Master STIC Informatique M.S.I – Option 2 SIC
ISIMA 3 – Filières 1, 2 et 5

Systèmes Répartis

Aspects logiciels

E. Mesnard
2010 - 2011

Sommaire

I-1 Introduction - Architecture des Systèmes

Classification de Flynn :

		Flot de Données	
		Unique (S)	Multiple (M)
Flot d'Instructions (contrôles)	Unique (S)	**SISD** (Von Neumann)	**SIMD** (Vectoriel)
	Multiple (M)	**MISD** (Systolic)	**MIMD** → SM Shared Memory (multi-processeur) / **DM Distributed Memory (multi-ordinateur)**

Systèmes Répartis

Flynn, Michael J,
"Some Computer Organizations and Their Effectiveness",
IEEE Transactions on Computing,
Vol C-21, No. 9, Sept 1972, pp 948-960.

I-2 Définition d'un Système Réparti

Système Réparti =

- Ensemble fini de **sites** (machines) interconnectés;
- **Pas de mémoire** commune entre les sites (« loosely coupled »);
- **Pas d'horloge** physique commune;
- Réseau de communication **connexe**; (ils peuvent tous se communiquer)
- La communication se fait par **messages**;
- Aucune machine n'a l'information d'un **état global**;
- Les machines prennent des décisions qu'à partir des informations localement disponibles.

I-3 Critères de conception des Systèmes Répartis

Critère 1 : La **Transparence**

Donner l'illusion que l'ensemble des machines se comporte exactement comme un système monoprocesseur

1. Accès :
   Les utilisateurs ne doivent pas pouvoir dire où sont placées les ressources
2. Concurrence :
   Des accès « simultanés » aux ressources doivent pouvoir se faire, sans interférence entre eux
3. Migration :
   Les processus et données mobiles, sans modification du nom et du chemin d'accès
4. Réplication :
   Les utilisateurs accèdent à des copies multiples des données, sans le savoir
5. Parallélisme :
   Les processus devraient pouvoir s'exécuter en parallèle sans que les utilisateurs le sachent

I-3 Critères de conception des Systèmes Répartis

Critère 2 : La **Fiabilité** et la **disponibilité**

Faire en sorte que l'ensemble des machines soit plus fiable et plus disponible qu'un système monoprocesseur.

Critère 3 : La **Performance**

Faire en sorte que les temps de réponses d'un système réparti soient plus faibles que ceux d'un système monoprocesseur.

Critère 4 : Le **Dimensionnement** et le **facteur d'échelle**

Faire en sorte que les applications qui s'exécutent sur le système réparti s'adaptent automatiquement à la modification du nombre de sites ou du nombre d'applications.

II-1 Temps physique et temps logique

Temps physique :

temps universel (absolu), temps atomique international et temps universel coordonné
mesure du temps dans les ordinateurs : horloges (absolues)
Cf. Paradigme **NTP** « client/serveur »

Temps logique :

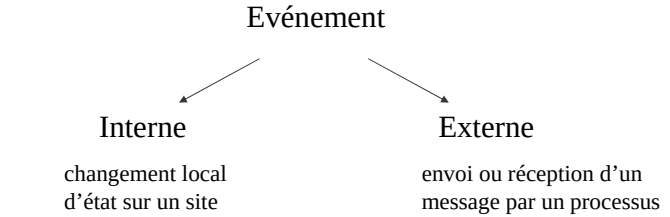
le temps logique est lié à l'ordre causal (la délivrance causale de messages)
définition d'horloges logiques et mécanismes de datation des événements

Car, le système de communication est tel que :

a) le temps de communication est potentiellement long par rapport aux temps de traitement,
b) les temps de transmission sont variables, l'ordre des messages n'est pas nécessairement préservé.
c) et donc, perception différente des mêmes événements (émission ou réception des messages) depuis des sites distincts.

II.2 L'ordre causal – précédence causale

Causal : Ordre correct dans les **événements**



Exemple de causalité respectée :

date d'émission < date de réception

II.2 L'ordre causal – précédence causale

Temps logique :

Comparaison logique d'événements du point de vue de leur exécution

Principe de construction d'horloges logiques :

Interne :

Datation locale par une horloge locale

Externe :

Re-synchronisation d'horloge en vérifiant que l'émission précède toujours la réception

II.2 L'ordre causal – précédence causale

Précédence **directe** :

« a Précède directement b » : $a \overset{\bullet}{\rightarrow} b$

- Soit a et b se sont produits sur le même site, et a est antérieur à b sur ce site,
- Soit a est l'envoi d'un message m depuis un site, et b la réception de ce message m sur le site destinataire.

Précédence **causale** :

« a Précède b » : $a \rightarrow b$

Fermeture réflexive et transitive de la précédence directe

- Réflexivité :
  Soit a = b
- Transitivité :
  Soit $\exists e_1, e_2, ..., e_m$ tel que $e_1 = a$ et $e_m = b$ et $\forall i, e_i \rightarrow e_{i+1}$

II.2 L'ordre causal – précédence causale

Concurrence – indépendance causale : « // »

Aucun des deux événement ne précède causalement l'autre :

$a // b \Leftrightarrow \overline{(a \rightarrow b)}.\overline{(b \rightarrow a)}$

A un événement « a », on associe 3 ensembles d'événements :

1) **Passé(a)** ou **hist(a)** (historique) :
   ensemble des événements antérieurs à a dans l'ordre causal (a appartient à cet ensemble),
2) **Futur(a)** :
   ensemble des événements postérieurs à a dans l'ordre causal (a appartient aussi à cet ensemble),
3) **Concurrent(a)** :
   ensemble des événements concurrents avec a.

II.2 L'ordre causal – précédence causale

Délivrance causale – précédence causale des messages :

La délivrance d'un message est l'opération consistant à rendre accessible le message aux applications clientes.

Exemple : TCP ne rend accessible un caractère que lorsque tous les caractères envoyés précédemment ont été effectivement reçus.

