Systèmes et Applications Répartis

(cours B4 code 19302)

année 2003-2004

Bases pour l'algorithmique répartie

C Kaiser

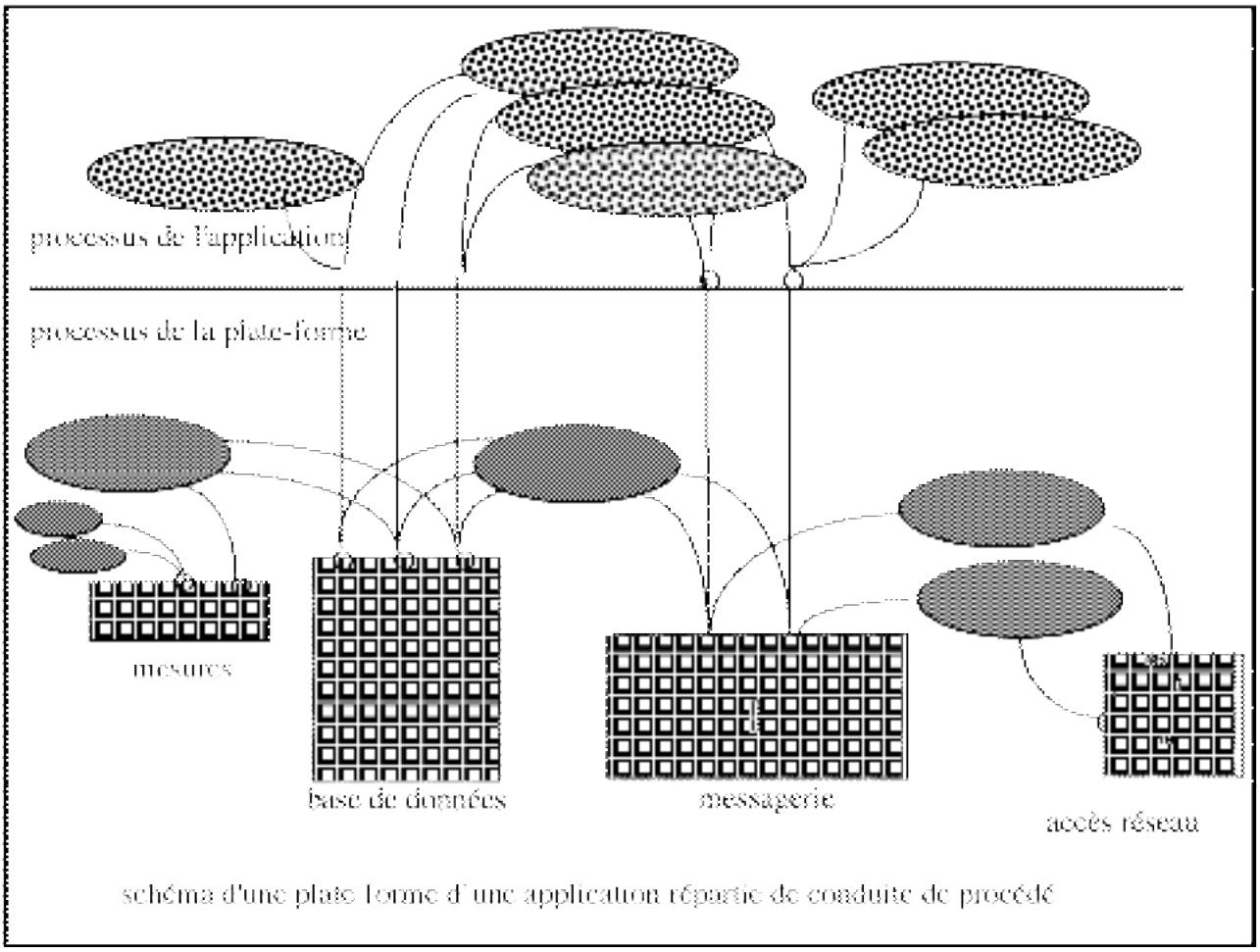
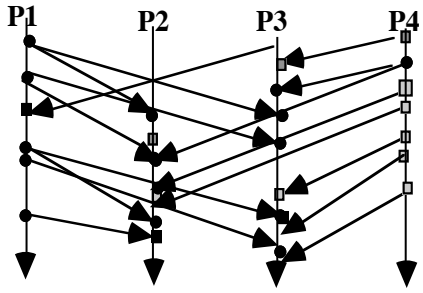


schéma d'une plate forme d'une application répartie de conduite de procédé



trace de messages entre 4 acteurs

BASES POUR L'ALGORITHMIQUE RÉPARTIE
ORDRES, ÉTAT GLOBAL, HORLOGES, SYNCHRONISATION, ALGORITHMIQUE, CONTRÔLE ET REPRISE DANS LES SYSTÈMES RÉPARTIS

1. ASPECTS DES APPLICATIONS RÉPARTIES : TYPES DE COOPÉRATION, MODÈLES DE COMMUNICATION ÉLÉMENTAIRE, DÈPENDANCE CAUSALE, MODÈLES DE DIFFUSION FIABLE ET COMMUNICATION DE GROUPE, PROPRIÉTÉS D'ORDRE DANS LES GROUPES. COMPLÉMENTS ET PROPRIÉTÉS LIÉES AUX DÉFAILLANCES.

2. ÉTAT GLOBAL D'UN SYSTÈME RÉPARTI : PASSÉ ET COUPURES COHÉRENTES, DÉTERMINATION D'UN ÉTAT GLOBAL COHÉRENT (CHANDY - LAMPORT).

3. DATATION CAUSALE ET HORLOGES VECTORIELLES : HORLOGES VECTORIELLES ET COUPURES COHÉRENTES, DIFFUSION FIABLE AVEC ORDRE CAUSAL.

4. ORDRE TOTAL PAR HORLOGES LOGIQUES : EXCLUSION MUTUELLE RÉPARTIE, AVEC HORLOGE LOGIQUE.

5. POSE DE POINTS DE REPRISE REPARTIS□ CHEMINS EN ZIGZAG, SAUVEGARDE ADAPTATIVE

6. ETUDE DE CAS : CALCUL COOPERATIF ET OBJET REPLIQUE SUR 4 SITE

7. EXERCICES AVEC SOLUTIONS

8. SOLUTIONS DES EXERCICES

9. ANNEXE. POINTS DE CONTRÔLE COHÉRENTS DANS LES S.R. ASYNCHRONES.

10. ANNEXE. DIFFUSION ORDONNÉE RESPECTANT UN ORDRE TOTAL. DIFFUSION PAR ÉCHANGE DISTRIBUÉ

11. ANNEXE. ÉTUDE DE CAS DE COHÉRENCE CAUSALE

Bibliographie :

R.Balter, J.P.Banâtre, S.Krakowiak, éditeurs. ***Construction des systèmes d'exploitation répartis.*** **Collection didactique INRIA 1991 (350 pages)**

G.Coulouris, J.Dollimore, T.Kindberg. ***Distributed Sytems (2nd edition).*** **Addison Wesley 1995 (601□pages)**

S.Mullender. ***Distributed Sytems (2nd ed.).*** **Addison Wesley 1994 (644□.)**

A.Tanenbaum. ***Distributed Operating Sytems.*** **Prentice Hall 1995 (614 p.)**

J.Besancenot et al. ***Les systèmes transactionnels.*** **Hermès 1997 (415 p.)**

G.Blair, J.B.Stéfani. ***Open Distributed Processing and Multimedia.*** **Addison Wesley 1998 (452 p.)**

1. ASPECTS DES APPLICATIONS REPARTIES
TYPES DE COOPÉRATION

RÉPARTITION SIMPLE (CLIENT SERVEUR)

Accès local ou distant aux bases de données, chaque BD cohérente individuellement

Abonnement à BD primaire : copies secondaires pour lecture, sur autres sites

Transactions avec accès à une seule BD primaire à la fois

Transactions avec accès emboîtés à plusieurs BD ou serveurs d'objets :

problèmes de cohérence globale et d'interblocage

Messagerie entre les processus de divers sites

RÉPARTITION SIMPLE (COOPÉRATION DE PAIR À PAIR)

Avec serveur de rendez-vous puis coopération à deux (à plusieurs, devient complexe)

RÉPARTITION PLUS COMPLEXE (APPLICATION COOPÉRATIVE)

Ensemble de transactions coopérantes. Problèmes de synchronisation :

- **démarrage dans un état cohérent**
- **coordination par un site fixe, mais si absent (panne, maintenance) alors élection d'un nouveau coordinateur**
- **terminaison de la coopération**
- **mise au point répartie**
- **points de reprise cohérents**
- **diffusion d'information dans un groupe**

Copies multiples d'une même BD avec écritures sur chaque copie : cohérence faible ou forte (problème des caches multiples)

Mobilité des sites

Caches Web

LE REEL DES COMMUNICATIONS ELEMENTAIRES

DÉSORDRES

POINT A POINT MODE MESSAGE

message d'un émetteur vers un récepteur sur un canal

***perte* de message, absence de récepteur (panne, maintenance,...)**

contrôle d'erreur par acquit et délais de garde, mais duplication possible

numérotation des messages successifs

réception *désordonnée* d'une suite de messages

contrôle de flux pour asservir les vitesses de l'émetteur et du récepteur (producteur-consommateur)

***incertitude* sur l'état du canal, de l'émetteur et du récepteur**

durées de transfert *variable*, asynchrone (mais il existe des bus synchrones)

POINT A POINT MODE CLIENT SERVEUR

l'émetteur est le client et le récepteur est le serveur

message requête d'un client vers un serveur sur un canal

message réponse du serveur au client

le client n'envoie pas d'autre requête avant d'avoir reçu la réponse

mêmes problèmes *d'erreurs*, *d'incertitudes* et de *variabilité* des délais

DIFFUSION A UN GROUPE

Un émetteur et N récepteurs sur le canal

exemples : Ethernet, Token ring

***perte* de message pour R récepteurs avec 0 <= R <= N**

d'une émission à l'autre perte sur des récepteurs différents

pas le même ordre sur tous les sites pour une suite de réceptions

pas de perception unique de l'ordre d'émission si émetteurs différents

durées de transfert *variable*

***incertitudes* sur l'état du canal, de l'émetteur et des récepteurs**

***incertitude* sur la composition du groupe**

LES SYSTÈMES RÉPARTIS

Ensemble fini de sites interconnectés par un réseau de communication
Pas de mémoire commune
Pas d'horloge physique partagée par 2 processus ou plus

CHAQUE SITE

• processeur, mémoire locale, mémoire stable (permanence des données en dépit de défaillances du processeur)

RÉSEAU DE COMMUNICATION

• connexe : tout processus peut communiquer avec tous les autres
• communication et synchronisation entre processus par messages via le réseau de communication

DIFFICULTÉS CARACTÉRISTIQUES

OBSERVATIONS :
deux observations faites sur deux sites distincts peuvent différer
- **par l'ordre de perception des événements**
- **par leur date**

DÉCISIONS COHÉRENTES ENTRE PLUSIEURS SITES
• visions différentes de l'état des ressources du système
• pas d'état global unique servant de référence et utilisable en temps réel au moment où il faut prendre une décision concernant l'ensemble des sites
• risque accru de défaillance (plus de composants)
=> la défaillance d'un élément n'est pas un événement rare
=> importance des hypothèses sur les défaillances possibles

attention, à ne pas confondre avec les
SYSTÈMES PARALLÈLES CENTRALISÉS
encore appelés des multiprocesseurs à mémoire commune
• existence d'une mémoire commune (physique, réflexive, virtuelle)
• existence d'une horloge ou d'un rythme commun

LES MODELES DE LA COMMUNICATION ELEMENTAIRE

MODÈLE DE COMMUNICATION SYNCHRONE FIABLE

hypothèses les plus fortes

Tout message arrive avant le délai d_{max}, qu'il soit point à point ou diffusé.

Communication fiable : pas de perte de message, pas de panne de site

Réseau connexe : tout site peut communiquer avec tous les autres
parfois réseau isotrope : même délai d_{max} pour tous les canaux

Les horloges des sites sont aussi synchronisées
(leur écart est borné par dh_{max})

Les vitesses des processeurs sont supérieures à un minorant connu

Exemple : réseaux locaux avec heure reçue par radio sur chaque site, réseaux locaux industriels avec synchronisation d'horloge véhiculée par le réseau, réseaux locaux avec redondance des bus et des processeurs

- **Mais cette hypothèse, valable avec une certaine probabilité, n'est pas toujours réaliste :**

probabilité trop faible pour l'application,
probabilité variable dans le temps (cas des réseaux dont la charge instantanée est totalement imprévisible)

ELECTION EN COMMUNICATION SYNCHRONE FIABLE

Les sites S1, S2,..., Si, ..., SN doivent élire un site coordonnateur

Chaque site Si a un identificateur unique uid(i)

et peut diffuser un message <élection, i, uid(i))>

Tous les sites démarrent l'élection en même temps à P

(à dates fixes, par exemple, élection périodique)

Soit $T = d_{max} + dh_{max}$,

Chaque site Si

(i) s'attend à recevoir un message d'élection à partir de P

(ii) s'il n'a rien reçu au bout de T * uid(i) secondes, mesurées sur son horloge, il diffuse son message d'élection.

Résultat : le premier site qui diffuse son message est l'élu

Cet algorithme synchrone élit le site de plus petit uid

commentaire : simple n'est-il pas? mais l'hypothèse synchrone est restrictive (pas de pannes, horloges communes, faibles écarts dh) la "nature" est asynchrone.

ELECTION EN COMMUNICATION SYNCHRONE NON FIABLE

hypothèses : processeurs et canaux à silence sur défaillance

(pas de panne transitoire ou byzantine, mais silence après i, ii ou iii)

Chaque site Si

(i) à P, diffuse son uid(i) aux autres sites, lui compris

(ii) à P + T, il diffuse le min(i) des uid(j) qu'il reçoit avant P + T,

(iii) à P + 2*T, s'il note un consensus des min(j) reçus (selon les cas, la majorité, ou l'unanimité des valeurs reçues), il élit la valeur de consensus (si l'élu est tombé en panne après sa phase i, il faut recommencer)

EXEMPLE DE RAISONNEMENT SYNCHRONE

L'EMPEREUR ET LES TROIS BRIGANDS

L'EMPEREUR

"Voici cinq disques. Trois sont blancs, deux sont noirs.

Je vais accrocher un disque dans le dos de chacun de vous.

Chacun peut voir la couleur du disque que portent les deux autres
Aucun ne peut voir la couleur du disque qu'il porte

Le premier qui par un raisonnement
saura trouver la couleur de son disque et me l'expliquer
sera gracié"

L'EMPEREUR
ACCROCHE UN DISQUE BLANC DANS LE DOS DE CHAQUE BRIGAND

UN BRIGAND, LE PLUS INTELLIGENT, TROUVE LA SOLUTION

Expliquez en détail le raisonnement qu'il a fait

L'EMPEREUR ET LES TROIS BRIGANDS

CONNAISSANCE PARTIELLE COMMUNE À CHACUN

• Le groupe comprend trois membres A, B, C
• Il y a cinq disques seulement : trois blancs, deux noirs

CONNAISSANCE PARTIELLE PROPRE À CHACUN (ici A)

• connaissance de la couleur de B et C (resp. C et A, A et B)
• ignorance de la couleur de son disque A (resp. B, C)

RAISONNEMENT DE A

H1 "si mon disque était NOIR,
alors B (respectivement C) raisonnerait ainsi :
"comme le disque de A est NOIR,
H2 si le mien était NOIR lui aussi,
alors C (resp. B) saurait de façon certaine
que son disque est BLANC
et il le dirait à l'empereur,
C2 comme C (resp. B) ne dit rien, j'en *conclus*
que mon disque est BLANC"
C1 comme ni B ni C ne donnent la solution, j'en conclus que
mon disque n'est pas NOIR
donc mon disque est BLANC"

RAISONNEMENT SYNCHRONE
il faut attendre le temps que tous puissent atteindre la connaissance C2 pour en déduire la connaissance C1

ALGORITHME PAR VAGUE

• le nombre des membres est connu
• même algorithme partout
• pas de panne (ni de processeur, ni de transmission)
• un certain fonctionnement synchrone
vitesse minimum de traitement
(ici, le raisonnement)
délai maximum de transmission
(ici, observation du comportement des autres)

MODÈLE DE COMMUNICATION ASYNCHRONE

CHAQUE SITE

• processeur, mémoire locale, mémoire stable

(permanence des données en dépit de défaillances du processeur)

RÉSEAU DE COMMUNICATION

• connexe : tout processus peut communiquer avec tous les autres

• communication et synchronisation entre processus par messages

via le réseau de communication

PROPRIÉTÉS CARACTÉRISTIQUES

• Pas de mémoire commune

• Pas d'horloge physique partagée par 2 processus ou plus

• Absence de majorant connu sur le temps de transfert des messages

• Absence de minorant connu sur les vitesses des processeurs

Nota : appelé aussi modèle à délais non bornés

Exemple : cas de Internet et des réseaux longue distance (WAN)

PROBLÈME. Un processus Pi ne peut pas savoir si un autre processus Pj est défaillant ou si la réponse qu'il attend de Pj est en préparation par Pj (processus très lent) ou encore en route (message très lent). En pratique, il est essentiel d'introduire une notion de temps (temps réel et non temps du processus Pi) : combien de temps Pi doit-il attendre avant de suspecter Pj de défaillance

suspecter une défaillance ≠ détecter une défaillance

MODÈLE DE COMMUNICATION ASYNCHRONE FIABLE

réseau connexe : tout site peut communiquer avec tous les autres
communication fiable : pas de perte de message, pas de panne de site
réaliste sur une courte durée et avec un faible taux de pannes

PROPRIÉTÉ DE CAUSALITE ÉLÉMENTAIRE

< base de la synchronisation répartie>

Par la nature physique de la communication, l'émission d'un message sur un site précède nécessairement la réception du message sur le site destinataire. Toute réception d'un message est causée par une émission antérieure. (il ne peut y avoir de réception spontanée de message)

Cette relation causale permet d'établir, dans un système réparti, une relation d'ordre partiel entre l'événement d'émission d'un message sur un site et l'événement de réception du message sur un autre site destinataire. cette relation se note -> (on lit 'a précédé') :

$$\forall m, \text{ÉMISSION}(m) \rightarrow \text{RÉCEPTION}(m)$$

(notée "happened before" par Lamport)

Plus exactement la relation est établie lorsque le message a été reçu

$$\forall m, \text{RÉCEPTION}(m) \Rightarrow (\text{ÉMISSION}(m) \rightarrow \text{RÉCEPTION}(m))$$

RELATION DE PRÉCÉDENCE ENTRE DES ÉVÉNEMENTS RÉPARTIS

événement : instruction exécutée par un processeur

(précédence : "happened before", L.Lamport, CACM 21,7, 1978)

a) A "a précédé" A' si A et A' sont des événements qui ont été générés dans cet ordre sur le même site S (ordre d'exécution local) : A -> A'

b) A "a précédé" A' si A est l'événement d'émission d'un message M par le site P et que A' est l'événement de réception du message M sur Q : A ->A' (ordre causal pour chaque message)

La relation de précédence dans un système réparti est la fermeture transitive des deux relations précédentes.

si A -> B et B -> C alors A -> C

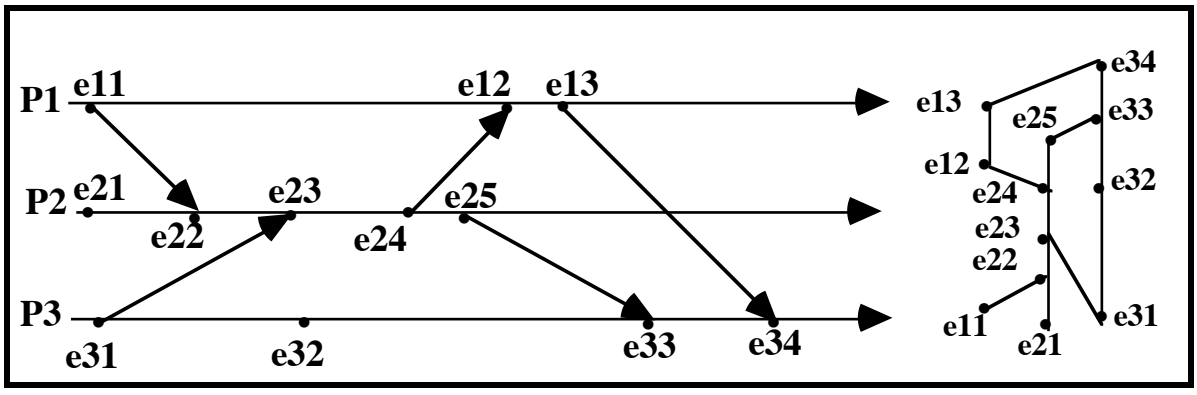
La précédence est un ordre partiel entre les événements du système réparti

La causalité entre événements implique la précédence entre eux :

A cause B => A -> B

Une précédence entre événements exprime une causalité potentielle :

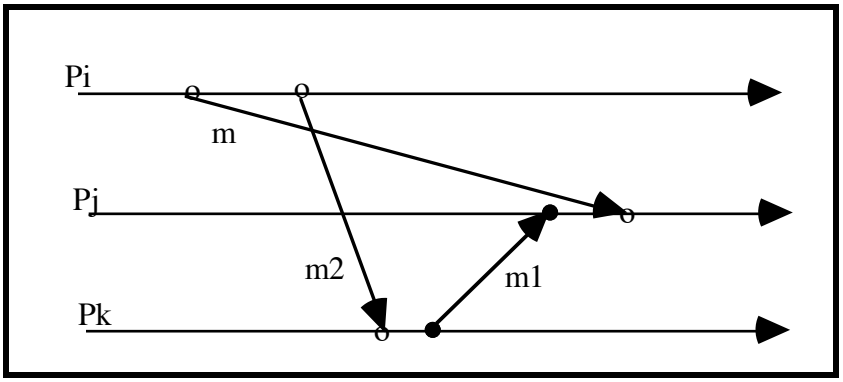
(si A -> B, A peut avoir influencé B).



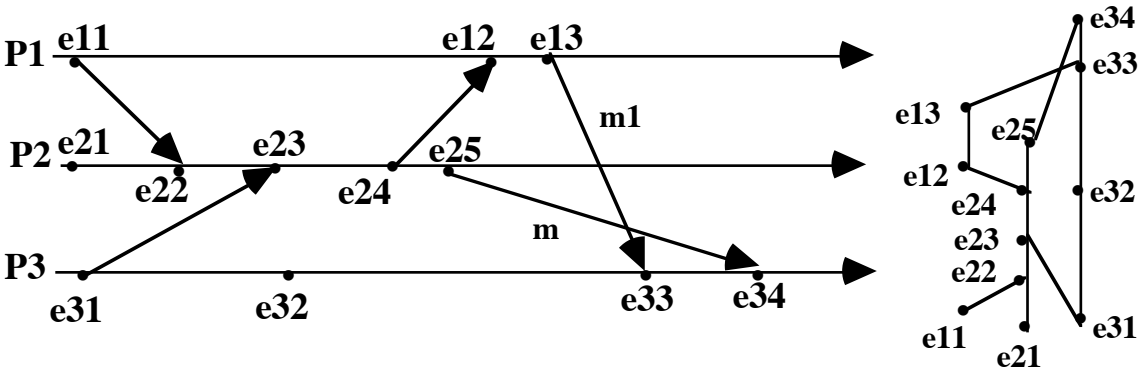
DÉPENDANCE CAUSALE ENTRE MESSAGES

Les messages respectent la dépendance causale si et seulement si :

$\forall$Pi, $\forall$Pj, $\forall$Pk, $\forall$m émis sur Cij, $\forall$m1 émis sur Ckj,
$EMISSION_i(m) \rightarrow EMISSION_k(m1) \Rightarrow RECEPTION_j(m) \rightarrow RECEPTION_j(m1)$

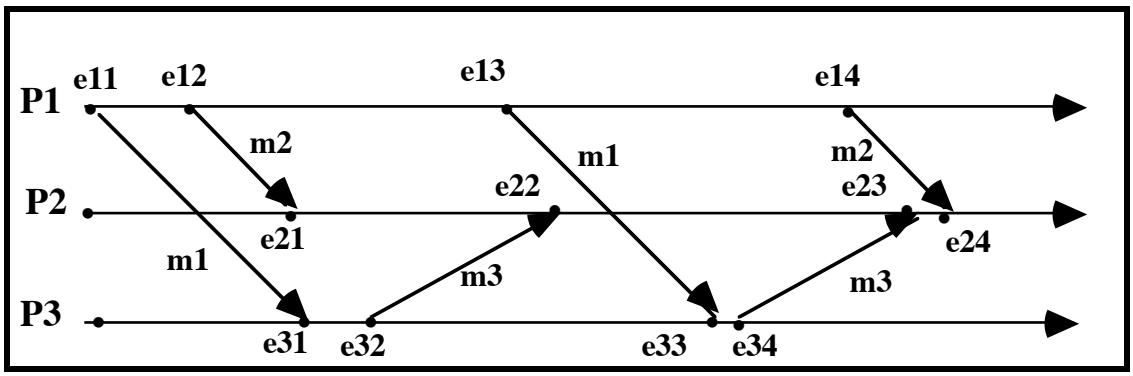


Premier exemple : la réception sur Pj ne respecte pas la dépendance causale



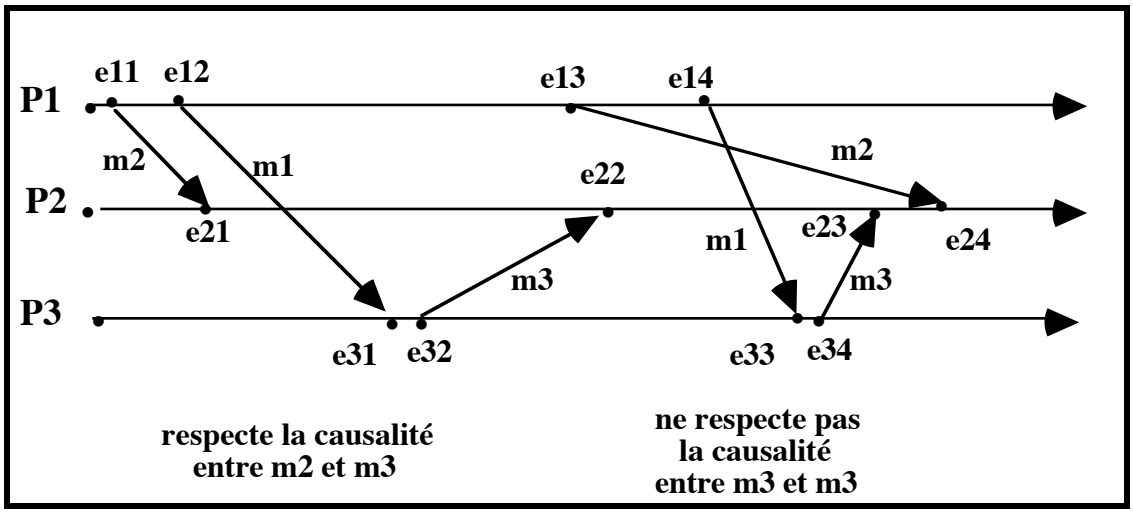
Deuxième exemple : la réception respecte la dépendance causale
car les émissions e13 et e25 ne sont pas en relation de précédence

DÉPENDANCE CAUSALE ENTRE MESSAGES
Un problème de logique



pas de dépendance causale entre EMISSION1(m2) EMISSION3(m3)
Donc il n'y a pas de dépendance causale entre m2 de P1 et m3 de P3

m1 : "Je <u>vais demander</u> à P2 de te rencontrer. Appelle le de ma part"



EMISSION1(m2) -> EMISSION3(m3)=> RECEPTION2(m2) -> RECEPTION2(m3)
Donc il y a dépendance causale entre m2 de P1 et m3 de P3

m1 : "<u>J'ai demandé</u> à P2 de te rencontrer. Fixe lui une date et un lieu"

HYPOTHESES PROPRES A UN CANAL C_{ij}

(canal : liaison point à point entre un émetteur et un récepteur)

DEFINITION : m1 double m2 dans le canal C_{ij} si et seulement si

$$\text{EMISSION}_i(m2) \rightarrow \text{EMISSION}_i(m1)$$

$$\text{et RECEPTION}_j(m1) \rightarrow \text{RECEPTION}_j(m2)$$

PROPRIÉTÉS D'ORDRE DES MESSAGES DANS LES CANAUX

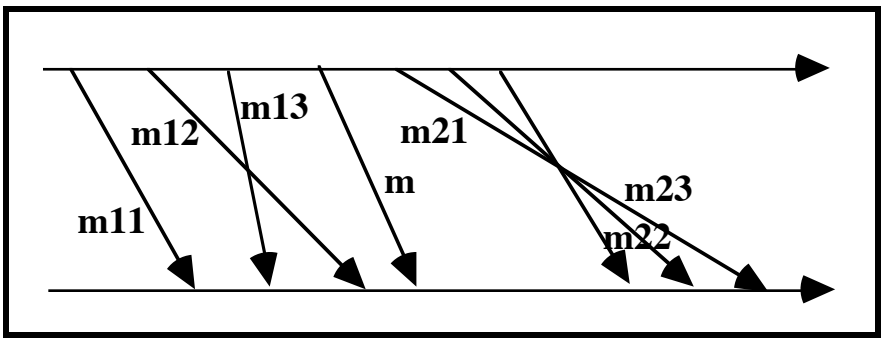
1• un message marqueur m ne peut ni doubler ni être doublé sur C

$\forall$m1, $\forall$m2,

EMISSION(m1) ->EMISSION(m) => RECEPTION(m1) -> RECEPTION(m)

EMISSION(m) -> EMISSION(m2) => RECEPTION(m) -> RECEPTION(m2)

2• un message ordinaire m n'impose pas de condition de réception, mais respecte celles des autres (il ne peut doubler les marqueurs et ne peut être doublé par les marqueurs).



Tout marqueur m sépare les messages du canal en deux sous-ensembles

$<_m$ = {m1 | EMISSION(m1) -> EMISSION(m) }

et m est un marqueur => RECEPTION(m1) -> RECEPTION(m)

$>_m$ = {m2 | EMISSION(m) -> EMISSION(m2) }

et m est un marqueur => RECEPTION(m) -> RECEPTION(m2)

TYPES DE COMPORTEMENT D'UN CANAL

1• le moins contraint : tous les messages sont ordinaires

2• le plus contraint : tous les messages sont des marqueurs (canal FIFO)

MODELES DE DIFFUSION FIABLE ET COMMUNICATION DE GROUPE

Un message émis doit être reçu par n destinataires.

MODELES DE COMMUNICATION

Communication inclusive ou non (l'émetteur reçoit le même message - le sien enrichi par le réseau- que les récepteurs)

Communication interne ou externe (l'émetteur, client du groupe, n'appartient pas au groupe)

CLASSIFICATION SELON LES EMETTEURS ET LES RECEPTEURS (classification OSI)

MODE CENTRALISE ("multicast")

Un seul émetteur (toujours le même) et n récepteurs.

MODE CENTRALISE A CENTRE MOBILE

L'émetteur est unique par périodes.

MODE MULTI-CENTRE

N processus émetteurs peuvent à tout instant effectuer une diffusion vers P récepteurs.

MODE DECENTRALISE OU MODE CONVERSATION

Un ensemble de N sites peuvent être à tout instant émetteurs et sont tous destinataires des messages.

PROPRIETES D'ORDRE DANS LES GROUPES

DIFFUSION RESPECTANT L'ORDRE LOCAL

Pour deux diffusions successives du même processus, les messages sont délivrés dans le même ordre sur chaque site distant

DIFFUSION RESPECTANT L'ORDRE CAUSAL

diffusion + causalité (Birman et Joseph 1987)

Relation de dépendance causale entre les messages, généralisée à la diffusion

Toute suite de diffusions de messages en relation de causalité implique la délivrance des messages sur tous les sites destinataires dans la même relation de causalité

$\forall$Pi, $\forall$Pk, $\forall$m

DIFFUSION$_i$(m) précède causalement DIFFUSION$_k$(m1)

=> RECEPTION$_j$(m) précède RECEPTION$_j$(m1) pour tout P_j.

