Horloges logiques et datation des évènements.

Johanne Cohen[1]

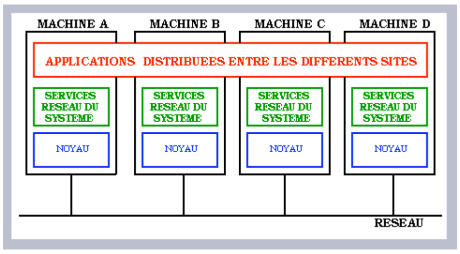
[1]LORIA/CNRS, Nancy, France.

Référence

inspiré du cours

- Jean-Marie Rifflet (http ://www.pps.jussieu.fr/ rifflet/)
- Sacha Krakowiak (http ://proton.inrialpes.fr/ krakowia/)

Système réparti



Être capable de raisonner sur un système réparti

- définir des prédicats (état)
- coordonner des instructions (ordre)

Système réparti

- Un système réparti est constitué de N sites (ou processus) communiquant par messages.
- Chacun de ces sites agit comme un automate : il réalise des opérations qui modifient son état.
  - Sur un même site : ordre des événements = ordre exécution des instructions
  - sur plusieurs sites : ordre des événements = ? ? ?

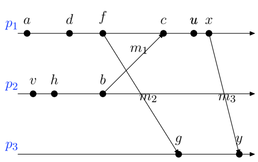
Difficile car....

- pas de mémoire commune, pas d'horloge commune
- Asynchronisme des communications et des traitements.

Modèle asynchrone

- Les processus ne communiquent que par des messages
- pas d'hypothèses
  - sur la durée de transmission des messages
  - sur la vitesse d'exécution des processus
- 3 types d'événements

local	changement de l'état d'un processus
du à une communication	émission d'un message
	réception d'un message



Les communications

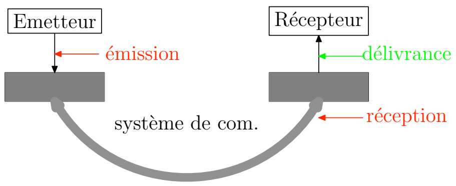
Échange d'information entre sites : envoi de messages.

Hypothèses

- Aucune perte de message (mais le temps de transmission peut être infini)
- Aucune erreur de transmission (une message arrive intacte)
- les canaux de communications peuvent être FIFO ou non.

Délivrance

La communication entre les noeuds est en général assurée par une couche de communication spécifique utilisée par des entités de la couche supérieure pour émettre/recevoir des messages.



La délivrance d'un message : l'opération consistant à le rendre accessible aux applications clientes.
Par exemple :le protocole TCP dans la couche transport.

Relation d'ordre sur les évènements : précédence.

- Sur un même site : ordre des événements = ordre exécution des instructions
- sur plusieurs sites : il faut définir un ordre des événements tel que
  - il soit global
  - il doit être calculer entre deux évènements en fonction de l'information local.

Solution : Définition de la causalité [Lamport78].

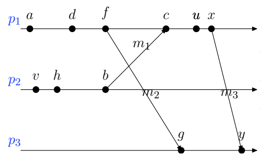
Plan

Ordre causal

L'ordre causal est l'ordre partiel sur les événements :

Si a et b sont deux événements, a précède b ($a \rightsquigarrow b$) si et seulement si l'une des trois conditions est vraie :

- a et b ont lieu sur le même site avec a avant b.
- $a = Envoyer(< M >)$ et $b = Recevoir(< M >)$ du même message.
- Il existe un évènements c tel que $a \rightsquigarrow c$ et $c \rightsquigarrow b$.



- a, d, f, c précèdent u
- b précède c
- $v \rightsquigarrow u$ car $b \rightsquigarrow u$ et $c \rightsquigarrow u$

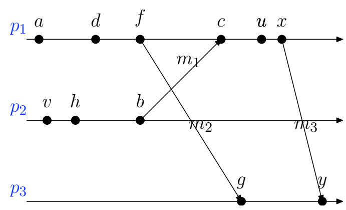
Ordre causal =ordre partiel

A un événement e trois ensembles d'événements sont associés :

- $Passe(e)$: ensemble des événements antérieurs à e dans l'ordre causal (e appartient à cet ensemble) ;
- $Futur(e)$: ensemble des événements postérieurs à e dans l'ordre causal (e appartient à cet ensemble) ;
- $Concurrent(e)$: ensemble des événements concurrents avec e.

