TD sur les horloges logiques.

La diffusion est l'opération qui consiste pour un site donné à envoyer un même message à tous les autres sites d'un système. Nous considérons un système constitué de n sites $P_0, P_1, \ldots, P_{n-1}$ tous interconnectés.

Correction : Ce td est construit à partir de la page web suivante :

- http ://www.pps.jussieu.fr/ rifflet/enseignements/AlgoProgSysRep/abcast.html
- http ://www.pps.jussieu.fr/ rifflet/enseignements/AlgoProgSysRep/cbcast.html
- Plus généralement de http ://www.pps.jussieu.fr/ rifflet/enseignements/AlgoProgSysRep/index.html

Exercice : Diffusion ne respectant pas l'ordre FIFO des messages

On suppose maintenant que les sites sont tous fiables, mais que le réseau de communications peut ne pas respecter l'ordre FIFO. On suppose maintenant que P_0 diffuse plusieurs messages vers les autres sites. Proposer une méthode pour garantir que ces sites traitent les messages en respectant l'ordre d'envoi par P_0.

Correction :

Description des variables

- `num_envoi(p) le numéro d'envoi`
- `seq(i) signifie que le noeud` i `a reçu le message de numéro` $1, \ldots, seq(i) - 1$.

Code du site p

```
Procédure Init
  num_envoi(p) := 0;

Procédure diffuser (M)
  num_envoi(p) := num_envoi(p) + 1 ;
  Pour tout noeud q dans V différent de p faire
        Envoyer(<M, num_envoi(p) >) à q ;
```

Code du site $u \neq p$

```
Procédure Init
  seq(u) := 1 ;


Lors de la réception de <M, num_envoi(p) > de p
  Stocker (M) ;
  Attendre (num_envoi(p)= seq(u)) ;
  Délivrer(M) ;
  seq(u) := seq(u) + 1 ;
  Détruire(M) ;
```

Inconvénient du protocole

- un site i peut avoir à stocker beaucoup de messages avant de pouvoir les délivrer et de pouvoir les détruire (par exemple si un des messages est très lent par rapport aux suivants et arrive bien après eux).

– Le numéro de séquence des messages croît au delà de toute limite raisonnable si p diffuse beaucoup de messages. C'est un problème de taille de messages.
– Ici on utilise explicitement le fait que le réseau ne perd pas de message. Dans le cas contraire,le protocole peut être bloqué et ne plus délivrer de messages.

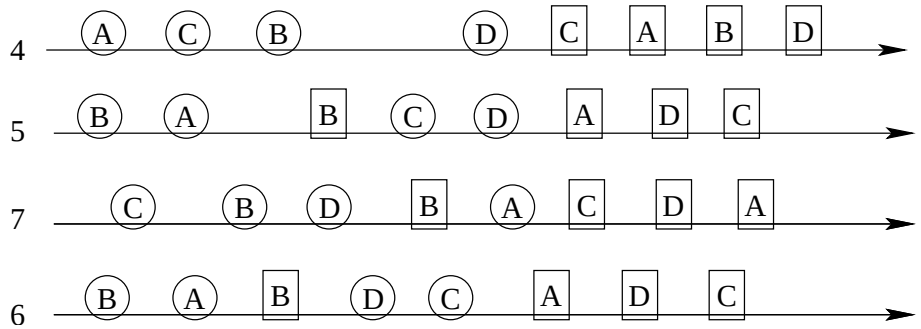
Pour résoudre les deux premiers points on peut mettre en place un système `d'acquittements dans une fenêtre de taille t fixée`. Dans ce système, le site p fait au plus t diffusions de suite avant de recevoir des acquittements. Chaque fois qu'un site i a pu délivrer un message, il envoie un acquittement à p, comprenant le numéro du message acquitté. Lorsque p a reçu les acquittements de tous les sites, pour les derniers messages envoyés non encore acquittés, il peut continuer à diffuser. Dans ce protocole, les numéros de séquence des messages diffusés et les numéros de séquence en réception sont construits modulo t, en commençant à 0. Du coup, les numéros de séquence ne dépassent pas une certaine valeur et le deuxième problème est résolu. Chaque site n'aura à stocker qu'au plus t messages avant de les détruire et si t est suffisamment petit, le premier problème est résolu.