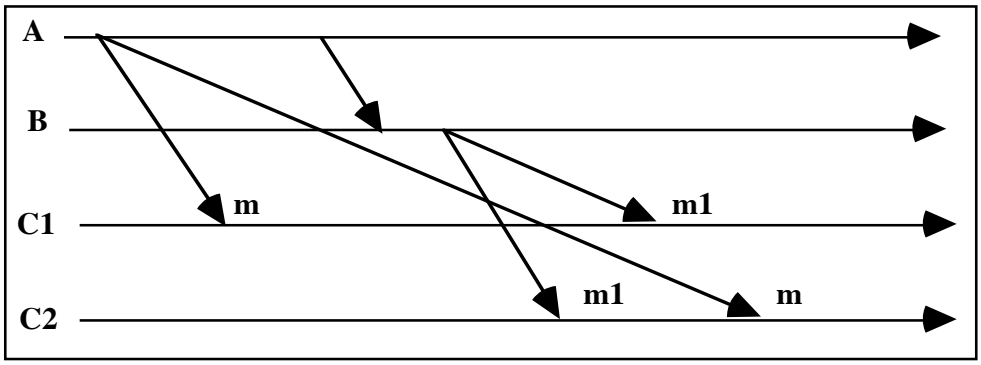
EXEMPLES D'UTILISATION DE LA DIFFUSION CAUSALE

Exemple 1:

A diffuse un courrier électronique m à C1 et C2 qui contient: "je vais demander à B de vous diffuser du travail par courrier électronique ".

Pour respecter l'ordre causal, C1 et C2 ne doivent pas recevoir le courrier m1 de B avant celui m de A.

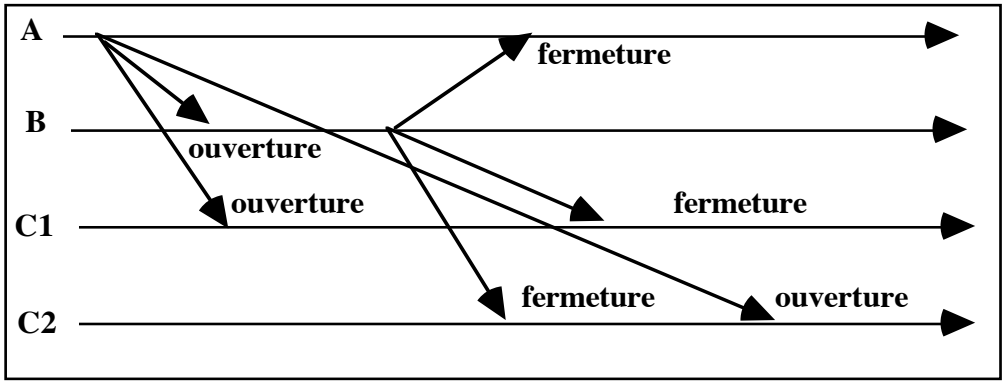
émission$_A$(m) -> émission$_B$(m1) => réception$_C$(m) -> réception$_C$(m1)



Exemple 2:

A envoie à C1 et C2 une commande d'ouverture de vanne, en en rendant compte à B. Plus tard B envoie à C1 et C2 l'ordre de fermeture de la vanne, en en rendant compte à A.

Respect de la causalité : le message de A (ouverture) doit être enregistré avant celui de B (fermeture).



DIFFUSION RESPECTANT UN ORDRE TOTAL SIMPLE

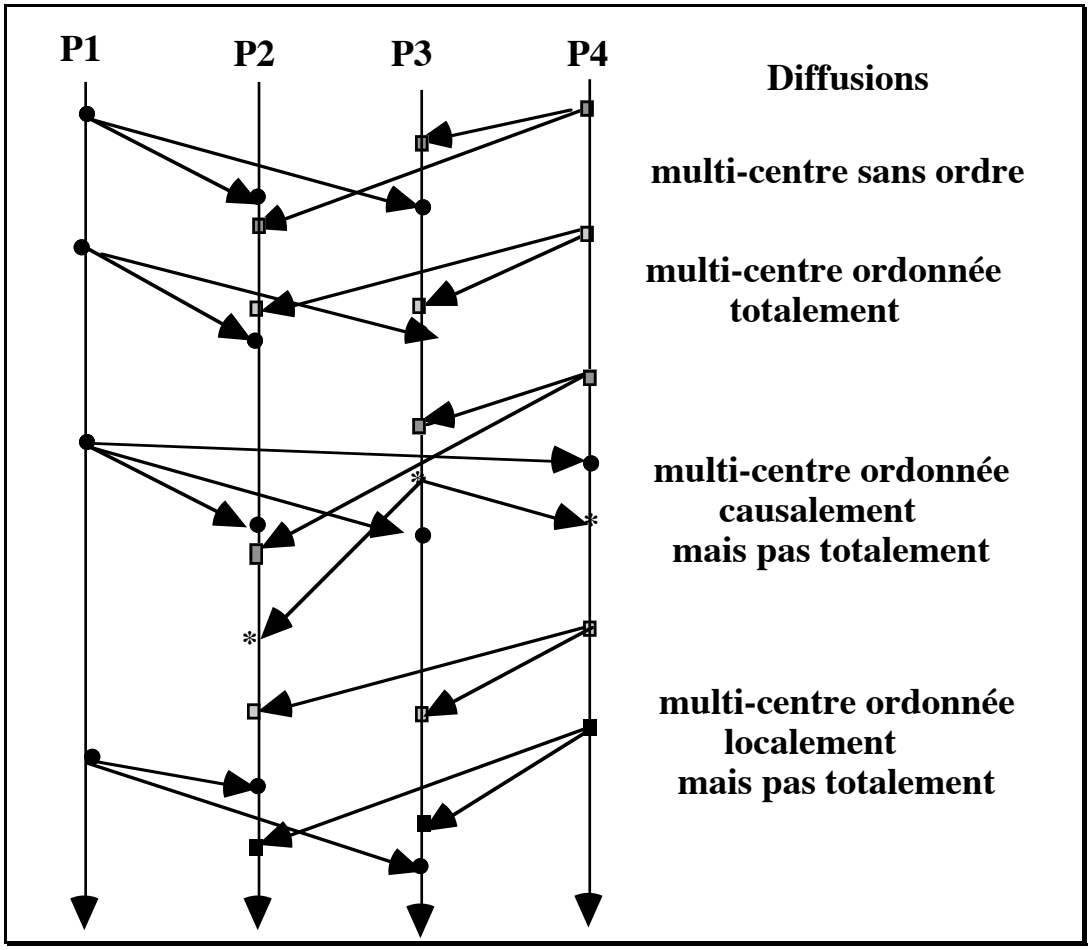
Si plusieurs diffusions ont lieu concurremment de différents processus vers le même groupe de diffusion, alors tous les messages sont délivrés aux applications réceptrices dans le même ordre sur tous les récepteurs.

Exemple d'utilisation : copies multiples.

Le fait que toutes les opérations de modifications d'un ensemble de données en copies multiples soient effectuées dans le même ordre sur toutes les copies suffit à assurer le maintien de la cohérence (faible) des copies.

DIFFUSION RESPECTANT L'ORDRE TOTAL CAUSAL

L'ordre total respecte aussi la relation d'ordre causal entre messages



QUELQUES COMPLÉMENTS

PROPRIÉTÉS GÉNÉRALES

SÛRETÉ : rien de "mauvais" ne peut se produire

- **une base de données réparties n'est jamais dans un état incohérent**
- **si un processus correct livre un message en diffusion fiable, tous les processus corrects livrent aussi ce même message**
- **absence d'interblocage**

VIVACITÉ : quelque chose de "bon" finit par se produire

- **si un processus correct diffuse un message en diffusion fiable, il livre ce message**
- **absence de famine**

PONCTUALITÉ : les activations et les terminaisons de processus, les remises de messages se font selon les contraintes temporelles définies

- **échéance de terminaison au plus tard**
- **échéance de remise de message au plus tard**
- **gigue bornée**
- **écart entre les temps de réponse minimum et maximum**
- **déviation maximum par rapport à une suite de valeurs périodiques**
- **en général, absence de faute temporelle**

CONFIANCE : comportement du système en présence de défaillances partielles

- **un processus est soit exécuté par tous les serveurs, soit par aucun**
- **le système réalise les fonctions vitales spécifiées**
- **le système reste dans un état sécuritaire**

Nota : voir aussi QUALITÉ DE SERVICE (QoS)

MODÈLES DE DÉFAILLANCES

MODÈLE À PANNES FRANCHES ET CANAUX FIABLES

• Les processeurs :

panne franche si arrêt immédiat et prématuré d'un processus

• Les processus :

défaillant s'il subit une panne franche

correct sinon

un processus défaillant le reste (pas de rétablissement "recovery")

• Le réseau :

fiable : ni perte ni altération de message

• Conclusion : deux processus corrects peuvent toujours communiquer

MODÈLE PAR OMISSION

ex : non réception d'un message lorsque les tampons de réception sont pleins

• En point à point :

• nombre maximum d'omissions connu sur une tranche de temps (ex. durée de la mission du système)

• tout message émis par un processus correct finit par être reçu par son destinataire correct (et si l'émetteur défaille après sa première émission et que le message est perdu par suite d'omission sur le canal)

• En liaison multipoint, comptabilité des omissions

• une omission par destinataire correct n'ayant pas reçu le mesage

• une omission si le message n'a pas été reçu par tous les destinataires corrects

DÉTECTEURS NON FIABLES DE DÉFAILLANCES DANS LA COMMUNICATION ASYNCHRONE

• L'asynchronisme des processus et des communications, dans un SR asynchrone, rend impossible, pour un processus donné, la distinction entre un processus défaillant et un processus extrêmement lent ou avec lequel les communications sont extrêmement lentes.

• Il n'est donc pas possible depuis l'intérieur d'un SR asynchrone de détecter les défaillances en assurant à la fois :

la PRÉCISION (propriété de sûreté) : seuls les processus défaillants doivent être détectés (on ne fait pas de fausse détection de défaillance),

la COMPLÉTUDE (propriété de vivacité) : les processus défaillants doivent être détectés (on ne doit pas oublier d'en détecter quelques uns).

On équipe chaque processus avec un *détecteur de défaillance* qui maintient la liste des processus qu'il suspecte de défaillances.

• Mise en oeuvre avec des horloges de garde qui donnent l'alarme lorsqu'une réponse n'est pas parvenue au bout d'un certain délai.

• Détecteurs non fiables car on ne peut pas garantir que les délais de garde ont été bien choisis pour tous les cas de figure.

PROPRIÉTÉS DES DÉTECTEURS NON FIABLES DE DÉFAILLANCE

COMPLÉTUDE FORTE (FAIBLE) : tout processus défaillant sera suspecté de façon permanente par tout (au moins un) processus correct.

PRÉCISION FORTE (FAIBLE) : tout (il existe un) processus correct (qui) n'est jamais suspecté par les processus corrects

PRÉCISION FORTE (FAIBLE) INÉLUCTABLE : il existe un instant à partir duquel la propriété de précision forte (faible) sera satisfaite

CLASSES DE DÉTECTEURS DE DÉFAILLANCE [Chandra 96]

détecteurs fiables : précision forte et complétude forte
S : complétude forte et précision faible
◊ S : complétude forte et précision faible inéluctable
◊ W : complétude faible et précision faible inéluctable
◊ S et ◊ W sont les classes les plus faibles permettant de résoudre le consensus

Nota
propriétés supposées avec une certaine probabilité
propriétés supposées pendant un certain délai de fonctionnement
suspecter une défaillance ≠ détecter une défaillance
vérification a posteriori de la véracité des propriétés (est-ce possible?)

SYSTÈMES RÉPARTIS À COMMUNICATIONS ASYNCHRONES
RÉSULTAT D'IMPOSSIBILITÉ
[Fischer 85]

Il est impossible de concevoir un protocole déterministe pour résoudre le problème du consensus dans un système asynchrone dans lequel (seulement) un processus peut être défaillant par panne franche.

Intuitivement ceci est dû à l'impossibilité de distinguer un processus défaillant d'un processus extrêmement lent.

Remarques

1. Il y a des solutions relativement simples dans les systèmes répartis synchrones

2. Il existe des protocoles probabilistes utilisant des tirages aléatoires. Ces protocoles sont non-déterministes

3. Il existe des solutions dans un système réparti asynchrone équipé de détecteurs de défaillances non fiables satisfaisant certaines propriétés relatives à la sûreté et à la vivacité des détections [Chandra 96].

IMPORTANCE DE CE RÉSULTAT D'IMPOSSIBILITÉ

On a montré que toute une série de problèmes d'accord réparti sont équivalents :

accord : offrir à chaque site correct une vision commune de certaines variables (états ou messages) du système et cela en présence de défaillance des sites

consensus = généraux byzantins = problème des deux armées

consensus <=> diffusion atomique <=>validation atomique

consensus <=> synchronisme virtuel <=> diffusion atomique sélective

LE PROBLÈME DU CONSENSUS

On considère un ensemble de n processsus (sur n sites) qui peuvent être défaillants par panne franche. Chaque processus Pi est doté d'une valeur potentielle notée Vi. Chaque processus correct propose sa valeur à l'ensemble des autres processus.

Problème : concevoir un protocole tel que tous les processus corrects se mettent d'accord de façon irrévocable sur une valeur résultat appelée valeur décidée. La valeur décidée doit être une des valeurs proposées;

SPÉCIFICATION DU CONSENSUS

par 4 propriétés :

TERMINAISON : si les processus corrects lancent le consensus alors tout processus correct décide d'une valeur au bout d'un temps fini.

INTÉGRITÉ : tout processus décide au plus une fois.

VALIDITÉ : toute valeur décidée est l'une des valeurs potentielles.

ACCORD : la valeur décidée est la même pour tous les processus corrects.

Le CONSENSUS UNIFORME suppose une autre propriété d'accord;

ACCORD UNIFORME : tous les processus qui décident d'une valeur décident de la même valeur.

Un processus défaillant qui a décidé avant de subir une défaillance ne peut décider une valeur différente de celle qui est choisie par les processus corrects. Important si on rétablit un processus après défaillance ("recovery").

TRANSACTIONS RÉPARTIES

Propriétés ACID d'une transaction [Bernstein 87]

ATOMICITÉ :

soit la transaction est validée et ses effets sur la base sont permanents, soit la transaction est avortée et elle n'a aucun effet sur la base

COHÉRENCE :

une transaction préserve la cohérence des objets qu'elle manipule

ISOLATION :

les effets d'une transaction sont cachés aux transactions concurrentes

DURABILITÉ :

les effets d'une transaction validée sont permanents

TRANSACTION RÉPARTIE [Besancenot 97] :

accède à des objets gérés par des processus sur des sites différents

Problèmes additionnels pour le contrôle de concurrence réparti :

- **détection de défaillances**
- **verrouillage réparti**
- **interblocage réparti**
- **estampillage réparti**
- **certification répartie**
- **validation atomique répartie**

VALIDATION ATOMIQUE DES TRANSACTIONS RÉPARTIES

par un vote de l'ensemble des processus participants

vote OUI si le participant est d'accord pour valider la transaction

vote NON sinon

La décision ne peut être de valider que si tous les votes sont OUI.

La décision prise est irréversible et suivie par les processus

Propriétés nécessaires (qualité de service)

UNANIMITÉ :

deux participants ne peuvent décider différemment (sûreté)

VALIDITÉ :

la transaction n'est validée que si tous les participants votent oui (sûreté)

TERMINAISON :

si toutes les défaillances sont réparées et aucune défaillance ne survient, tous les participants doivent décider (vivacité)

NON-TRIVIALITÉ :

les participants doivent décider de valider si tous les votes sont oui et si aucun site n'est défaillant ou suspecté d'être défaillant

(éviter le protocole trivial où l'abandon est systématique)

NON-BLOCAGE :

tout participant correct décide

si défaillance de certains participants, les autres doivent décider de la terminaison et libérer les ressources (relâcher les verrous)

Protocole de validation atomique en deux phases (2PC)

Protocole de validation atomique en trois phases (3PC)

COMMUNICATION DE GROUPES

UTILISATEURS : applications coopératives, quasi video à la demande, transactions boursières, vente aux enchères, gestion des alarmes dans un système de contrôle-commande.

EXIGENCES DES UTILISATEURS DU GROUPE (qualité de service)
nombre d'émetteurs : un ou plusieurs
nombre de récepteurs : problème d'échelle ("scalability") si beaucoup
possibilité ou non de se joindre à une communication en cours
possibilité ou non d'exclure les récepteurs trop lents
absence de sous-groupes autonomes et vue unique du groupe
défaillances tolérées : aucune, omission réseau uniquement, arrêt des récepteurs, arrêt des émetteurs, arrêt des récepteurs et arrêt des émetteurs, omission réseau et arrêt des récepteurs, omission réseau et arrêt des récepteurs et arrêt des émetteurs,...

MEMBRES D'UN GROUPE : participants, défaillants, absents
mise au courant (de l'état du groupe = vue + état de l'application répartie) des nouveaux participants après absence ou défaillance

COEXISTENCE de plusieurs protocoles de groupe

DIFFUSION

DIFFUSION SÉLECTIVE ("multicast")

DIFFUSION GÉNÉRALE ("broadcast")

vue du groupe = ensemble des membres effectivement présents

= membres non défaillants, non partis ou non exclus

un algorithme de diffusion peut faire intervenir :

- seulement les processus source et destinataires du message

- la vue du groupe

DIFFUSION CONTRÔLÉE : tout message émis par un processus correct est remis par tout destinataire correct, membre de la vue du groupe. Si l'émetteur défaille, certains destinataires corrects peuvent remettre le message et d'autres non.

DIFFUSION FIABLE : d'une part si un émetteur correct diffuse un message, alors ce message est remis par au moins un destinataire correct, et d'autre part si un destinataire correct remet un message alors tous les destinataires corrects remettent ce même message.

DIFFUSION A L'ANCIENNETE ("FIFO") : l'ordre de remise des messages préserve l'ordre d'émission des messages par la source

DIFFUSION CAUSALE : l'ordre de remise des messages préserve l'ordre causal

DIFFUSION ATOMIQUE : fiable et satisfaisant un ordre total

ORDRE CAUSAL

ORDRE TOTAL

ORDRE CHRONOLOGIQUE : l'ordre de remise des messages préserve l'ordre chronologique (horloges physiques) d'émission des messages

Ordre local/global : à un groupe /à tous les groupes simultanément

RECHERCHE DE MODÈLES INTERMÉDIAIRES

DEUX MODÈLES EXTRÊMES :

- **communication synchrone fiable**
- **communication asynchrone**

TRAVAUX DE RECHERCHE ACTUELS

Définir des modèles intermédiaires fondés sur des hypothèses probabilistes

- **de bon fonctionnement sur une durée supposée ou observée a posteriori,**
- **de taux de défaillances pendant une durée donnée,**
- **d'intervalle de temps entre défaillances, etc...**

EXEMPLES DE NOUVEAUX MODÈLES

- **Modèle asynchrone temporisé ("timed asynchronous system" [Cristian 96])**
  - **Pas de mémoire commune**
  - **Synchronisation logicielle des horloges physique**
  - **Le temps de transfert des messages n'est pas borné, mais la plupart des messages arrivent avant un délai constant connu d**
  - **Pas de minorant pour les vitesses des processeurs, mais la plupart des traitement de message sont bornés par un délai constant connu s**
  - **Le service de message de bout en bout est donc borné, la plupart du temps, par b = s + d + s**
  - **Un système asynchrone temporisé est en fonctionnement stable pendant un intervalle temporel s'il n'y a pas de panne de processeur et si chaque message est délivré avant un délai b**

- **Modèle quasi-synchrone, avec un canal synchrone pour véhiculer l'heure ([Verissimo])**

2. ETAT GLOBAL D'UN SYSTEME REPARTI

• ETAT LOCAL ELi D'UN SITE Si

état initial et séquence d'événements locaux sur Si

• ETAT LOCAL ECij D'UN CANAL Cij

ensemble des messages en transit sur le canal Cij

émis par Si et non encore reçus par Sj

• EVENEMENTS (ATOMIQUES) FAISANT EVOLUER LE SYSTEME

{ELi} événement interne sur Si {EL'i}

{ELi, ECij} émission de m par Si sur Cij {EL'i, EC'ij = ECij ∪ {m}}

{ELi, ECki} réception de m par Si sur Cki {EL'i, EC'ki = ECij - {m}}

• ETAT GLOBAL S = { pour tout i, j, ∈ ELi, ∈ ECij}

• DIFFICULTES

ELi n'est immédiatement observable que sur Si

ECij n'est jamais directement observable, ni sur Si, ni sur Sj

• OBJECTIF : définir un état global déterminable <u>localement</u> par tout site

REMARQUE : toute définition d'état doit respecter la dépendance causale

nota : on dit indifféremment site ou processus

(dans ce cas il y a un processus par site)

PASSÉ D'UN ÉVÉNEMENT

Le passé (ou historique) d'un événement e, c'est par définition :

hist(e) = e ∪ ensemble des événements e' tels que e' -> e

Seul le passé de e peut avoir influencé e

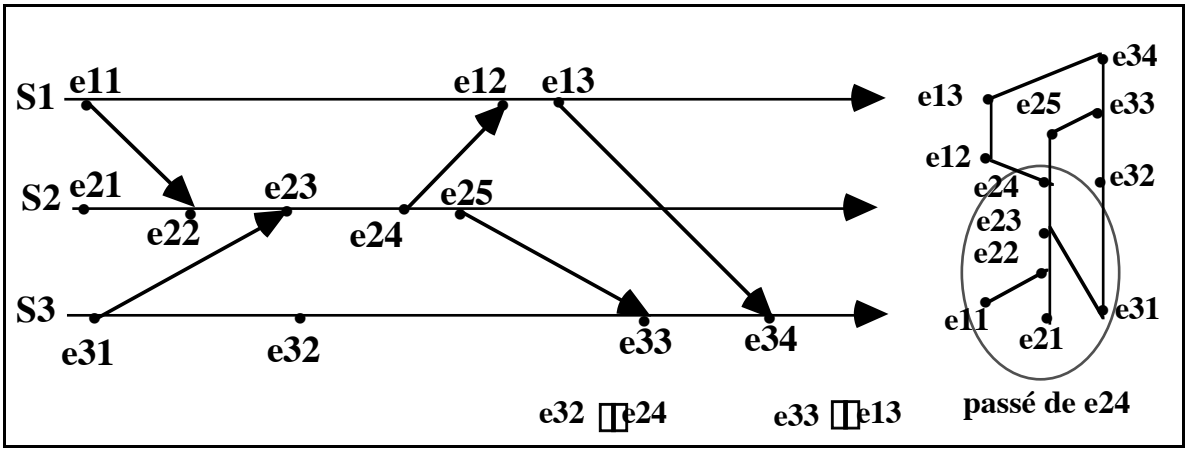
Chaîne causale : e_0 ... e_n tels que e_{i-1} -> e_i pour tout i ∈ (1...n)

Evénements concurrents (ou encore causalement indépendants)

a ∥ b <=> non (a -> b) et non (b -> a)

aucun des 2 événements n'appartient au passé de l'autre

aucun des 2 événements ne peut influencer l'autre



Applications

définition de la cohérence d'un état, d'une observation

mise au point répartie

mesure du parallélisme

COUPURES

soit E un ensemble d'événements constituant une application répartie

Coupure = sous-ensemble fini C de E tel que pour a, b ∈ E

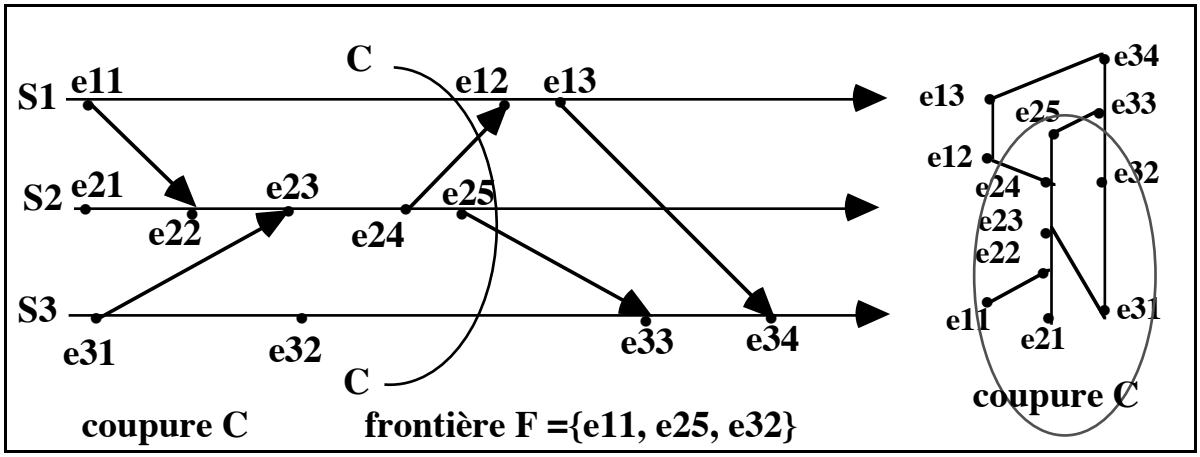
a ∈ C et (b précède localement a) -> (b ∈ C)

Photographie instantanée d'un système obtenue en prenant un événement par site et tous les événements du site qui le précède.

C'est un sous-ensemble de l'histoire de l'application qui contient toute l'histoire qui le précède : cela permet de définir un passé et un futur
(par rapport à la coupure)

Frontière F d'une coupure C :

ensemble des événements les plus récents de la coupure, un par site
a ∈ F <=> a ∈ C et il n'existe pas de b ∈ C tel que a -> b



COUPURES COHERENTES

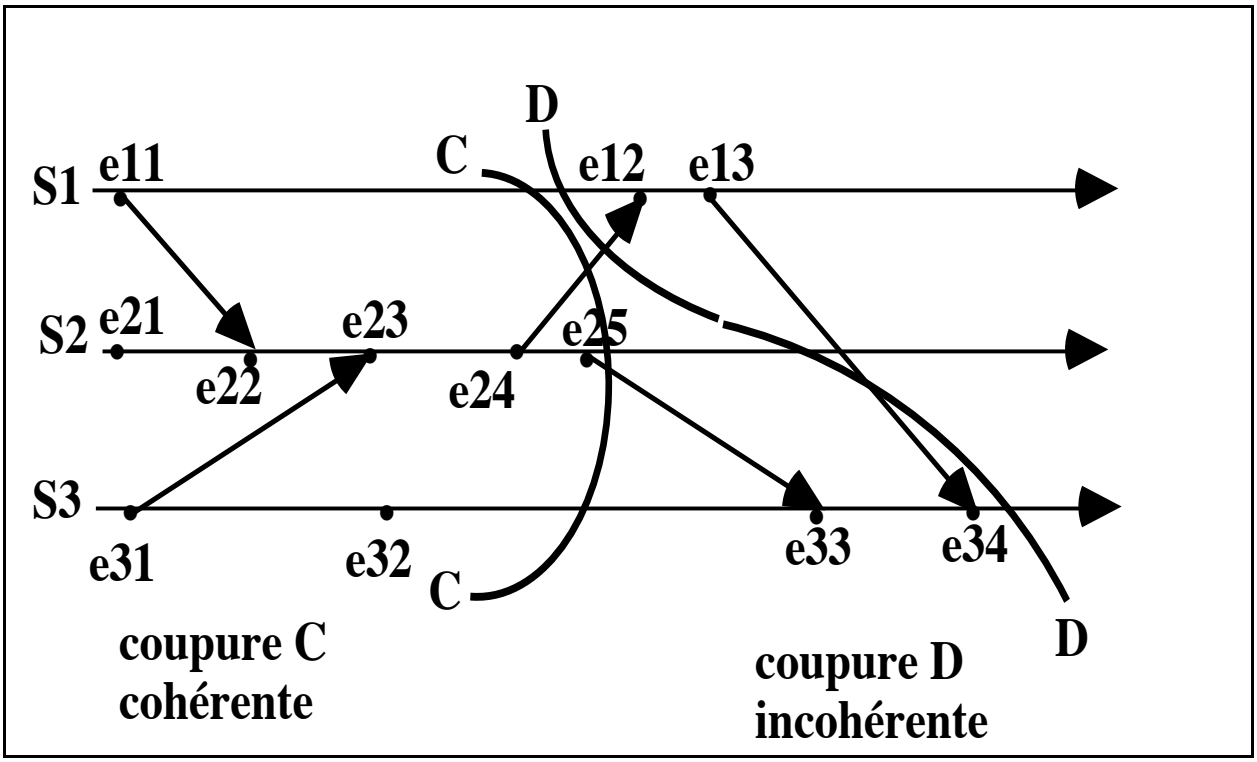
Cohérence = respect de la causalité dans la coupure

une chaîne causale ne peut sortir et re-entrer dans la coupure

un message ne peut pas venir du futur en franchissant la frontière

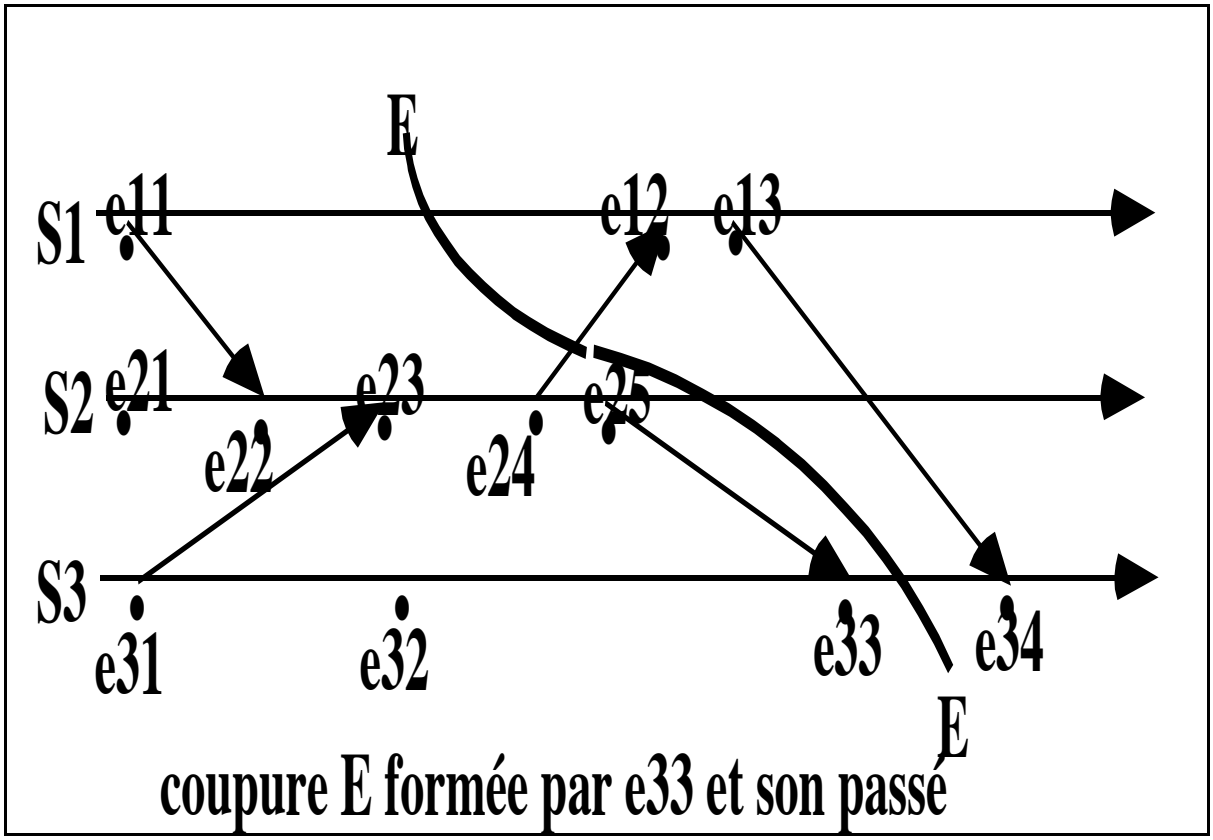
Coupure cohérente = coupure fermée par la relation de dépendance causale

a ∈ C et (b -> a) => (b ∈ C)



Etat global cohérent = état associé à une coupure cohérente

COUPURES COHERENTES



Exemple de coupure cohérente : le passé d'un événement

ETAT GLOBAL COHÉRENT D'UN SYSTEME REPARTI

• soit ELi état local du site Si pour tout i (histoire locale de Si)

• soit ECij état local du canal Cij pour tout (i, j)

état global S = { pour tout i, j, ∈ ELi, ∈ ECij }

coupure associée = { pour tout i, ∈ ELi }

état global S cohérent si coupure associée cohérente

La frontière de la coupure associée définit un passé et un futur

Il en résulte deux conditions nécessaires pour les messages m qui traversent une frontière de coupure

condition C1 :

si EMISSION$_i$(m) ∈ ELi alors

- **soit RECEPTION$_j$(m) ∈ ELj,**
- **soit m ∈ ECij**

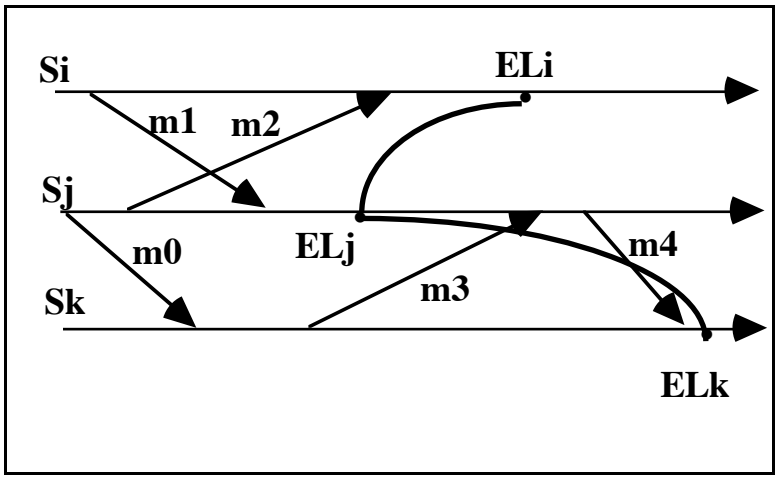
tout message émis dans le passé est soit reçu dans le passé soit en transfert

condition C2 :

si EMISSION$_i$(m) ∈ ELi alors RECEPTION$_j$(m) ∈ ELj

tout message émis dans le futur reste dans le futur

UNE EXECUTION REPARTIE



état global {ELi, ELj, ELk}

1• EMISSION$_k$(m3) ∉ ELk mais

comme RECEPTION$_j$(m3) ∈ ELj,

il faut que m3 ∈ ECkj

donc la condition C1 est vraie seulement si m3 ∈ ECkj

2• EMISSION$_j$(m4) ∈ ELj mais RECEPTION$_k$(m4) ∉ ELk

la condition C2 est fausse car

C2 : si EMISSION$_j$(m4) ∈ ELj alors RECEPTION$_k$(m4) ∈ ELk

conclusion

{ELi, ELj, ELk} n'est pas un état global cohérent

DETERMINATION D'UN ETAT GLOBAL COHERENT

(Chandy - Lamport 1985)

Hypothèses

- **le réseau de communication est connexe**
- **les canaux respectent l'ordre d'émission des messages (canaux FIFO)**
- **un site "élu" particulier lance la détermination d'état global**

Principe

- **association d'un message marqueur à chaque enregistrement d'état local d'un site pour que les autres sites puissent repérer les messages qui sont avant ou après cet enregistrement (les messages du passé et du futur)**
- **chaque site détermine son état local et celui de ses canaux en réception, puis envoie ces états au site "élu"**

Propriétés

- **l'algorithme se termine**
- **l'état global enregistré correspond à une coupure cohérente**
- **les états enregistrés des canaux sont corrects pour cette coupure**

SOLUTION DE CHANDY et LAMPORT (1985)

HYPOTHÈSE : CANAUX FIFO

tout message mk envoyé sur un canal FIFO sépare les messages du canal en deux sous-ensembles : $<_{mk}$ (avant mk) et $>_{mk}$ (après mk).