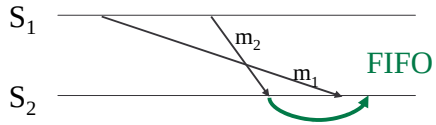
Propriété 1 : Ordre de délivrance **FIFO**

Si deux messages sont envoyés successivement depuis un même site $\mathbf{S_i}$ vers un même destinataire $\mathbf{S_j}$, le premier sera délivré sur le site $\mathbf{S_j}$ avant le second :

$$snd_i(m_1, j) \rightarrow snd_i(m_2, j) \Rightarrow rcv_j(m_1) \rightarrow rcv_j(m_2)$$

$snd_i(m, j)$ envoi (« send ») d'un message m d'un site S_i vers un site S_j.
$rcv_j(m)$ délivrance (« receive ») du message m sur le site S_j.

Contre-exemple :



II.2 L'ordre causal – précédence causale

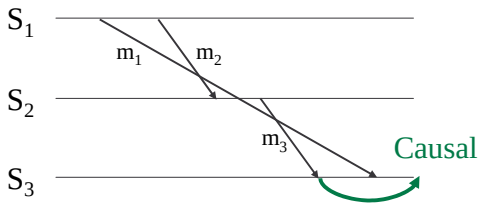
Propriété 2 : Ordre de délivrance **causale**

Extension à plusieurs sites.

Si l'envoi du message $\mathbf{m_1}$ par le site $\mathbf{S_i}$ à destination du site $\mathbf{S_k}$ précède (causalement) l'envoi du message $\mathbf{m_2}$ par le site $\mathbf{S_j}$ à destination du site $\mathbf{S_k}$, le message $\mathbf{m_1}$ sera délivré avant le message $\mathbf{m_2}$ sur le site $\mathbf{S_k}$.

$$\mathrm{snd}_i(m_1,k) \rightarrow \mathrm{snd}_j(m_2,k) \Rightarrow \mathrm{rcv}_k(m_1) \rightarrow \mathrm{rcv}_k(m_2)$$

Contre-exemple :



II.3 Les horloges logiques scalaires

Principe :

HLi : horloge logique du site Si (un nombre scalaire, entier)

ELm : estampille logique attribuée au message m lors de son envoi

Trois types d'événements :

1) Evénement purement local sur Si :
   HLi++
2) Envoi d'un message m par le site Si :
   HLi++
   puis : ELm = HLi
   (le message m est envoyé avec la nouvelle valeur de l'horloge comme estampille)
3) Réception d'un message m d'estampille ELm sur Si :
   HLi = max(HLi, ELm)+1

II.3 Les horloges logiques scalaires

Ordre total strict : « $\Rightarrow$ » ou « << »

Date d'un événement = (horloge logique , numéro du site)

Date d'un événement = (HLi , i)

a $\Rightarrow$ b si et seulement si :

- HLi(a) < HLj(b)
- ou
- HLi(a) = HLj(b) et i<j.

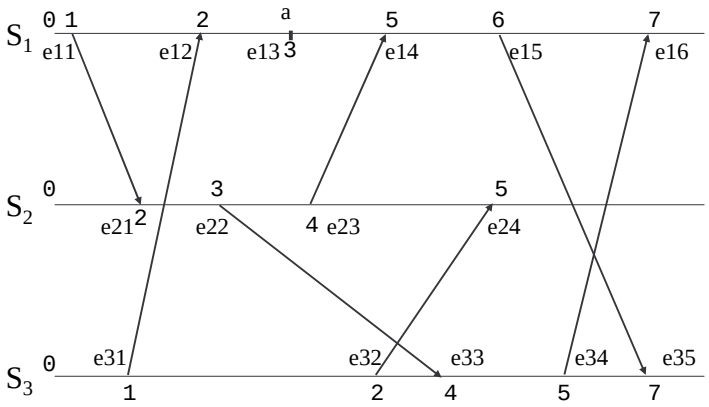
Propriétés :

1) Tous les événements dans **Passé(p)** apparaissent **avant** p dans la chaîne des événements.
2) Tous les événements dans **Futur(p)** apparaissent **après** p dans cette chaîne.
3) Des événements **concurrents** sont artificiellement ordonnés.

II.3 Les horloges logiques scalaires

Chaîne des événements (ordre logique) :

e11 e31 e12 e21 e32 e13 e22 e23 e33 e14 e24 e34 e15 e16 e35



II.4 Les horloges logiques vectorielles

Evitent les inconvénients des horloges scalaires :

1) Plus d'ordre artificiel sur les événements concurrents,
2) Correction des défaillances FIFO des canaux de communication.

Principe :

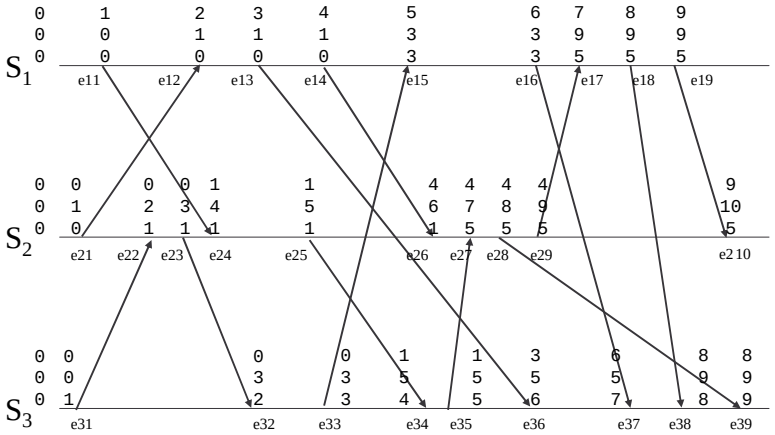
HVi[n] : horloge vectorielle du site Si, constituée de n entiers

EVm : estampille vectorielle attribuée au message m lors de son envoi

1) Evénement purement local sur Si :
   HVi[i]++
2) Envoi d'un message m par le site Si :
   HVi[i]++
   puis : EVm = HVi
   (le message m est envoyé avec la nouvelle valeur de l'horloge comme estampille)
3) Réception d'un message m d'estampille EVm sur Si :
   HVi[i]++
   $\forall j, \text{avec } j = 1,\ldots,n \text{ et } j \neq i, HVi[j] = \max(HVi[j], EVm[j])$

