Chapitre 2

Le temps logique dans les systèmes distribués

Objectif

- **Déterminer un ordre des événements :**
  « Est-ce qu'un message m_1 en provenance du site 1 a été envoyé avant ou après l'envoi d'un message m_2 en provenance du site 2 ? »
- Exemple du « make » distribué sous Unix [Tanenbaum]
- **Concepts d'ordre et de délivrance causale [Rifflet]**
- **Horloges et estampilles : [Lamport]**
  - scalaires
  - vectorielles
  - matricielles (TD)

1. Ordre causal

1.1. Relation de précédence directe

Un événement e ***précède directement*** un événement e' : **$e \rightarrow e'$** , si l'une des deux conditions suivantes est vraie :

1. les événements e et e' ont lieu dans le même processus, et e se produit exactement avant e' sur ce processus.

1. l'événement e correspond à l'envoi d'un message par un processus, et l'événement e' correspond à la réception du même message par un autre processus.

1.2. Relation de précédence causale

- Elle est définie comme la **fermeture réflexive et transitive** de la relation de précédence directe
- Un événement e **précède causalement** un événement e' : $e \overset{\bullet}{\rightarrow} e'$, si et seulement si :
  1. ou bien e = e' (réflexivité)
  2. ou bien $\exists\ e_1, e_2, \ldots, e_m$ tels que $e_1 = e$ et $e_m = e'$ et $\forall i,\ e_i \rightarrow e_{i+1}$ (transitivité)

- La relation de précédence causale définit un ***ordre partiel*** des événements.

- Si deux événements x et y se produisent dans des processus différents, il se peut qu'aucune des deux relations x $\xrightarrow{\bullet}$ y et x $\xrightarrow{\bullet}$ y ne soit vraie.
- De tels événements x et y non comparables par la relation de précédence causale, sont dits ***concurrents*** : on note **x||y**
- A un événement e on peut associer trois ensembles d'événements :
  - **Passé(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 dans l'ordre causal.

=> Exemple

2. Livraison causale

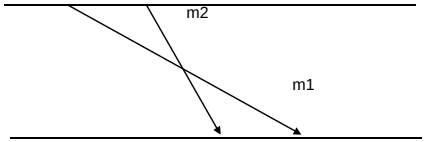
- La livraison **d'un message m**, notée **del(m)**, est l'opération consistant à le rendre accessible aux applications utilisatrices
- La livraison d'un message reçu pourra être **retardée** lors de sa réception pour garantir un ordre de livraison souhaité

Ordre de livraison FIFO

Si deux messages sont envoyés successivement depuis un même site S_i vers un même destinataire Sj, alors le premier sera délivré sur le site Sj avant le second, i. e :

$\mathbf{snd_i(m_1,j)} \xrightarrow{\bullet} \mathbf{snd_i(m_2,j)} \Rightarrow \mathbf{del_j(m_1)} \xrightarrow{\bullet} \mathbf{del_j(m_2)}$

Exemple : dans la figure suivante l'ordre FIFO n'est pas garanti



$\Rightarrow$ Pour respecter l'ordre FIFO **la livraison de m_2 doit être retardée** jusqu'à ce que le message m_1 arrive et soit délivré

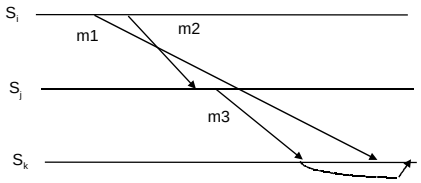
$\Rightarrow$ La couche de communication dispose d'un **tampon ordonné** permettant de mettre les messages en attente

Ordre de livraison causale

- Cette propriété étend l'ordre de livraison FIFO à des communications à destination d'un même site en provenance de plusieurs autres sites.
- Si l'envoi d'un 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, alors le message m_1 sera délivré avant le message m2 sur le site S_k :

  $\mathbf{snd_i(m_1,k)} \xrightarrow{\bullet} \mathbf{snd_j(m_2,k)} \Rightarrow \mathbf{del_k(m_1)} \xrightarrow{\bullet} \mathbf{del_k(m_2)}$

Exemple : la figure suivante illustre un cas où la propriété de livraison causale est violée si les messages sont délivrés dans l'ordre où ils sont reçus



⇒ Pour respecter l'ordre de livraison causale, la **livraison de m_3 doit être retardée** jusqu'à l'arrivée et la livraison du message m_1

3. Horloges et estampilles scalaires

Les processus qui interagissent doivent se « mettre d'accord » sur l'ordre dans lequel les événements se produisent

3.1. Définition et principe des estampilles scalaires

Si les sites datent les événements au moyen d'horloges HL_i (HL_i est l'horloge du site i) et si les horloges sont utilisées pour dater les événements (HL_e est la date de l'événement e), les horloges doivent pouvoir être comparées (ordre noté $\subset$) de telle manière que si :

$$e \dashrightarrow e' \Rightarrow HL_e \subseteq HL_{e'},$$

Principe :

- Chaque site gère un compteur dont la valeur est un entier.
- Sur chaque site ce compteur est initialisé à 0 au lancement du système.
- La valeur de l'horloge logique d'un site est incrémentée chaque fois qu'un **événement local** s'y produit. Un tel événement est :
  - ⇒ soit une opération purement locale,
  - ⇒ soit **l'envoi d'un message** :

    dans ce cas la valeur courante (après incrémentation) de l'horloge de l'émetteur est embarquée avec le message et sert à l'estampiller.