Exercice : diffusions respectant l'ordre total (ABCAST).

Construire un algorithme de diffusion garantissant que les messages sont délivrés en respectant un ordre total. Le protocole ABCAST utilise un mécanisme d'estampillage scalaire. C'est un protocole en deux phases.

– Chaque message contient une estampille qui peut être provisoire ou définitive (choisit par le l'émetteur de la diffusion).
– A la réception d'un message, chacun des sites du groupe lui attribue une estampille provisoire (à l'aide de son horloge logique). Le message reçu est rangé dans une file d'attente dans le site et son estampille est renvoyé à l'émetteur.
– L'émetteur choisit l'estampille définitive une fois qu'il a reçu toutes les estampilles proposées par chacun du membre du groupe. puis la transmet à tous les membres du groupe.
– Lors de la réception d'un message ayant une estampille définitive,. . ..

1. Terminer d'écrire l'algorithme et appliquer l'algorithme sur la figure 1.



2. Écrire plus formellement l'algorithme.

Exercice : diffusions respectant l'ordre causal (CBCAST).

L'objectif de cette exercice est de construire un algorithme de diffusions garantissant que les messages sont délivrés en respectant l'ordre causal.

Rappel : Un événement sera soit l'envoi d'un message, soit la réception d'un message, soit un événement interne. L'ordre suivant est l'ordre partiel sur les événements (ordre causal). Si a et b sont deux événements, a précède b ($a \rightarrow b$) 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 $a \rightarrow c$ et $c \rightarrow b$.

Définition : Si m_1 et m_2 sont deux messages, on dira que m_1 précède immédiatement m_2, noté $m_1 \rightsquigarrow m_2$ si,

- $Envoyer(m_1) \rightarrow Envoyer(m_2)$,
- il n'existe pas de message m tel que $Envoyer(m_1) \rightarrow Envoyer(m)$ et $Envoyer(m) \rightarrow Envoyer(m_2)$.

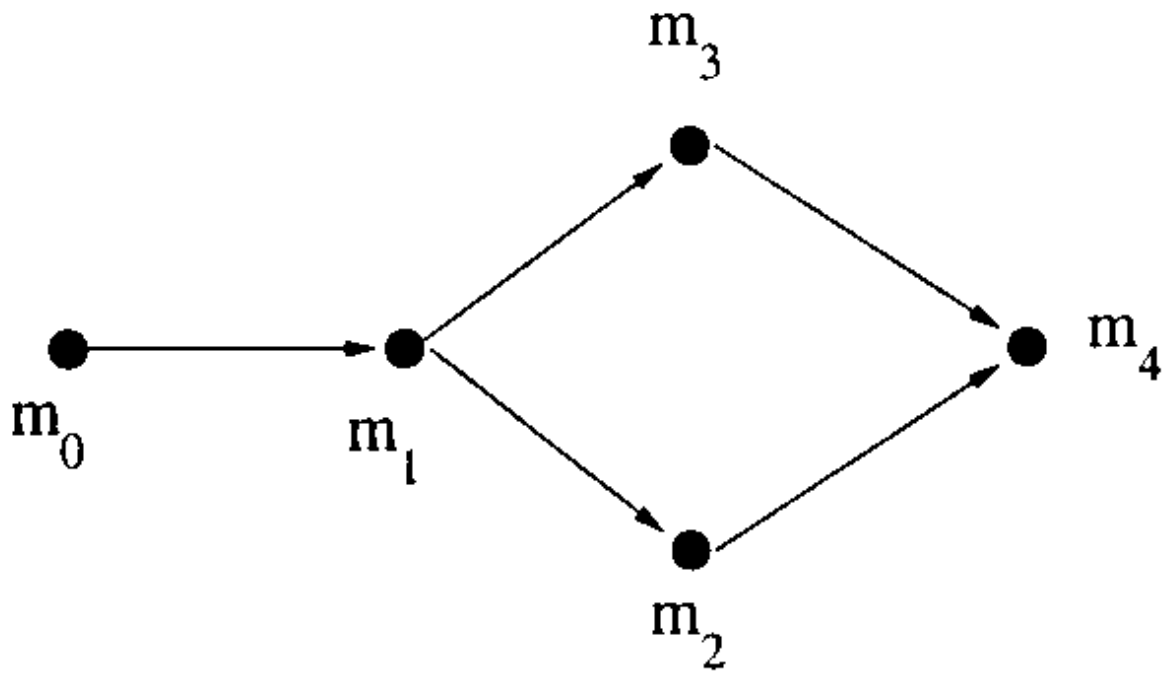
Objectif : définir un protocole de diffusion respectant l'ordre causal