II.4 Les horloges logiques vectorielles

Exemple à 3 sites :



II.4 Les horloges logiques vectorielles

Propriétés des horloges vectorielles :

Propriété 1 : $\forall i, HVe[i] = \text{Card}(\{e' / e' \in Si \text{ et } e' \rightarrow e\})$

Composante i du vecteur est le nombre d'événements qui appartient au passé de l'événement « e » considéré

Propriété 2 : $HVe \subseteq HVe' \Leftrightarrow \forall i,\ HVe[i] \leq HVe'[i]$

Extension de la relation d'ordre

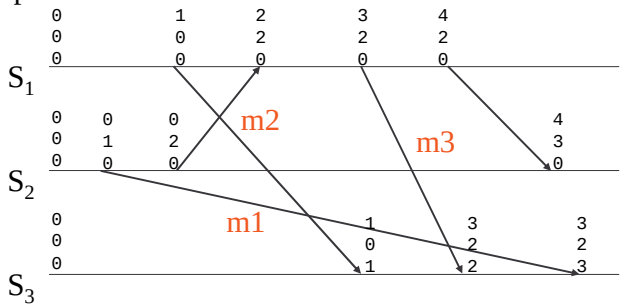
Propriété 3 : $e \rightarrow e'$ si et seulement si $HVe \subseteq HVe'$
$e // e'$ si et seulement si $HVe // HVe'$

HV vérifie la relation de précédence causale entre les événements

II.4 Les horloges logiques vectorielles

Propriété 4 : Détection possible, **mais *a posteriori***, de la violation de l'ordre causal dans la réception des messages.

Exemple :



Problème

Le message m1 a pour estampille (0,**1**,0)
Ce message arrive sur le site S3 avec une horloge vectorielle qui vaut (3,**2**,2)

⇨ Un message postérieur au message m1 est donc déjà arrivé sur S3.

Ici, il s'agit bien sûr du message m3 d'estampille (3,2,0).

II.5 Les horloges logiques matricielles

L'horloge HMi du site Si est matricielle et carré d'ordre n

Objectif des HM :

Obtenir, pour un site, des informations sur les horloges, sur les messages échangés et sur les événements internes des autres sites **entre eux**.

Connaître, pour un site :

Version 1 : **tous les événements** internes et externes (donc, les horloges) que les sites connaissent les uns des autres.

Version 2 : les **messages émis** entre tous les sites, avec vérification de l'ordre causal.

Version 3 : les **messages** effectivement **délivrés**.

II.5 Version 1 : Les HM de W, B, S et L

Wuu, Gene ; Bernstein, Arthur / Sarin, Sunil ; Lynch, Nancy

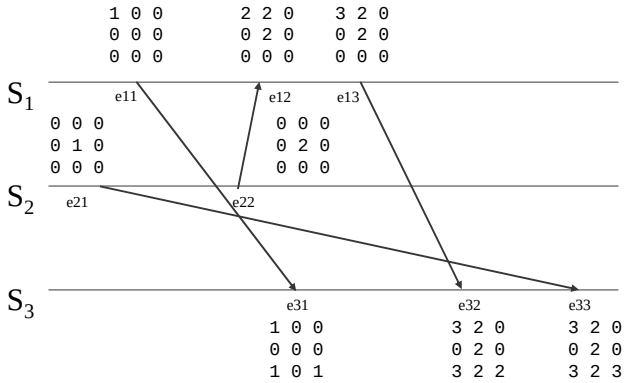
Algorithme (principe) :

1) Evénement purement local sur Si :
   HMi[i,i]++
2) Envoi d'un message m par le site Si vers le site Sj :
   HMi[i,i]++
   puis : EMm = HMi
3) Réception d'un message m (EMm) sur Si depuis Sj :
   HMi[i,i]++

   $\forall k, HMi[i,k] = \max(HMi[i,k], EMm[j,k])$

   $\forall k$ et $\forall l, HMi[k,l] = \max(HMi[k,l], EMm[k,l])$

II.5 Version 1 : Les HM de W, B, S et L



II.5 Version 1 : Les HM de W, B, S et L

Signification de la matrice :

a) HMi[j,j] est le nombre d'événements sur Sj
   (si j=i, c'est le nombre d'événements locaux)

b) La ligne i est le nombre d'événements que Si « pense »
   qu'il y a eu sur les autres sites

c) Les autres lignes j sont les informations que le site Sj
   a sur les autres sites, d'après « ce que croit » Si.

II.5 Version 2 : Les HM de R, S, T

Raynal, Michel ; Schiper, André ; Toueg, Sam

Algorithme (principe) :

1) Evénement purement local sur Si :
   HMi[i,i]++

2) Envoi d'un message m par le site Si vers le site Sj :
   HMi[i,i]++
   HMi[i,j]++
   puis : EMm = HMi

3) Réception d'un message m (EMm) sur Si depuis Sj :

   Après vérification de l'ordre causal dans les arrivées :

   HMi[i,i]++
   $\forall k$ et $\forall l$, HMi[k,l] = max(HMi[k,l], EMm[k,l])

II.5 Version 2 : Les HM de R, S, T

Avec, le principe de vérification de l'ordre causal suivant :

Condition 1 :

Si doit avoir reçu tous les messages précédents de Sj

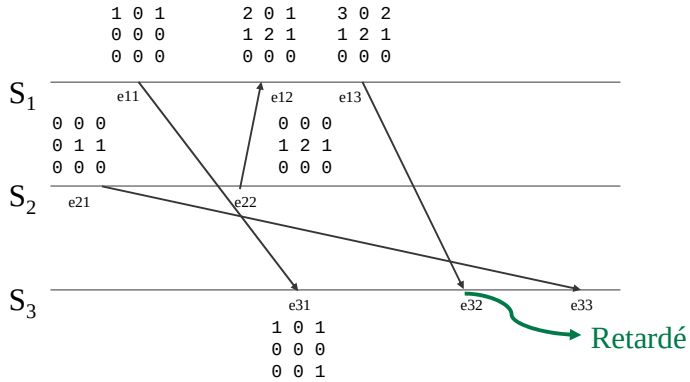
EMm[j,i] = HMi[j,i] + 1 (ordre FIFO sur le canal (j,i))

Condition 2 :

Si doit avoir reçu tous les messages qui lui ont été envoyés plus tôt depuis d'autre sites

pour tout k≠j EMm[k,i] ≤ HMi[k,i]

II.5 Version 2 : Les HM de R, S, T



Pour l'événement e32, on a une réception :

i=3 et j=1

C1:EMm[1,3]=HM3[1,3]+1 OK

C2: k≠1, EMm[k,3]≤HM3[k,3] FAUX car EMm[2,3]=1 alors que HM3[2,3]=0

⇨ Il manque un message de S2 vers S3.
Le message **n'est pas délivrable**.
Le message émis en e13 est retardé.

