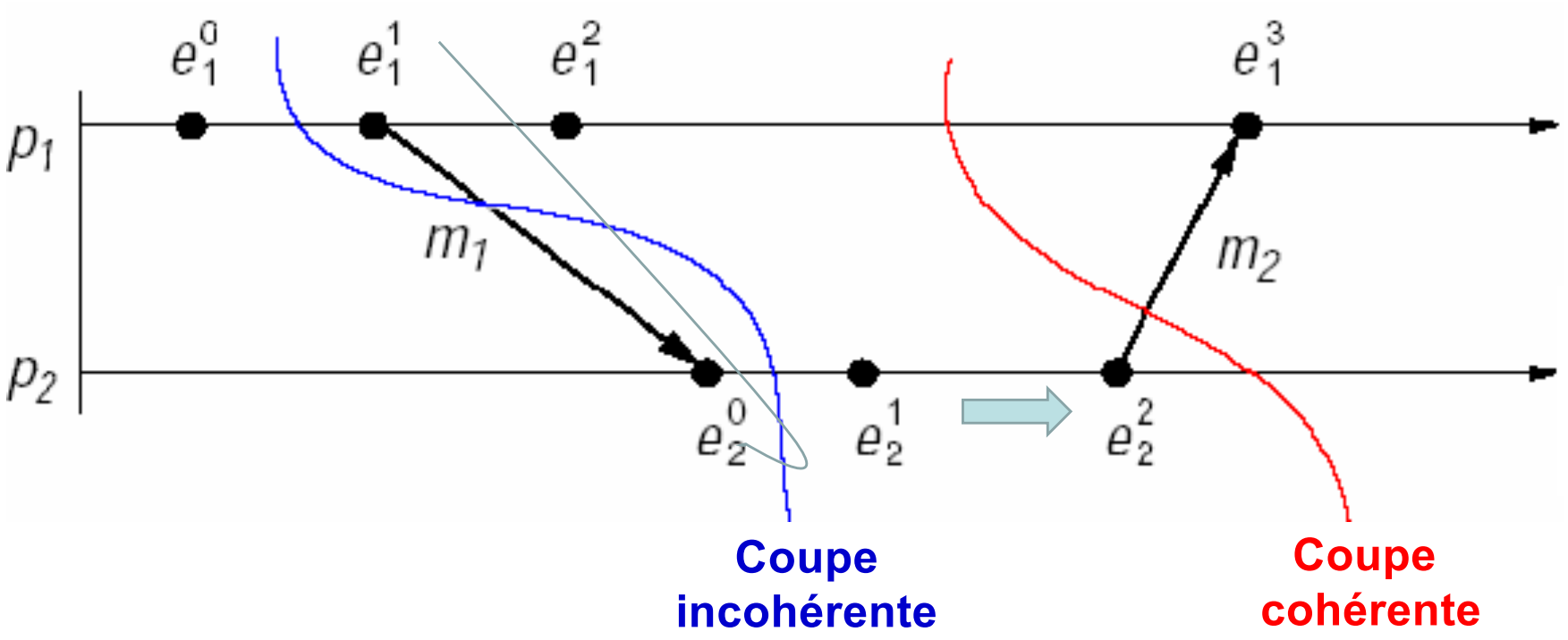
- **Coupe cohérent de l'état global d'un système réparti**

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

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

$$\forall \text{ événement } \mathbf{e} \in C, f \rightarrow e \Rightarrow f \in C$$

- **Coupe cohérent de l'état global d'un système réparti (suite) :**
  - **Un état global cohérent correspond à une coupe cohérente.**



Algorithme pour l'obtention d'une coupe cohérente

- 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'une coupe cohérente possible

- Il est possible d'enregistrer les événements (changements d'état) de tous les processus. Chaque processus envoie une trace `a 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, chaque entrée des vecteurs de compteurs d'événements est inférieure ou égale à la même entrée pour le dernier événement inclus pour le processus correspondant. $V(s_i)[i] \geq V(s_j)[i]$ $i,j=1,2,...,N$

- Dans un système synchrone, un état est cohérent si les différences entre les temps des événements sont inférieures à l'imprécision des horloges.

Services de temps et de coordination

1. Le temps
2. Les horloges logiques
3. **Coordination**
4. Conclusion

Coordination et temps

- Un des problèmes fondamentaux des systèmes répartis : le consensus!
  - Comment faire pour mettre d'accord plusieurs processus indépendants?
  - Comment faire pour coordonner leurs actions?