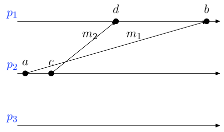
Notation $e \parallel e'$: deux événements e et e' sont concurrents

Exemple



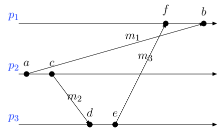
- $Passe(u) = \{a, b, c, f, v, h\}$
- $Futur(u) = \{x, y\}$
- $Concurrent(u) = \{g\}$

Remarque sur certains points incohérents



Canal non FIFO :

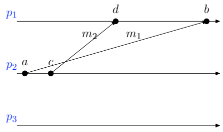
- $c \rightsquigarrow d$
- $d \rightsquigarrow b$
- intérêt de recevoir m_1 ?



Autre scénario :

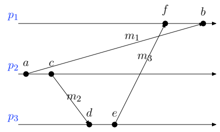
- $a \rightsquigarrow f$
- $f \rightsquigarrow b$
- intérêt de recevoir m_1 ?

Remarque sur certains points incohérents



Canal non FIFO :

- $c \rightsquigarrow d$
- $d \rightsquigarrow b$
- intérêt de recevoir m_1 ?



Autre scénario :

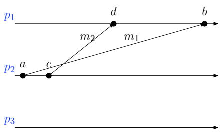
- $a \rightsquigarrow f$
- $f \rightsquigarrow b$
- intérêt de recevoir m_1 ?

Délivrer à l'application cliente de manière correcte

Délivrance FIFO

La délivrance FIFO
assure que si 2 messages sont envoyés successivement depuis un même site S_i vers un même destinataire S_j, alors le premier envoyé sera délivré sur le site S_j avant le second. Formellement, on a

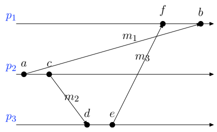
$$send_i(m_1, j) \rightsquigarrow send_i(m_2, j) \text{ implique } del_j(m_1) \rightsquigarrow del_j(m_2)$$



Délivrance causale

La délivrance causale
assure que si l'envoi du message m_1 par le site S_i à destination du site S_k précède (causalement) l'envoi du message m_2 par le site S_j à destination du site S_k, le message m_1 sera délivré avant le message m_2 sur le site S_k . Formellement, on a

$$send_i(m_1, k) \rightsquigarrow send_j(m_2, k) \text{ implique } del_k(m_1) \rightsquigarrow del_k(m_2)$$



Plan

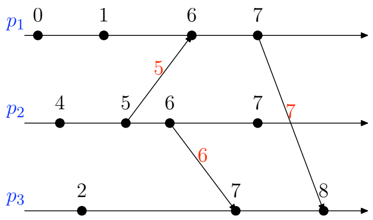
Horloges scalaires

- Chaque site i gère un compteur (une horloge HL_i) dont la valeur est un entier.
- chaque message m envoyé est estampillé (daté) (EL_m) par la date de son évènement.
  - Si un événement est local, alors $HL_i \leftarrow HL_i + 1$
  - Si l'événement est l' envoi d'un message m alors

$$\begin{cases} HL_i \leftarrow HL_i + 1 \\ EL_m \leftarrow HL_i \end{cases}$$

  - Si l'événement est la réception du message m alors
$$HL_i \leftarrow \max(HL_i, EL_m) + 1$$

illustration



L'ordre des événements n'est pas un ordre strict : plusieurs événements peuvent porter la même valeur.

Ordre strict si modification suivante

- l'estampille logique $HL(e)$ de e sur le site i est un couple (HL_i, i).
- L'ordre sur les estampilles est le suivant :

$$(HL_i, i) \prec (HL_j, j) \text{ si et seulement si } \begin{cases} HL_i < HL_j \\ ou \quad HL_i < HL_j \wedge i < j \end{cases}.$$

Propriété : respect de l'ordre causal ?