II.5 Version 2 : Les HM de R, S, T

Signification de la matrice :

a) HMi[j,j] est le nombre d'événements de Sj (pour j=i : événements locaux)

b) La ligne i (HMi[i,k]) est le nombre de messages émis par Si vers les autres sites Sk

c) Les autres lignes j (HMi[j,k]) sont les nombres de messages que le site Si sait que les sites Sj ont émis vers les sites Sk

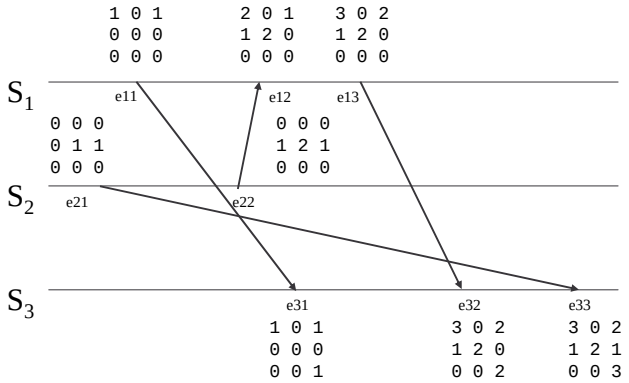
II.5 Version 3 : Les HM de C, D, K

Coulouris, George; Dollimore, Jean ; Kindberg, Tim

Algorithme (principe) :

1) Evénement purement local sur Si :
   HMi[i,i]++
2) Envoi d'un message m par le site Si vers le site Sj :
   HMi[i,i]++
   HMi[i,j]++
   puis : EMm = HMi
3) Réception d'un message m (EMm) sur Si depuis Sj :
   HMi[i,i]++
   HMi[j,i]++
   pour tout **k≠i** (diagonale : horloges locales)
   $HMi[k,k] = \max(HMi[k,k], EMm[k,k])$
   pour tout **k≠j** et $\forall l$:
   $HMi[k,l] = \max(HMi[k,l], EMm[k,l])$

II.5 Version 3 : Les HM de C, D, K



II.5 Version 3 : Les HM de C, D, K

Signification de la matrice :

a) HMi[i,i] est le nombre d'événements locaux

b) La ligne i (HMi[i,k]) est le nombre de messages émis par Si vers Sk

c) Les autres lignes j (HMi[j,k]) indique au site Si le nombre de messages que le site Sj a émis vers Sk, et que Sk a effectivement reçu.

II-6 Etat global et coupure cohérente

Chandy, K. Mani ; Lamport, Leslie

Etat global = vision instantanée d'un S.R.

= état des différents sites et canaux de communication

Difficile sans figer les systèmes

Un site possède *l'image* de l'état des autres sites.

L'image est construite par des *messages* qui lui sont envoyés.

→ Ce ne peut être que l'état du **passé** des sites

Etude de **l'histoire** des sites

II-6 Etat global et coupure cohérente

Définitions :

Histoire **locale** (hi) :

Pour Si : hi = <ei1, ei2, ..., eik, ...>
suite ordonnée des événements sur le site

Histoire **globale** (H) :

Vision complète, des N sites
Pour N sites : H = <h1, ...,hN>

Coupure de l'histoire globale (C) :

C = <c1, ...,ci, ...,cN>
où ci = hi = <ei1, ei2, ..., eik, ...>

« photographie » instantanée des histoires locales

II-6 Etat global et coupure cohérente

Définitions :

ELi : état local du site Si

ECij : état local du canal Cij

Etat Global EG : EG = { pour tout i, j ; ∪ELi, ∪ECij }

« Un **état global** d'un système réparti est constitué
d'un état local de chacun des sites
et d'un état de chacun des canaux de communication »

Coupure C associée à un EG :

C = { pour tout i ; ∪ELi }

Recherche d'un état global **cohérent**

➡ Recherche d'une **coupure cohérente**

II-6 Etat global et coupure cohérente

Conditions de cohérence (basées sur le respect de la causalité) :

m : message concerné (à la frontière de coupure)

Condition 1 :

Si EMISSIONi(m) ∈ ELi, alors :
Soit RECEPTIONj(m) ∈ ELj
Soit m ∈ ECij

« Tout message émis dans le passé est :
soit déjà reçu,
soit encore en transfert »

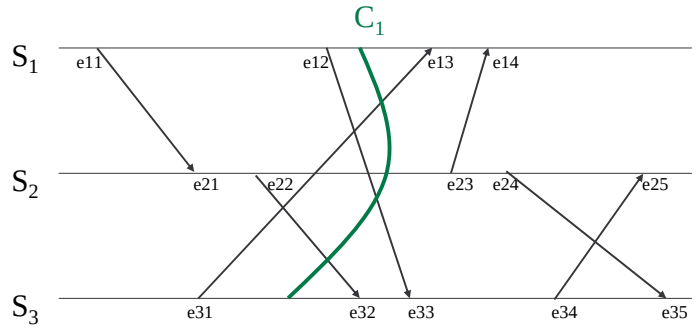
Condition 2 :

Si EMISSIONi(m) ∉ ELi,
alors RECEPTIONj(m) ∉ ELj

« Tout message émis dans le futur reste dans le futur »
« Un message ne peut pas venir du futur »

II-6 Etat global et coupure cohérente

Exemple à 3 sites :