CONTRAINTES DE COHÉRENCE

• Quand Si enregistre son état ELi, toutes les émissions de messages faites par Si avant ELi sont captées dans ELi et les réceptions de ces messages doivent être captées dans les ELj et Cij des sites Sj (condition C1) et celles-là seulement (condition C2).

MISE EN OEUVRE

• Dès que Si enregistre ELi, il émet un message marqueur mk sur chaque canal Cij. Sj doit enregistrer ELj au plus tard à la réception de mk et Sj doit capter tous les messages $<_{mk}$, soit dans ELj soit dans Cji (condition C1). Les messages de $>_{mk}$ ne doivent pas être captés car ils viennent du futur de ELi sur Si (condition C2).

ALGORITHME SUR CHAQUE SITE Si

Début (Si "élu") ou première réception d'un marqueur mk (émis par Sj)

(1) enregistrer ELi

(2) enregistrer ECji comme vide car Si a déjà reçu tous les messages $<_{mk}$ et ceux de $>_{mk}$ ne font pas partie de l'état global.

Au démarrage sur "élu", aucun ECji n'est enregistré.

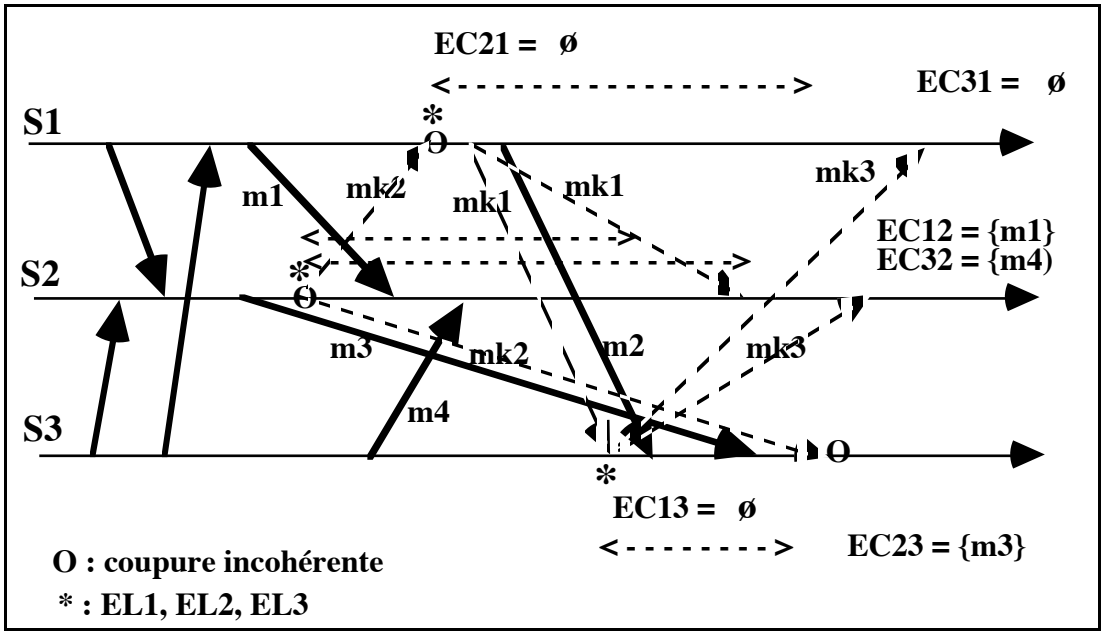
(3) diffuser mk à tous ses voisins sur les canaux Cij

(1-2-3) doit être atomique

Réceptions suivantes du marqueur mk (émis par Sj)

enregistrer ECji comme constitué des messages <mk reçus par Si entre l'enregistrement de ELi et l'arrivée de mk (envoyés par Pj avant ELj mais pas reçus par Pi au moment de ELi)

EXEMPLE DE DETERMINATION D'UN ETAT COHERENT



S2 lance la détermination d'état global et diffuse mk2

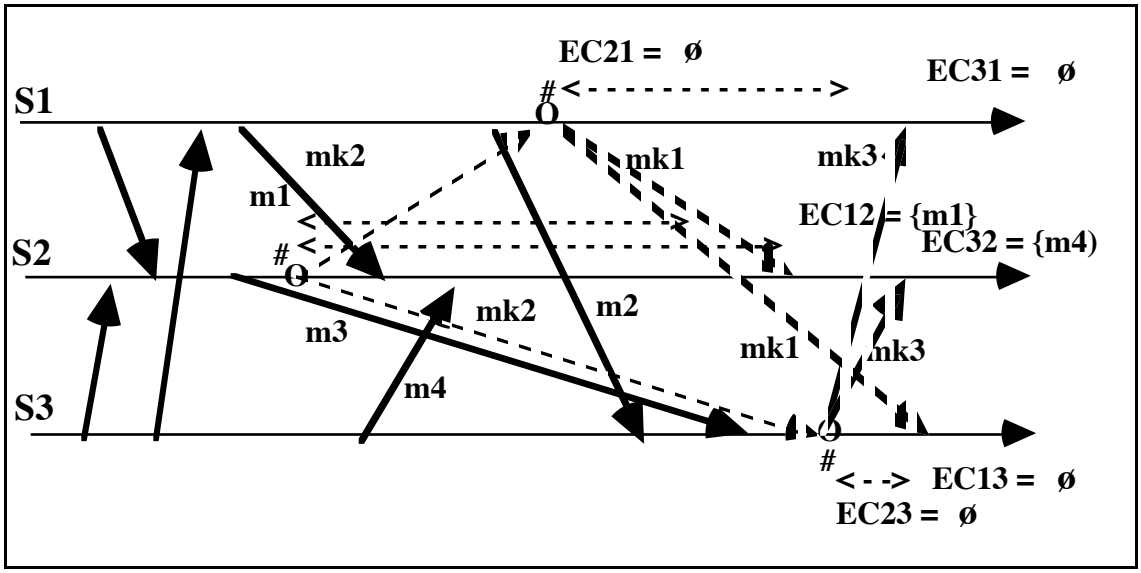
Les 3 événements émission2(mk2), réception1(mk2) et réception3(mk2) déterminent une coupure incohérente.

Les marqueurs mk1 et mk3 permettent :

- **de forcer la transitivité de la causalité et d'avoir une coupure cohérente.**
- **de noter dans les ECij les messages qui traversent la coupure cohérente.**

EXEMPLE DE DETERMINATION D'UN ETAT COHERENT

AUTRE EXÉCUTION POSSIBLE => AUTRE ÉTAT COHÉRENT



autre trace et autre état cohérent

DETERMINER UN ETAT GLOBAL DANS UN SYSTEME REPARTI

éléments de bibliographe

Solution pour un canal FIFO :

K.M. Chandy and L.Lamport, ***Distributed Snapshots : Determining Global States of Distributed Systems*****. ACM TOCS, Vol 3,1, (1985) pp. 63-75**

solutions pour les canaux non FIFO

méthode cumulative (et rapide) de Lai et Yang : T.H.Lai and T.H.Yang, ***On Distributed Snapshots*****; Inf. Proc. Letters, Vol. 25, (1987), pp. 153-158**

méthode non cumulative (mais lente) de Mattern : F.Mattern, ***Virtual Time and Global States of Distributed Systems*****. Proc. of Int. Workshop on Parallel and Dist. Systems, North Holland, 1988, pp. 215-226**

solution utilisant des messages de contrôle : M.Ajuha, ***Global Snapshots for Asynchronous Distributed Systems with non FIFO Channels*****. Tec. Rep. #CS92-268, U. of Calif., San Diego (1992), 7 p.**

solutions fondées sur l'ordre causal

méthode centralisée d'Acharya et Badrinath : A.Acharya and B.R.Badrinath, ***Recording Distributed Snapshots Based on Causal Order of Message Delivery*****. Inf. Proc. Letters, Vol. 44, (1992), pp.317-321**

méthode répartie d'Alagar et Venkatesan : S.Alagar and S.Venkatesan, ***An Optimal Algorithm for Distributed Snapshots with Causal Message Ordering*****, Tec. Rep, U. of Texas, Dallas (193), 7 p.**

Synthèse

J.M.Helary, A.Mostefaoui, M.Raynal, ***Déterminer un état global dans un système réparti*****, RR. 2090, INRIA (1993), 21 p.**

3. DATATION CAUSALE ET HORLOGES VECTORIELLES

ENREGISTREMENT D'UNE TRACE

• Objectifs :

• Reconstruire la séquence des messages échangés pour faire une analyse post mortem ou pour préparer la réexécution d'une situation à analyser.
Il faut donc conserver la dépendance causale entre les événements dépendants et indiquer les événements causalement indépendants.

• On date chaque événement e du système avec une méthode de datation causale (l'horloge causale). Soit D(e) la date ainsi fournie.
Elle permettra de reconstituer la trace si et seulement si :

$$a \rightarrow b \Leftrightarrow D(a) < D(b)$$

$$a \parallel b \Leftrightarrow \text{non } (D(a) < D(b)) \text{ et non } (D(b) < D(a))$$

C'est la condition forte des horloges car elle inclut : $D(a) < D(b) \Rightarrow a \not\parallel b$

Rappel : condition pour que deux événements a et b soient concurrents, ou encore causalement indépendants :

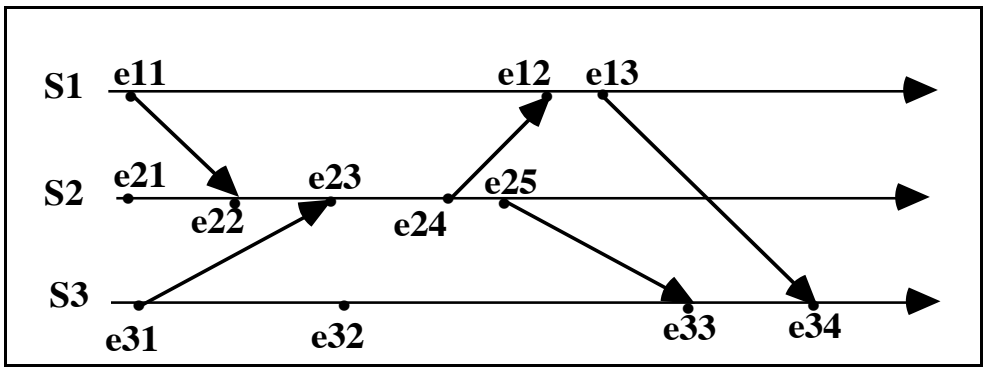
$$a \parallel b \Leftrightarrow \text{non } (a \rightarrow b) \text{ et non } (b \rightarrow a)$$

UNE METHODE DE DATATION CAUSALE
les historiques

Rappel : passé d'un événement e

• Le passé (ou historique) d'un événement e, c'est par définition :

hist(e) = e ∪ ensemble des événements e' tels que e' -> e



hist(e33) = {e11 e25 e24 e23 e22 e21 e31 e32 e33}

Idée : utiliser le passé de e pour la datation car le passé permet de représenter la dépendance causale :

a -> b <=> a ∈ hist(b)

a ∥ b <=> (a ∉ hist(b)) et (b ∉ hist(a))

Gros inconvénient : la taille de hist(e)

Remède : on observe que, pour définir hist(e), un événement par site suffit.

HORLOGES VECTORIELLES

(Fidge, Mattern 1988)

• Projection de hist(e) sur Si :

$$hist_i(e) = \{ a \in hist(e) \mid a \in Si \}$$

• Propriété : $e_{i,k} \in hist_i(e)$ => pour tout $j < k$: $e_{i,j} \in hist_i(e)$

Si on indice les événements de $hist_i(e)$ et si un événement avec un indice k appartient à $hist_i(e)$, alors tous les événements d'indice inférieur à k font aussi partie de $hist_i(e)$.

• Alors un seul entier suffit pour représenter $hist_i(e)$, c'est le nombre d'événements de $hist_i(e)$.

Comme $hist(e) = \bigcup hist_i(e)$, on représente tout hist(e) avec un vecteur V(e)

$$1 \leq i \leq n : V(e)[i] = k \text{ tel que } e_{i,k} \in hist_i(e) \text{ et } e_{i,k+1} \notin hist_i(e)$$

V(e)[i] = nombre d'événements de Si "connus de e"

(i. e. connus sur le site de e immédiatement après l'occurrence de e)
(nombre d'événements de l'historique de e qui sont localisés sur Si)

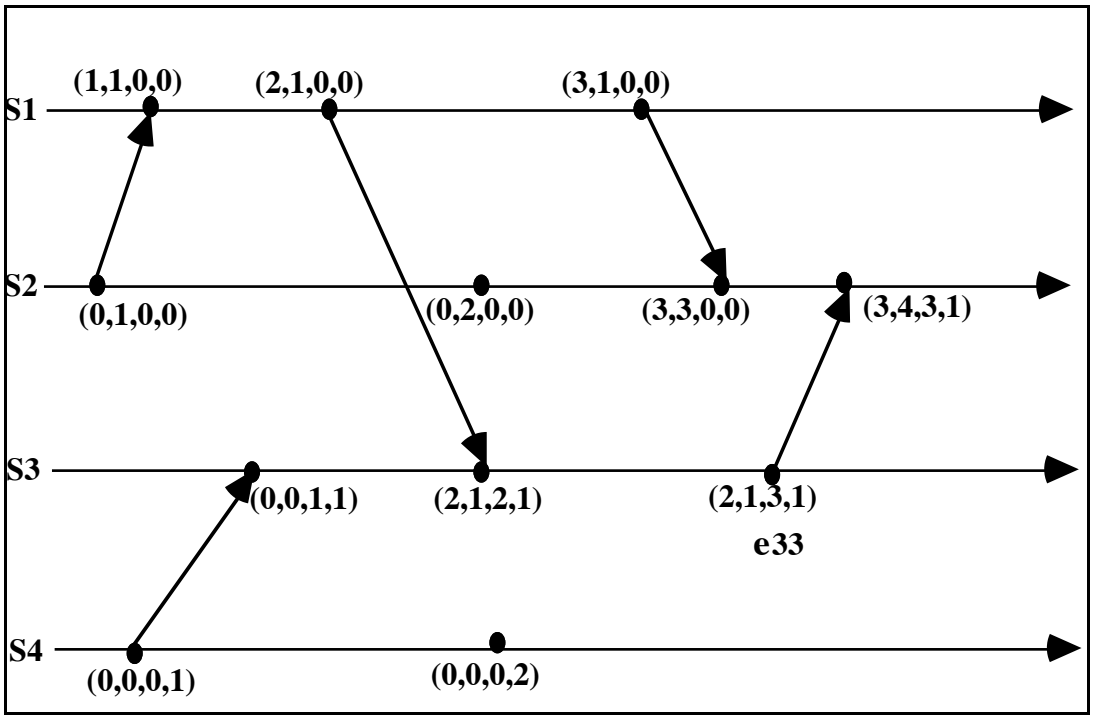
REALISATION DES HORLOGES VECTORIELLES

On associe une horloge vectorielle V_i à chaque site Si

- **Initialement Vi = (0,...,0)**
- **A chaque événement à dater local à Si, on fait $V_i[i] := V_i[i] + 1$**
- **Chaque message m porte une estampille Vm (Vm = Vi de l'émetteur)**
- **A la réception de (m, Vm) par un site Si, on enrichit l'historique connu par Si avec l'historique transporté par m :**

  $V_i[i] := V_i[i] + 1$

  $V_i[j] := \max(V_i[j], Vm[j])$ pour tous $j = 1,...n, j \neq i$



- **Autre façon de connaître l'"heure" de e33:**

  l'histoire de e comprend

  2 événements sur S1

  1 événement sur S2

  3 événements sur S3

  1 seul événement sur S4

PROPRIETES DES HORLOGES VECTORIELLES

• Relation d'ordre partiel sur les horloges vectorielles :

V ≤ V' défini par : quel que soit i, V[i] ≤ V'[i]

V < V' défini par V ≤ V' et V ≠ V'

V □□ V' défini par ¬ (V < V') et ¬ (V' < V)

Les horloges vectorielles représentent <u>exactement</u> la dépendance causale :

pour tout a, b

a -> b <=> V(a) < V(b)

a □□ b <=> V(a) □□ V(b)

• Les horloges vectorielles sont "denses" :

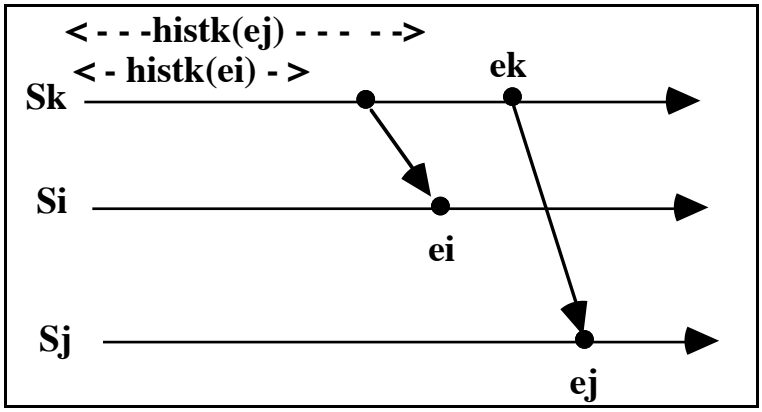
Soit ei □ Si, ej □ Sj,

si V(ei)[k] < V(ej)[k], pour k ≠ j, alors il existe ek sur Sk tel que

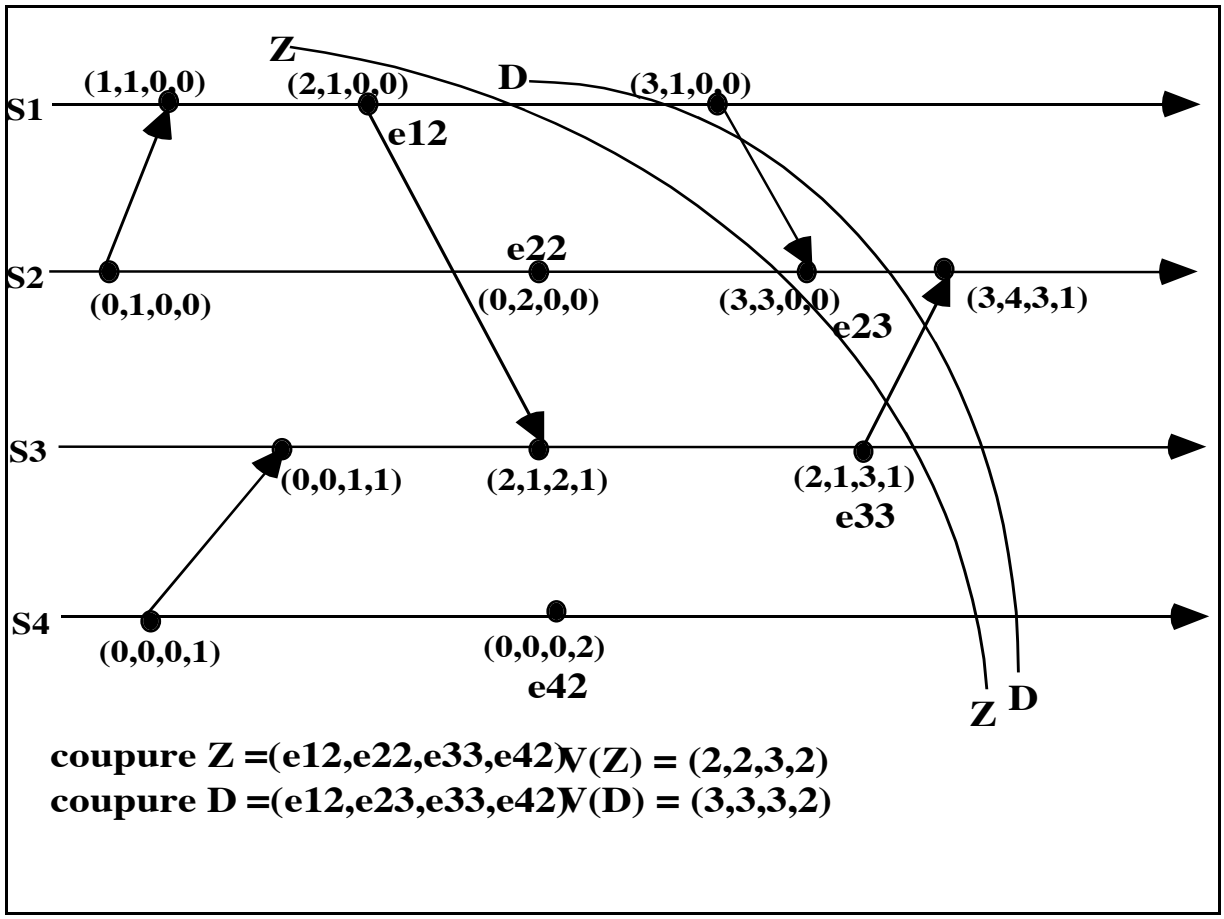
¬(ek -> ei) et (ek -> ej)

En effet V(ei)[k] = □$hist_k$(ei)□= nombre d'événements de hist(ei) sur Sk

V(ej)[k] = □$hist_k$(ej)□= nombre d'événements de hist(ei) sur Sk



HORLOGES VECTORIELLES ET COUPURES COHERENTES



• Date d'une coupure C = (c1,c2,...,cn) ⊡ union des histoires des événements de F

V(C) = sup (V(c1),...,V(cn))

soit pour tout i, V(C)[i] = sup (V(c1)[i],...,V(cn)[i])

La coupure est cohérente si et seulement si c'est l'union des histoires des projections

V(C) = (V(c1)[1],...,V(cn)[n])

V(Z) = (2, 2, 3, 2) = (V(e12)[1], V(e22)[2], V(e33)[3], V(42)[4])

V(D) = (3, 3, 3, 2)

(V(e12)[1], V(e23)[2], V(e33)[3], V(42)[4]) = (2, 3, 3, 2)

• Z est cohérente, D ne l'est pas

DIFFUSION AVEC ORDRE CAUSAL

(Birman, Schiper, Stephenson 1990)

• Utilisation des horloges vectorielles pour garantir la causalité

• Si un message arrive trop tôt pour la causalité, on le fait attendre

• On ne compte que les diffusions :

1) avant diffusion de m, le site Si exécute $V_i[i] := V_i[i] + 1$

2) estampille de m par Si : Vm = Vi

3) à la réception sur Sj de (m, Vm) diffusé par Si, on attend que :

a) toutes les diffusions précédentes de Si soient arrivées sur Sj

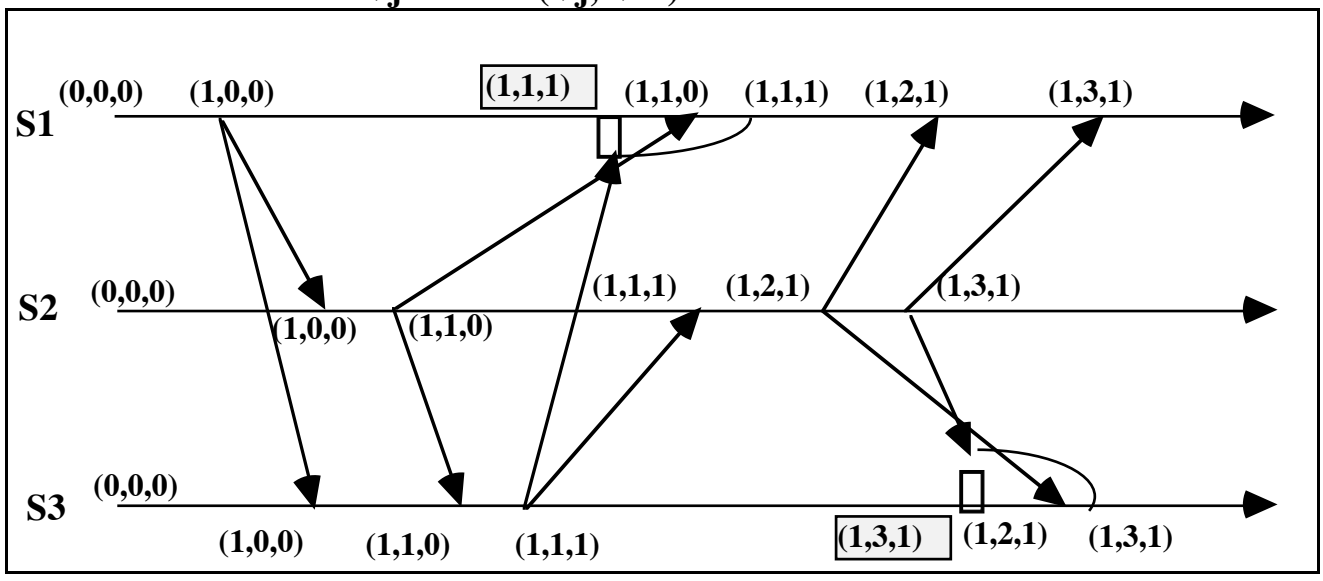
soit $Vj[i] = Vm[i] - 1$

b) toutes les diffusions antérieures à m et reçues sur Si aient été aussi reçues par Sj

soit pour tous $k \neq i$, $Vj[k] \geq Vm[k]$

4) après remise de m, on enregistre l'historique connu grâce à m

soit $Vj := \max(Vj, Vm)$



en □ : V1 = (1,0,0) et Vm = (1,1,1) soit V1[3] = Vm[3] - 1; <u>V1[2] < Vm[2]</u>

en □ : V3 = (1,1,1) et Vm = (1,3,1) soit <u>V3[2] ≠ Vm[2] - 1</u>; V3[1] ≥ Vm[1]

Bibliographie

[Mattern 88] F. Mattern, Virtual time and global states in distributed systems, *International Conference on Parallel and Distributed Algorithms*, North Holland (1988)

[Mattern 92] F. Mattern, On the relativistic structure of logical time in distributed systems, in *Datation et contrôle des exécutions réparties*, Bigre, n° 78, édité par l'IRISA, Rennes, mars 1992

4. DATATION TOTALE
LINEARISATION DE L'OBSERVATION D'UN SYSTEME

• Linéarisation de l'observation d'un système réparti S:

- **"vue" en séquence des événements de S par un observateur interne**
- **"vue" en séquence des événements de S par un observateur externe**

• C'est la définition d'un ordre total, soit <<, sur les événements de S

• Linéarisation valide si elle est compatible avec la relation de précédence causale

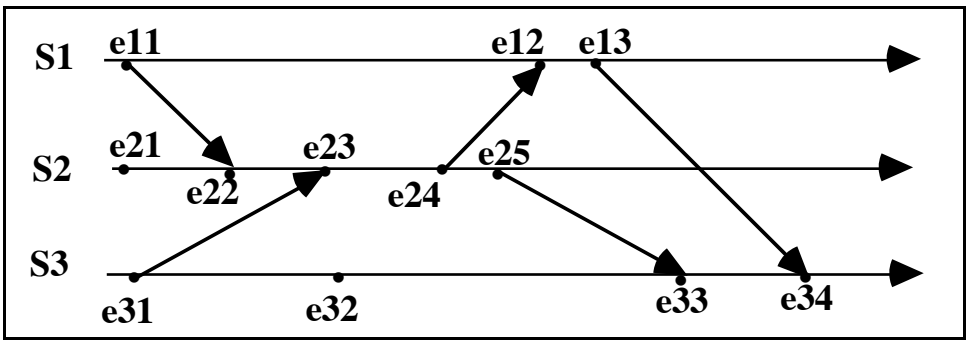
$$e, e' \in S : e \rightarrow e' \Rightarrow e << e'$$

• Exemple :

e11 e21 e31 e22 e23 e32 e24 e25 e12 e33 e13 e34 valide

e11 e21 e31 e22 e32 e23 e24 e25 e12 e33 e13 e34 valide (car e32 \|\| e23)

e11 e21 e31 e22 e23 e32 e24 e25 e12 e33 e34 e13 invalide (car e13 -> e34)



• On date chaque événement e du système avec une méthode de datation totale (l'horloge logique). Soit H(e) la date ainsi fournie qui est valide ssi:

$$e \rightarrow e' \Rightarrow H(e) < H(e')$$

Nota. $H(e) < H(e') \Rightarrow non(e' \rightarrow e')$

C'est la condition faible des horloges car, ou bien e -> e', ou bien e \|\| e'

REALISATION DES HORLOGES LOGIQUES

(Lamport 1978)

• Objectif : réaliser une datation totale des événements

• respectant la dépendance causale

• déterminable par consultation locale

Un compteur entier Hi, initialisé à 0, est maintenu sur chaque site Si.

• A chaque événement e, localisé sur Si :

• Hi := Hi + 1

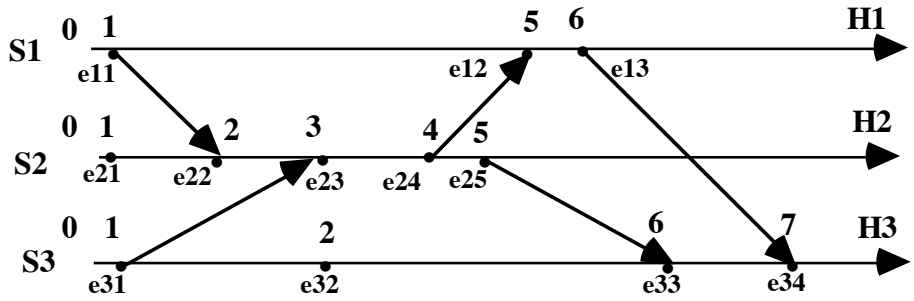
• e est daté par Hi : H(e) := Hi

• Si e est l'émission d'un message m, celui-ci est estampillé par la date de son émission sur Si, obtenue en lisant Hi :

• E(m) = H(émission(m)) := Hi

• Si e est la réception d'un message m venant de Sj, l'estampille E(m) est utilisée par le compteur Hi (l'horloge logique locale sur Si) pour rattraper le retard sur Hj, s'il en a, avant de dater d'incrémenter Hi pour dater e.