Correction : le but est le suivant :

Si $Envoyer(m_1) \rightarrow Envoyer(m)$ (délivrer en i), alors on veut qu'en site i, Délivrer (m_1) avant Délivrer(m_2)

1. Sur l'exemple de la page 4, définissez les messages qui précèdent chacun des messages. On suppose pour simplifier que tout message qui arrive à sa destination est délivré au moment de sa réception.

Correction :



Chaque site P_i possède
- une variable locale num_envoi_i : numéro du dernier message diffusé par le site P_i.
- un tableau de n cases DEL_i :

  $DEL_i[j] = d \Leftrightarrow$ le dernier message diffusé à partir de P_j et délivré à P_i avait pour numéro d.

Chaque message sera identifié par la paire (id, num_envoi) où id est l'identité de l'envoyeur et num_envoi le numéro de ce message lorsqu'il a été envoyé par le site P_{id}. En outre, chaque message M transportera un ensemble CB_M (barrière causale) des identificateurs des messages qui précédent immédiatement M.

2. Lorsqu'un site P_i reçoit un message M avec de telles données, sous quelles conditions peut-il délivrer M en prenant en compte les contraintes de précédente immédiates ?
   Correction : la condition est : $\forall (k,d) \in CB_M : d \leq DEL_i[k]$. (pour tout message de diffusion dont son origine est k et son numéro est d).

   Cela exprime que tous les prédécesseurs immédiats de M ont été déjà délivrés.

3. Écrivez le protocole de diffusion pour qu'il respecte les contraintes suivantes :
   Correction :

   `Procédure d'initialisation :`
   $CB(i) = \emptyset$;
   $num_envoi(i) = 0$;

   `Procédure diffuser(M) :`
   $num_envoi(i) = num_envoi(i) + 1$;

`Pour tout` $(x \in V)$ `faire Envoyer(`$< M, num_envoi(i), CB(i) >$`) à x ;`
$CB(i) = (i, num_envoi(i))$;

`Lors de la réception de` $< M, num_envoi_M, CB_M >$ `de` j
`Attendre` $(\forall (k,d) \in CB_M : d \leq DEL_i[k])$.
$DEL(i)[j] = num_envoi_M$;
$CB(i) = CB(i) - CB_M \cup (j, num_envoi_M)$;`(*)`
`Délivrer (M)`

$(*)$ A faire de manière atomique.

- m est un message identifié par (k,d) et un message m_1 tel que $m \rightsquigarrow m_1$, on a $(k,d) \in CB_{m_1}$.
  Correction : Soit i l'envoyeur de m_1. Comme on a $m \rightsquigarrow m_1$, deux cas sont à considérer. Avant d'envoyer m_1,
  - m a été délivré à i
  - m a été envoyé par i,

  Dans les deux cas, le site i met à jour CB_i, en prenant en compte m. La délivrance d'un autre message m' entre m et l'envoi de m_1 soit ne change rien soit est impossible (car $m \rightsquigarrow m_1$).
- les délivrances de messages respectent l'ordre causal.
  Correction : Considérons deux messages m_0 et m_x tels que :
  - Envoyer$(m_0) \rightarrow$ Envoyer(m_x)
  - et m_0 et m_x sont délivré à i.