Propriété

Si $e \rightsquigarrow e$, alors $HL(e) \prec HL(e')$

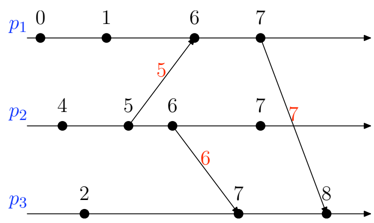
Démonstration.

Raisonnement par récurrence : sur la longueur n de la suite d'évènement reliant e à e'

- si $n = 0$: alors $e = e'$ et $HL(e) \preceq HL(e')$
- si $n > 1$: soit $e = e_0, e_1, \ldots, e_n = e'$ une suite d'évènement tel que $\forall i \in [0, \ldots, n-1]$, $e_i \longrightarrow e_{i+1}$
  - par récurrence on a $e \rightsquigarrow e_{n-1}$ et $HL(e) \prec HL(e_{n-1})$
  - Comme $e_{n-1} \longrightarrow e_n$, on a
    - Soit e_{n-1}, e_n appartiennent au même site $HL(e_{n-1}) \prec HL(e_n)$
    - Soit il existe une message m tel que $send(m) = e_{n-1}$: par définition on a e $HL(e_{n-1}) \prec HL(e_n)$
  - Par transitivité de $\preceq$, on a $HL(e) \prec HL(e_n)$

□

illustration



L'ordre des événements n'est pas un ordre strict : plusieurs événements peuvent porter la même valeur.

Ordre strict si modification suivante

- l'estampille logique $HL(e)$ de e sur le site i est un couple (HL_i, i).
- L'ordre sur les estampilles est le suivant :

$$(HL_i, i) \prec (HL_j, j) \text{ si et seulement si } \begin{cases} HL_i < HL_j \\ ou \quad HL_i < HL_j \wedge i < j \end{cases}.$$

Propriété : respect de l'ordre causal ?

Propriété

Si $e \rightsquigarrow e$, alors $HL(e) \prec HL(e')$

Démonstration.

Raisonnement par récurrence : sur la longueur n de la suite d'évènement reliant e à e'

- si $n = 0$: alors $e = e'$ et $HL(e) \preceq HL(e')$
- si $n > 1$: soit $e = e_0, e_1, \ldots, e_n = e'$ une suite d'évènement tel que $\forall i \in [0, \ldots, n-1]$, $e_i \longrightarrow e_{i+1}$
  - par récurrence on a $e \rightsquigarrow e_{n-1}$ et $HL(e) \prec HL(e_{n-1})$
  - Comme $e_{n-1} \longrightarrow e_n$, on a
    - Soit e_{n-1}, e_n appartiennent au même site $HL(e_{n-1}) \prec HL(e_n)$
    - Soit il existe une message m tel que $send(m) = e_{n-1}$: par définition on a e $HL(e_{n-1}) \prec HL(e_n)$
  - Par transitivité de $\preceq$, on a $HL(e) \prec HL(e_n)$

□

Exemple d'application : exclusion mutuelle

- Un seul site à la fois peut accèder à la ressource critique
- Algorithme distribué : gestion d'une file d'attente distribuée.

  Hypothèse : tous les sites connaissent le nombre total de sites dans le système.

  1. Chaque site i voulant accèder à la ressource critique date sa demande
  2. Il envoie un message de demande d'accès + sa date à tout le monde
  3. tous les sites ne voulant pas y accèder + ayant une date de demande postérieure à la sienne lui donnent leur accord.
  4. Pour les autres, ils attendent qu'ils ont accédé et libéré à la ressource pour lui donné leur accord.
  5. Dès qu'il a reçu l'accord de tout le monde il rentre dans la section critique.

Exemple : 3 sites.