• Hi := max (Hi, E(m)) + 1



• Réalisation d'un ordre total (<<) : soit a sur Si, b sur Sj

donc H(a) = $H_i(a)$ et H(b) = $H_j(b)$

a << b <=> e' (H(a) < H(b) ou (H(a) = H(b) et i < j)

l'ordre total sur les sites est arbitraire (il doit être le même partout)

EVALUATION DES HORLOGES LOGIQUES

Hypothèses de validité

- **le réseau de communication est connexe et fiable**
- **le temps de transmission des messages est borné supérieurement**
- **pas de communication cachée qui donneraient un autre ordre total**
  (via des canaux comme le téléphone entre utilisateurs,
  ou comme une heure universelle fournie par satellite).

Avantages

- **première datation répartie introduite**
- **économique : datation par un seul nombre et non par un vecteur**
- **causalité des messages respectée par remise à l'heure du récepteur**

Utilisation importante des estampilles

- **ordre total : e11 e21 e31 e22 e32 e23 e24 e12 e25 e13 e33 e34**
- **exclusion mutuelle répartie**
- **mise à jour de copies multiples**
- **diffusion fiable ordonnée totalement**
- **datation des transactions réparties pour gérer les verrous des BD**
- **datation des transactions réparties pour la prévention d'interblocage**
- **gestion cohérente de fichiers répartis**
- **génération répartie de noms uniques pour la désignation interne**
- **génération répartie de noms uniques pour l'authentification**

Limitation de la datation par estampilles

- **la notion de dépendance causale est effacée artificiellement, car**
- **les estampilles ne sont pas denses : si H(e) < H(e'),**
  on ne peut pas savoir s'il existe e'' tel que e -> e'' ou e'' -> e'

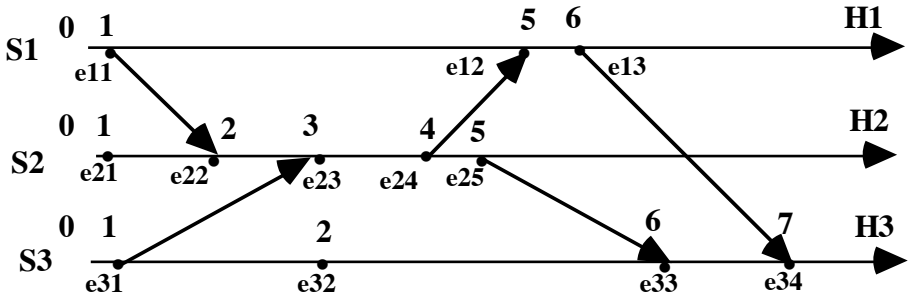
LIMITATION DE LA DATATION PAR HEURE LOGIQUE

e -> e' => H(e) < H(e')

• c'est la condition faible des horloges car

(R) **H(e) ≤ H(e') => non(e' → e)** **[(a => b) <=> (non b => non a)]**

c'est à dire : ou bien e -> e', ou bien e||e'



a) **• e11 → e33 se traduit en H1(e11) < H3(e33)** **e11 << e33**

• e||e' donne n'importe quoi :

e32||e12 donne H3(e32) < H1(e12) **e32 << e12**

e32||e22 donne H3(e32) = H2(e22) **e32 >> e22**

e32||e11 donne H3(e32) > H1(e11) **e32 >> e11**

b) **• H(e22) < H(e33) => non(e33 -> e22)** **ici** **e22-> e33**

• H(e12) < H(e33) => non(e33 -> e12) **ici** **e32|| e12**

c) **• seule certitude** **H(e) = H(e') => e||e'**

H(e) = H(e') <=> (H(e) ≤ H(e')) et (H(e) ≥ H(e'))

(H(e) ≤ H(e')) et (H(e) ≥ H(e')) => non(e' -> e) et non(e -> é) **[par (R)]**

→ e||e' **CQFD**

exemples **H(e13) = H(e33) = 6 . On a bien e13||e33**

H(e22) = H(e32) = 2 . On a bien e22||e32

• Les estampilles ne sont pas denses : soit e et e' tels que H(e) < H(e'),

on ne peut pas savoir s'il existe e'' tel que e -> e'' et/ou e'' -> e'

dans l'exemple : H(e22) = 2; H(e32) = 2; H(e33) = 6

H(e22) < H(e33) et il existe e24 tel que e22 -> e24 et e24 -> e33

H(e32) < H(e33) et il n'existe pas e'' tel que e -> e'' et/ou e'' -> e'

EXCLUSION MUTUELLE REPARTIE
(Lamport 1978)

Hypothèses

- **le nombre N des sites est connu**
- **les canaux sont FIFO**
- **ordre total réalisé par horloge logique**

Principe : connaissance mutuelle répartie acquise par chaque site"

Demande d'entrée d'un site Si en section critique :
diffusion de (req, H(req), i) à tous les sites, Si compris
entrée en section critique quand Si sait que:
a) tous les sites ont reçu sa demande ou qu'ils en ont émis une aussi
b) sa demande est la plus ancienne de toutes

Réception par Si d'une demande d'entrée en section critique de Sj :
réponse systématique par (acq, H(acq), Si)

Libération de la section critique par Si :
diffusion de (lib, H(lib), Si) à tous les sites, Si compris

Structure de données sur chaque site : tableau de N messages, 1 par site

- **initialement sur Si** **Mij := (lib, 0, Sj) pour tout j**
- **réception de M = (req, H(req), j) ou (lib, H(lib), Sj) =>**
  Mij := M
- **réception de M = (acq, H(acq), Sj) =>**
  si Mij ≠ (req, H(req), j) alors Mij := M

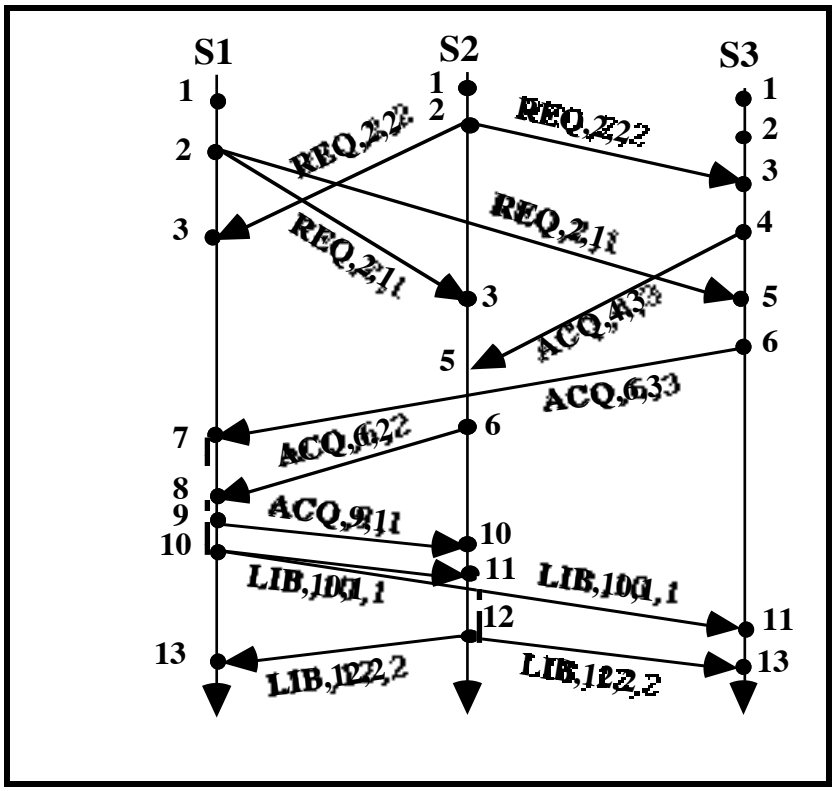
si l'acquittement vient après une requête de Sj, on n'oublie pas celle-ci

Règle de décision pour chaque site Si :
Si entre en section critique quand sa demande Mii << Mj pour tout j
En effet, comme les canaux sont FIFO,
il n'y a plus de requête plus ancienne en chemin dans un canal

Propriétés :

- **Si est seul en section critique.**
- **Il n'y a pas de coalition car les entrées suivent l'ordre total**
- **Mais il faut 3(n - 1) messages**

EXCLUSION MUTUELLE REPARTIE
(Lamport 1978)
EXEMPLE



	H1		H2		H3	
M11	LIB,0,1	0	M21 LIB,0,1	0	M31 LIB,0,1	0
M12	LIB,0,2	0	M22 LIB,0,2	0	M32 LIB,0,2	0
M13	LIB,0,3	0	M23 LIB,0,3	0	M33 LIB,0,3	0
M11	REQ,2,1	2	M21 REQ,2,1	3	M31 REQ,2,1	5
M12	REQ,2,2	3	M22 REQ,2,2	2	M32 REQ,2,2	3
M13	ACQ,6,3	7	M23 ACQ,4,3	5	M33 LIB,0,3	0
			M21 LIB,10,1	11	M31 LIB,10,1	11
			M22 REQ,2,2	2	M32 LIB,12,2	13
			M23 ACQ,4,3	5	M33 LIB,0,3	0

S1 entre en s.c. à H1 = 7

S2 entre en SC à H2 = 11

EXCLUSION MUTUELLE REPARTIE

(Ricart, Agrawala 1981)

Hypothèses

- **le nombre N des sites est connu**
- **les canaux sont quelconques (hypothèse FIFO non nécessaire)**
- **ordre total réalisé par horloge logique**

Principe : file d'attente répartie par morceau sur certains sites

Demande d'entrée d'un site Si en section critique :

diffusion de (req, H(req), i) à tous les sites, Si compris

entrée en section critique quand tous les sites ont donné leur accord

Réception par Si d'une demande d'entrée en section critique de Sj :

- **Si n'est ni en section critique ni candidat : accord (acc, H(acc), Si)**
- **Si est lui-même candidat :**

  accord (acc, H(acc), Si) si demande de Sj antérieure

  sinon mise en attente par Si de la demande de Sj
- **Si est en section critique :**

  mise en attente par Si de la demande de Sj

Libération de la section critique par Si :

accord (acc, H(acc), Si) à toutes les demandes mises en attente par Si

Propriétés :

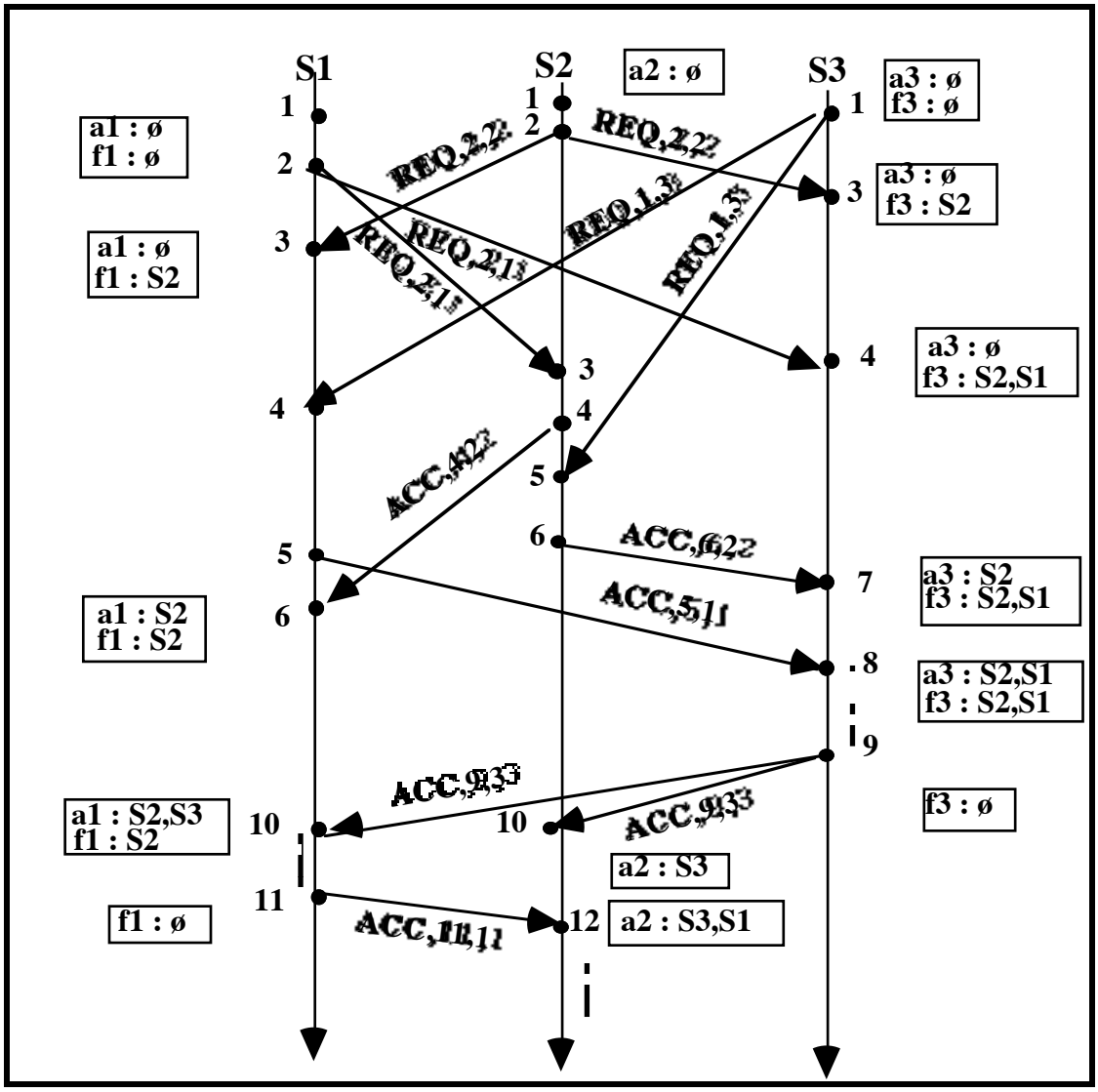
- **Si est seul en section critique.**
- **Il n'y a pas de coalition car les entrées en S.C. suivent l'ordre total**

  (équité faible car les messages peuvent se doubler)
- **Il faut 2(n - 1) messages**

EXCLUSION MUTUELLE REPARTIE

(Ricart, Agrawala 1981)

EXEMPLE



légende : **ai : ensemble des ACC reçus par le site i**
fi : ensemble des sites mis en attente par Si

5. POSE DE POINTS DE REPRISE RÉPARTIS

*** tolérance aux pannes et reprise arrière (ex. longs calculs scientifiques)**

*** validation en transactionnel (ex. cohérence et prévention d'interblocage)**

• DEUX OPÉRATIONS A CONSIDÉRER

sauvegarde des information nécessaires à la reprise (points et messages)

retour arrière à un état global cohérent et ré-exécution de l'application

• TECHNIQUES DE POSE DES POINTS DE REPRISE

sauvegarde (pose) périodique indépendamment sur chaque site

effet domino

sauvegarde cohérente déclenchée périodiquement par un site initiateur
(exemple : voir l'algorithme de Chandy, Lamport)

synchronisation explicite
coûteux en message
initiateur quelconque (tolérance aux pannes)

sauvegarde adaptative de Xu et Netzer (1993)

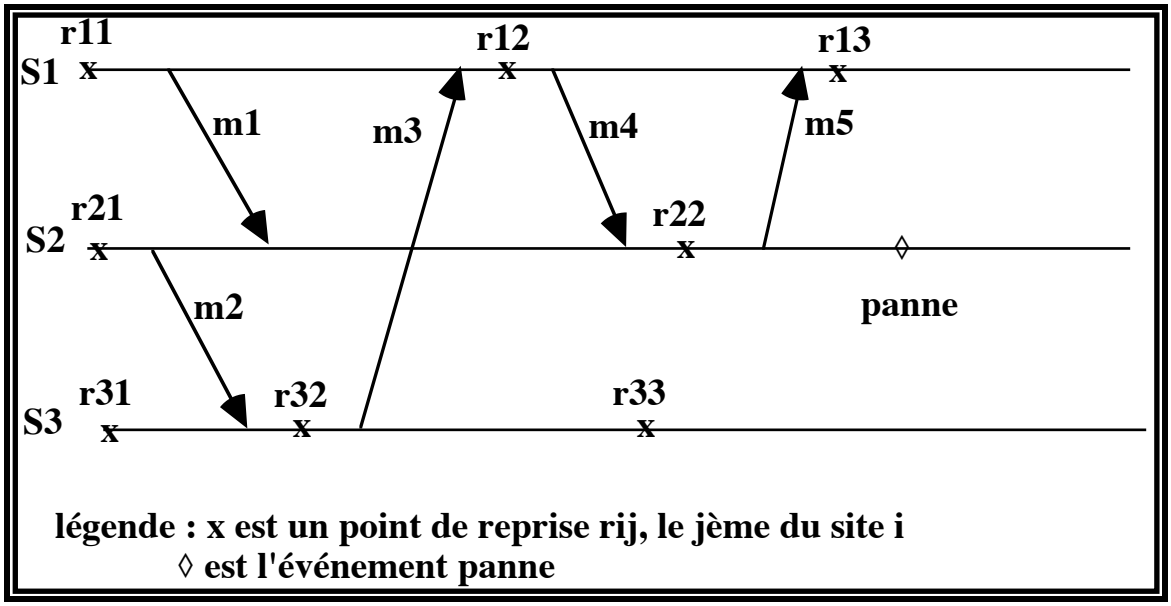
synchronisation implicite entre sites
points de contrôle forcés sur un site récepteur

Référence : [XU 93] J. XU, R. NETZER, Adaptative Independant Checkpointing for Reducing Rollback Propagation, 5th IEEE Symp. on Parallel and Distributed Processing, Dec. 93, Dallas TX, USA, 8 pages, 1993.

SAUVEGARDE INDÉPENDANTE SUR CHAQUE SITE

• effet domino : le retour arrière d'un site entraîne le retour des autres au delà du dernier point de reprise et cela parfois jusqu'au début du traitement

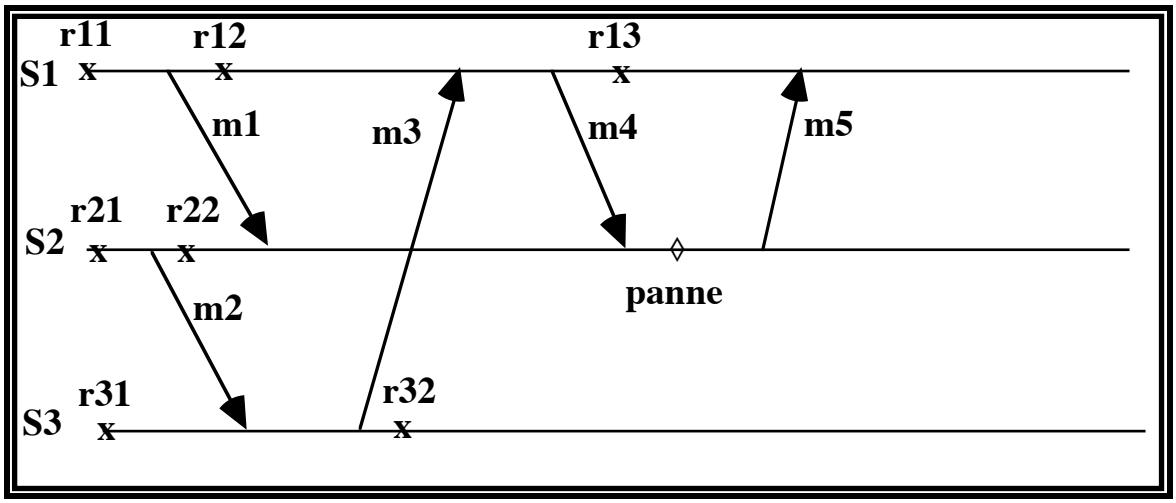
Premier exemple : points pris au hasard



Pour retrouver un point cohérent et repartir après réparation (ou avec un processeur en redondance passive sur S2), S2 remonte à r22, mais comme l'émission de m5 est perdue, il faut faire remonter S1 à r12 ; puis comme l'émission de m4 est perdue, il faut faire remonter S2 à r21 ; etc.

Aucune coupure r1a, r2b, r3c n'est cohérente sauf la coupure initiale r11, r21, r31.

Deuxième exemple : sauvegarde après émission de chaque message



On peut repartir avec r13, r22, r32 si on a noté l'émission de m1 et m4

MODÉLISATION

• On note -> la dépendance causale

• On note $r_{p,j}$ le jème point de reprise sur le site S_p;

Chaque Sp pose un point de reprise initial r_{p1} au début de l'exécution.

Une ligne de reprise est un ensemble de points de reprise, un par site.

Une ligne de reprise R est cohérente si elle forme un état global cohérent : tout message enregistré reçu sur un site a été enregistré émis sur un autre.

Enregistré <=> fait partie du passé, de la coupure

• L'intervalle de reprise $I_{p,i}$ est la suite d'événements entre $r_{p,i}$ et $r_{p,i+1}$

(il contient $r_{p,i}$ mais pas $r_{p,i+1}$)

CHEMINS EN ZIGZAG

• Il y a un chemin en zigzag de $r_{p,i}$ à $r_{q,j}$ si et seulement si il existe des messages m_1, m_2, ...,m_n (n ≥ 1) tels que

(1) m_1 est envoyé par le site S_p après $r_{p,i}$

(2) si m_k ($1 \le k < n$) est reçu par le site S_r, alors m_{k+1} est envoyé par S_r dans le même intervalle de reprise ou dans un intervalle postérieur
(noter que m_{k+1} peut être envoyé avant ou après l'arrivée de m_k),

(3) m_n est reçu par le site S_q avant $r_{q,j}$

CYCLE EN ZIGZAG

Le point de reprise r appartient à un cycle en zigzag si et seulement si il y a un chemin en zigzag de r à lui-même.

CHEMIN CAUSAL ENTRE POINTS DE REPRISE

si $r_{p,i}$ -> $r_{q,j}$, alors on a un chemin causal

chemin causal => chemin en zigzag

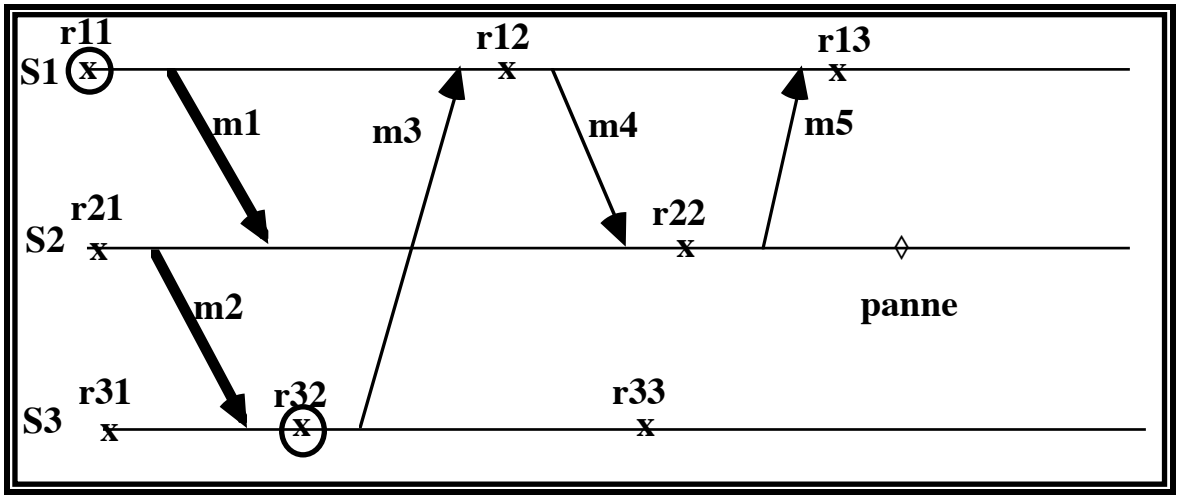
chemin en zigzag ≠> chemin causal

THÉORÈME

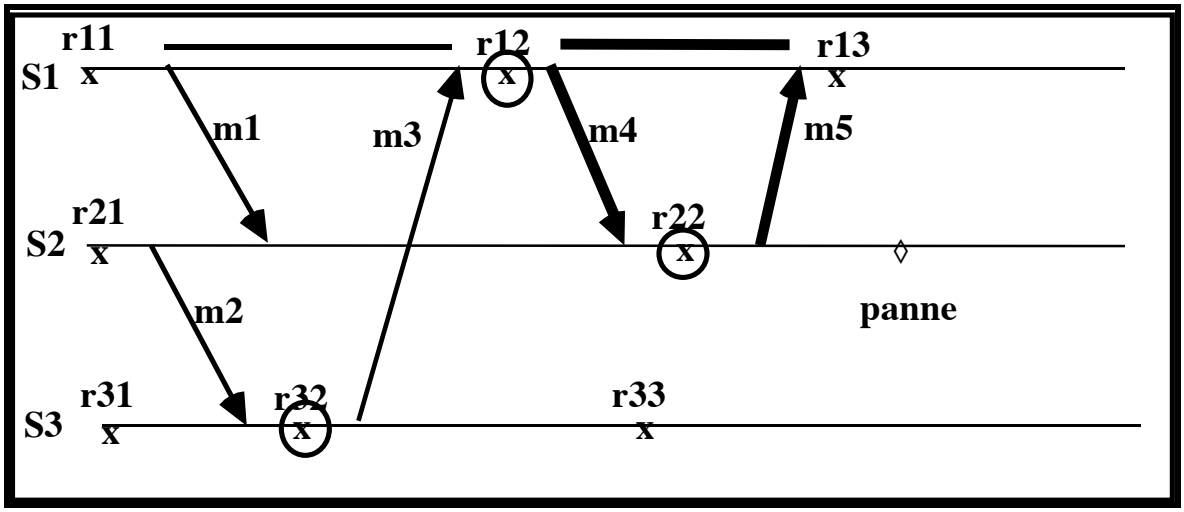
Un ensemble R de points de reprise sur différents sites peut être étendu faire partie d'une ligne de reprise cohérente si et seulement si aucun point de reprise de R n'appartient à un chemin en zigzag vers un autre point de reprise de R (ou vers lui-même, formant un cycle en zigzag).

COROLLAIRE : S'il n'y a pas de chemin en zigzag (ni cycle) dans R on peut toujours construire une ligne de reprise cohérente incluant R

EXEMPLES DE CHEMINS ET CYCLES EN ZIGZAG



chemin en zigzag de r11 à r32, chemin non causal
chemin causal de r32 à r22, chemin causal de r21 à r12



cycle en zigzag autour de r22 (messages m5 et m4)
cycle en zigzag autour de r32 (messages m3, m1 et m2)
cycle en zigzag autour de r12 (messages m4, m2 et m3)

PRINCIPE DE LA SAUVEGARDE ADAPTATIVE

POINTS DE REPRISE UTILISABLE

• Un point de reprise $r_{p,i}$ est utilisable s'il appartient à une ligne de reprise cohérente, c'est à dire s'il appartient à un état global cohérent.

COROLLAIRE

Un point de reprise $r_{p,i}$ est utilisable s'il n'y a pas de cycle en zigzag qui le contienne.

PRINCIPE DE LA SAUVEGARDE ADAPTATIVE

Empêcher les cycles en zigzag de se former et pour cela :

• détecter si un message reçu sur un site achève de construire un cycle,

• si c'est le cas, forcer un point de reprise avant la réception du message.

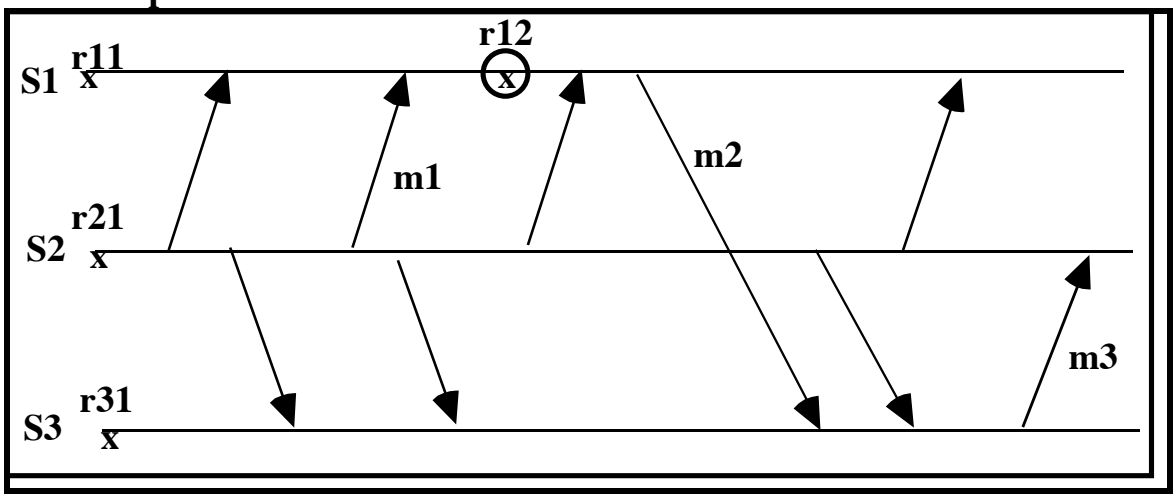
MÉTHODE APPROCHÉE

En fait on ne peut pas détecter si le cycle est en zigzag. Alors en approximation on détecte si le chemin est causal.

(à l'expérience, il y a très peu de chemins en zigzag non causaux)

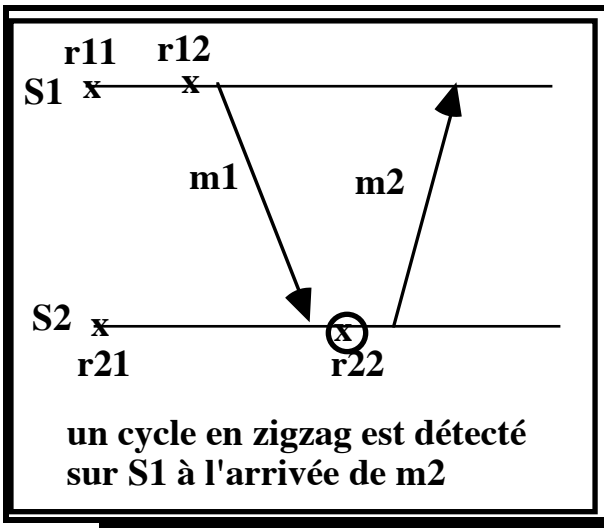
DÉTECTION DES CYCLES EN ZIGZAG

• difficulté car parfois la connaissance du futur lointain est nécessaire

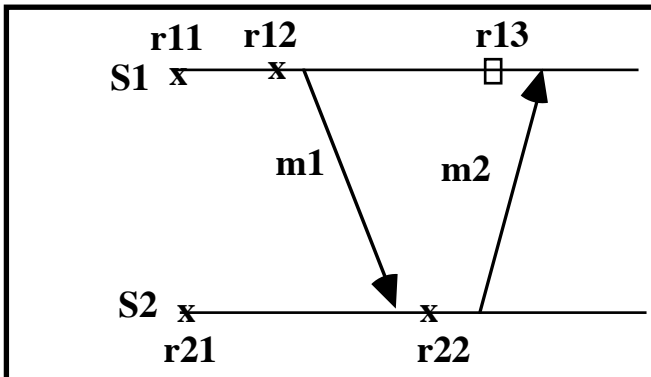


Le cycle en zigzag autour de r12 apparaît bien après r12 quand m3 arrive

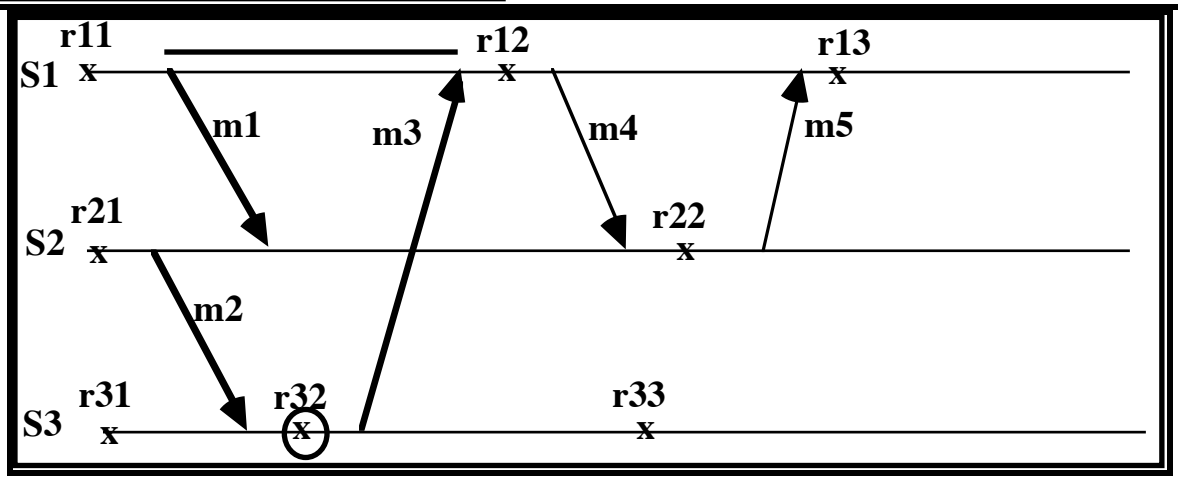
APPROXIMATION : DÉTECTER LES CHEMINS CAUSAUX



un cycle en zigzag est détecté sur S1 à l'arrivée de m2



un point de reprise r13 est forcé sur S1 ce qui rend utilisable r13 sur S2



r32 : cycle en zigzag non détecté (il n'est pas causal)

ALGORITHME DE SAUVEGARDE ADAPTATIVE

• Détecter les dépendances causales :

• On utilise les horloges vectorielles pour dater les points de reprise

On associe une horloge vectorielle V_i à chaque site Si

• Initialement Vi = (0,0,...,0,1,0,...,0) nul partout sauf la ième composante

• A chaque point de reprise nouveau local à Si, on fait $V_i[i] := V_i[i] + 1$

• Chaque message m porte une estampille Vm (Vm = Vi de l'émetteur)

• A la réception de (m, Vm) par un site Si, on enrichit l'historique connu par Si avec l'historique transporté par m :

$$V_i[j] := \max(V_i[j], Vm[j]) \text{ pour tous } j = 1,...n, j \neq i$$

• $V_i[j]$ indique le passé de Si situé sur Sj et tel qu'il est connu par Si

indique donc aussi le numéro du dernier point de reprise sur Sj qui est en dépendance causale avec le point de reprise courant sur Si.