  On montre ceci par récurrence sur la longueur ℓ du chemin entre m_0 et m_x dans le graphe de précédente immédiate $(\rightsquigarrow)$ des messages.
  Cas de base $\ell = 1$: $m_0 \rightsquigarrow m_x$ et d'après le résultat de la question précédente $m_0 \in CB_{m_x}$. Or le site i va attendre d'avoir délivrer m_0 avant de délivrer m_x. (C'est l'attente de l'algorithme lors de la réception de m_x).
  hypothèse de récurrence : Lorsque le chemin de causalité est de longueur strictement inférieure à $\ell \geq 2$, les contraintes de précédente sont respectées.
  Considérons maintenant un chemin de longueur $\ell + 1$ de m_0 à m_x

  $$m_0 \rightsquigarrow m_1 \rightsquigarrow \dots m_\ell \rightsquigarrow m_x$$

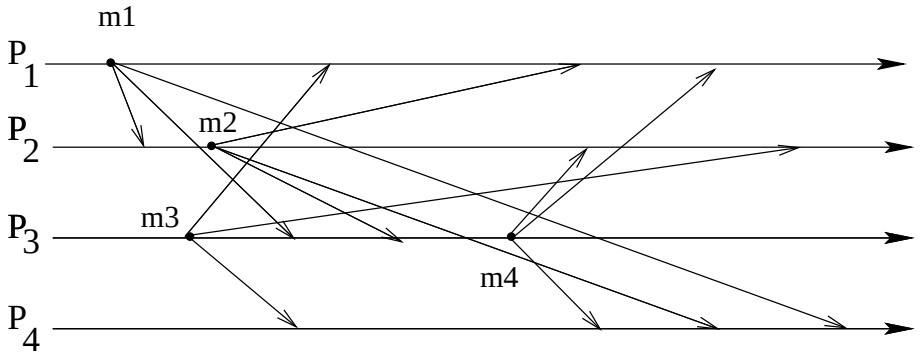
  Ainsi par hypothèse : tous les messages $m_0, m_1, \dots m_\ell$ délivré à i le sont suivant l'ordre causal. m_ℓ est délivré à i. Comme $m_\ell \rightsquigarrow m_x$, m_ℓ est délivré avant m_x (cas de base) . Ainsi m_0 est délivré avant m_x en i.

Objectif : définir un protocole de diffusion respectant l'ordre causal utilisant la notion d'horloge vectorielle.

L'objectif est de modifier l'algorithme précédent en remplaçant les barrières causales par les horloges vectorielles.

1. Que peut-on dire sur les "dates" à laquelles les messages m_1, m_2 tels que $Envoyer(m_1) \rightarrow Envoyer(m_2)$, ont été émis ?
   Correction :
   - $send(m_1) \rightarrow send(m_2)$, alors $VT(m_1) < VT(m_2)$.
   - si i émet le message m_1, $VT(m_1)[i] \leq VT(m_2)[i]$.
2. Lorsqu'un site P_i reçoit un message M, sous quelles conditions peut-il délivrer M en prenant en compte les horloges vectorielles ?
   Correction :

   $$\forall k : 1 \dots, n \{ \begin{array}{ll} VT_M[k] = VT_i[k] + 1 & \text{si } k = j \\ VT_M[k] \leq VT_i[k] & \text{sinon} \end{array}$$



3. Écrire l' algorithme et appliquer l'algorithme sur la figure 2.

Correction :

– $VT(i)$: horloge vectorielle du site i.

`Procédure d'initialisation :`
`Tous les éléments de` $VT(i)$ `sont initialisés à 0 ;`

`Procédure diffuser(M) sur le site` P_i `:`
`VT(i)[i] = VT(i)[i] +1 ;`
`Pour tout (`$x \in V$`) faire Envoyer(<M,VT(i)>) à x ;`

`Lors de la réception de` $< M, VT_M >$ `de` j `sur le site` i
`Attendre jusqu' à`

$$\forall k : 1 \ldots, n\{ \begin{array}{ll} VT_M[k] = VT_i[k] + 1 & \texttt{si } k = j \\ VT_M[k] \leq VT_i[k] & \texttt{sinon} \end{array}$$

`Délivrer (M)`
$VT_i[j] = VT_i[j] + 1$

Correction : Considérons que le site j reçoivent deux messages m_0 et m_x tel que Envoyer(m_0) $\rightarrow$ Envoyer(m_x).

case 1. m_0 et m_x ont même émetteur i. Par construction, nous avons $VT(m_0) < VT(m_x)$. Donc m_0 est délivré après m_x

case 2. m_0 et m_x ont pour émetteur i, et i'. Raisonnons par récurrence sur le nombre de message reçus ℓ par le site j que m_x ne peut pas être délivré avant m_0 .