- Comment faire pour résoudre ce problème?
  - Une solution naïve est d'implémenter le modèle maître-esclave (primaire-secondaire);

- Est-ce que le problème du consensus est le même pour les systèmes synchrones et asynchrones?

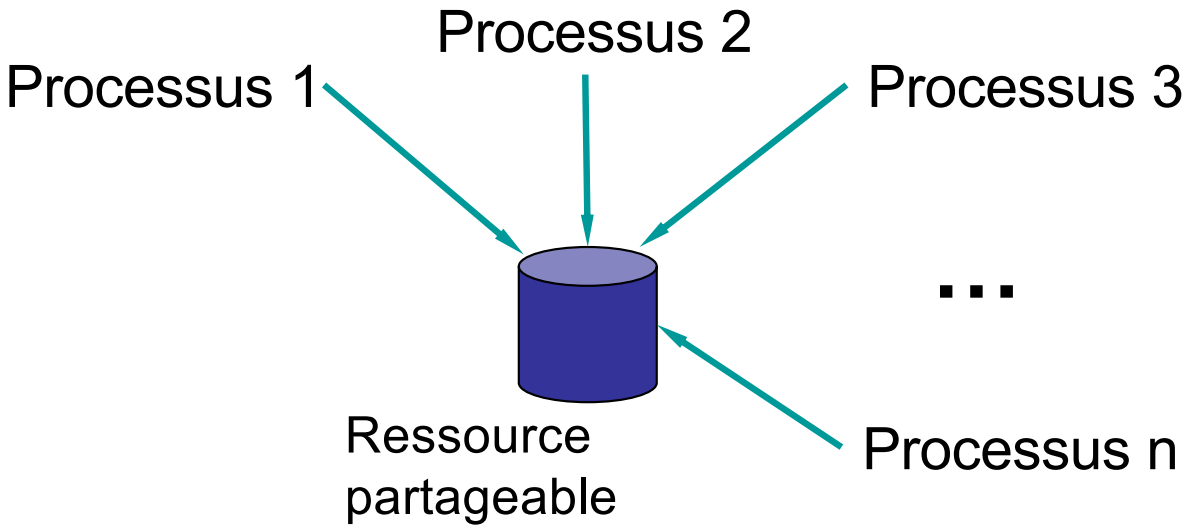
Problème de l'exclusion mutuelle répartie

- Généralisation de l'exclusion mutuelle aux systèmes répartis

- Hypothèses de base

  - Les paires de processus sont connectés par des liens fiables
  - Les processus sont indépendants les uns des autres
  - Le réseau ne se déconnecte pas
  - La panne totale est prise en compte dans les types de pannes considérées
  - Il est possible de détecter localement une panne

Problème de l'exclusion mutuelle répartie



- **Exclusion mutuelle est nécessaire pour**
  - Prévenir les interférences
  - Assurer la cohérence en cas d'accès simultanés aux ressources

Problème de l'exclusion mutuelle répartie

- **Notion de section critique**

Entrer()	*entrer dans la section critique (bloque les autres processus)*
• • •	*accès aux ressources partagées dans la section critique*
Quitter()	*quitter la section critique*

Problème de l'exclusion mutuelle répartie

- Difficulté d'assurer l'exclusion mutuelle répartie
  - Le système ne possède pas de variables globales partagées
  - La communication se fait par transmission de messages
    - La gestion des ressources se fait donc par transmission de messages

Propriétés de Problème de l'exclusion mutuelle répartie

- **Sûreté :** un seul processus à la fois s'exécute en section critique (obtient le verrou);
- **Vivacité :** un processus qui demande d'entrer en section critique se verra éventuellement accorder cette permission
  - Absence d'impasse ou de privation.
- **Ordre :** les demandes sont traitées selon la relation d'ordre causale. Premier arrivé, premier servi.

Décrivez une situation où un serveur central d'exclusion mutuelle ne traite pas les requêtes en ordre causal.