Site 1		Site 2		Site 3	
(Accès,14)					
	(D,0ok)	R(Accès,14,1)		R(Accès,14,1)	
		$\leftarrow (ok, 1)$		$\leftarrow (ok, 1)$	
	(D,2 ok)				
Ressource		(Accès,21)	(D,0ok)		
R.+ R(Accès,21,2)	(D,21,2)			(Accès,20)	(D,0ok)
R.+ R(Accès,20,3)	(D,21,2),(D,20,3)	R(Accès,20,3)	(D,0oK)		(D,0ok),
Ressource	(D,21,2),(D,20,3)	$(ok, 3) \rightarrow$	(D,0oK)	R(Accès,21,2)	(D,0ok),(D,21,2)
liberation	(D,21,2),(D,20,3)		(D,0ok)		(D,1ok),(D,21,2)
$(ok, 2), (ok, 3) \rightarrow$			(D,0ok)		(D,1ok),(D,21,2)
$(ok, 2), (ok, 3) \rightarrow$			(D,0ok)		(D,2ok),(D,21,2)
			(D,1ok)	Ressource	(D,21,2)
			(D,1ok)	libération	(D,21,2)
			(D,1ok)	$\leftarrow (ok, 2)$	
			(D,2ok)		
			ressource		

Horloges vectorielles

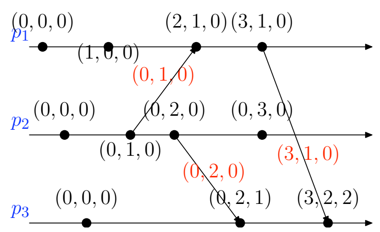
- Chaque site i gère un vecteur d'entiers de n éléments (une horloge HV_i) avec n le nombre de sites
- chaque message m envoyé est estampillé (daté) (EV_m) par la date de son évènement.
  - Si un événement est locale, alors $HV_i[i] \leftarrow HL_i[i] + 1$
  - Si l'événement est l'envoi d'un message m alors

$$\begin{cases} HV_i[i] \leftarrow HL_i[i] + 1 \\ EV_m \leftarrow HV_i \end{cases}$$

  - Si l'événement est la réception du message m alors

$$\begin{cases} HV_i[i] \leftarrow HL_i[i] + 1 \\ HL_i[j] \leftarrow \max(HL_i[j], EL_m[j]) \quad \text{pour tout } j \neq i \end{cases}$$

illustration



Remarque

Propriété :

la valeur de la i-ème composante de EV_e correspond au nombre d'évènements du site S_i appartenant au passé de e.

$$\forall i, EV_e[i] = card(\{e' : e' \in S_i \wedge e' \rightsquigarrow e\})$$

Démonstration.

Soit $e' \in PASSE(e)$ un évènement du site S_i : $EV_{e'}[i] \leq EV_e[i]$.

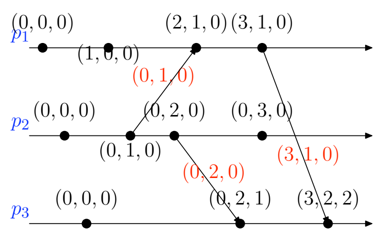
- Considérons e' tel que $EV_{e'}[i] = EV_e[i]$.
- $card(PASSE(e') \cap \{e''$ appartenant à $S_i\}) = EV_{e'}[i] - 1$
- $Futur(e') \cap Passe(e) = \emptyset$ sinon $EV_{e'}[i] \neq EV_e[i]$.

$\square$

la relation d'ordre.

La relation d'ordre suivante peut par ailleurs être définie sur les estampilles vectorielles :

$$EV_e \preceq EV_{e'} \text{ si et seulement si } \forall i, EV_e[i] \preceq EV_{e'}[i]$$



Propriété

Propriété

- $e \rightsquigarrow e'$ si et seulement si $EV_e \preceq EV_{e'}$
- $e \parallel e'$ si et seulement si $EV_e \parallel EV_{e'}$

Démonstration.