Il est à noter que par définition, seul émission d'un message sur un site i et la reception $\oplus$ la délivrance sur un site j peut augmenter d'horloge vectorielle $VT(i)[j]$.

Cas de base : $\ell = 1$ $m_0 \rightsquigarrow m_x$

D'après la question précédente, on a

– Si $Envoyer(m_1) \rightarrow Envoyer(m_2)$, alors $VT(m_1) < VT(m_2)$.

– Si $Envoyer(m_1) \rightarrow Envoyer(m_2)$ et si i émet le message m_1, $VT(m_1)[i] \leq VT(m_2)[i]$.

Si m_x est délivré, alors on a

$$\forall k : 1 \ldots, n\{ \begin{array}{ll} VT_{m_x}[k] = VT_j[k] + 1 & \text{si } k = i' \\ VT_{m_x}[k] \leq VT_j[k] & \text{sinon} \end{array}$$

En particulier, ceci signifie que $VT_{m_x}[i] = VT_j[i]$. Seul un message m émis par i et délivré sur le site j a pu modifié l'élément $VT_j[i]$.

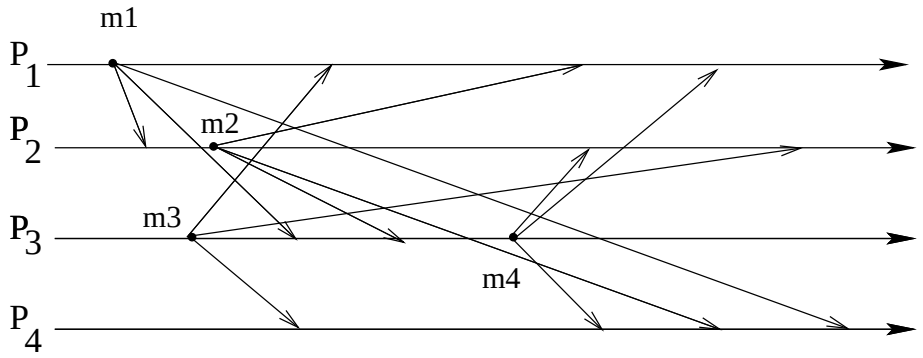
– si $m \rightarrow m_0$, alors $VT_m < VT_{m_0}$ et $VT_m[i] < VT_{m_0}[i]$, impossible.

– si $m_0 \rightarrow m$, alors impossible (cas précédent).

hypothèse de récurrence : Lorsque le chemin de causalité est de longueur strictement inférieure à $\ell \geq 2$, les contraintes de précédente sont respectées.

Considérons maintenant un chemin de longueur $\ell + 1$ de m_0 à m_x

$$m_0 \rightsquigarrow m_1 \rightsquigarrow \ldots m_\ell \rightsquigarrow m_x$$



Correction :

– $VT(i)$: horloge vectorielle du site i.

`Procédure d'initialisation :`
  `Tous les éléments de` $VT(i)$ `sont initialisés à 0 ;`

`Procédure diffuser(M) sur le site` P_i `:`
  `VT(i)[i] = VT(i)[i] +1 ;`
  `Pour tout (`$x \in V$`) faire Envoyer(<M,VT(i)>) à x ;`

`Lors de la réception de` $< M, VT_M >$ `de` j `sur le site` i
  `Attendre jusqu' à`

$$\forall k : 1 \ldots, n\{ \begin{array}{ll} VT_M[k] = VT_i[k] + 1 & \texttt{si } k = j \\ VT_M[k] \leq VT_i[k] & \texttt{sinon} \end{array}$$

  `Délivrer (M)`
  $VT_i[j] = VT_i[j] + 1$

Correction : Considérons que le site j reçoivent deux messages m_0 et m_x tel que Envoyer(m_0) $\rightarrow$ Envoyer(m_x).

case 1. m_0 et m_x ont même émetteur i. Par construction, nous avons $VT(m_0) < VT(m_x)$. Donc m_0 est délivré après m_x

case 2. m_0 et m_x ont pour émetteur i, et i'. Raisonnons par récurrence sur le nombre de message reçus ℓ par le site j que m_x ne peut pas être délivré avant m_0 .