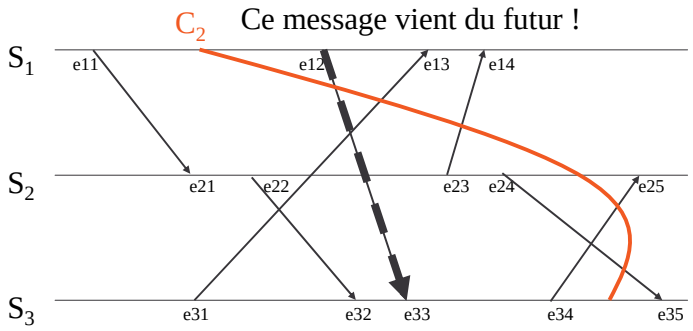
Coupure C_1 : $C_1 = < C_1 1, C_1 2, C_1 3 >$

$C_1 1$ = <e11,e12> Passé(e12) = {e11,e12}
$C_1 2$ = <e21,e22> Passé(e22) = {e11,e21,e22}
$C_1 3$ = <e31> Passé(e31) = {e31}

Tous présents ➡ Coupure **cohérente**

II-6 Etat global et coupure *in*cohérente

Exemple à 3 sites :



Coupure C_2 : $C_2 = < C_21, C_22, C_23 >$

C_21 = <e11> Passé(e11) = {e11}
C_22 = <e21,e22,e23,e24> Passé(e24) = {e11,e21,e22,e23,e24}
C_23 = <e31,e32,e33,e34> Passé(e34) = {e11,e12,e21,e22,e31,e32,e33,e34}

Problème : il manque e12 ⇒ Coupure **incohérente**

II-6 Etat global et coupure cohérente

Cohérence d'une coupure par les Horloges Vectorielles :

Estampille vectorielle d'une coupure (date d'une coupure) :

EV(C) = sup(EV(e1), ...,EV(ej), ...,EV(eN))

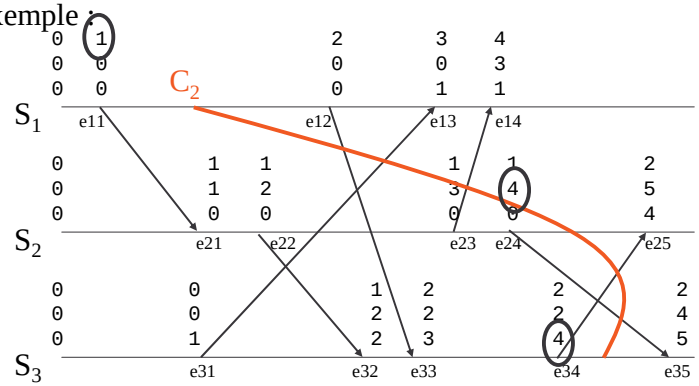
Soit, pour tout k :
EV(C)[k] = sup(EV(e1)[k], ...,EV(ej)[k], ...,EV(eN)[k])

La coupure C est **cohérente** si et seulement si :

<EV(e1)[1], ...,EV(ej)[j], ...,EV(eN)[N]> = EV(C)

II-6 Etat global et coupure *in*cohérente

Exemple :



Coupure C_2 : $C_2 = < C_21, C_22, C_23 >$

C_21 = <e11>
C_22 = <…,e24>
C_23 = <…,e34>

$EV(C_2) = SUP\left\{\begin{matrix}1&1&2\\0&4&2\\0&0&4\end{matrix}\right\} = \begin{matrix}2\\4\\4\end{matrix}$ ⇒ Coupure **incohérente**

<EV(e11)[1], EV(e24)[2],EV(e34)[3]> = (1,4,4)

II-6 Etat global et coupure cohérente

Algorithme de Chandy et Lamport

Objectif :

Définir un état global cohérent

Rappel: « Un **état global** d'un système réparti est constitué des états de tous les sites (en coupure cohérente) et de l'état des canaux de communication »

Principe :

Un « marqueur » demande aux sites de mémoriser leur état.
Le marqueur sépare « avant » et « après » enregistrement.

Hypothèses :

1) Canal de communication **unidirectionnel et FIFO** entre chaque paire de sites,
2) Réseau connexe avec communication fiable,
3) Tout site peut initier l'algorithme : c'est l'élu,
4) Le système continue à fonctionner.

II-6 Etat global et coupure cohérente

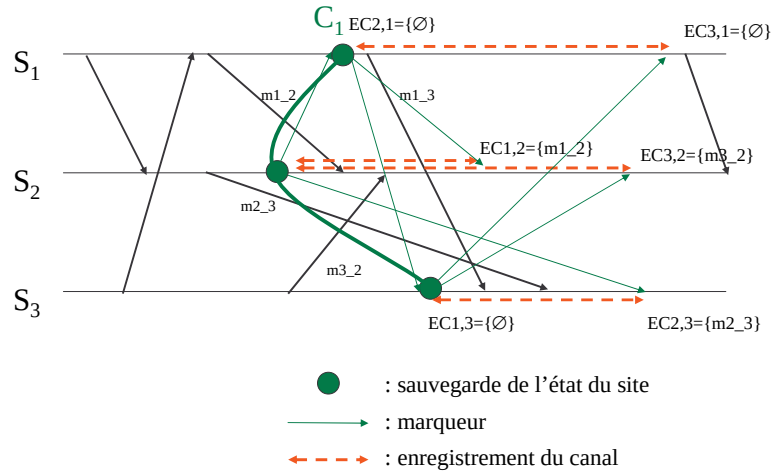
Algorithme :

- Un site Si diffuse le marqueur initialisant ainsi l'algorithme.
- Sur Sj, première réception du marqueur depuis Si :
  - Sauvegarder l'état local du site : Cj = ELj
  - Marquer le canal i vers j à vide : ECij={∅}
  - Diffuser le marqueur vers tous les sites
  - Enregistrer les messages provenant des autres canaux
- Sur Sk, réceptions suivantes du marqueur depuis Sj :
  - Arrêter l'enregistrement, en figeant ainsi le canal j vers k :
    ECjk={messages_enregistrés}

II-6 Etat global et coupure cohérente

Exemple d'application de l'algorithme de Chandy et Lamport :

S2 initialise l'algorithme



III-1 L'Exclusion Mutuelle

Monoprocesseur : sémaphores par Edsger W. Dijkstra, en 1965

Multiprocesseur : absence de mémoire partagée

➡ Solution centralisée…

Un processus contrôleur gère la file (FIFO) des demandes.