- Si $e \rightsquigarrow e'$ alors,
  - $Passe(e) \subseteq Passe(e')$,
  - $\forall i,\ EV_e \leq EV_{e'}$
  - $EV_e \subseteq EV_{e'}$
- Si $e \parallel e'$ alors

$\square$

Propriété

Propriété

- $e \rightsquigarrow e'$ si et seulement si $EV_e \preceq EV_{e'}$
- $e \parallel e'$ si et seulement si $EV_e \parallel EV_{e'}$

Démonstration.

- Si $e \rightsquigarrow e'$ alors, $EV_e \preceq EV_{e'}$
  - $Passe(e) \subseteq Passe(e')$,
  - $\forall i,\ EV_e \leq EV_{e'}$
  - $EV_e \subseteq EV_{e'}$
- Si $e \parallel e'$ alors

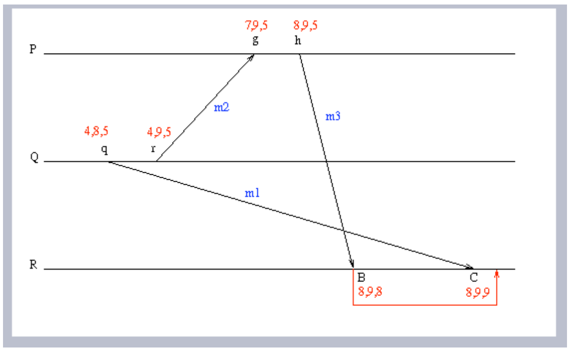
$\square$

Suite de la preuve : si $e \parallel e'$, alors

Soit S_i et S_j les sites appartenant à e et e'

- si $j = i$ alors, e et e' ne peuvent pas être concurrents.
- Donc $j \neq i$
- Soit $x(e,j)$ le plus grand élément du $Passe_j(e)$
- on a $x(e,j) \rightarrow^+ e'$ (car si $e' \rightsquigarrow x(e,j)$, alors $e' \in Passe(e)$)
- Donc : $Passe_j(e) = Passe_j(x(e,j)) \subset Passe_j(e')$
- Par conséquent : $EV_e[j] < EV_{e'}[j]$
- Utilisant le même raisonnement : on a $EV_{e'}[i] < EV_e[i]$

Cet ordre permet-il la délivrance causale ? NON



Horloges et estampilles matricielles

- Chaque site i gère une horloge (notée HM_i) matricielle ($n \times n$) avec n le nombre de sites
- chaque message m envoyé est estampillé (daté) (EM_m) par la valeur courante de HM_i.
- Que signifie $HM_i[j,k]$?
  - $HM_i[j,k] =$ nbre de messages issus de p_j vers p_k dont p_i a connaissance.
  - $HM_i[j,j]$: correspond au nbre d'évènements locaux du site j.

Comment HM_i est-elle modifiée ? (1/2)

- Évènement local à p_i : $HM_i[i,i] \leftarrow HM_i[i,i] + 1$
- Émission de m vers p_j : $\left\{ \begin{array}{l} HM_i[i,i] \leftarrow HM_i[i,i] + 1 \\ HM_i[i,j] \leftarrow HM_i[i,j] + 1 \\ m \text{ est estampillé : } EM_m = HM_i \end{array} \right.$
- Réception de (m, EM_m) en provenance p_j :

Comment HM_i est-elle modifiée ? (1/2)

- Évènement local à p_i : $HM_i[i,i] \leftarrow HM_i[i,i] + 1$
- Émission de m vers p_j : $\begin{cases} HM_i[i,i] \leftarrow HM_i[i,i] + 1 \\ HM_i[i,j] \leftarrow HM_i[i,j] + 1 \\ m \text{ est estampillé : } EM_m = HM_i \end{cases}$
- Réception de (m, EM_m) en provenance p_j :

Toutes les informations sont connues pour assurer la délivrance causale

Comment HM_i est-elle modifiée ? (2/2)

- Réception de (m, EM_m) en provenance p_j :
  - Quand un message peut-il être délivré ?
  - Lors de la délivrance du message (m, EM_m) en provenance de p_j , mettre à jour l'horloge

Comment HM_i est-elle modifiée ? (2/2)