Il est à noter que par définition, seul émission d'un message sur un site i et la reception $\oplus$ la délivrance sur un site j peut augmenter d'horloge vectorielle $VT(i)[j]$.

Cas de base : $\ell = 1$ $m_0 \rightsquigarrow m_x$

D'après la question précédente, on a

– Si $Envoyer(m_1) \rightarrow Envoyer(m_2)$, alors $VT(m_1) < VT(m_2)$.

– Si $Envoyer(m_1) \rightarrow Envoyer(m_2)$ et si i émet le message m_1, $VT(m_1)[i] \leq VT(m_2)[i]$.

Si m_x est délivré, alors on a

$$\forall k : 1 \ldots, n\{ \begin{array}{ll} VT_{m_x}[k] = VT_j[k] + 1 & \text{si } k = i' \\ VT_{m_x}[k] \leq VT_j[k] & \text{sinon} \end{array}$$

En particulier, ceci signifie que $VT_{m_x}[i] = VT_j[i]$. Seul un message m émis par i et délivré sur le site j a pu modifié l'élément $VT_j[i]$.

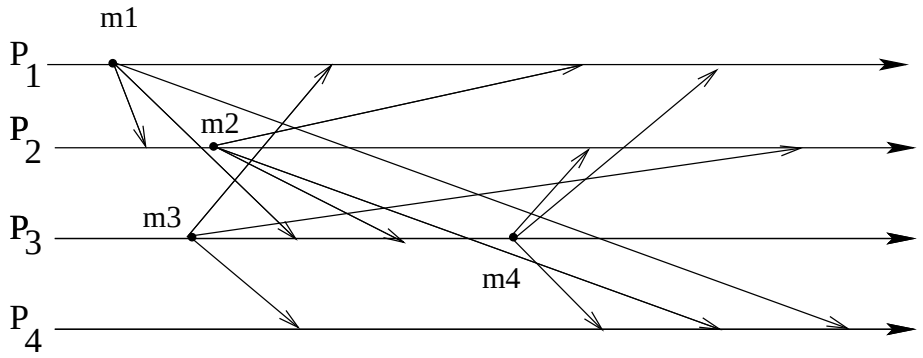
– si $m \rightarrow m_0$, alors $VT_m < VT_{m_0}$ et $VT_m[i] < VT_{m_0}[i]$, impossible.

– si $m_0 \rightarrow m$, alors impossible (cas précédent).

hypothèse de récurrence : Lorsque le chemin de causalité est de longueur strictement inférieure à $\ell \geq 2$, les contraintes de précédente sont respectées.

Considérons maintenant un chemin de longueur $\ell + 1$ de m_0 à m_x

$$m_0 \rightsquigarrow m_1 \rightsquigarrow \ldots m_\ell \rightsquigarrow m_x$$



Correction :

– $VT(i)$: horloge vectorielle du site i.

`Procédure d'initialisation :`
  `Tous les éléments de` $VT(i)$ `sont initialisés à 0 ;`

`Procédure diffuser(M) sur le site` P_i `:`
  `VT(i)[i] = VT(i)[i] +1 ;`
  `Pour tout (`$x \in V$`) faire Envoyer(<M,VT(i)>) à x ;`

`Lors de la réception de` $< M, VT_M >$ `de` j `sur le site` i
  `Attendre jusqu' à`

$$\forall k : 1 \ldots, n\{ \begin{array}{ll} VT_M[k] = VT_i[k] + 1 & \texttt{si } k = j \\ VT_M[k] \leq VT_i[k] & \texttt{sinon} \end{array}$$

  `Délivrer (M)`
  $VT_i[j] = VT_i[j] + 1$

Correction : Considérons que le site j reçoivent deux messages m_0 et m_x tel que Envoyer$(m_0) \rightarrow$ Envoyer(m_x).

case 1. m_0 et m_x ont même émetteur i. Par construction, nous avons $VT(m_0) < VT(m_x)$. Donc m_0 est délivré après m_x

case 2. m_0 et m_x ont pour émetteur i, et i'. Raisonnons par récurrence sur le nombre de message reçus ℓ par le site j que m_x ne peut pas être délivré avant m_0 .

Il est à noter que par définition, seul émission d'un message sur un site i et la reception $\oplus$ la délivrance sur un site j peut augmenter d'horloge vectorielle $VT(i)[j]$.