Un processus qui se voit attribuer l'autorisation du contrôleur accède à la section critique.

Dès la fin du traitement, le processus informe le contrôleur.

Ce dernier donne alors la permission à un autre processus.

… puis, répartir cette solution !

III-2 L'E.M. de Lamport

Hypothèses :

1) Le nombre de sites **N est connu** ;
2) Les canaux de communication sont **FIFO** et **fiables** ;
3) **Horloge Logique** scalaire de Lamport sur chaque site, fournissant alors un **ordre total strict**.

Messages échangés pour réaliser l'E.M :

Format des messages : (MsgType, Estampille, Emetteur)
3 types « MsgType » :

- REQUETE (**REQ**) :
  un site veut entrer en section critique ;
- REPONSE (**ACQ**, acquittement) :
  Un site qui reçoit une requête renvoie au site désireux un accusé de réception (quittance ou acquittement) sous la forme d'un message de type ACQ ;
- LIBERATION (**LIB** ou REL, relâchement) :
  Un tel message est envoyé par un site lorsqu'il quitte la section critique.

III-2 L'E.M. de Lamport

Principe : Le site Si veut entrer en section critique :

➡ Diffusion de (REQ,HL(REQ),Si)

Le site Si **entre effectivement** en section critique quand :

1) Tous les sites ont reçu sa demande,
   ou qu'ils en ont également émis une,
2) **ET** que sa demande est la plus ancienne de toutes.

Algorithme :

Les sites gèrent des files d'attente Mi.

A) Réception par Si d'un message m = (REQ ou LIB, HLj, Sj) :*

1) Il met à jour son horloge logique HLi : HLi = max(HLi, ELm)+1
2) Il place le message dans sa file d'attente : Mij := m
3)* Si répond à Sj, par l'émission du message (ACQ, HLi(ACQ), Si).

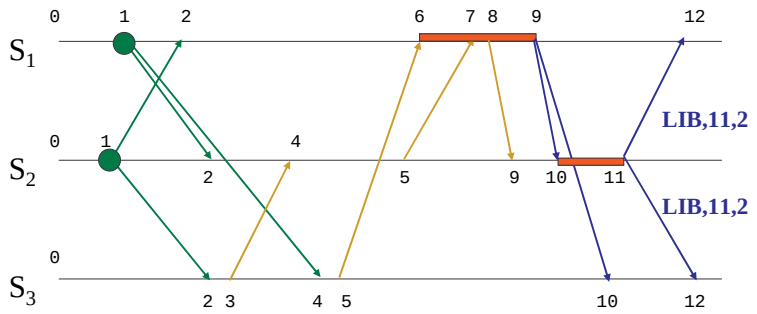
B) Réception par Si d'un message m = (ACQ, HLj(ACQ), Sj) :

Le message m est placé en tête (selon les horloges) de la file (si cette dernière ne contient pas un message de type REQ) ou est ignoré (si présence d'un REQ).

1) Il met à jour son horloge logique HLi : HLi = max(HLi, ELm)+1
2) Si Mij ≠ (REQ, HLj(REQ), Sj) alors Mij := m

III-2 L'E.M. de Lamport

Exemple :



	HL1
M11 LIB,9,1	9
M12 LIB,11,2	12
M13 ACQ,5,3	6

	HL2
M21 LIB,9,1	10
M22 LIB,11,2	11
M23 ACQ,3,3	4

	HL3
M31 LIB,9,1	10
M32 LIB,11,2	12
M33 LIB,0,3	0

III-2 L'E.M. de Lamport

Propriétés :

1) Un seul site ne peut entrer en section critique à la fois…

2) Canaux FIFO :
   si un site a reçu un message d'accord du site j,
   toute requête antérieure de ce même site lui est
   déjà nécessairement arrivé.

3) Pas d'interblocage possible.

4) **3*(N-1) messages** pour traiter une entrée en section critique (N-1 messages de chacun des types).

III-3 L'E.M. de Ricart/Agrawala

But : minimiser le nombre de messages nécessaires

➡ éliminer les messages de type LIB.

2*(N-1) messages au lieu de 3*(N-1)

Hypothèses : même que Lamport

2 types de messages :

- REQUETE (**REQ**) : envoyé quand le site veut entrer en section critique ;
- ACCORD (**ACQ**, acquittement) : envoyé soit immédiatement à la réception d'un message REQ, soit ultérieurement, à la sortie de section critique du site récepteur.

3 états possibles :

- DEHORS : il n'est ni en section critique, ni demandeur pour y entrer ;
- DEDANS : il est en section critique ;
- ENATTENTE : il a demandé à entrer en section critique et attend d'obtenir l'accord de tous les autres sites.

III-3 L'E.M. de Ricart/Agrawala

Principe :

Chaque site Si gère :

- un indicateur **Etat_Site_i**, initialisé à DEHORS.
- une variable **Date_Demande_i**, initialisée à 0, qui sert à mémoriser l'instant de la demande.
- une file d'attente **FILEi**, de capacité N messages, dans laquelle le site place toutes les requêtes des sites demandeurs qu'il ne peut satisfaire immédiatement. Cette file est initialisée à {Φ}.
- un vecteur **Accord_Attendu_i**, de taille N, servant à mémoriser les accords reçus. Ce vecteur est initialisé avec {Sk}, pour k variant de 1 à N, avec k≠i.

Le site Si veut entrer en section critique :

➡ Diffusion de (REQ,HL(REQ),Si)

Un site donne son accord que s'il n'est pas lui-même en attente.

Un site entre en section critique quand Accord_Attendu est vide.