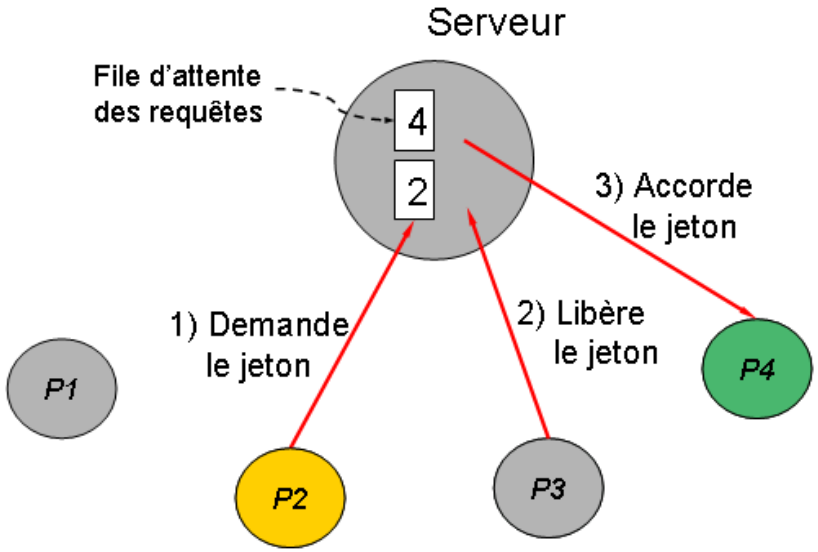
- Un ordinateur A envoie une requête RA au serveur, puis un message à un ordinateur B.
- B envoie alors une requête RB au serveur.
- Il pourrait arriver que les délais différents ou retransmissions fassent que RB arrive avant RA au serveur.

Algorithmes de l'exclusion mutuelle répartie

- Hypothèses de base
  - Le système est asynchrone
  - Les processus ne tombent pas en panne
  - La transmission de messages est fiable

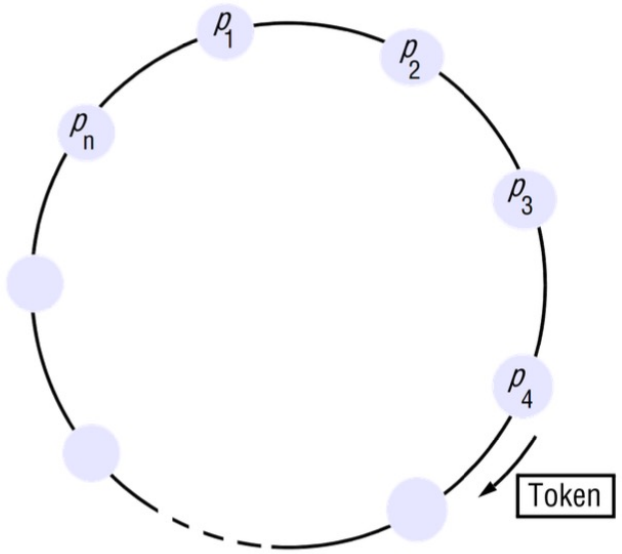
- Types d'algorithmes l'exclusion mutuelle répartie
  - Exclusion mutuelle avec un serveur centralisé
  - Exclusion mutuelle avec horloge logique

Serveur central



- Un client envoie une demande de verrou, le serveur attend que le verrou soit libre, le donne au client, et le client le relâche éventuellement, ce qui permet au serveur de le donner à un autre client.
- Si le serveur est en panne, élection d'un nouveau serveur (majorité de clients), et vérification de tous les clients pour voir qui possède des verrous ou a soumis une requête (problème si clients non rejoignables en raison de division du réseau).
- Exemple: serveur lockd sous NFS.

Anneau à jeton



- Le verrou passe de processus en processus constamment.
- Inutile lorsque personne ne requiert le verrou.
- Problème dès qu'un client fait défaut.
- Ne respecte pas premier arrivé, premier servi.

Exclusion par envoi à tous

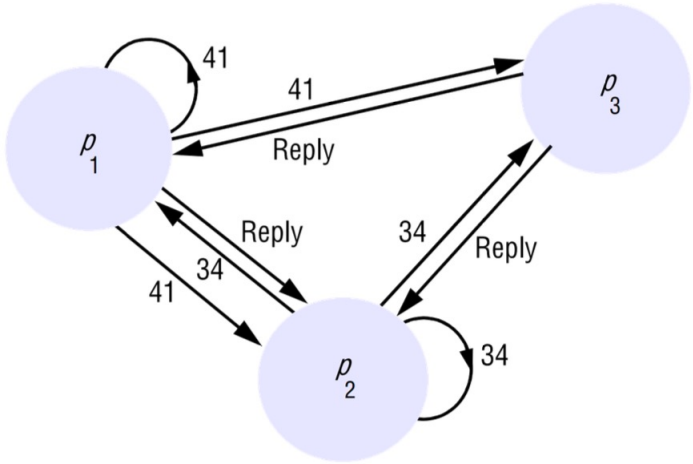
- Envoyer une requête à tous (Il y a N processus).
- Recevoir le O.K. de tous.
- Si on a le verrou, attendre d'en avoir terminé avant de répondre O.K.
- Si on veut le verrou et notre demande est antérieure (estampille de temps), attendre le verrou et d'en avoir fini avant de répondre O.K.
- Demande N messages avec multidiffusion (ou 2N - 2 sans message à tous).
- Tous les clients doivent être actifs.
- Moins bon que serveur central.