- Réception de (m, EM_m) en provenance p_j :
  - Quand un message peut-il être délivré ? quand tous les messages antérieurs à lui ont été délivrés, i.e :
    - Respect de l'ordre FIFO sur le canal $j \rightarrow i$: $EM_m[j,i] = HM_i[j,i]$
    - Respect de l'ordre de réception : $EM_m[k,i] = HM_i[k,i], \forall k \neq i$
  - Lors de la délivrance du message (m, EM_m) en provenance de p_j , mettre à jour l'horloge

Comment HM_i est-elle modifiée ? (2/2)

- Réception de (m, EM_m) en provenance p_j :
  - Quand un message peut-il être délivré ? quand tous les messages antérieurs à lui ont été délivrés, i.e :
    - Respect de l'ordre FIFO sur le canal $j \rightarrow i$: $EM_m[j,i] = HM_i[j,i]$
    - Respect de l'ordre de réception : $EM_m[k,i] = HM_i[k,i], \forall k \neq i$
  - Lors de la délivrance du message (m, EM_m) en provenance de p_j , mettre à jour l'horloge
    - $HM_i[i,i] \leftarrow HM_i[i,i] + 1$
    - $HM_i[i,j] \leftarrow HM_i[i,j] + 1$
    - pour tout $k \neq i$, pour tout $\ell \neq i$:$HM_i[k,\ell] \leftarrow \max(HM_i[k,\ell], EM_m[k,\ell])$

illustration (début)

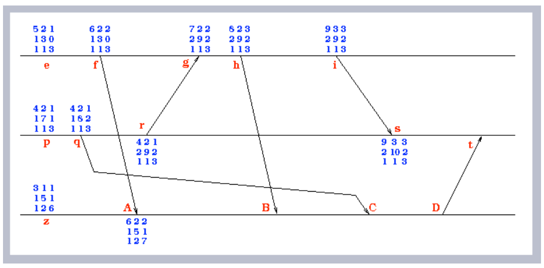


Fig.: datation avec l'horloge matricielle

illustration (attention à la délivrance des messages)

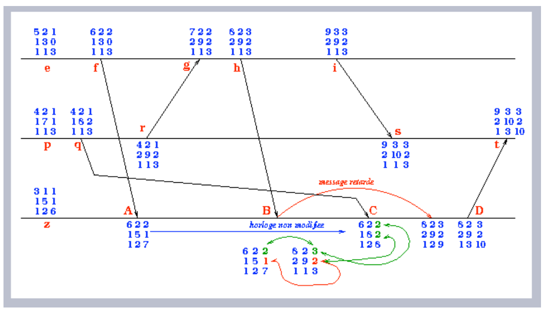


Fig.: datation avec l'horloge matricielle

En résumé

- Horloge scalaire :
  HL_i : ce que p_i connaît du système (nbre d'évènements)
- Horloge vectorielle :
  $HV_i[j]$:ce que p_i connaît du site p_j
- Horloge matricielle :
  $HM_i[j,k]$:ce que p_i connaît de la connaissance du site p_j sur le site p_k

Plan

État d'un système réparti

- Notion non-triviale : observateur universel ayant un accès instantané à l'ensemble de systèmes

  Irréalisable
- notion de coupure : capture l'état du dernier évènement avant "la photo" sur chaque site.

Coupure

Définition
Une coupure C est l'ensemble des d'évènements tels que

Si $e \in C$ et (e' précède localement e) , alors $e' \in C$

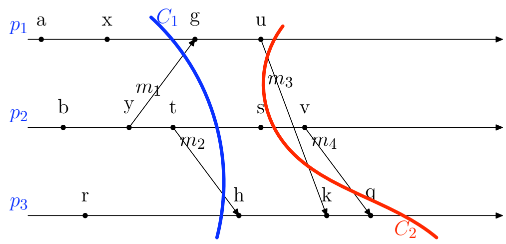


FIG.: la coupure C_1 contient les évènements $\{a, x, b, y, t, r\}$

Coupures cohérentes

Définition
Une coupure cohérente C est une coupure cohérente si :