III-3 L'E.M. de Ricart/Agrawala

Algorithme :

A) Réception par Si d'un message m = (REQ, HLj, Sj) :

Réponse éventuelle par un ACQ :

1) Il met à jour son horloge logique HLi : HLi = max(HLi, ELm)+1
2) Il traite le message, suivant son état :

Cas 1 : DEHORS

Emission à Sj de (ACQ, HLi(ACQ), Si)

Cas 2 : ENATTENTE

Si (HLj(REQ),j) << (Date_Demande_i,i) Alors

Emission à Sj de (ACQ, HLi(ACQ), Si)

Sinon

FILEi = FILEi + {Sj}

FinSi

Cas 3 : DEDANS

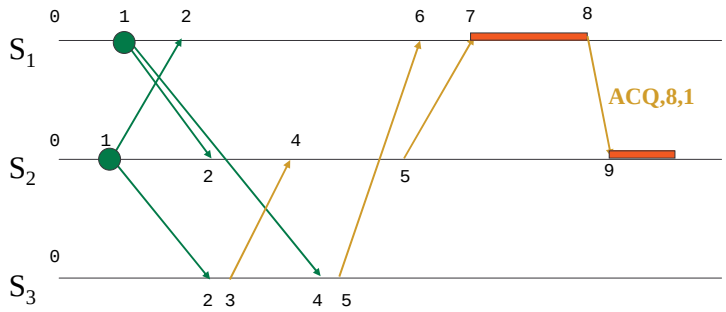
FILEi = FILEi + {Sj}

B) Réception par Si d'un message m = (ACQ, HLj(ACQ), Sj) :

1) Il met à jour son horloge logique HLi : HLi = max(HLi, ELm)+1
2) Accord_Attendu_i = Accord_Attendu_i – {Sj}
3) Si Accord_Attendu_i = {Φ}, alors le site Si entre en section critique

III-3 L'E.M. de Ricart/Agrawala

Exemple :



HL2 = 9 S2 reçoit le dernier accord attendu

Accord_Attendu_2 = {S1}-{S1}={∅}

Etat_Site_2 = DEDANS

III-4 L'E.M. de Suzuki/Kasami

But : minimiser encore le nombre de messages nécessaires

➡ Circulation d'un **jeton** : le détenteur peut entrer en S.C.

N messages au lieu de 2*(N-1) et 3*(N-1)

Hypothèses : même que Lamport et Ricart/Agrawala

3 états possibles, comme pour Ricart/Agrawala :

- DEHORS : il n'est ni en section critique, ni demandeur pour y entrer ;
- DEDANS : il est en section critique ;
- ENATTENTE : il a demandé à entrer en section critique et **attend d'obtenir le jeton.**

L'algorithme n'est plus basé sur l'Horloge Logique.

Utilisation d'un vecteur d'horloges « RN », proche des horloges vectorielles.

Seules les émissions sont comptabilisées.

III-4 L'E.M. de Suzuki/Kasami

Principe :

Chaque site Si gère :

- un indicateur **Etat_Site_i**, initialisé à DEHORS.
- un indicateur **Jeton_Présent_i**. Le site possédant le jeton le conserve jusqu'à ce qu'il lui soit demandé par un autre.
- un vecteur **RNi** d'horloges de N entiers, tel que :
  - RNi[j] est le numéro de la plus grande demande reçue de la part du site Sj.
  - RNi[i] est le nombre de demandes formulées par le site Si.

  RN comptabilise le Nombre d'émissions de Requêtes.

Le jeton contient les structures de données Q et LN, telles que :

- Q est une file d'attente, de capacité N noms de sites :
  File contenant les demandes des sites pour entrer en section critique, et ordonnée selon les instants d'arrivée de ces demandes.
- LN est un vecteur de taille N :
  Historique des entrées en section critique des différents processus.
  LN[i] : numéro de la dernière demande d'entrée en section critique du site Si qui a été satisfaite.

III-4 L'E.M. de Suzuki/Kasami

Algorithme :

A) Demande d'entrée de Si en section critique :

```
// Diffusion de la demande estampillée et attente de l'arrivée du jeton
Etat_Site_i = ENATTENTE
RNi[i] = RNi[i] + 1                          // Incrémentation du n° demande
Si (Jeton_Présent_i == FAUX) Alors
                Diffusion de (REQ,RNi,Si) // Diffusion si pas de jeton
                Attente Jeton_Présent_i     // Blocage…
Finsi
Etat_Site_i = DEDANS                         // … jusqu'à ce que le jeton arrive
```

B) Réception par Si d'un message m = (REQ, RNj, Sj) :

```
RNi[j] = Max(RNi[j],RNj[j])          // Mise à jour « horloge » n° demande
Si (Jeton_Présent_i == VRAI)         // envoi éventuel du jeton…
      et (Etat_Site_i == DEHORS)     // … s'il n'est pas en section critique, …
      et (RNi[j] == LN[j]+1)         // et que la demande de Sj est plus récente
Alors
      Jeton_Présent_i = FAUX
      Envoyer (Jeton, Sj)
Finsi
```


III-4 L'E.M. de Suzuki/Kasami

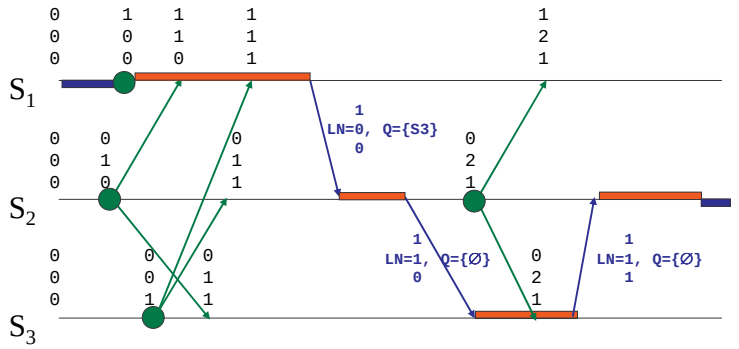
Algorithme :

C) Libération de la section critique par Si :

```
Etat_Site_i = DEHORS                 // Liberation de la section critique
LN[i] = RNi[i]                       // Mise à jour de l'historique
// Construction de la file Q (incluse dans le jeton) pour prendre en compte
// les demandes qu'il a reçues depuis qu'il détient le jeton
Pour S variant de 1 à N Faire
      Si (S != i) et (S ∉ Jeton.Q) et (RNi[S]= Jeton.LN[S] +1) Alors
            Ajouter_en_Fin(Jeton.Q,S)
      Finsi
FinPour
// Comparaison de son vecteur RNi avec le vecteur LN du jeton :
// Si différence et s'il y a des sites demandeurs (file Q non vide), il envoie
// le jeton au premier site dans la file Q (le plus petit).
Si (Jeton.Q != {∅}) Alors
      Jeton_Présent_i = FAUX
      Site = Extraire_la_Tête(Jeton.Q)
      Envoyer (Jeton, Site)
Finsi
```