Exclusion par envoi à tous (suite)

- p3 ne veut pas entrer dans la section critique
- p1 et p2 demandent accès à la section critique en même temps:
  - requête de p1 à T=41
  - requête de p2 à T=34

Processus:

- p3 répond immédiatement aux requêtes
- p2 reçoit la requête de p1, il voit que T(p2) < T(p1) et ne répond pas (hold p1)
- p1 reçoit la requête de p2, il voit que T(p2) < T(p1) et répond immédiatement
- p2 reçoit la réponse de p1 et entre dans la section critique; quand il la quitte, il répond à p1
- p1 reçoit la réponse de p2 et entre dans la section critique



• Exclusion mutuelle avec horloges logiques

– **Différentes étapes de l'algorithme :**

- **Initialisation :**

```
État := Libre;
```

- **Le processus Pi désire entrer en section critique**

```
État  := Attente;
T := estampille de la requête;
Diffuser la requête <T, pi> à tous les autres processus;
Attendre jusqu'à (Nombre de réponses reçues = (N – 1));
État := Acquise;
```

Exclusion mutuelle avec horloges logiques (suite)

– **Différentes étapes de l'algorithme (suite) :**

- **Réception d'une requête <Ti, Pi> à Pj (i ≠ j):**

```
Si  (État = Acquise) OU
    (État = Attente ET (T, Pj) < (Ti, Pi))
Alors Placer la requête de Pi dans la file d'attente;
Sinon  répondre immédiatement à Pi;
```

- **Le processus Pi quitte la section critique :**

```
État  := Libre;
Répondre à toutes les requêtes placées dans la file d'attente;
```

Comparaison des algorithmes Exclusion mutuelle

Algorithme	Nombre de messages		Problèmes
	Entrer()/Quitter()	Avant Entrer()	
Centralisé	3	2	Panne du serveur
Anneau virtuel	1 à ∝	0 à N-1	Panne d'un processus Perte du jeton Ordonnancement non-satisfait
Horloges logiques	2(N-1)	2(N-1)	Panne d'un processus

Exercice 8.6

Un groupe de 21 processus, p1 . p21, utilisent un serveur central d'exclusion mutuelle. Les 21 processus demandent en même temps (e.g., à 16h00m00.000s) un même verrou v1.

i) Que peut-on dire de l'ordre dans lequel ces 21 processus obtiendront le verrou?

ii) Combien de messages seront échangés au total pour que tous ces processus obtiennent éventuellement le verrou demandé? On suppose qu'il n'y a pas de message perdu.

Solution

Si les horloges ne sont pas bien synchronisées, les demandes ne seront pas émises exactement en même temps, et l'ordre dans lequel elles sont envoyées dépendra de leur décalage de temps. Même si tous les processus demandent

i. le verrou en même temps, ces requêtes seront sérialisées sur le réseau et arriveront dans un certain ordre, un peu aléatoire, au serveur qui pourra les traiter séquentiellement.

Solution

Si les horloges ne sont pas bien synchronisées, les demandes ne seront pas émises exactement en même temps, et l'ordre dans lequel elles sont envoyées dépendra de leur décalage de temps. Même si tous les processus demandent

ii. Un message est requis pour qu'un processus demande le verrou, il reçoit un message de réponse du serveur qui le lui accorde, et le client enverra finalement un message lorsqu'il en a terminé avec le verrou. Le processus suivant en ligne, qui avait déjà envoyé une demande, recevra alors un message du serveur qui lui accorde le verrou. On voit donc qu'il faut 3 messages pour chaque client (demande, obtention, cession). Le nombre total de messages échangés est donc de 21 x 3 = 63 messages. Si des accusés de réception sont utilisés, car le délai n'est pas facilement prévisible, soit pour obtenir la réponse suite à une demande, ou soit pour que le client relâche le verrou après son obtention, alors le nombre des messages pourrait être doublé à 126.

Services de temps et de coordination

1. Le temps
2. Les horloges logiques
3. Coordination
4. **Conclusion**

- Chaque ordinateur a son horloge asynchrone et la synchronisation de temps est imparfaite.
- Pour plusieurs applications, un ordre relatif, une horloge logique, suffit.
- Exclusion mutuelle, élection ou consensus, les approches les plus distribuées ou les plus « démocratiques » ne sont pas nécessairement les plus efficaces.
- Attention aux pannes multiples, répétées, au partitionnement de réseau...