$$(e \in C) \text{ et } (e' \rightsquigarrow e) \Longrightarrow e' \in C$$

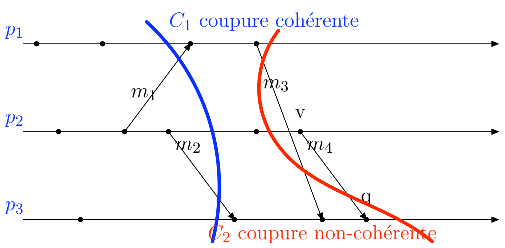


FIG.: C_2 non-cohérente car $v \notin C_2$ et $q \in C_2$

Propriétés sur les coupures cohérentes

Définition
Une coupure cohérente C est une coupure cohérente si :

$$(e \in C) \text{ et } (e' \rightsquigarrow e) \Longleftarrow e' \in C$$

Caractérisation des coupures cohérentes

Soit c_i les évènements définissant la coupure : $C = (c_1, \ldots, c_n)$.
Propriété : $(\forall i, j,\ c_j \mid\mid c_i) \Leftrightarrow C$ est une coupure cohérente

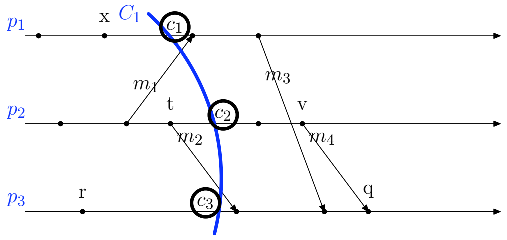


FIG.: $C_1 = (c_1, c_2, c_3)$

Caractérisation des coupures cohérentes

Soit c_i les évènements définissant la coupure : $C = (c_1, \ldots, c_n)$.
Propriété : $(\forall i, j,\ c_j \mid\mid c_i) \Leftrightarrow C$ est une coupure cohérente

Démonstration.

- Soit C est une coupure cohérente. Par contradiction, supposons $\exists i \neq j,\ c_i \rightsquigarrow c_j$.
  - $c_i \rightsquigarrow a_i \rightsquigarrow a_j \rightsquigarrow c_j$ (message entre les deux sites)
  - Comme $a_i \rightsquigarrow c_j$, on a $a_i \in C$.
  - Ceci est contradictoire avec le fait que $c_i \rightsquigarrow a_i$
- Soit $C = (c_1, \ldots, c_n)$ avec $\forall i, j,\ c_j \mid\mid c_i$.
  - Soit $a_i \rightsquigarrow c_i$, et $a_j \rightsquigarrow a_i$
  - Alors $a_j \rightsquigarrow c_i$, et $a_j \in C$
  - Supposons que $c_j \rightsquigarrow a_j$: on aurait $c_j \rightsquigarrow a_j \rightsquigarrow a_i \rightsquigarrow c_i$ (Contradictoire avec le fait que $c_j \mid\mid c_i$).
  - Donc C est une coupure cohérente.

□

Enregistrement l'état d'un système

- Motivation :
  - Observation, détection de propriétés.
  - reprise en cas de panne
- Contraintes :
  - L' état enregistré doit être cohérent
  - Le coût de l'enregistrement doit être raisonnable
- Difficulté :
  - opérations locales /propriété globale

Protocole de Chandy-Lamport [1985]

Hypothèses :

- les canaux de communication entre processus sont FIFO
- Un seul processus décide de lancer la procédure d'enregistrement

Objectif :

enregistrement d'un état cohérent en un temps fini

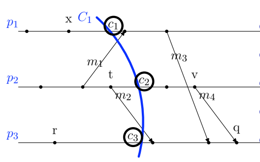
sous quelle forme ?

Définition d'un état.