III-4 L'E.M. de Suzuki/Kasami

Exemple :



IV-1 La problématique de l'Election

« Leader Election »

Trouver le site qui a le plus petit (ou grand) identificateur (ID)

G. Le Lann
Distributed Systems - Towards a Formal Approach
Congrès IFIP, Information Processing, 1977, pp 155-160.

Utilisation :

1) Réaction à une défaillance : un (et un seul) doit réagir

   Organisation Maître/Esclave

   Un seul Esclave doit prendre la place du Maître s'il défaille

2) Génération d'un jeton dans un « Token Ring » :

   Tous les sites détectent la perte (à l'aide d'une horloge de garde)

   Un seul régénère le jeton : Le « Seigneur de l'Anneau » !

3) Beaucoup d'algorithmes s'appuient sur un « First Step »

IV-2 L'Election de Le Lann

Hypothèses :

1) Le nombre de sites **N est connu** ;
2) Les sites ont des **ID uniques**;
3) Les canaux de communication sont **FIFO** et **fiables** ;
4) Les sites sont organisés en anneau **unidirectionnel**.

Idée :

Les sites doivent s'échanger (se transmettre) leurs ID.

Par comparaison (plus petit ou plus grand), ils savent qui est l'élu.

Principe :

Chaque site envoie un message contenant son ID à son voisin ;

Ce message est renvoyé par le voisin au sien, et ainsi de suite ;

Fin des transmissions lorsque le site reçoit son propre message.

IV-2 L'Election de Le Lann

Algorithme, à exécuter sur chaque site Si :

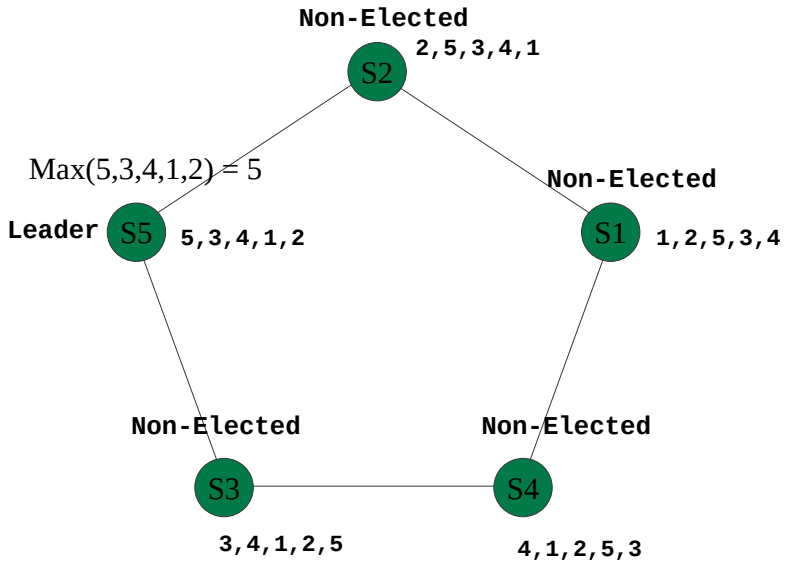
```
Liste_Si = {ID_Si};
SEND(ID_Si,Next(Si));
RECEIVE(ID);

TantQue (ID ≠ ID_Si) Faire
        Liste_Si = Liste_Si ∪ {ID};
        SEND(ID,Next(Si));
        RECEIVE(ID);
FinTantQue

Si (ID_Si = Max (Liste_Si) Alors
        Etat_Si = Leader
Sinon
        Etat_Si = Non-Elected
FinSi
```


IV-2 L'Election de Le Lann

Exemple :



IV-2 L'Election de Le Lann

Propriétés :

Complexité en **messages** échangés :

Pour N sites, chaque message est émis N fois

Algorithme en $O(N^2)$ messages

Complexité en **temps** :

2N-1 unités de temps après l'émission du premier ID

Algorithme en O(N) unités de temps

Etat final :

Garantie d'obtention d'un unique « Leader »

Les autres sites sont des « Non-Elected »

IV-3 L'Election de Chang/Roberts

E.J.H. Chang, R. Roberts
An improved algorithm for decentralized extrema-finding in circular configurations of processes
Communication of the ACM, Vol. 22, n°5, 1979, pp 281-283.

Variante de Le Lann :

Hypothèses (mêmes que Le Lann)

Principe (très proche de Le Lann) :

Les sites se transmettent leurs ID,
Mais, ils ne transmettent pas les ID inférieurs.
Chaque site Si ne transmet donc les messages que si :
ID > ID_Si
Le site qui reçoit son ID se proclame leader.

IV-3 L'Election de Chang/Roberts

Algorithme, à exécuter sur chaque site Si :

```
Etat_Si = Inconnu;
SEND(ID_Si,Next(Si));

Répéter
        RECEIVE(ID);
        Si (ID_Si == ID) Alors
                Etat_Si = Leader
        Sinon Si (ID > ID_Si) Alors
                Etat_Si = Non_Elected
                SEND(ID,Next(Si));
        FinSi
Jusqu'à (Etat_Si = Leader)
```

Il n'y a qu'un site qui va terminer « proprement » ce processus : le leader

IV-3 L'Election de Chang/Roberts

Propriétés :

Complexité en **messages** échangés :

Complexité ***moyenne*** en O(N LogN) messages

Mais, au pire (S5 émet sur S4, et S4 émet sur S3, etc.) :
Algorithme en $O(N^2)$ messages, car N*(N+1)/2 messages

Complexité en **temps** :

2N-1 unités de temps après l'émission du premier ID

Algorithme en O(N) unités de temps

Etat final :

Garantie d'obtention d'un unique « Leader »

Les autres sites savent « rapidement » qu'ils sont Non-Elected

Gestion de la fin des processus « infinis » sur les Non-Elected