• A la création d'un nouveau point de reprise $r_{i,k}$ sur Si, le site Si sauvegarde dans RVi les dépendances causales de $r_{i,k}$. Soit $RVi = Vi(r_{i,k})$.

• A l'envoi d'un message m de Si vers Sj, Si surcharge le message m avec Zid = RVi[j], c'est à dire avec le numéro du dernier point de reprise sur Sj qui est sur un chemin causal avec $r_{i,k}$

• A la réception d'un message (m, Vm, Zid) par Si et avant de traiter le message, l'algorithme regarde s'il existe un cycle causal entre le point de reprise courant sur Si, daté $RV_i[i]$ et le point de reprise sur Sj précédant l'émission. Il y a un cycle causal point de reprise sur Sj précédant l'émission si $RV_i[i] = Zid$.

Puis Vi est mis à jour.

ALGORITHME :

1 : si $Zid = RV_i[i]$ alors forcer un nouveau point de reprise;

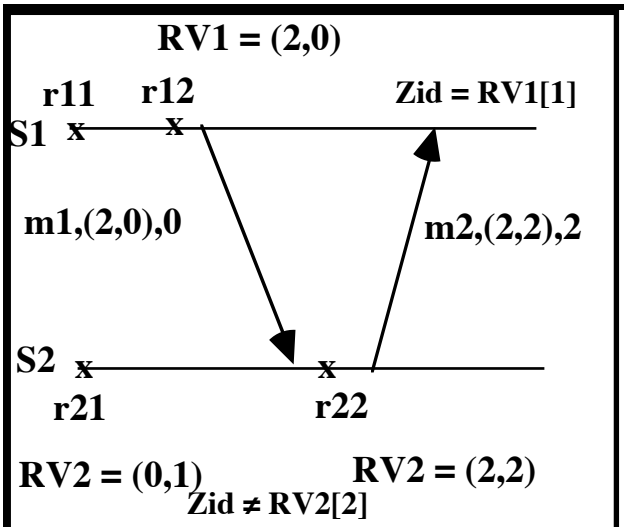
$V_i[i] := V_i[i] + 1$;

$RV_i = V_i$

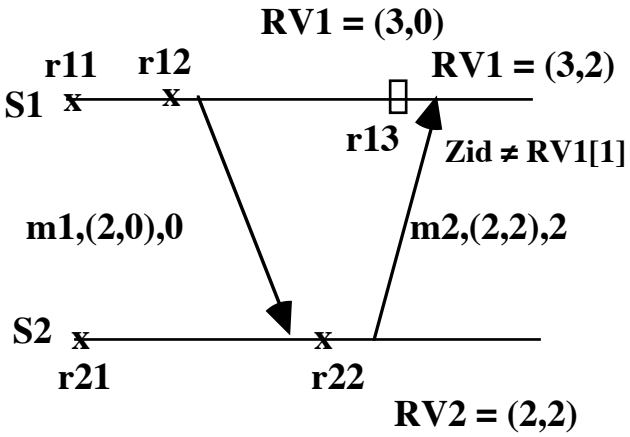
finsi;

2 : $V_i[j] := \max(V_i[j], Vm[j])$ pour tous $j = 1,...n, j \neq i$; -- historique

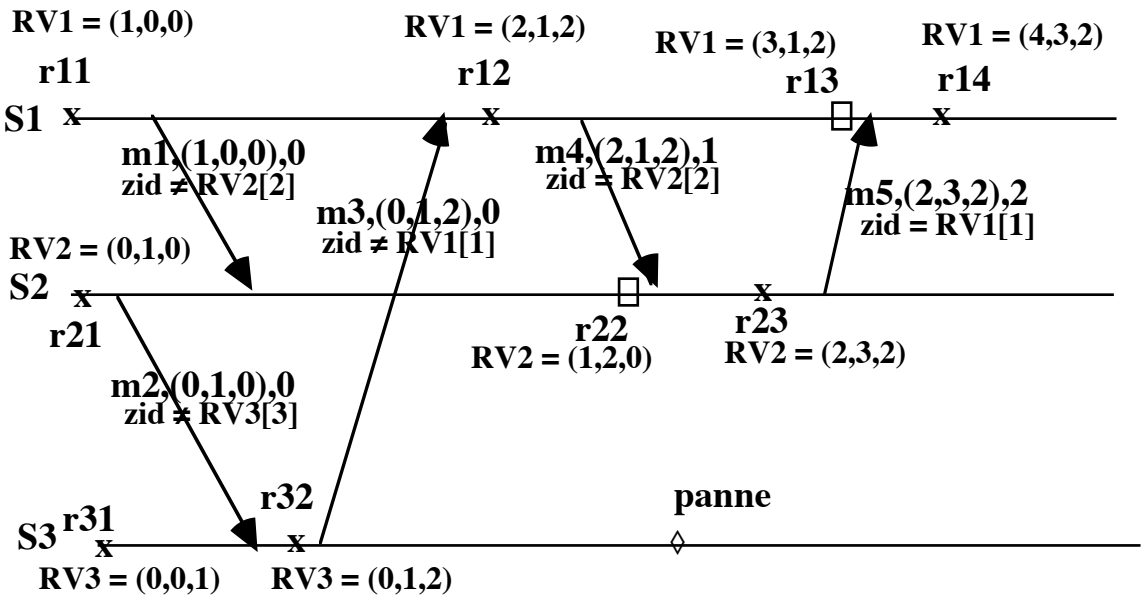
EXEMPLES DE SAUVEGARDE ADAPTATIVE



un cycle en zigzag pour r22 est détecté sur S1 à l'arrivée de m2



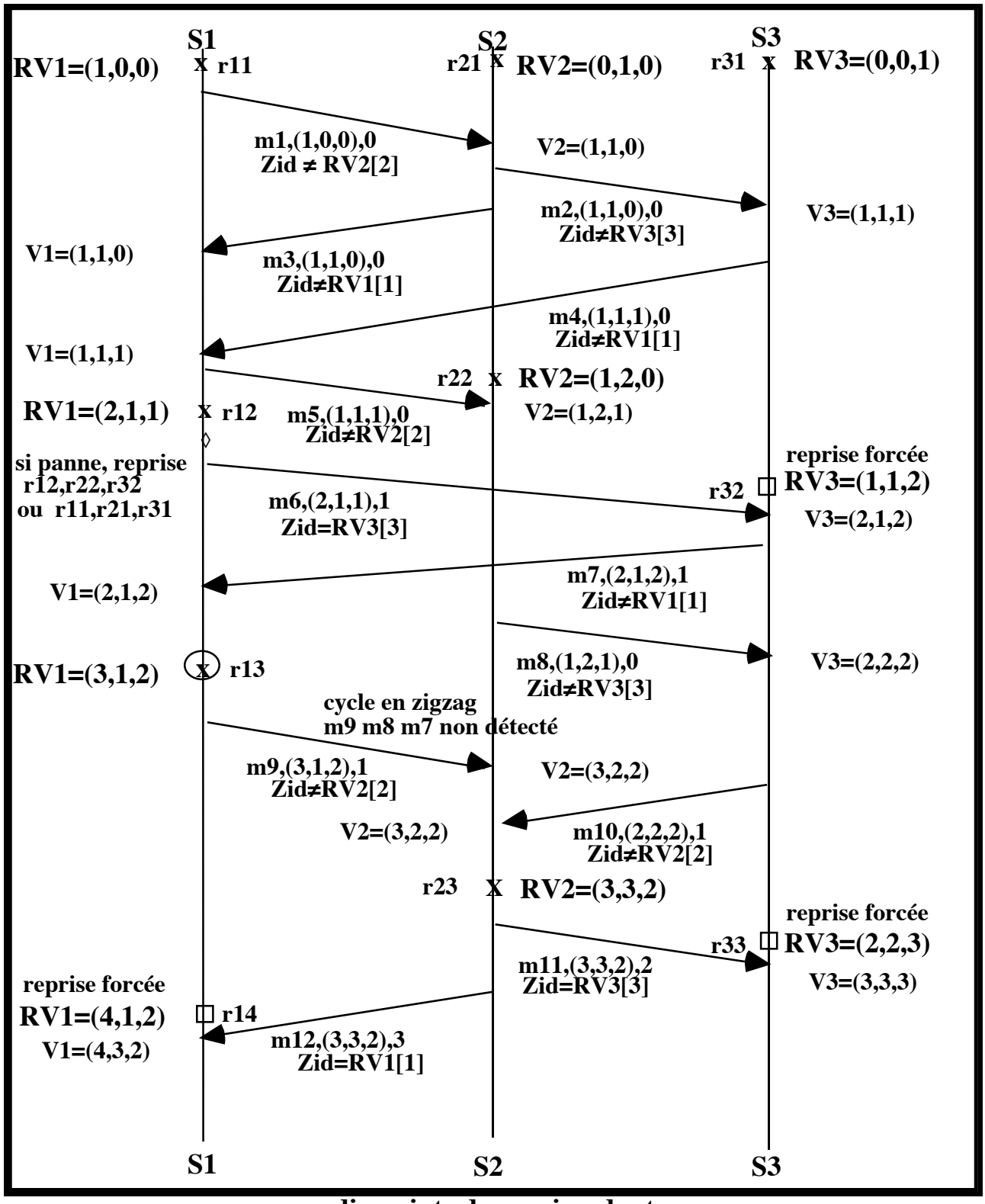
un point de reprise r13 est forcé sur S1 ce qui rend utilisable r22 sur S2



le cycle en zigzag autour de r32 n'est pas détecté, il n'est pas causal, la reprise arrière remontera à r31

un cycle autour de r12 est détecté sur S2, à l'arrivée de m4,1 on force la prise de r22

EXEMPLE DE SAUVEGARDE ADAPTATIVE



dix points de reprise, dont

trois reprises sont forcées, sept reprises spontanées

le cycle en zigzag autour de r13 n'a pas été détecté

CONCLUSION SUR LA SAUVEGARDE ADAPTATIVE

RÉSULTATS EXPÉRIMENTAUX :

• Conditions d'expérimentation

6 programmes de tests (de 13 000 à 370 000 messages échangés au total)

Hypercube Intel iPSC/860 avec 16 noeuds utilisés pour les tests

• Réduction de la propagation arrière pour atteindre une ligne cohérente

Avec une sauvegarde périodique déclenchée indépendamment sur chaque site, il faut en moyenne faire un retour arrière de 3 à 4 points de reprise par site pour obtenir une ligne de reprise cohérente (c'est l'effet domino).

En ajoutant la sauvegarde adaptative, on réduit ce retour arrière à un peu moins d'un point de reprise par site (l'effet domino est limité).

• Surcoût pour la sauvegarde adaptative

• faible en gestion de l'horloge vectorielle et de tests sur les messages

• nombre de points de reprise additionnels : pose 4% de points de reprise de plus que la sauvegarde périodique indépendante

BILAN

• Condition nécessaire et suffisante pour qu'un ensemble de points de reprise soit cohérent : absence de chemins en zigzag entre ces points

• Méthode heuristique : on recherche les chemins causaux (leur absence est une condition nécessaire) au lieu de rechercher les chemins en zigzag

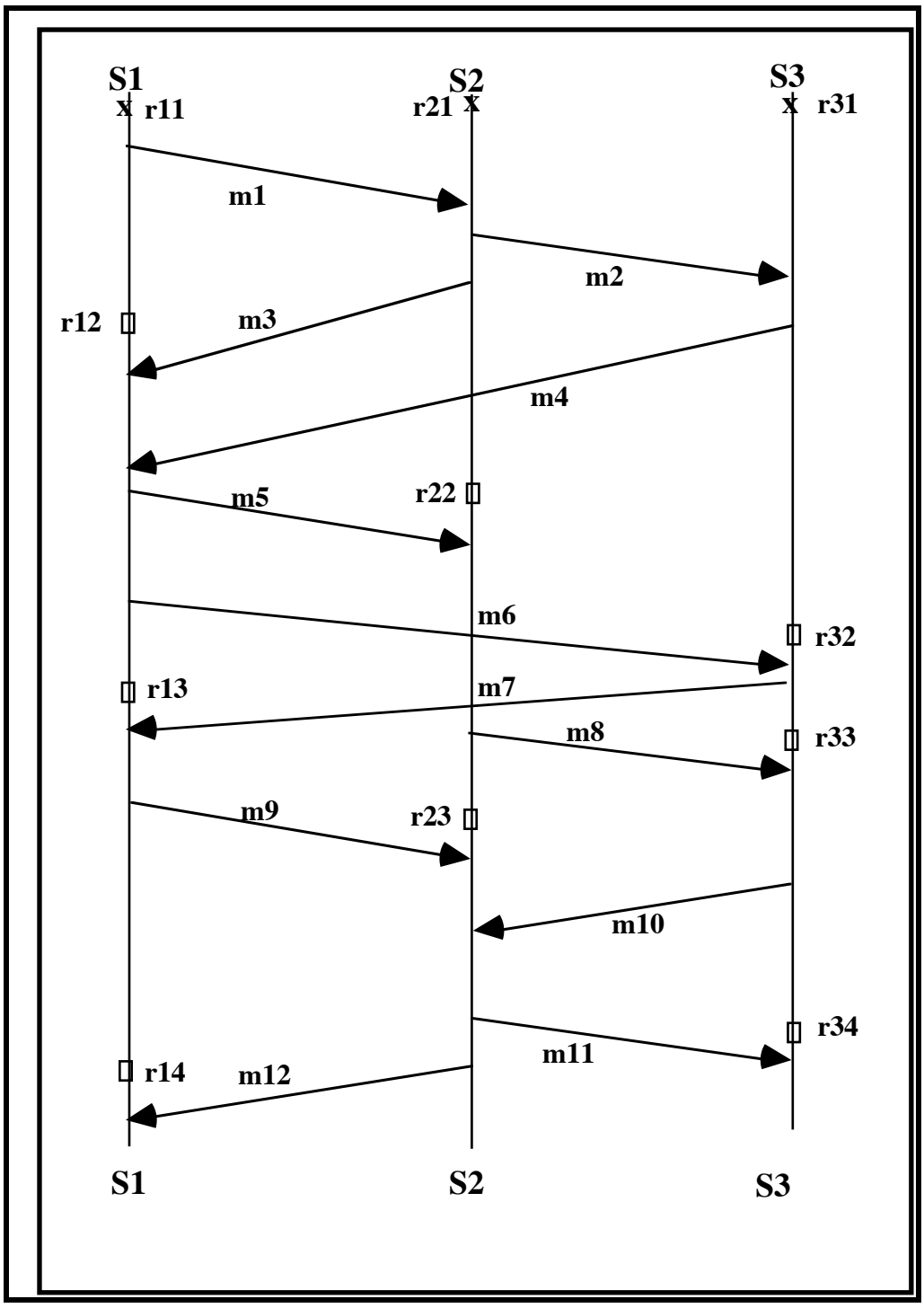
• Conclusion : gain obtenu par la réduction de l'effet domino

• ne pas oublier qu'il faut en plus faire le retour arrière, avoir sauvegardé les messages qui franchissent la coupure et réexécuter l'application.

RETOUR SUR L'EXEMPLE AVEC UNE AUTRE MÉTHODE

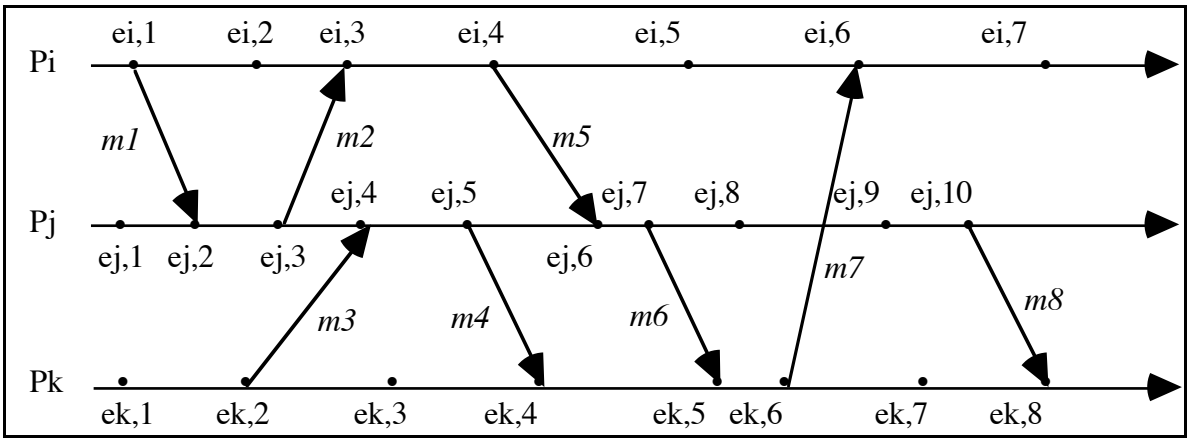
((RECEPTION)*(ÉMISSON)*SAUVEGARDE)

(heuristique de Russell 1980)



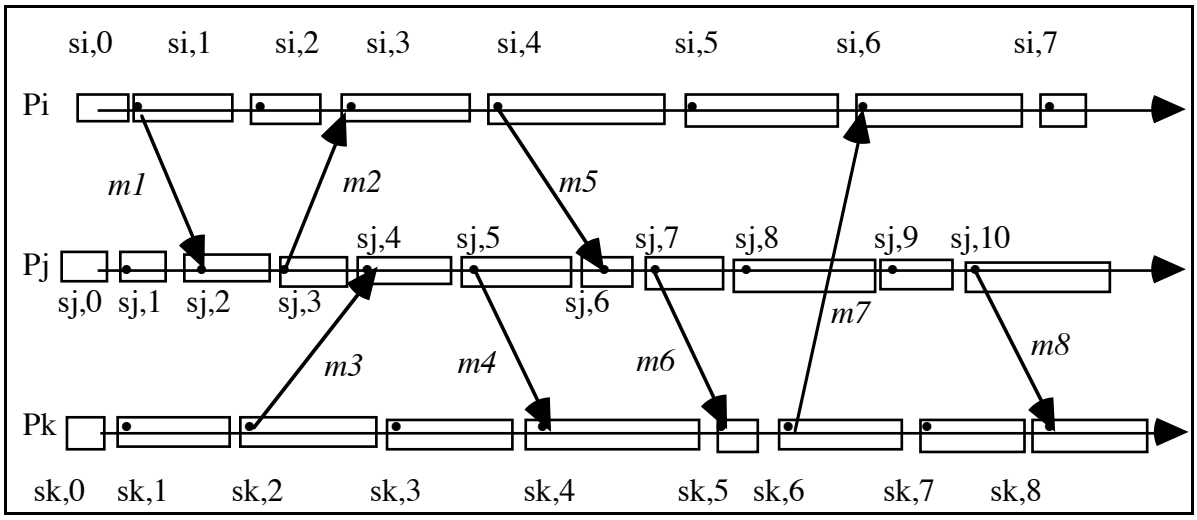
toutes les onze sauvegardes sont forcées

AUTRES MODÉLISATIONS D'UNE ÉXÉCUTION RÉPARTIE

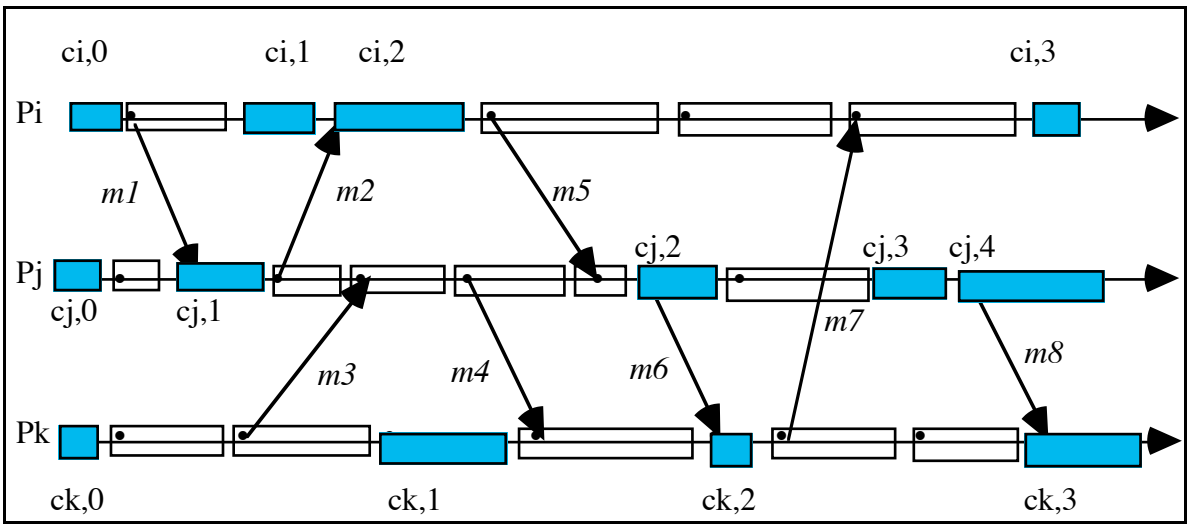


1. Exemple d'exécution répartie modélisée par des événements

événement = exécution d'instruction



2. Même exécution répartie modélisée par des états locaux



3. Même exécution répartie modélisée par des points de contrôle

6. ÉTUDE DE CAS

Objet réparti et répliqué sur 4 sites S1, S2, S3, S4

(avec même valeur initiale 100 Keuros)
méthodes : ajouter, retrancher, multiplier, lire

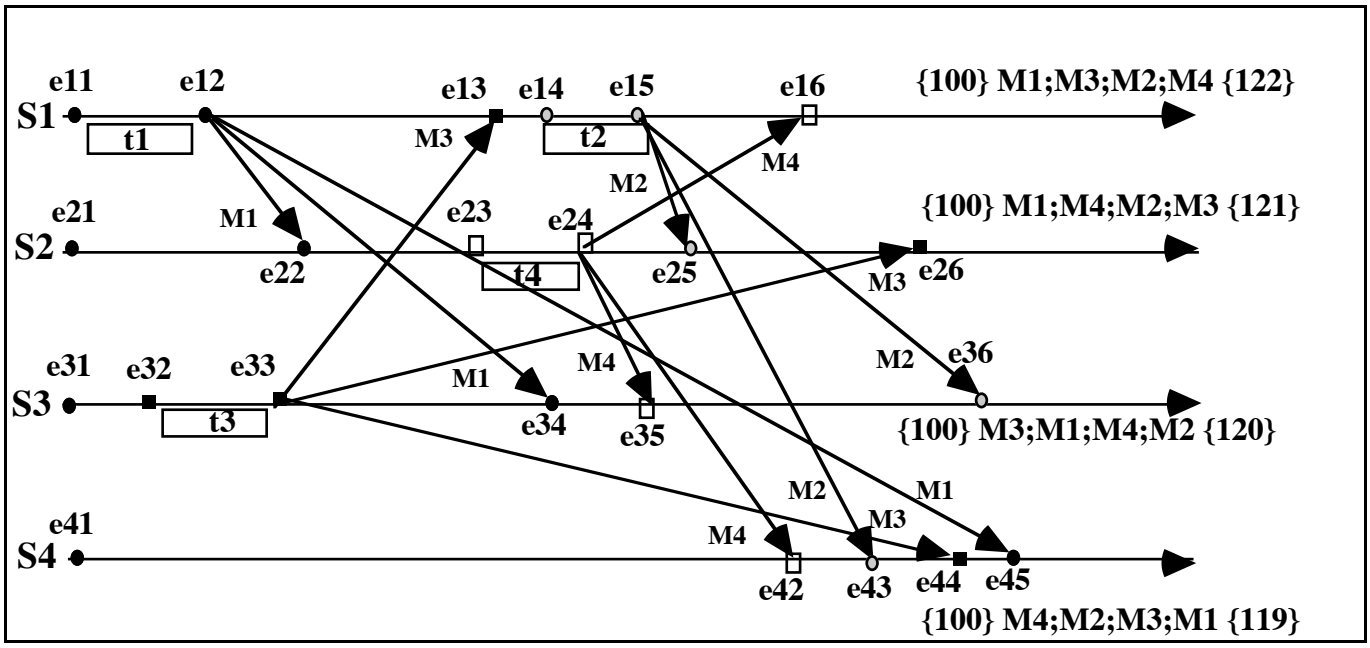
Application coopérative asynchrone

S1 : traitement t1 avec la copie locale; envoi M1; délai variable ; traitement t2 avec la copie locale; envoi M2;
S2 : traitement t4 avec la copie locale; envoi M4;
S3 : traitement t3 avec la copie locale; envoi M3;

t1 : ajouter 20
t2 : retrancher 10
t3 : multiplier la valeur par 110%
t4 : lire et sauvegarder (ou imprimer ou visualiser) la valeur locale

M1, M2, M3, M4 : messages de mise à jour diffusés à toutes les copies.
Mi : exécuter ti sur votre copie locale

DÉSORDRE NATUREL DES COMMUNICATIONS



Les sites visualisent 122, 120, 130, 100. Belle coopération!

Les répliques de l'objet valent 122, 121, 120, 119. Belle cohérence!

ORDRE CAUSAL

∀Sk,Mi émis par Si sur C_{ik}, Mj émis par Sj sur C_{jk},
émission$_i$(Mi) -> émission$_j$(Mj) => délivrance$_k$(Mi) -> délivrance$_k$(Mj)

ORDRE LOCAL
(c'est aussi un ordre causal)

∀Sk, Mi émis par Si sur C_{ik}, Mj émis par Si sur C_{ik},
émission$_i$(Mi) -> émission$_i$(Mj) => délivrance$_k$(Mi) -> délivrance$_k$(Mj)

ORDRE TOTAL

Mi diffusé sur Si et émis sur C_{ik},
∀ j diffusé sur Sj et émis sur C_{jk},
=>
∀Sk, délivrance$_k$(Mi) -> délivrance$_k$(Mj)
ou exclusif
∀Sk, délivrance$_k$(Mj) -> délivrance$_k$(Mi)

ORDRE TOTAL CAUSAL

∀) la précédence causale est respectée, si elle existe
∀Sk,Mi émis par Si sur C_{ik}, Mj émis par Sj sur C_{jk},
émission$_i$(Mi) -> émission$_j$(Mj) => délivrance$_k$(Mi) -> délivrance$_k$(Mj)

∀) si elle n'existe pas, on a un ordre total simple
Mi diffusé sur Si et émis sur C_{ik},
∀ j diffusé sur Sj et émis sur C_{jk},
émission$_i$(Mi) ∥ émission$_j$(Mj)
=>
∀Sk, délivrance$_k$(Mi) -> délivrance$_k$(Mj)
ou exclusif
∀Sk, délivrance$_k$(Mj) -> délivrance$_k$(Mi)

RELATIONS ENTRE ÉVÉNEMENTS DE DIFFUSION

On a 6 relations entre les événements de diffusion des 4 messages

ordre local d'émission :

émission$_1$(M1) -> émission$_1$(M2)

ordre causal d'émission

émission$_1$(M1) -> émission$_2$(M4)
émission$_3$(M3) -> émission$_1$(M2)
émission$_1$(M1) ∥ émission$_3$(M3)
émission$_2$(M4) ∥ émission$_3$(M3)
émission$_1$(M2) ∥ émission$_2$(M4)

QU'EN EST-IL AVEC CE DÉSORDRE NATUREL DES RÉCEPTIONS?

{100}M1;M3;M2;M4{122}; S1 respecte l'ordre causal et l'ordre local

{100}M1;M4;M2;M3{121}; S2 respecte l'ordre local, et pas l'ordre causal

{100}M3;M1;M4;M2{120}; S3 respecte l'ordre causal et l'ordre local

{100}M4;M2;M3;M1{119}; S4 ne respecte ni l'ordre causal ni l'ordre local

Il n'y a pas d'ordre total

D'autres variantes de réception ne sont pas meilleures :

{100}M3;M2;M1{120} respecte l'ordre causal et pas l'ordre local

{100}M2;M1;M3{121} ne respecte ni l'ordre causal ni l'ordre local

QUESTIONS :

Comment assurer le respect de l'ordre causal et de l'ordre local?

Comment obtenir un ordre total d'émission des messages?

Comment obtenir un ordre total d'émission et de délivrance des messages?

(synchronie de vue pour tous les sites)

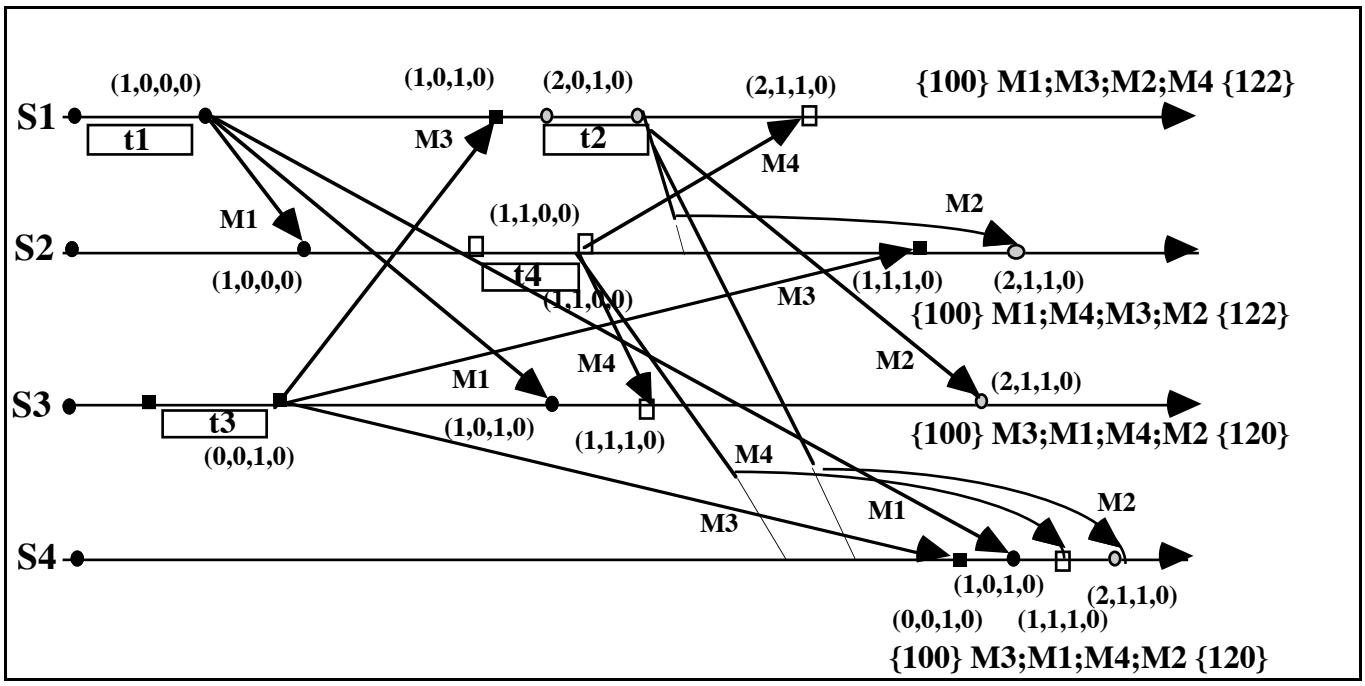
Comment obtenir une visualisation cohérente sans ordre total?