L'état d'un système de réparti est

- l'ensemble des états de chacun de ses sites
- et l'ensemble des états de chacun de ses canaux de communications

pouvant se déduire des déduire de l'état des sites

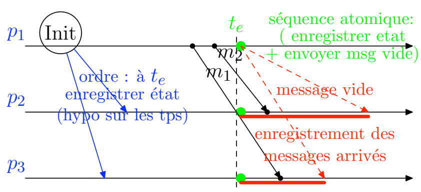


- état de p_1 :x
- état de p_2 : t
- état de p_3 : r
- état de $p_2 \rightarrow p_3$: m_2
- état de $p_2 \rightarrow p_1$: m_1

Première version :(1/2)

Hypothèses :

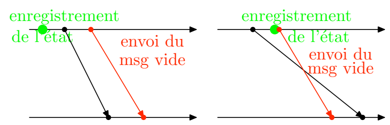
- horloge globale HR (temps réel).
- délai de transmission et rapport des vitesses des sites bornés.



Première version :(2/2)

Propriétés : la coupure C définit au tps t_e est une coupure cohérente

- si $e \in C$ et si $e' \rightsquigarrow e$ (localement) alors $HR(e') < HR(e)$ et $e' \in C$
- L'état du canal est correct du
  - au fait de l'atomicité des instructions `enregistrement état ; envoi message vide`
  - et du fait que le canal de communication est FIFO



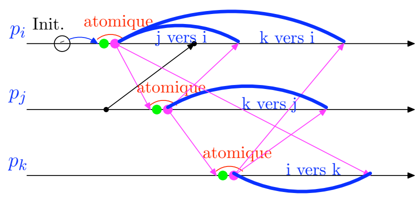
la non-atomicité   canal non FIFO

Deuxième version :(1/3)

Algorithme sur le site p_i :

1. 1ière réception du marqueur depuis p_j
   enregistrer état de p_i
   envoyer le marqueur à tous ses voisins   atomicité
   état du canal $j \rightarrow i := \emptyset$
   enregistrer les msgs sur les canaux entrants
2. Réception suivante du marqueur depuis p_k :
   état du canal $k \rightarrow i :=$ msg reçus depuis l'enregistrement.
   ne plus enregistrer les msg provenant de p_k

Deuxième version :(2/3)



Deuxième version :(3/3)

Propriétés : la coupure C ainsi calculée est une coupure cohérente

Démonstration.

En exercice ?

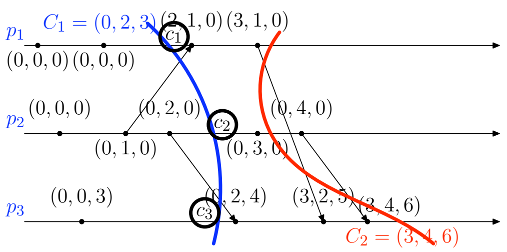
$\square$

Coupures cohérentes et horloges vectorielles

- La date de la coupure $C = (c_1, \ldots, c_n)$ est définie par

$$VH(C) = sup(VH(c_1), \ldots, VH(c_n))$$

- Illustration

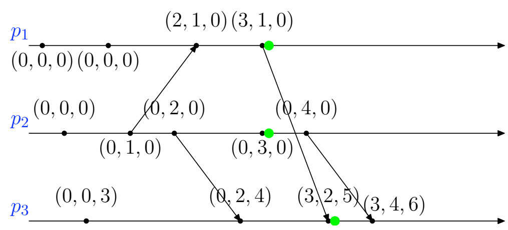


Condition pour les coupures soient cohérentes.

Propriété : C cohérente $\Leftrightarrow VH(C) = (VH(c_1)[1], \ldots, VH(c_n)[n])$

Démonstration.

- Soit C cohérente. $\forall i \neq j$, $VH(c_i)[i] \geq VH(c_j)[i]$.
  - $VH(c_i)[i]$ modifié par un évènement local ou d'un msg passé



$\square$

Suite de la démonstration

- Soit $C = (c_1, \ldots, c_n)$ non-cohérente.
  - $\exists i, j$ tels qu'un message m de p_i émis après C a été reçu par p_j avant C. Soit $EV(m)$ son estampille
  - Alors $VH(c_i)[i] < EV(m)[i] \leq VH(c_j)[i]$
  - Donc $(VH(c_1)[1], \ldots, VH(c_n)[n]) < VH(C)$