Cas de base : $\ell = 1$ $m_0 \rightsquigarrow m_x$

D'après la question précédente, on a

– Si $Envoyer(m_1) \rightarrow Envoyer(m_2)$, alors $VT(m_1) < VT(m_2)$.

– Si $Envoyer(m_1) \rightarrow Envoyer(m_2)$ et si i émet le message m_1, $VT(m_1)[i] \leq VT(m_2)[i]$.

Si m_x est délivré, alors on a

$$\forall k : 1 \ldots, n\{ \begin{array}{ll} VT_{m_x}[k] = VT_j[k] + 1 & \text{si } k = i' \\ VT_{m_x}[k] \leq VT_j[k] & \text{sinon} \end{array}$$

En particulier, ceci signifie que $VT_{m_x}[i] = VT_j[i]$. Seul un message m émis par i et délivré sur le site j a pu modifié l'élément $VT_j[i]$.

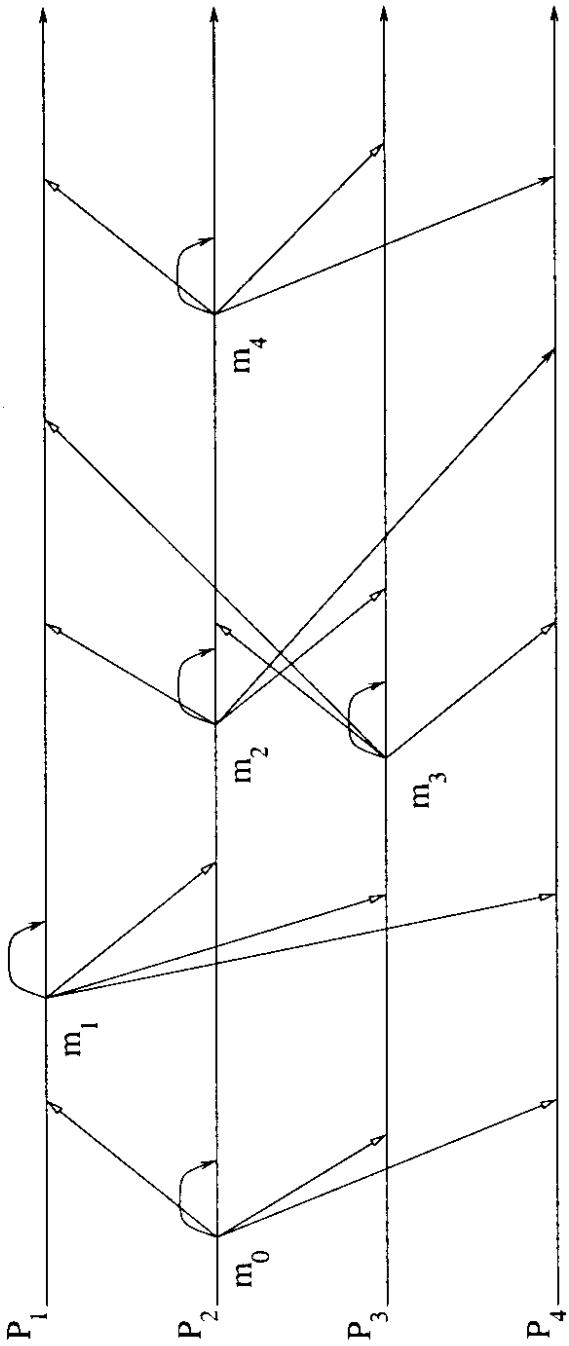
– si $m \rightarrow m_0$, alors $VT_m < VT_{m_0}$ et $VT_m[i] < VT_{m_0}[i]$, impossible.

– si $m_0 \rightarrow m$, alors impossible (cas précédent).

hypothèse de récurrence : Lorsque le chemin de causalité est de longueur strictement inférieure à $\ell \geq 2$, les contraintes de précédente sont respectées.

Considérons maintenant un chemin de longueur $\ell + 1$ de m_0 à m_x

$$m_0 \rightsquigarrow m_1 \rightsquigarrow \ldots m_\ell \rightsquigarrow m_x$$


P1
P2
P3
P4
m0
m1
m2
m3
m4