Comment prendre un ensemble cohérent de points de reprise

Comment installer la tolérance à une panne franche d'un processeur

RESPECT DE L'ORDRE CAUSAL

Comment assurer le respect de l'ordre causal et de l'ordre local?
utilisation des horloges vectorielles



Les sites visualisent 120, 122 ou 130.

Les répliques de l'objet valent 122 ou 120.

Problème : Comment obtenir un ordre total d'émission des messages?

Réponse : un site séquenceur, ou un jeton circulant ou exclusion mutuelle

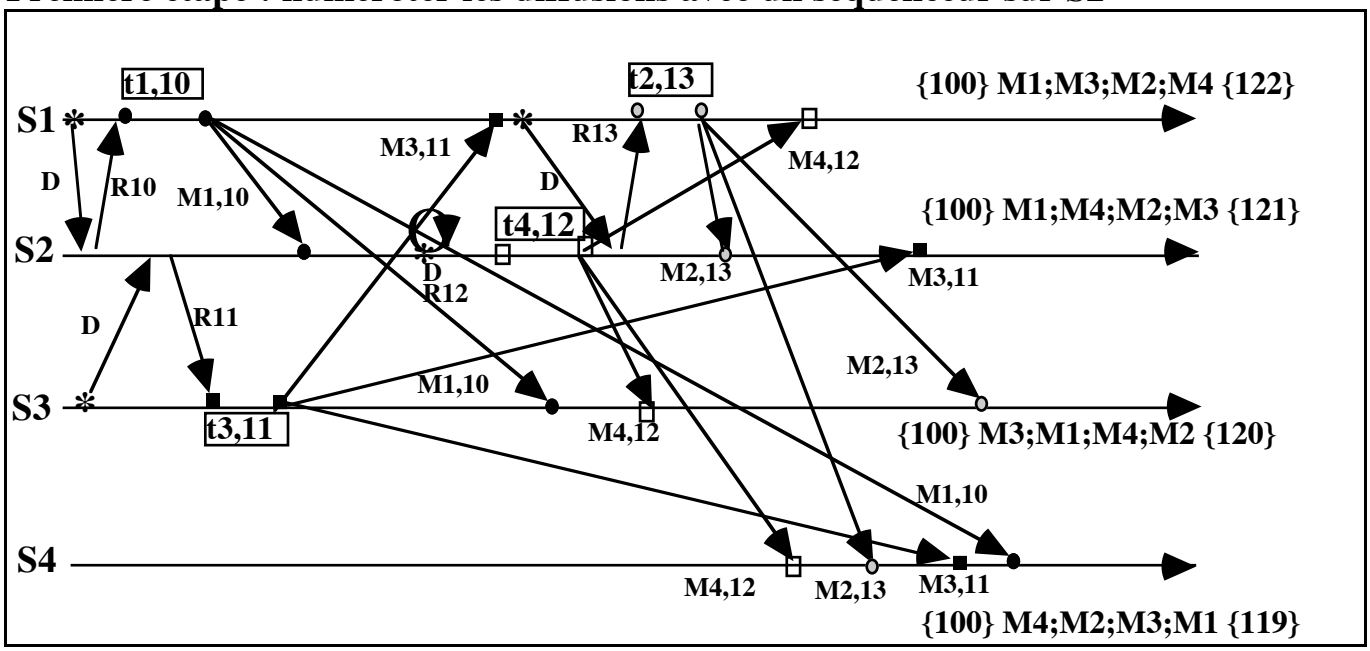
Comment obtenir un ordre total d'émission et de délivrance des messages?

numéroter les diffusions et diffuser le numéro dans le message

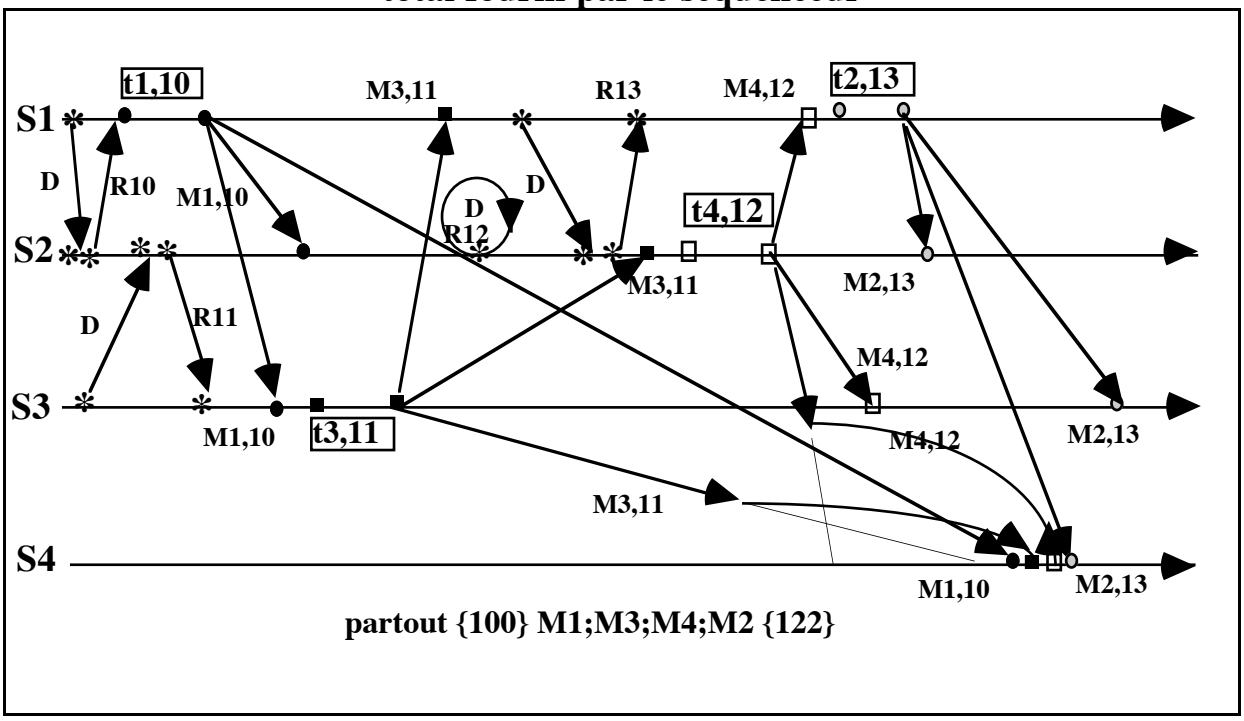
autre solution avec objet primaire et objets secondaires

ORDRE TOTAL DE DIFFUSION DES MESSAGES AVEC UN SITE SÉQUENCEUR

un site séquenceur fournit le numéro de diffusion
l'ordre total dépend de l'ordre de réception de la demande de numéro
Première étape : numéroter les diffusions avec un séquenceur sur S2



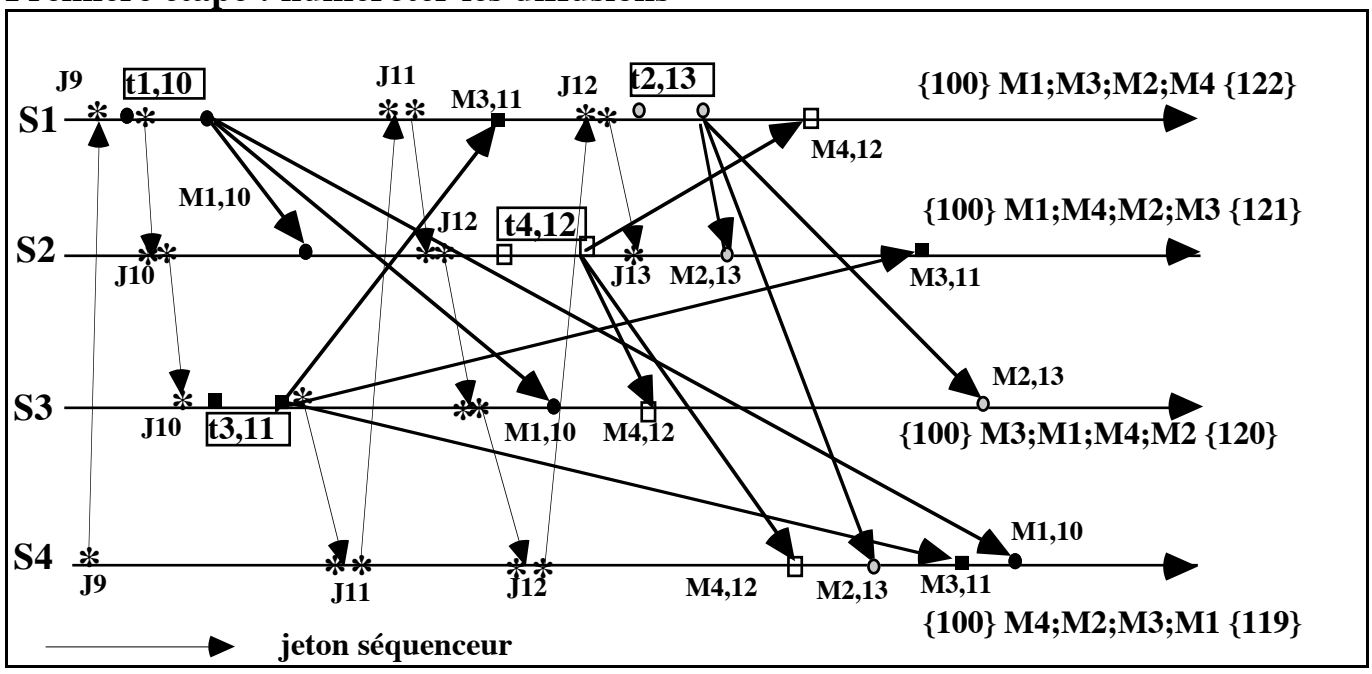
Cela ne suffit pas. Il faut réordonner les réceptions et les traitements selon l'ordre total fourni par le séquenceur



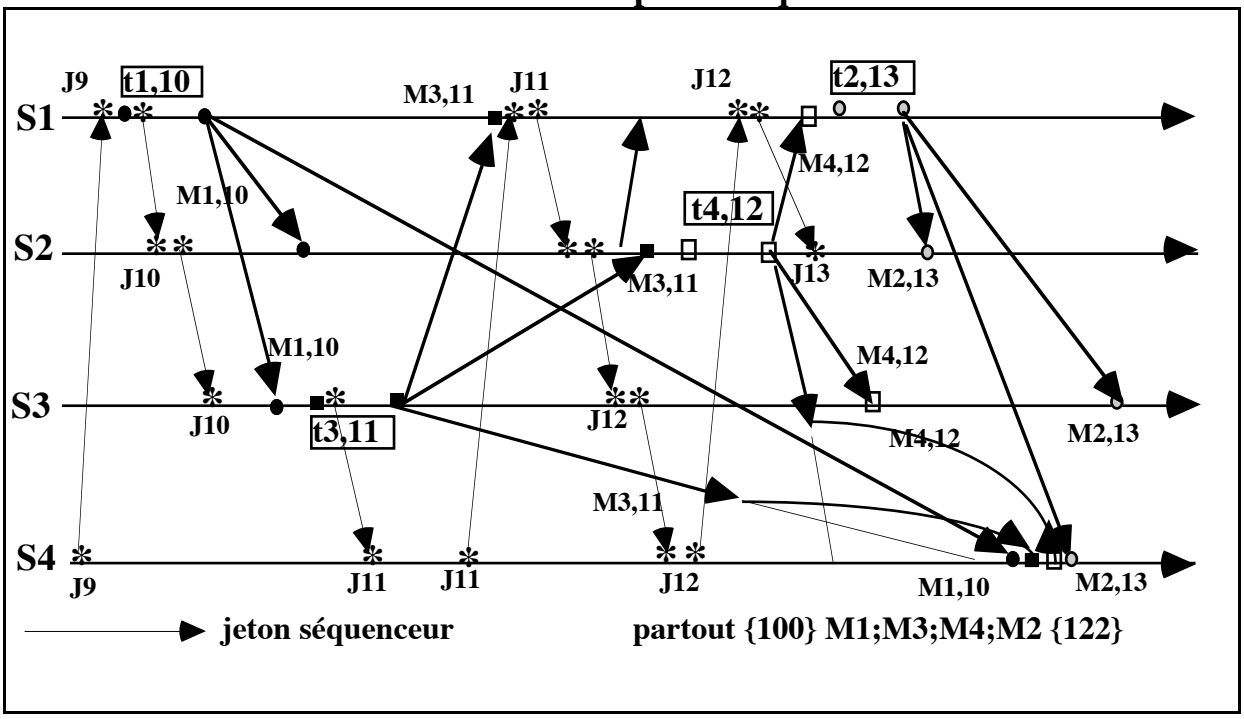
variante : le séquenceur reçoit le message à diffuser et fait la diffusion
(voir plus loin la solution avec objet primaire et objets secondaires)

ORDRE TOTAL DE DIFFUSION DES MESSAGES AVEC UN ANNEAU VIRTUEL ET UN JETON

le jeton contient un séquenceur qui fournit le numéro de diffusion
l'ordre total dépend de l'ordre de réception du jeton
Première étape : numéroter les diffusions



Cela ne suffit pas (même si les canaux sont FIFO). Il faut réordonner les réceptions et les traitements selon l'ordre total fourni par le séquenceur

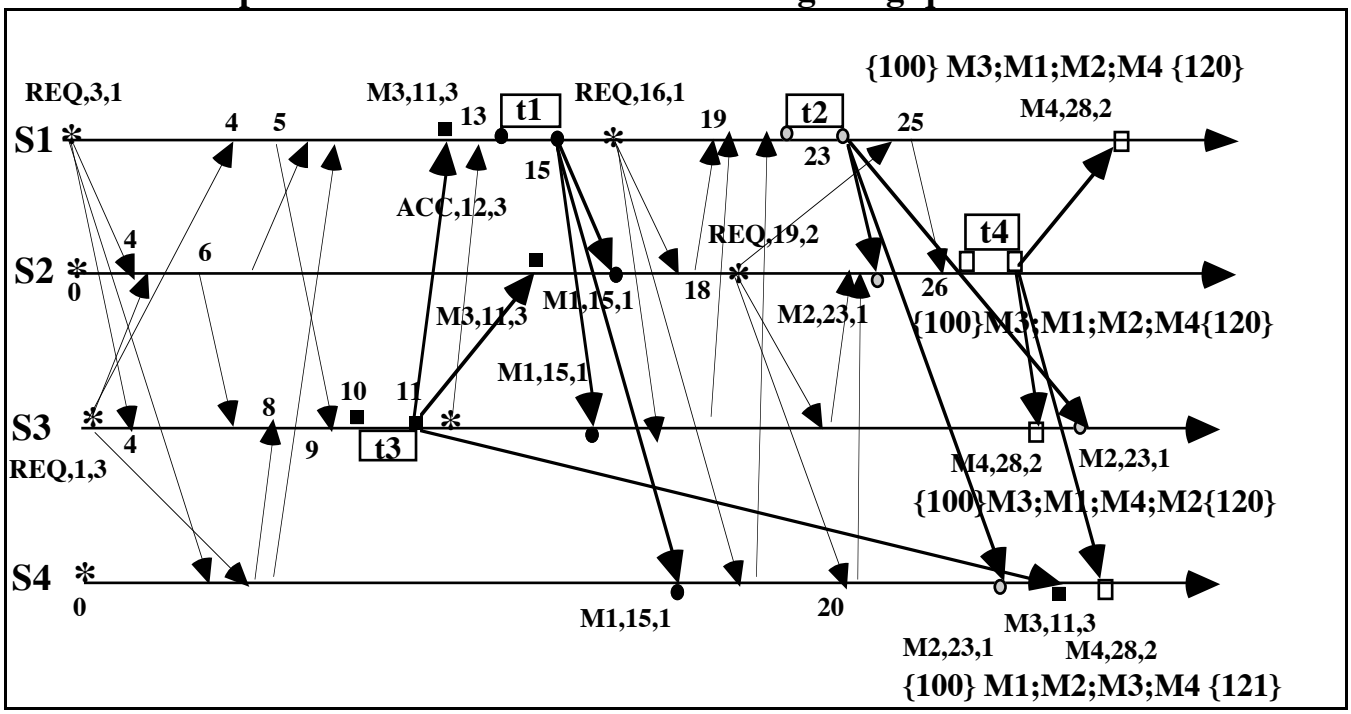


L'ordre total est causal. Peut-on avoir un ordre total non causal si les sites prennent des numéros en avance au passage du jeton ?

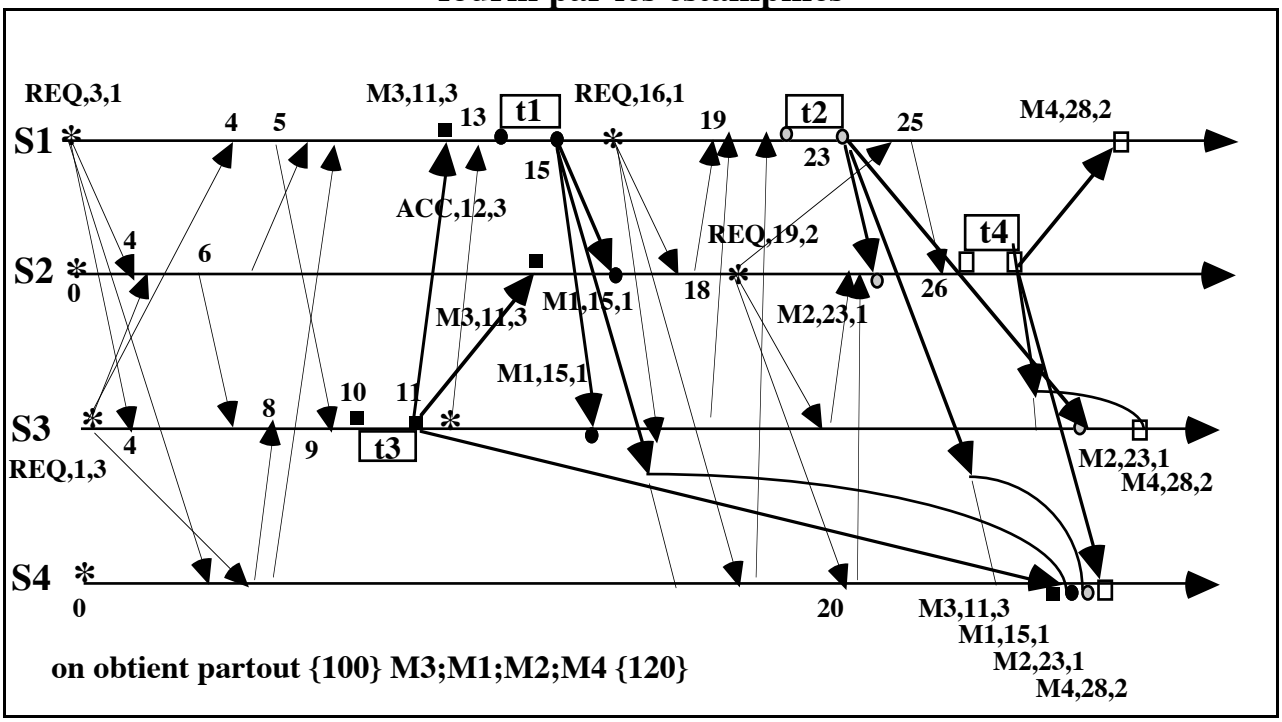
ORDRE TOTAL DE DIFFUSION DES MESSAGES
EXCLUSION MUTUELLE RÉPARTIE

on utilise Ricart-Agrawala

l'ordre total dépend des valeurs initiales des horloges logiques

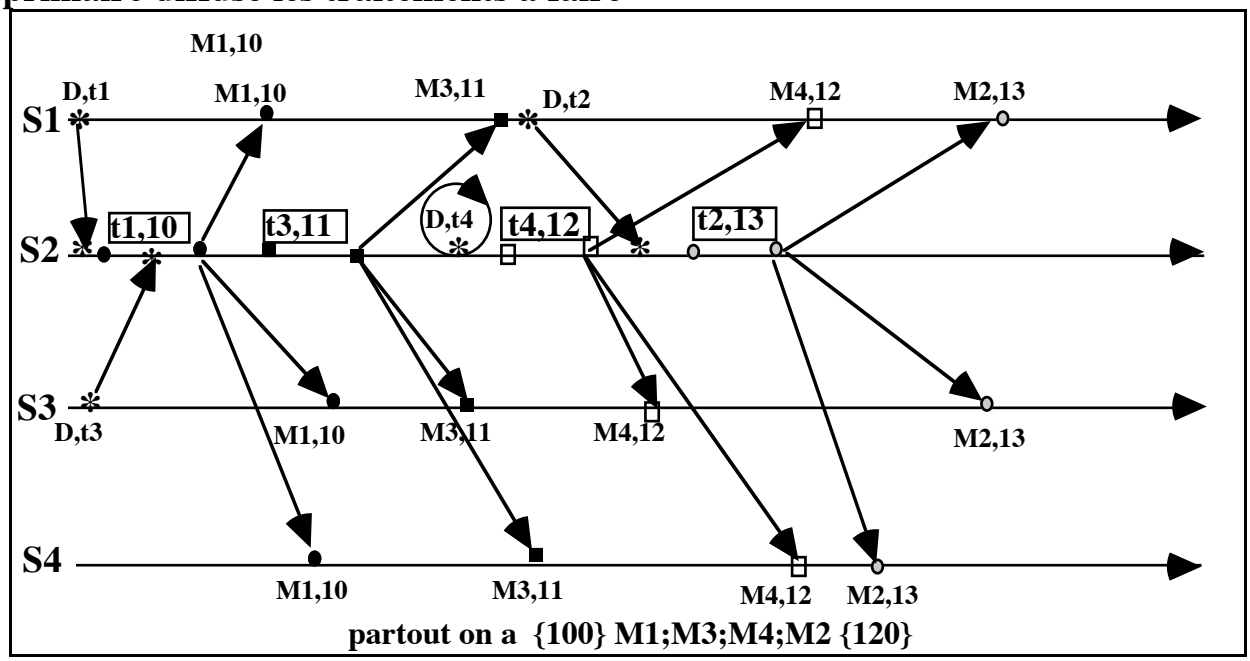


Cela ne suffit pas. Il faut réordonner les réceptions et les délivrer selon l'ordre total fourni par les estampilles



ORDRE TOTAL DE DIFFUSION DES MESSAGES
UN OBJET PRIMAIRE ET DES OBJETS SECONDAIRES

un site serveur primaire (est aussi séquenceur)
l'ordre total dépend de l'ordre de réception de la demande de service
Le primaire diffuse les traitements à faire



Si les canaux sont FIFO, on a l'ordre total
sinon, il faut numéroter les messages, comme ici
Permet des lectures concurrentes en cohérence faible des objets dupliqués
Problème en cas de panne du serveur primaire

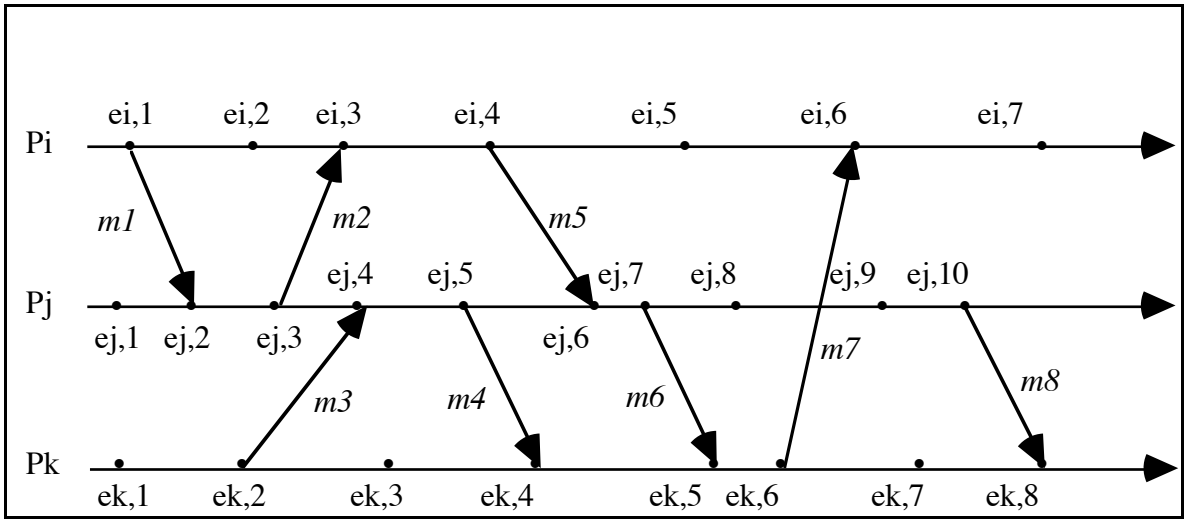
Plus généralement :
Comment tolérer une panne franche de processeur
Comment prendre un ensemble cohérent de points de reprise

autres études de cas
1. étude de cas avec deux objets répartis A et B et des traitements coopératifs répartis. Problèmes de cohérence transactionnelle.
P1 : A - 20 si A reste positif;
P2 : B + 20;
P3 : A + 10;
P4 : consulter A et B
2. traitements transactionnels répartis avec 3 objets répartis
P1 : A - 20; B+20;
P2 : B-10; A+10
P3 : C := A+B
visualiser A, B, C

7. EXERCICES

RELATION DE PRÉCEDENCE ENTRE DES ÉVÉNEMENTS RÉPARTIS

Exercice 1



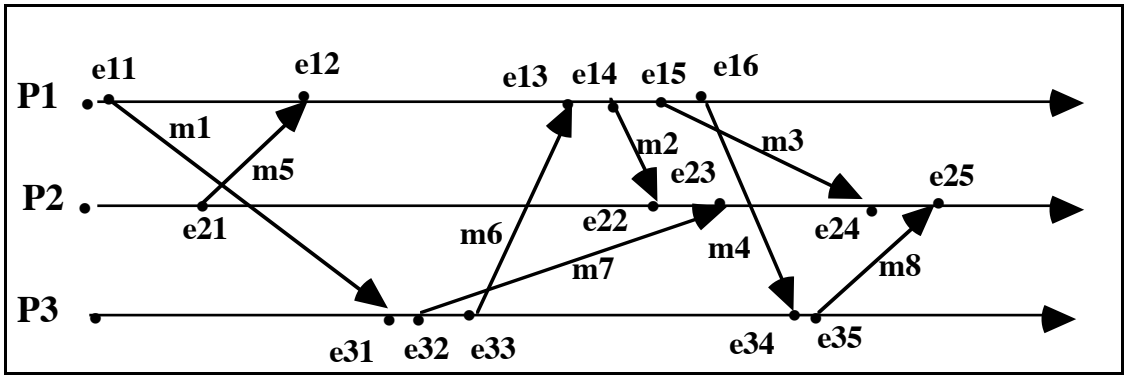
Comparer :

ei,2 et ek,4
ek,1 et ej,9
ek,2 et ei,7
ek,3 et ei,5
ek,3 et ej,10

DÉPENDANCE CAUSALE ENTRE MESSAGES

exercice 2

quels sont les messages qui ne respectent pas la dépendance causale?



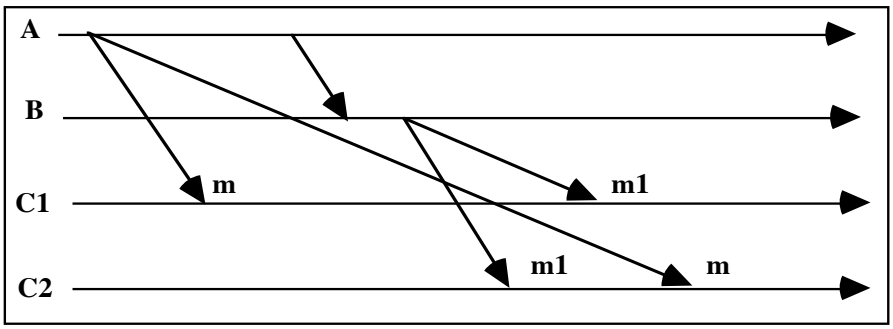
DÉPENDANCE CAUSALE ENTRE MESSAGES

Exemple 1:

A diffuse un courrier électronique m à C1 et C2 qui contient: "je demande à B de vous diffuser du travail par courrier électronique ".

Pour respecter l'ordre causal, C1 et C2 ne doivent pas recevoir le courrier m1 de B avant celui m de A.

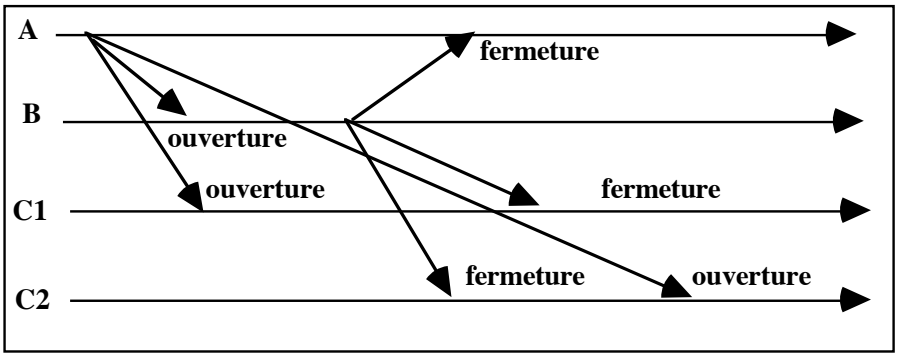
émission$_A$(m) → émission$_B$(m1) → réception$_C$(m) → réception$_C$(m1)



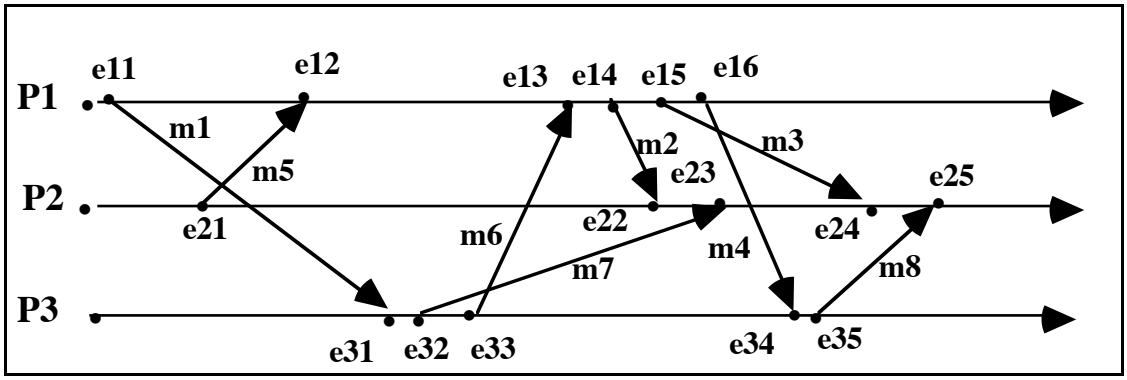
Exemple 2:

A envoie à C1 et C2 une commande d'ouverture de vanne, en en rendant compte à B. Plus tard B envoie à C1 et C2 l'ordre de fermeture de la vanne, en en rendant compte à A.

Respect de la causalité : le message de A (ouverture) doit être enregistré avant celui de B (fermeture).