- La réception d'un message permet de **synchroniser l'horloge du récepteur avec celle de l'émetteur** du message (qui est transportée dans le message).
- Le principe consiste à attribuer à l'horloge du récepteur une valeur supérieure à la fois à la valeur courante de l'horloge du site et à celle de l'estampille du message reçu.

Algorithme de Lamport :

⇒ si un événement local se produit sur le site i, HL_i est incrémentée : **HL_i ++**

⇒ si un événement correspondant à **l'envoi d'un message** se produit sur le site i, HL_i est incrémentée, et le message m est envoyé avec la nouvelle valeur de HL_i comme estampille : **$EL_m = HL_i$**

⇒ si un événement correspondant à **la réception d'un message** m d'estampille EL_m se produit sur le site i : **$HL_i = \max(HL_i , EL_m) + 1$**

- Exemple [Tanenbaum] :

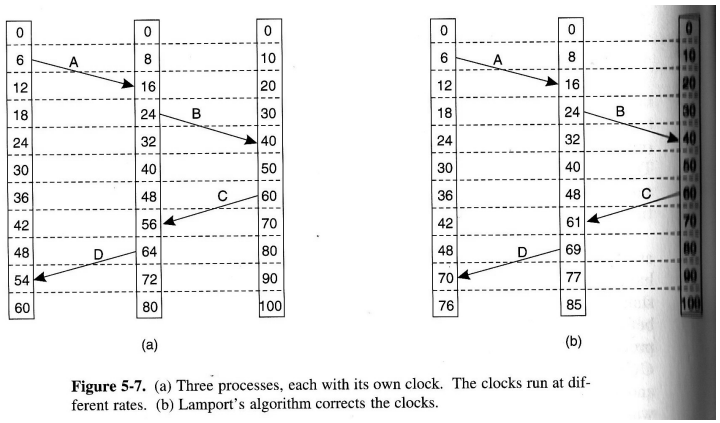


Figure 5-7. (a) Three processes, each with its own clock. The clocks run at different rates. (b) Lamport's algorithm corrects the clocks.

=> Application : TD

3.2. Propriétés des estampilles scalaires

1. L'ordre des événements n'est pas un ordre strict :

plusieurs événements peuvent porter la même valeur.

- Pour rendre cet ordre strict **on adjoint à la valeur de l'horloge logique d'un site l'identification du site**
- L'estampille logique d'un événement HL(e) d'un événement e du site i est un couple (HL_i , i).
- On a (HL_i , i) $\subset$ (HL_j , j) si et seulement si :
  - ou bien $HL_i < HL_j$
  - ou bien $HL_i = HL_j$ et i < j

2. **L'ordre ainsi défini est total :**

il induit une chaîne de tous les événements.

=> exemple

4. Horloges et estampilles vectorielles

4.1. Définition et principe des estampilles vectorielles

Inconvénient des estampilles scalaires :

Ordonnent artificiellement les événements concurrents

Principe des estampilles vectorielles :

- Chaque site gère une horloge vectorielle constituée de n entiers (le système comporte n sites).
- Les messages envoyés par un site sont estampillés en utilisant la valeur courante de l'horloge vectorielle.
- La réception d'un message permet au site récepteur de synchroniser son horloge vectorielle avec celle du site émetteur du message.

4.1. Définition et principe des estampilles vectorielles

- **HV_i** désigne l'horloge vectorielle du site i
- **EV_m** désigne l'estampille vectorielle attribuée au message m lors de son envoi :
  - si un **événement local** se produit sur le site i, HV_i [i] est incrémentée : **HV_i [i] ++**
  - si un **événement correspondant à l'envoi d'un message** se produit sur le site i, HV_i [i] est incrémentée, et le message m est envoyé avec la nouvelle valeur de HV_i comme estampille : **$EV_m = HV_i$**
  - si un message m d'estampille **EV_m** est reçu sur le site **i** :
    - **HV_i [i] est incrémentée,**
    - ← **$\forall j \neq i$, HV_i [j] = max (HV_i [j] , EV_m [j])**

    => exemple

4.2. Propriétés des estampilles vectorielles

1) Propriété fondamentale :

EV_e est l'estampille vectorielle de e :