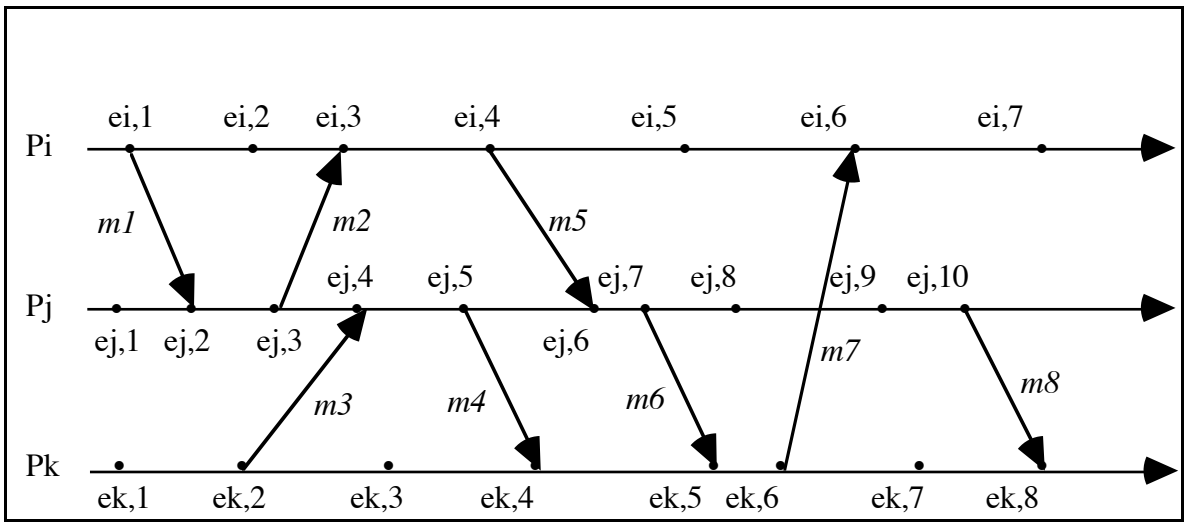
PASSÉ D'UN ÉVÉNEMENT



exercice 3 : quel est le passé de e13, e24, e23, e34 ?

e23 ∈ e32 est-ce vrai ou faux?
e24 ∈ e34 est-ce vrai ou faux?

PASSÉ D'UN ÉVÉNEMENT



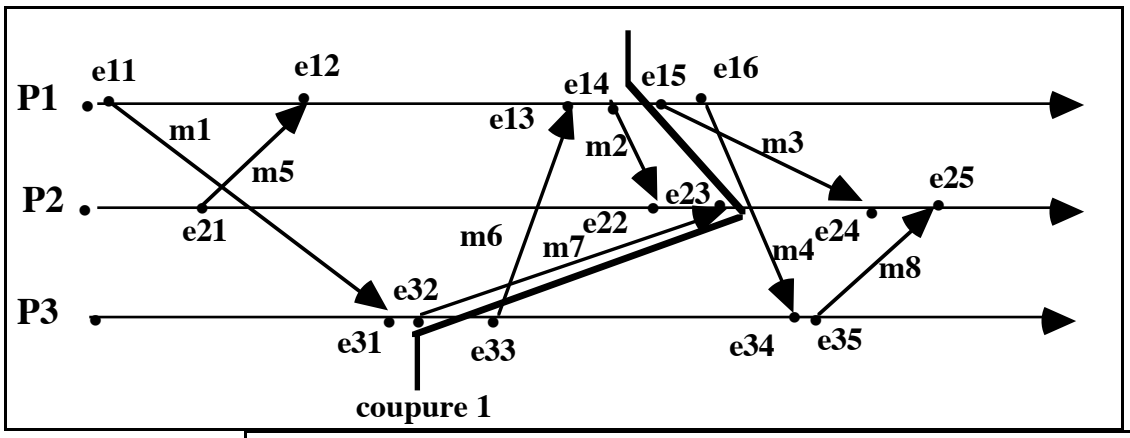
exercice 4 : donner le passé de
ei,2 ; ek,4 ; ek,1 ; ej,9 ; ek,2 ;
ei,7 ; ek,3 ; ei,5 ; ek,3 ; ej,10

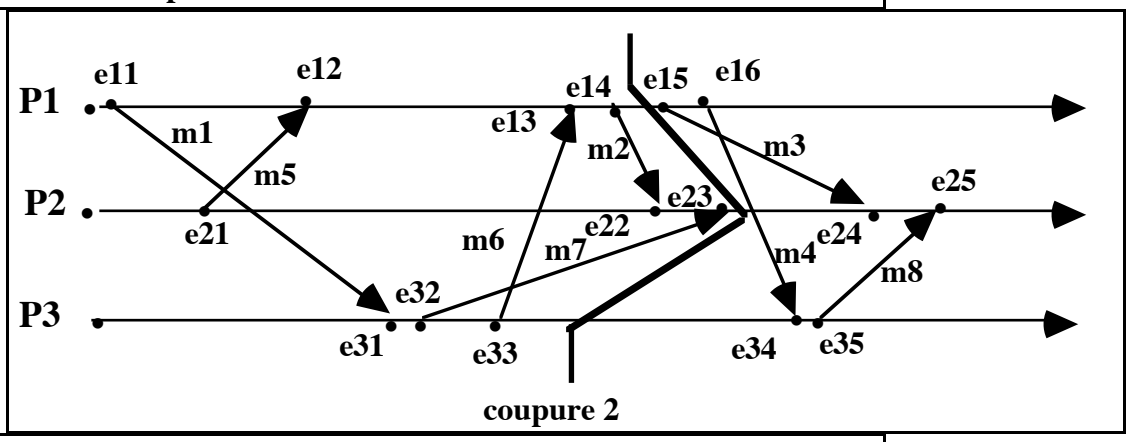
ek,1 ∈ ej,9 est-ce vrai ou faux?

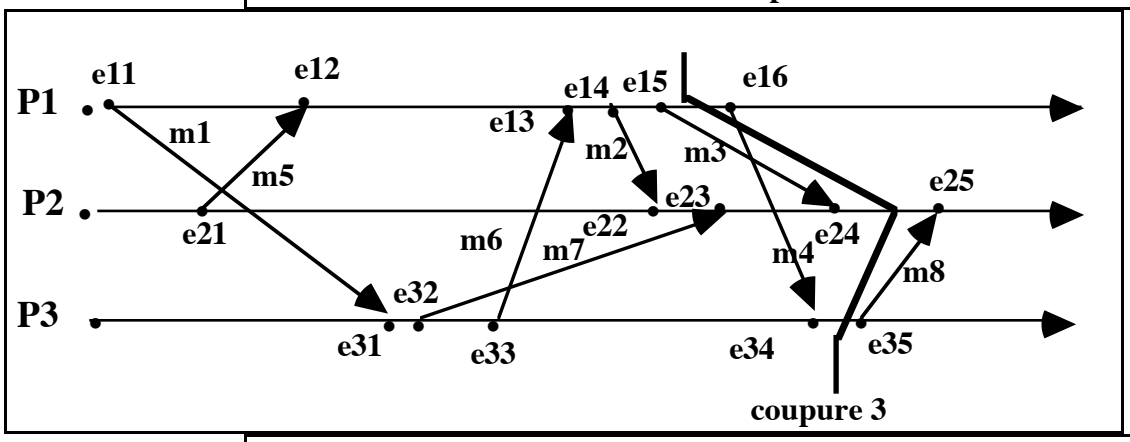
ek,2 ∈ ei,7 est-ce vrai ou faux?

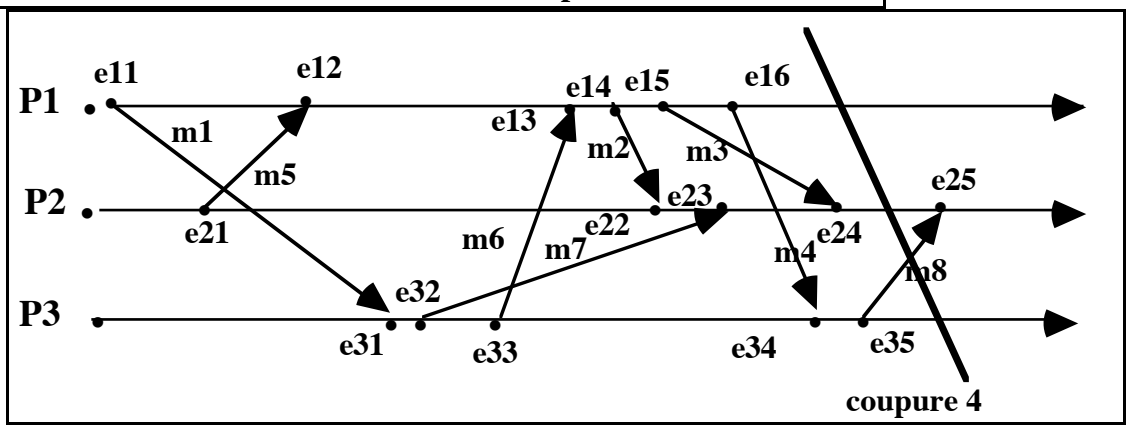
ek,3 ∈ ei,5 est-ce vrai ou faux?

COHÉRENCE DES COUPURES



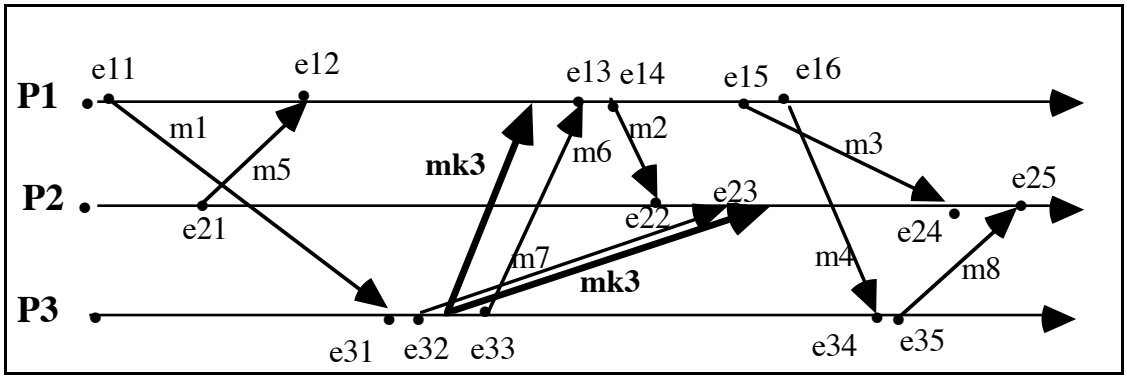






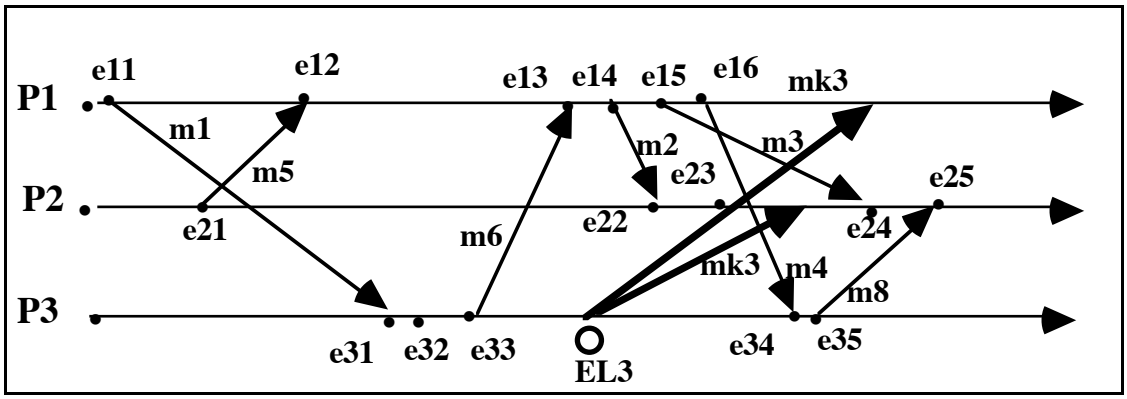
exercice 5: quelles sont les coupures cohérentes

CALCUL D'ÉTAT GLOBAL COHÉRENT



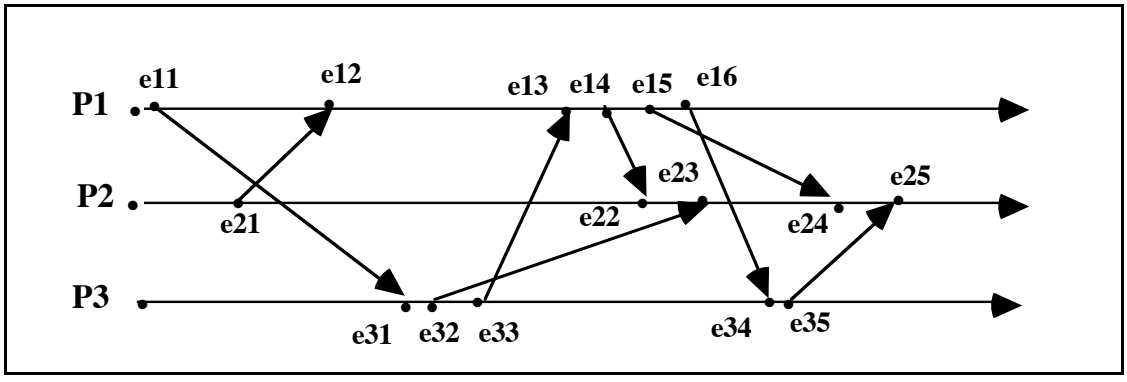
exercice 6:
P3 lance un calcul d'état cohérent après e32 selon Chandy Lamport compléter

CALCUL D'ÉTAT GLOBAL COHÉRENT



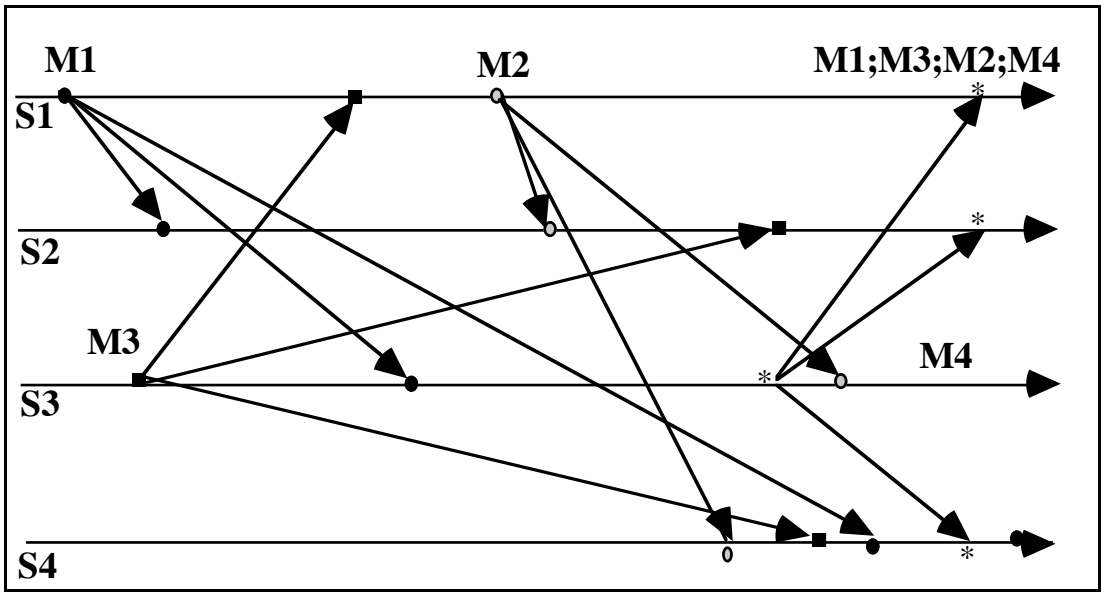
exercice 7 :
P3 lance un calcul d'état cohérent après e33 selon Chandy Lamport compléter

HORLOGES VECTORIELLES



exercice 8 : estampiller avec les horloges vectorielles

DIFFUSION AVEC ORDRE CAUSAL



exercice 9
délivrer les messages reçus selon l'ordre causal en mettant en oeuvre des horloges vectorielles

ORDRE TOTAL ET ESTAMPILLES DE LAMPORT

exercice 10

On veut analyser un programme réparti sur 4 sites. Pour cela, on enregistre sur chaque site les envois et arrivées de messages, ce qui permet de reconstituer la trace d'une exécution.

Une première exécution donne la première trace.

Une deuxième exécution du même programme donne une deuxième trace.

On note mij_k, le k^e message envoyé par le site i vers le site j. On utilise les horloges logiques et l'estampillage de Lamport pour donner un ordre total aux événements observés (envoi et arrivée de messages).

1. Que peut-on dire des événements :

A : arrivée sur le site 1 du message m41_1

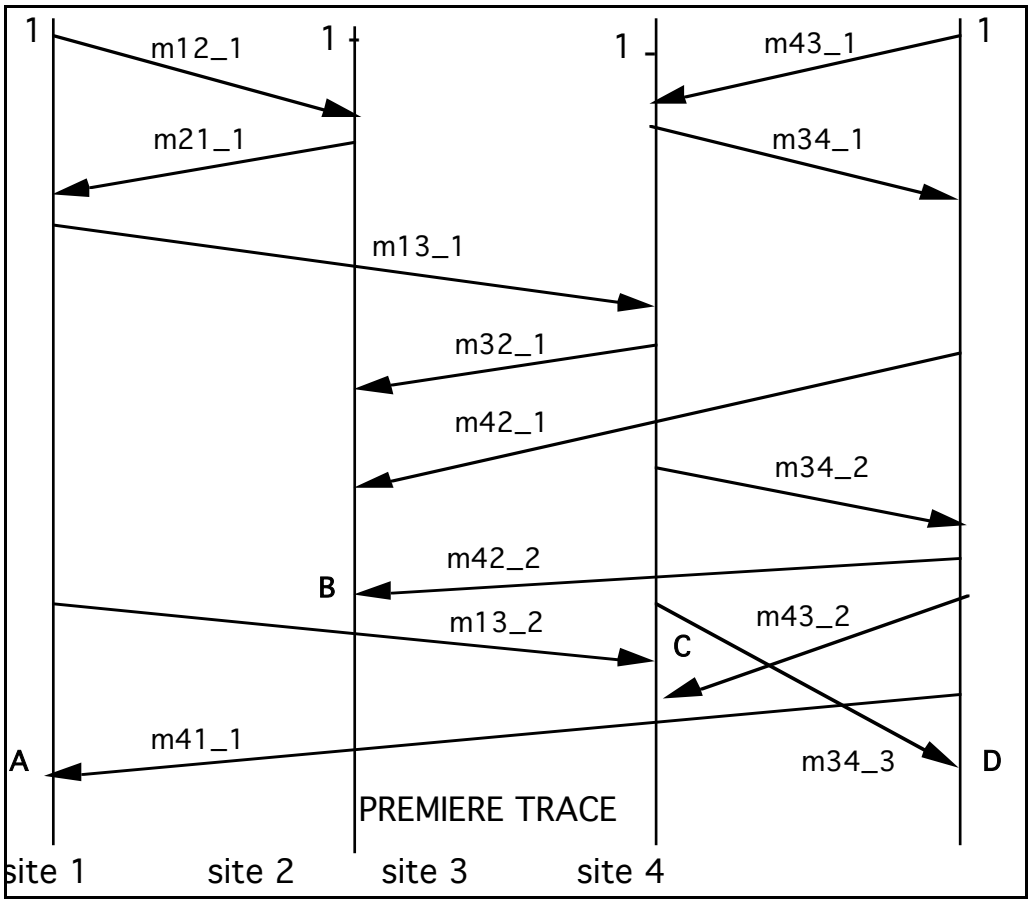
B : arrivée sur le site 2 du message m42_2

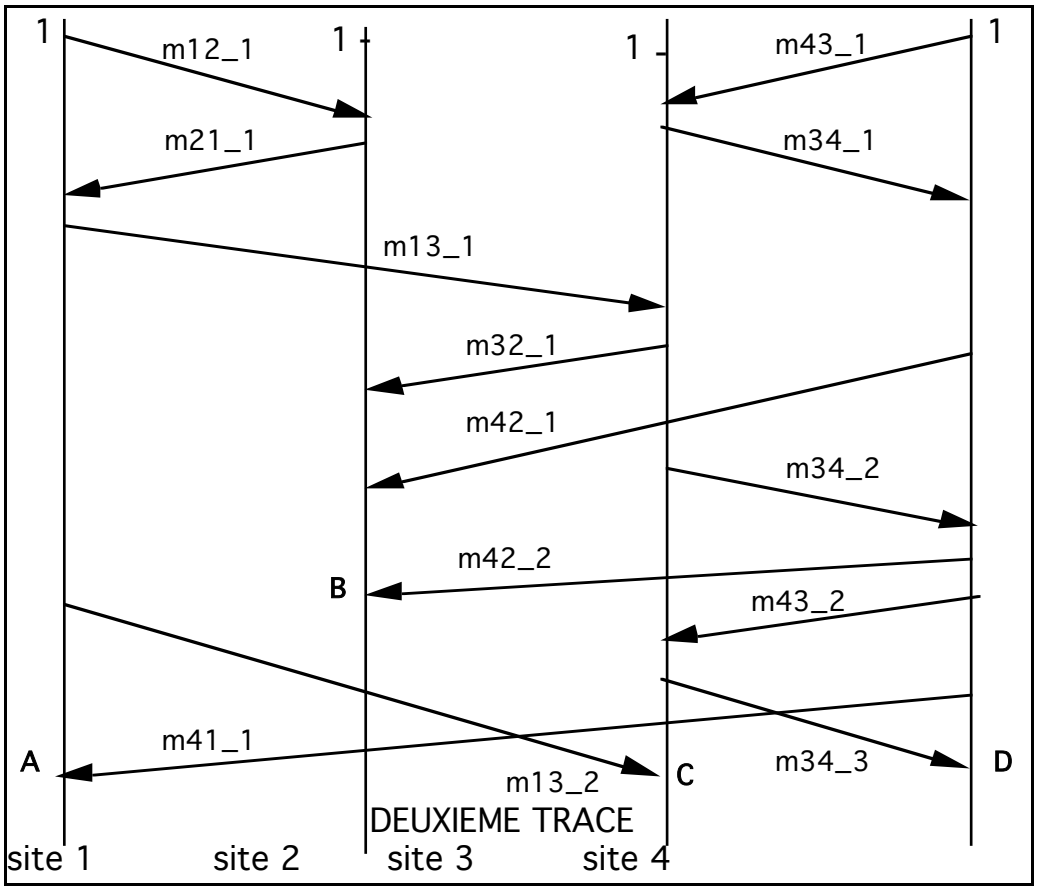
C : arrivée sur le site 3 du message m13_2

D : arrivée sur le site 4 du message m34_3

2. Dans chacune des traces, implanter des horloges vectorielles et retrouver la précédence entre A, B, C, D

3. Dans chacune des traces, implanter la date logique à la Lamport et classer ces événements selon l'ordre total, => , fourni par les estampilles.


1
1
1
1
m12_1
m43_1
m21_1
m34_1
m13_1
m32_1
m42_1
m34_2
m42_2
B
m13_2
m43_2
C
m41_1
A
m34_3
D
PREMIERE TRACE
site 1
site 2
site 3
site 4


1
1
1
1
m12_1
m43_1
m21_1
m34_1
m13_1
m32_1
m42_1
m34_2
m42_2
B
m43_2
m41_1
A
m13_2
C
m34_3
D
DEUXIEME TRACE
site 1
site 2
site 3
site 4

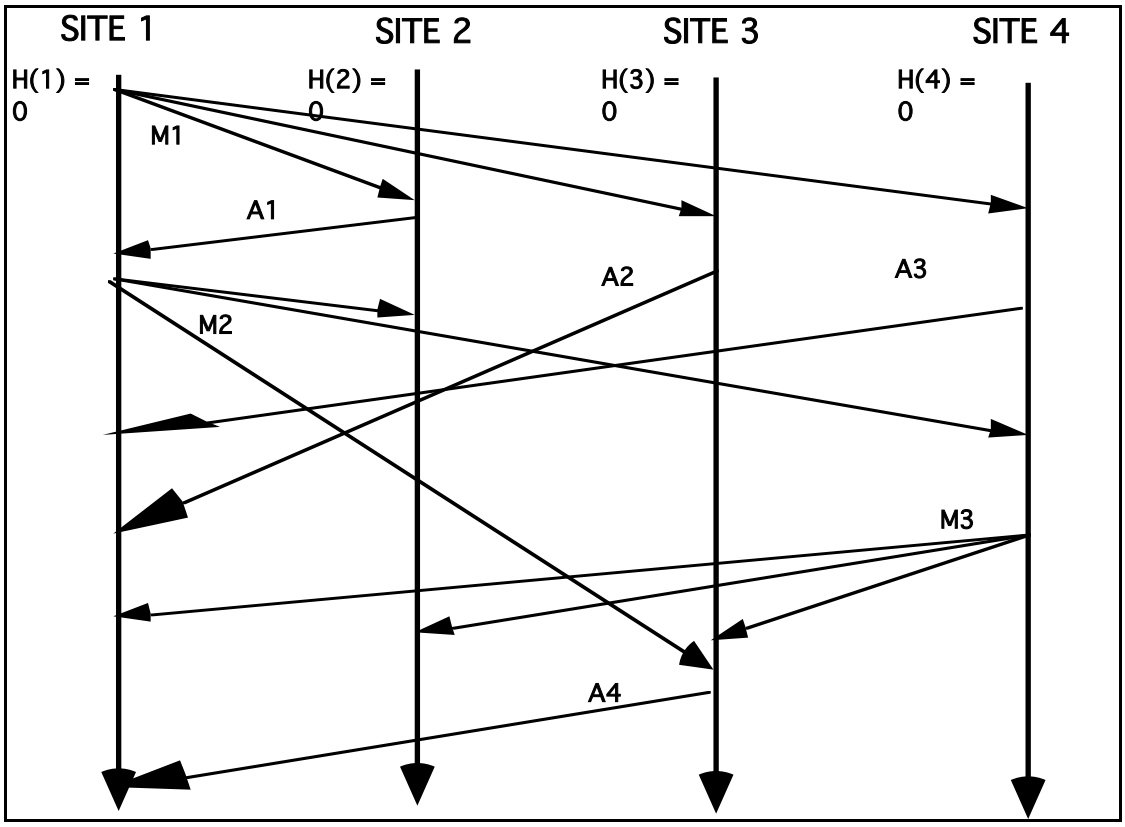
HORLOGES LOGIQUES A LA LAMPORT

exercice 11

Soit quatre sites qui diffusent des messages M1, M2, M3, ... ou qui répondent par des acquits A1, A2, A3, A4, ...Les acquits sont renvoyés à l'émetteur.

1 : A quelle heures locales des horloges logiques construites selon la règle de Lamport, le message M3 est-il envoyé par le site S4 et reçu par les sites S1, S2, S3 dans le fonctionnement ci-dessous?

2. Mettre des horloges vectorielles et délivrer les messages dans le respect de l'ordre causal



MISE A L'HEURE DES HORLOGES LOGIQUES

exercice 12

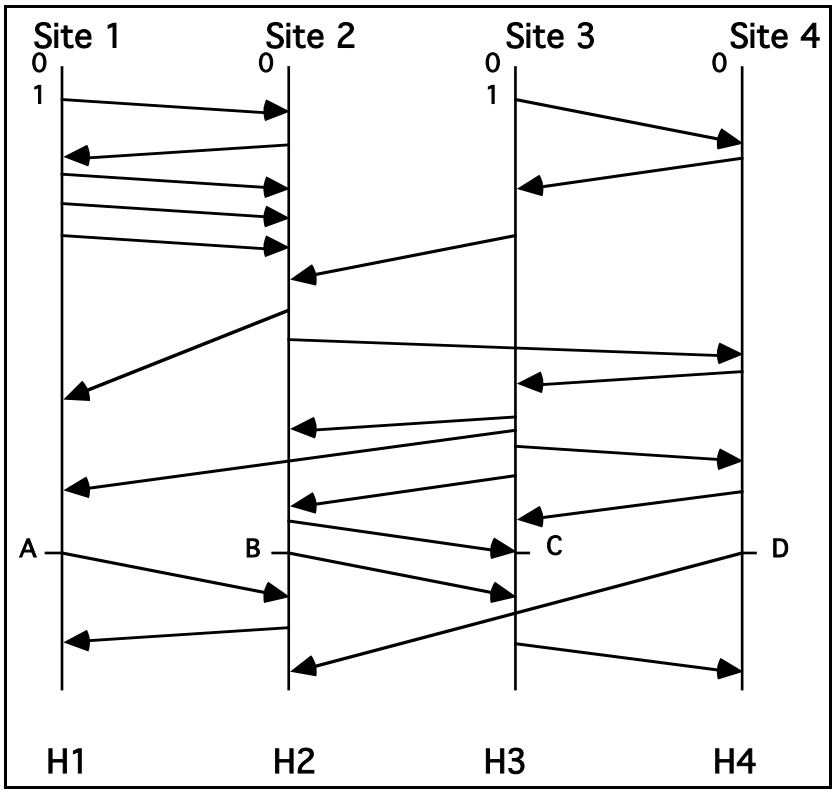
Un système qui comprend 4 sites a été l'objet des transferts de messages donnés dans la trace ci-dessous. On suppose que les horloges logiques de chaque site Hi étaient initialisées toutes à 0, que chaque message est estampillé selon la méthode de Lamport et que les horloges logiques ont été remises à l'heure pour respecter l'ordre total obtenu grâce à cet estampillage.

A). Dater tous les événements d'envoi et de réception de message qui apparaissent sur la trace.

B). Quelles sont les valeurs des horloges logiques de chacun des sites, à la fin de la trace.

C). En déduire une comparaison des événements A, B, C et D selon la relation d'ordre total => induit par l'estampillage à la Lamport.