$\Rightarrow$ **$\forall$ i , EV_e [i] = Card ({e' / e' $\in$ S_i et e' $\dashrightarrow$ e })**

i.e. la valeur de la i-ème composante de EV_e correspond au **nombre d'événements du site S_i** qui appartiennent au **passé de e.**

=> exemple

2) Relation d'ordre sur les estampilles vectorielles :

EV_e est l'estampille vectorielle de e :

$\Rightarrow$ **$EV_e \subseteq Ev_{e'} \Leftrightarrow \forall$ i, EV_e [i] $\leq Ev_{e'}$ [i]**

=> exemple

Cette relation d'ordre **reflète exactement la relation de précédence causale entre événements :**

Si EV_e est l'estampille vectorielle de e :

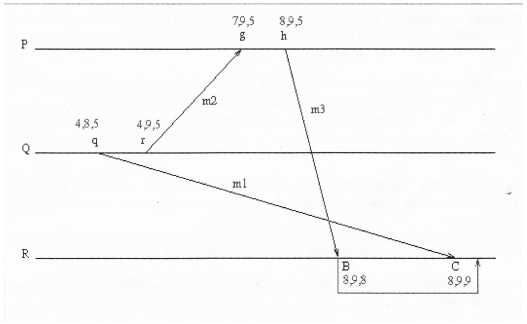
- **e $\dashrightarrow$ e' $\Leftrightarrow$ $EV_e \subseteq Ev_{e'}$**
- **e || e' $\Leftrightarrow$ EV_e || $Ev_{e'}$**

→ Exemple **:** montrer des estampilles inférieures, supérieures et incomparables

Preuve [Charron-Bost] : à faire en TD

Inconvénient des estampilles vectorielles :

=> ne permettent pas de garantir une livraison causale des messages.



- Dans l'exemple ci-dessus le message m3 est reçu trop tôt : pour respecter l'ordre causal de la livraison il ne doit être délivré qu'après le message m1.
- Les estampilles permettent de détecter **(a posteriori !)** la violation de l'ordre causal dans la réception des messages

5. Horloges et estampilles matricielles

5.1. Définition et principe des estampilles matricielles

Inconvénients des estampilles vectorielles :

$\Rightarrow$ ne permettent pas de corriger la défaillance vis-à-vis de la livraison causale des messages

Principe des estampilles matricielles :

- Dans un système de n sites, les horloges d'un site i et les estampilles des événements (et des messages) sont des matrices carrées d'ordre n.
- **HM_i** désigne l'horloge matricielle du site i (**S_i**)
- **EM_m** désigne l'estampille matricielle du message m,

Sur le site S_i, la matrice HM_i va :

- mémoriser le nombre de messages que le site S_i a envoyé aux autres sites => i-ème ligne de la matrice.
- mémoriser, pour chacun des autres sites j, le nombre de messages émis par le site j dont le site i a connaissance => j-ème ligne de la matrice.
- Ainsi sur le site i, la valeur EM_i [j,k] donne le nombre de messages en provenance du site j livrables sur le site k dont le site i a connaissance.

Remarque : La diagonale de HM_i représente la connaissance qu'a le site i du nombre d'événements locaux qui se sont déjà produits sur les différents sites j : correspond à **l'estampille vectorielle.**

La synchronisation des horloges des différents sites est réalisée de la manière suivante :

- lorsqu'un **événement local** se produit sur le site S_i : **HM_i [i,i] est incrémenté**;
- lorsqu'un **message est envoyé du site i vers le site j: HM_i [i,i] et HM_i [i,j] sont incrémentés** ;
- lorsqu'un **message m en provenance du site j est reçu par le site i**, il faut s'assurer que tous les messages envoyés antérieurement vers le site i sont effectivement arrivés. Cela suppose donc que le site i ait reçu :
  - d'une part tous les messages en provenance du site j,
  - d'autre part tous les messages envoyés depuis d'autres sites.

=> Cela correspond aux **conditions suivantes** (à vérifier dans l'ordre):

1. **EM_m [j,i] = HM_i [j,i] + 1** (ordre FIFO sur le canal (j,i))
2. **pour tout k ≠ i et k ≠ j, EM_m [k,i] ≤ HM_i [k,i]** (tous les messages en provenance des sites différents de j ont été reçus).

- Si toutes ces conditions sont vérifiées,le message est **livrable** et l'horloge du site i est mise à jour :
  1. **HM_i [i,i] ++ (incrémentation);**
  2. **HM_i [j,i] ++ (incrémentation);**
  3. **pour le reste de la matrice : HM_i [k,l] = max (HM_i [k,l], EM_m [k,l])**
- *Si les conditions ne sont pas toutes vérifiées,* **la livraison du message est retardée** jusqu'à ce qu'elles le deviennent et l'horloge n'est pas mise à jour.
- La livraison d'un message pourra ainsi provoquer celle des messages arrivés prématurément (retardés).
- Exemple : cf. TD

Exemple : estampilles matricielles