D). Dater avec des horloges vectorielles. Quelle relation causale y a-t-il entre les événements A, B, C et D



EXCLUSION MUTUELLE RÉPARTIE

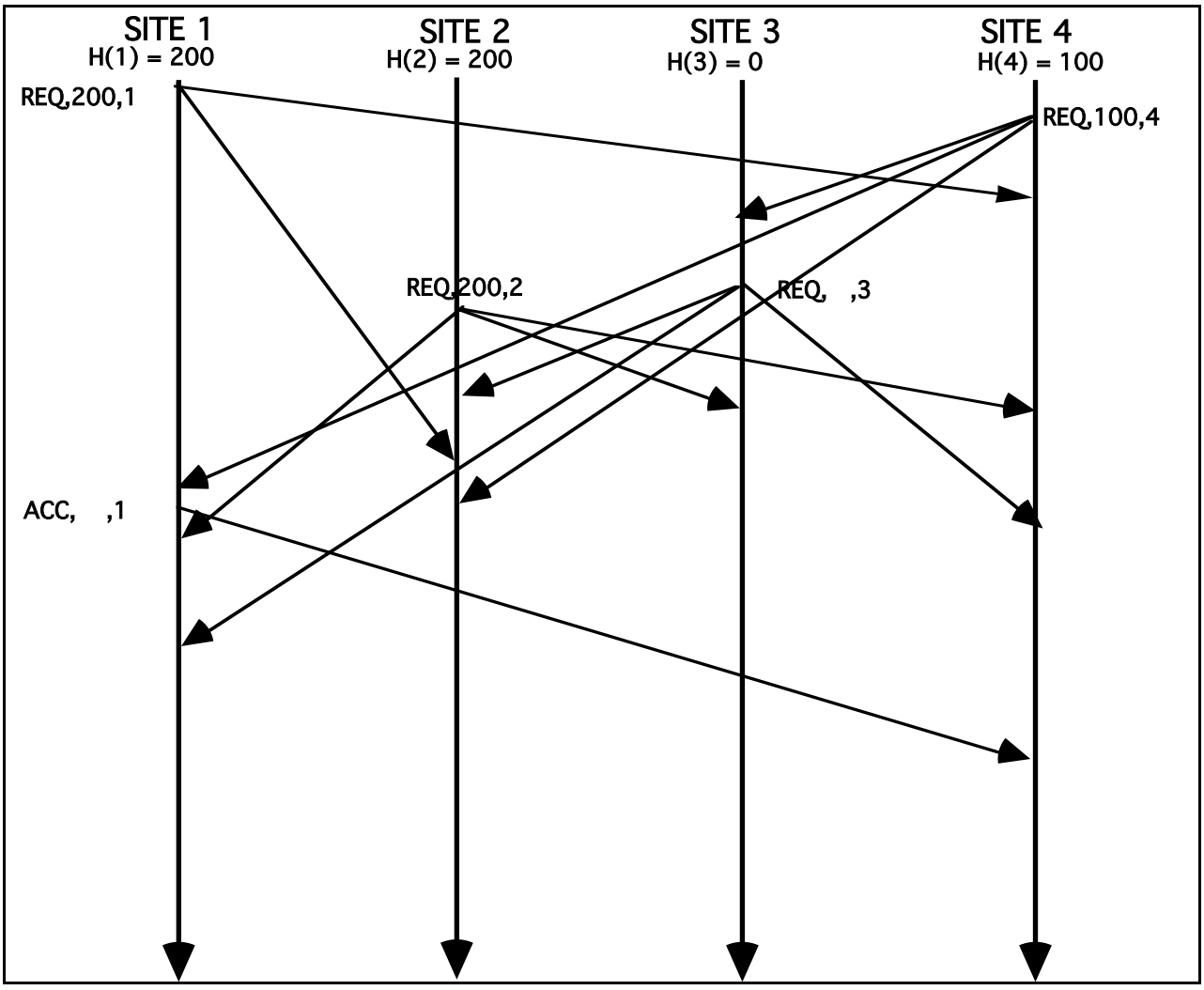
exercice 13

Les 4 sites envoient des demandes d'entrée en section critique comme indiqué ci-dessous. On vous demande de gérer cette situation :

1) avec l'algorithme de Lamport,

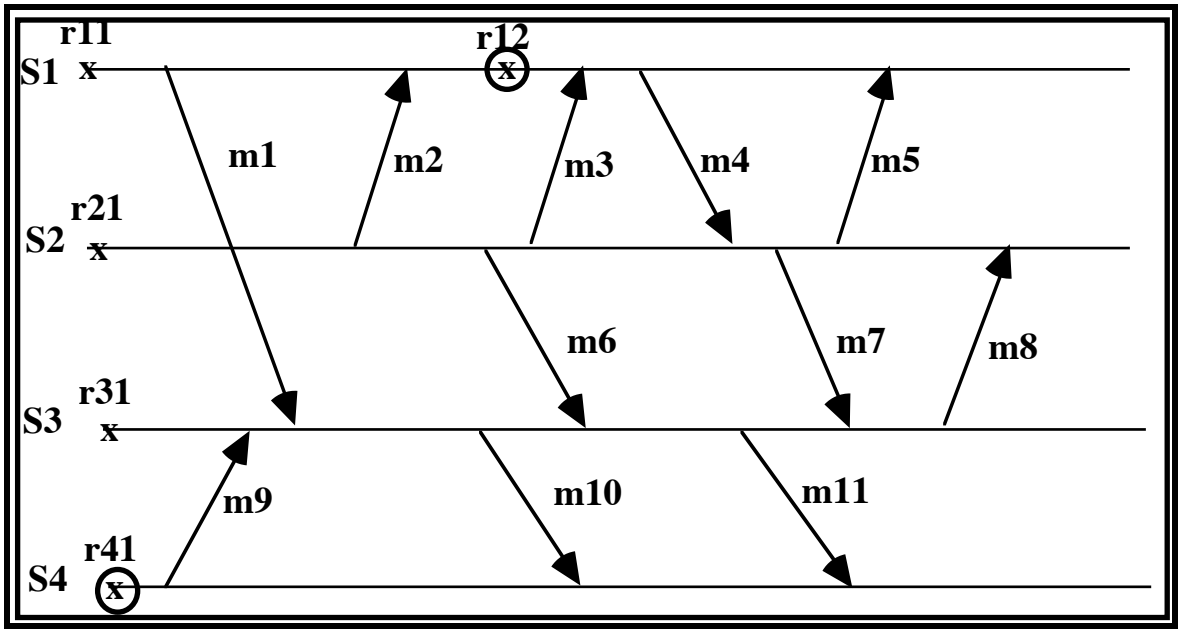
2) avec l'algorithme de Ricart et Agrawala

On suppose que les messages complémentaires pour chaque algorithme sont envoyés dès que c'est logiquement possible (Il n'y a pas d'attente locale et l'ordre local respecte l'ordre des demandes reçues. On rappelle que les canaux doivent être FIFO pour Lamport, mais non pour Ricart-Agrawala. On supposera pour ce dernier algorithme seulement que le canal C34 n'est pas FIFO et que l'ACC vers le site 4 est doublé par le message de demande REQ, ,3)

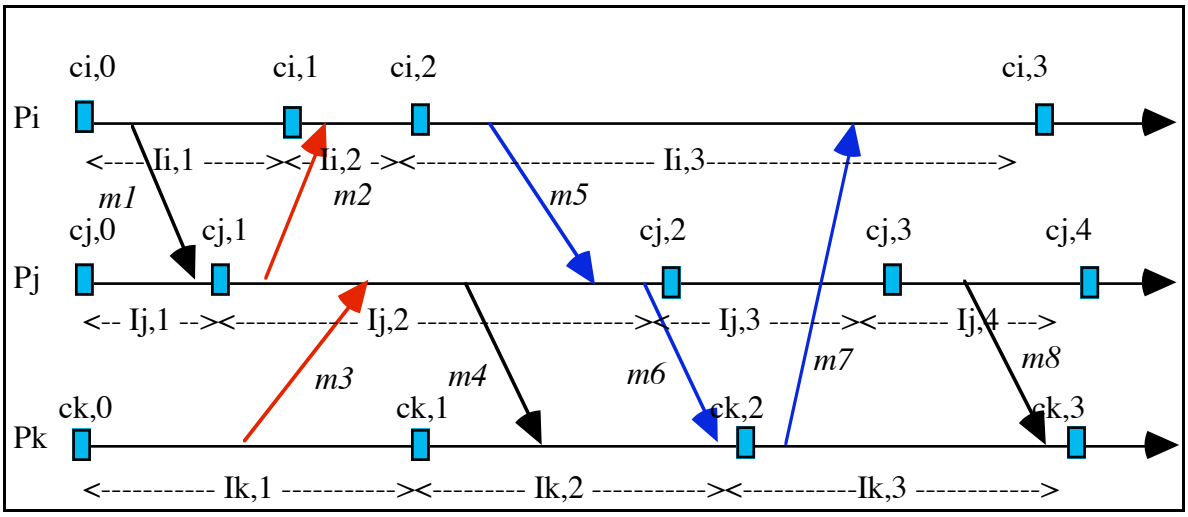


CHEMIN EN ZIGZAG

exercice 14.1 : montrer le chemin en zigzag de r41 à r12



14.2 rechercher les chemins et les cycles en zigzag



Autres exercices

1. étude de cas avec deux objets répartis A et B

et des traitements coopératifs répartis
P1 : A - 20 si A reste positif;
P2 : B + 20;
P3 : A + 10;
P4 : consulter A et B
problèmes de cohérence transactionnelle.

2. traitements transactionnels répartis avec 3 objets répartis

P1 : A - 20; B+20;
P2 : B-10; A+10
P3 : C := A+B
visualiser A, B, C

SOLUTIONS

SOLUTION DE L'EXERCICE 1:

ei,2 ∥ ek,4 ek,1 → ej,9 ek,2 → ei,7 ek,3 ∥ ei,5 ek,3 ∥ ej,10

SOLUTION DE L'EXERCICE 2 : m7 et m2

EMISSION3(m7) → EMISSION1(m2) mais RECEPTION2(m2) → RECEPTION2(m7)

SOLUTION DE L'EXERCICE 3

passé de e13 : e11 e12 e13
e21
e31 e32 e33

passé de e24 : e11 e12 e13 e14 e15
e21 e22 e23 e24
e31 e32 e33

passé de e23 : e11 e12 e13 e14
e21 e22 e23
e31 e32 e33
passé de e22 → {e23}

passé de e34 : e11 e12 e13 e14 e15 e16
e21
e31 e32 e33 e34

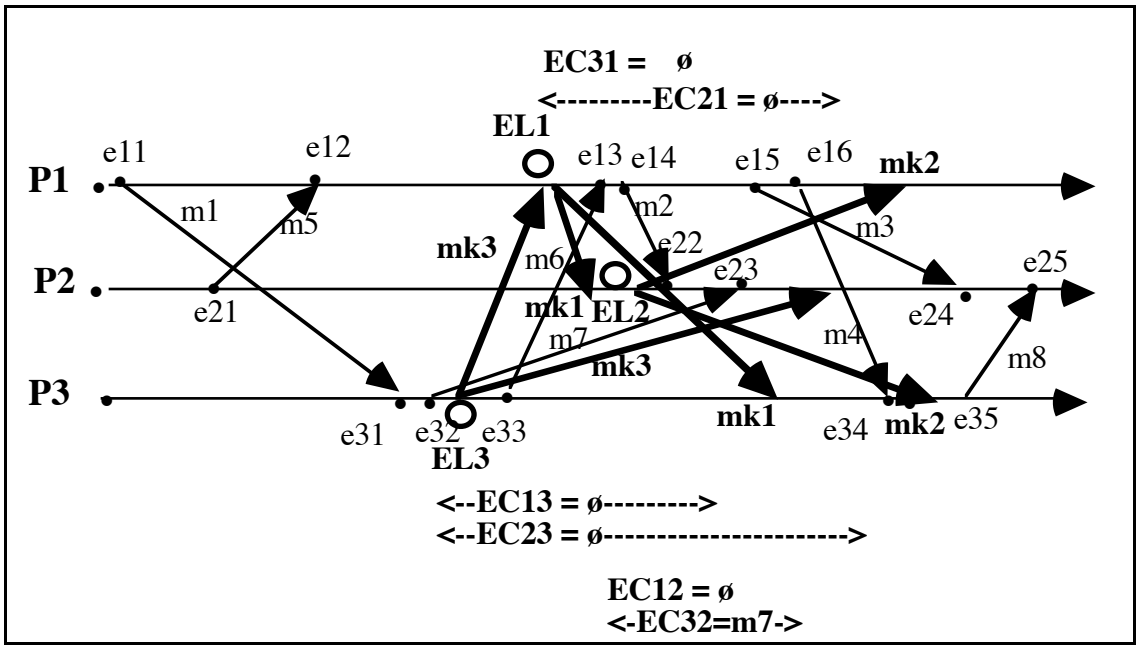
e24 ∥ e34 est vrai e23 ∥ e32 est faux

e32 → e23

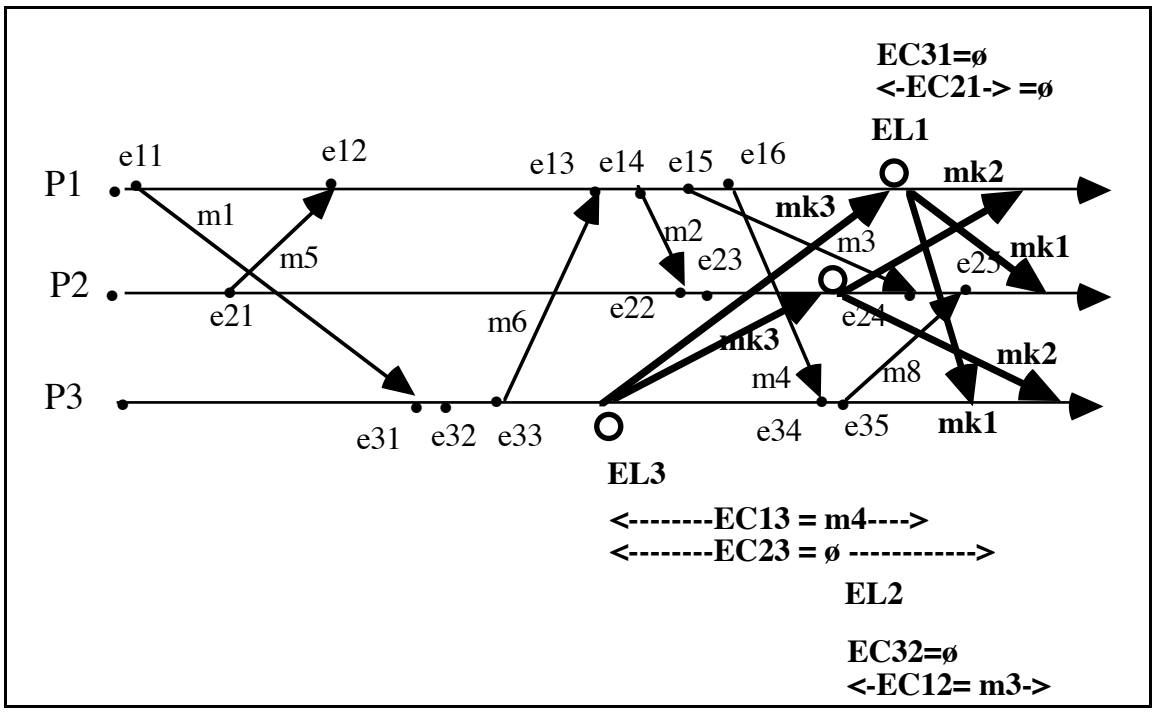
passé de e23 = passé de e32 → {e33 e13 e14 e22}

SOLUTION DE L'EXERCICE 5 : les coupures 2 et 4

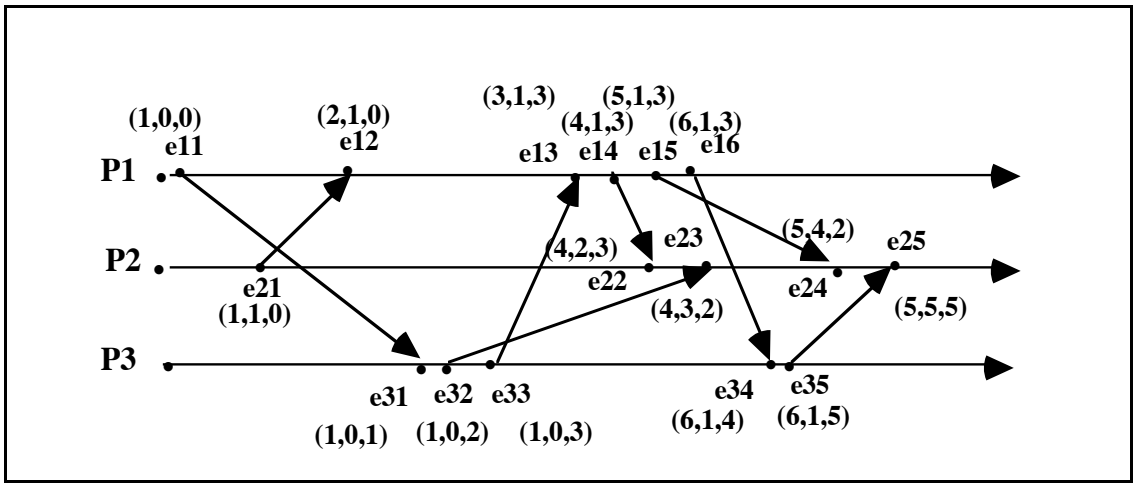
SOLUTION DE L'EXERCICE 6



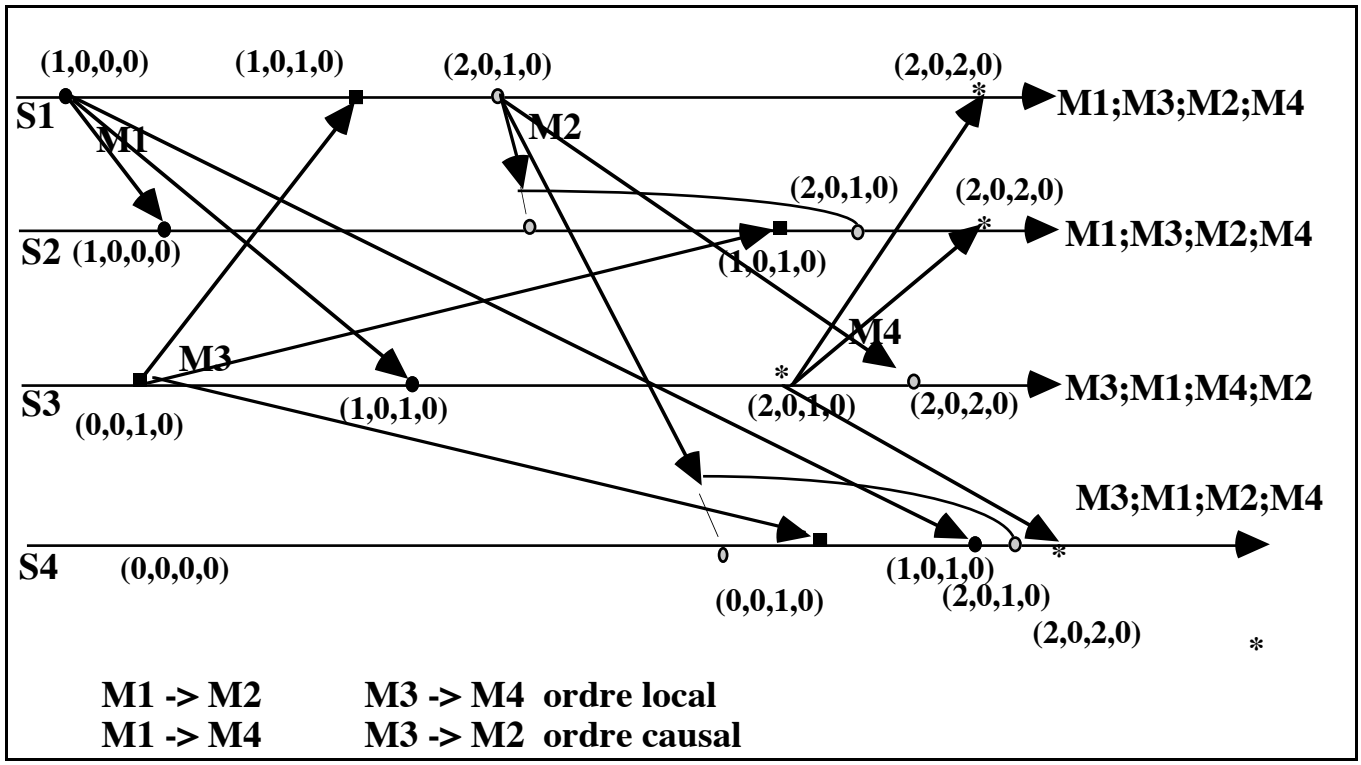
SOLUTION DE L'EXERCICE 7



SOLUTION DE L'EXERCICE 8



SOLUTION DE L'EXERCICE 9



SOLUTION DE L'EXERCICE 10 : première trace⊡C(10) => B(11) => A(13) => D(13)
Deuxième trace⊡B(11) => A(13) => C(14) => D(14)

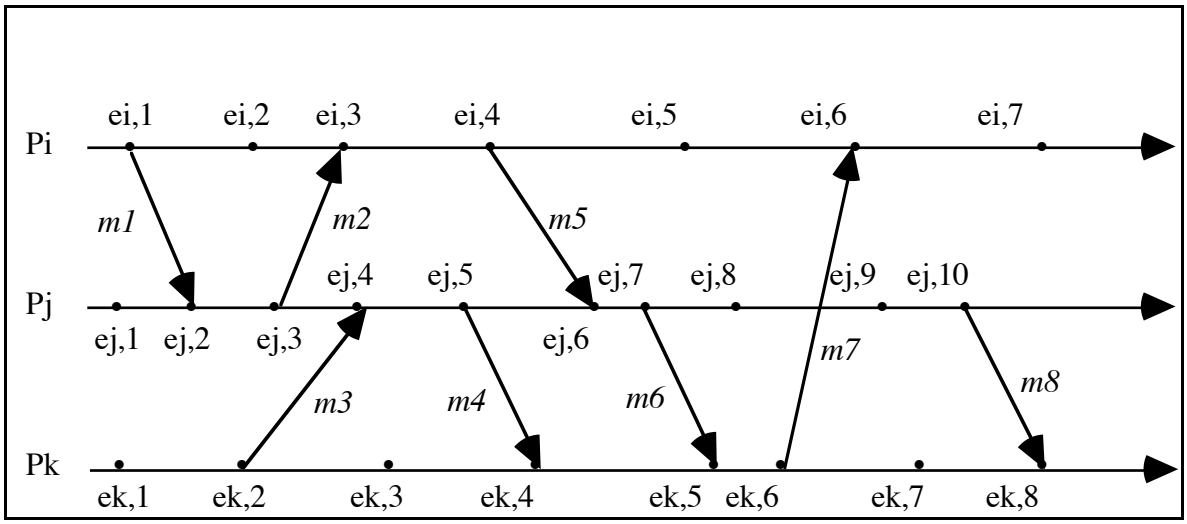
SOLUTION DE L'EXERCICE 12. CORRIGE DES HORLOGES LOGIQUES
A la fin H1 = 24, H2 = 24, H3 = 23, H4 = 24
H1(A) = 18, H4(D) = 20, H2(B) = 21, H3(C) = 21 donc A => D => B => C

SOLUTION DE L'EXERCICE 14.1 : m9,m8,m2
SOLUTION DE L'EXERCICE 14.2
ck,0 vers ci,2 : m3, m2
ci,2 vers ck,2 : m5, m4 ou encore m5, m6
ck,2 vers ck,2 : m7, m5, m6
ci,2 vers ci,2 : m5,m2

8. SOLUTIONS DES EXERCICES

RELATION DE PRÉCEDENCE ENTRE DES ÉVÉNEMENTS RÉPARTIS

Exercice 1



Comparer :

ei,2 et ek,4

ek,1 et ej,9

ek,2 et ei,7

ek,3 et ei,5

ek,3 et ej,10

solution de l'exercice 1 :

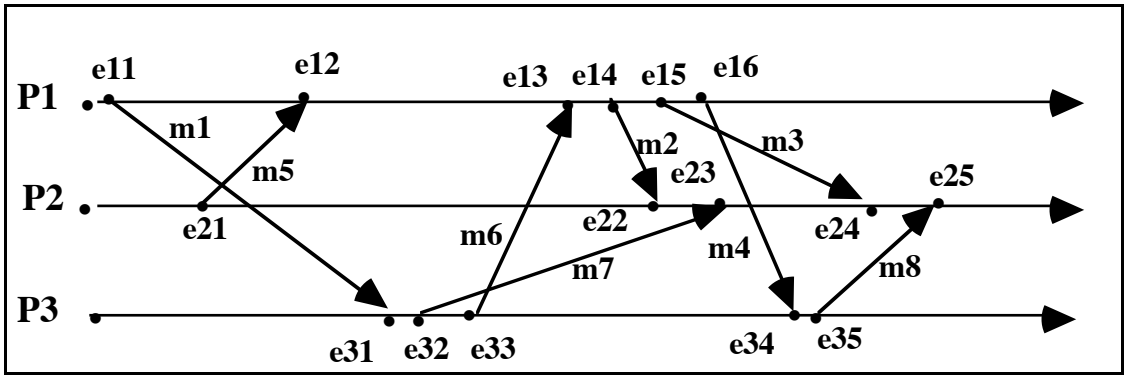
ei,2 ∥ ek,4 ek,1 → ej,9 ek,2 → ei,7 ek,3 ∥ ei,5 ek,3 ∥ ej,10

DÉPENDANCE CAUSALE ENTRE MESSAGES

Un exercice

exercice 2

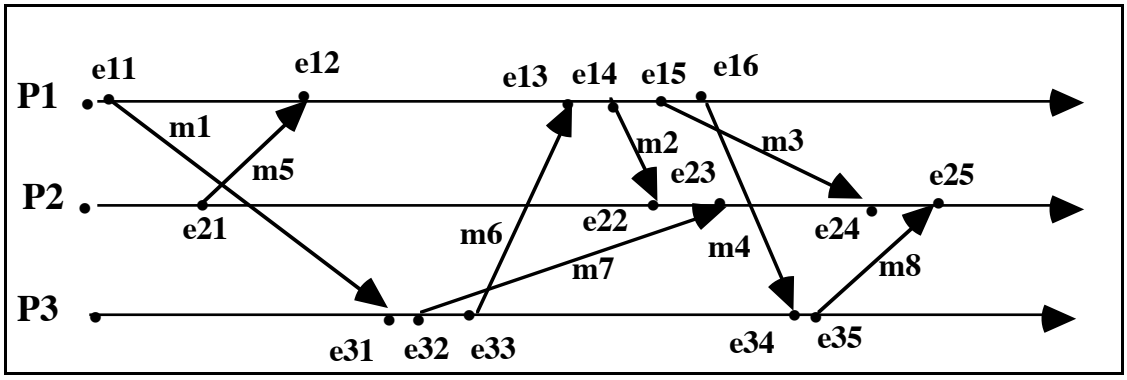
quels sont les messages qui ne respectent pas la dépendance causale?



solution de l'exercice 2 : m7 et m2

$EMISSION_3(m7) \rightarrow EMISSION_1(m2)$
mais
$RECEPTION_2(m2) \not\rightarrow RECEPTION_2(m7)$

PASSÉ D'UN ÉVÉNEMENT



exercice 3

quel est le passé de e13, e24, e23, e34 ?

e23 □ e32 est-ce vrai ou faux?
e24 □ e34 est-ce vrai ou faux?

solution de l'exercice 3

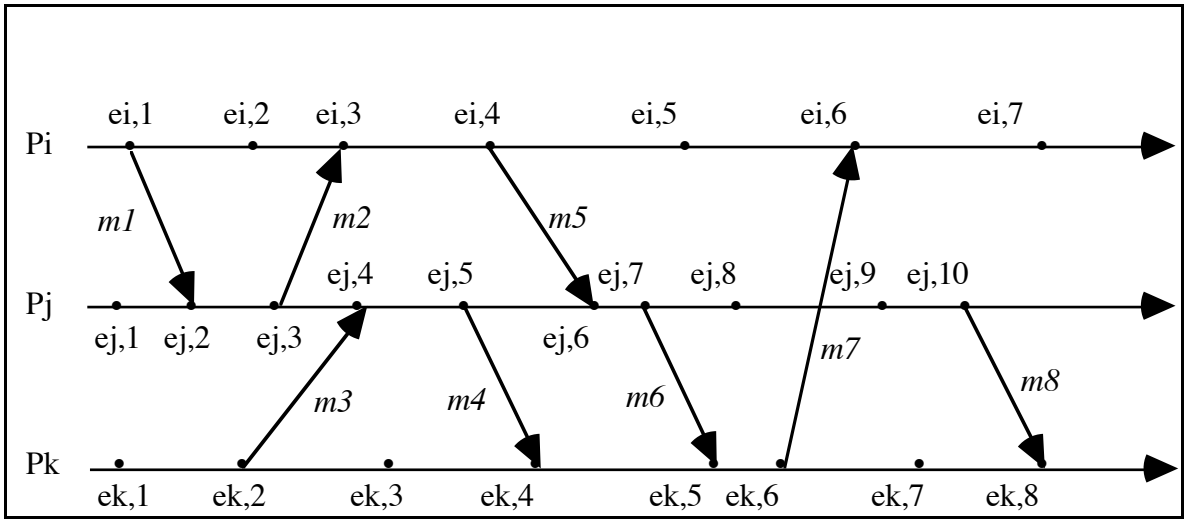
passé de e13 : e11 e12 e13
e21
e31 e32 e33
passé de e24 : e11 e12 e13 e14 e15
e21 e22 e23 e24
e31 e32 e33
passé de e23 : e11 e12 e13 e14
e21 e22 e23
e31 e32 e33
passé de e22 □ {e23}
passé de e34 : e11 e12 e13 e14 e15 e16
e21
e31 e32 e33 e34

e24 □ e34 est vrai

e23 □ e32 est faux

car e32 est dans le passé de e23 : e32 □ e23
passé de e23 = passé de e32 □ {e33 e13 e14 e22}

PASSÉ D'UN ÉVÉNEMENT



exercice 4

quel est le passé de

ei,2

ek,4

ek,1

ej,9

ek,2

ei,7

ek,3

ei,5

ek,3

ej,10

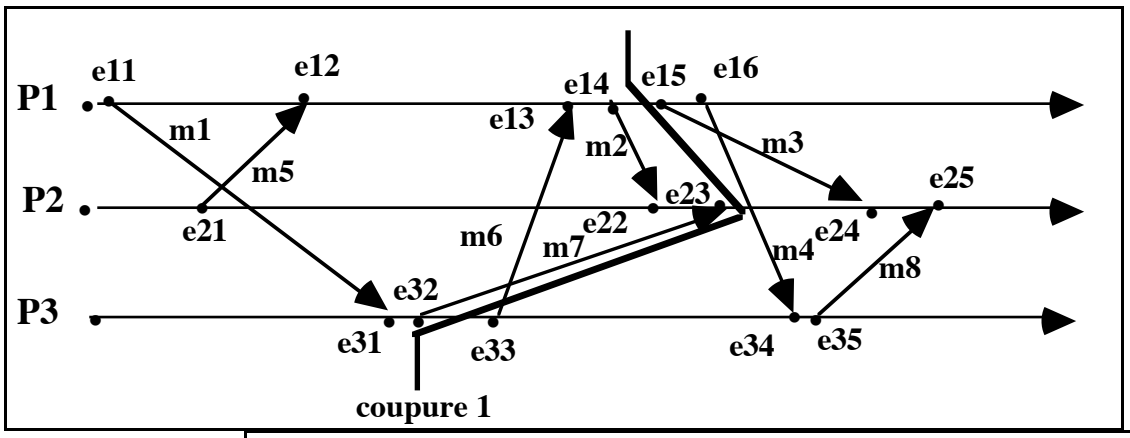
?

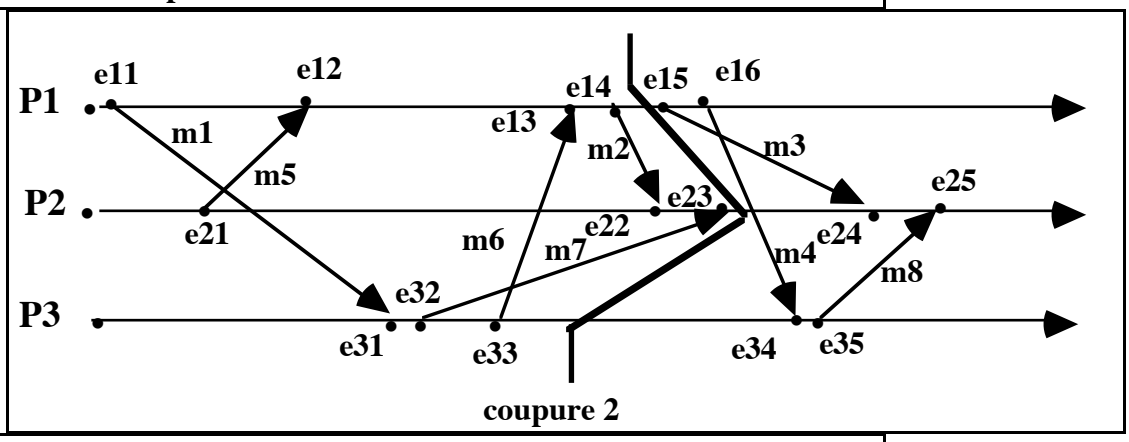
ek,1 □ ej,9 est-ce vrai ou faux?

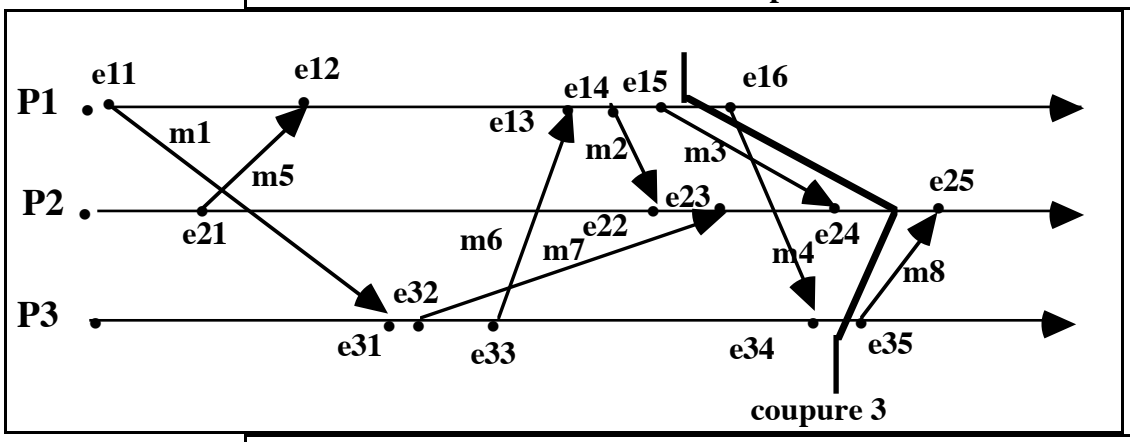
ek,2 □ ei,7 est-ce vrai ou faux?

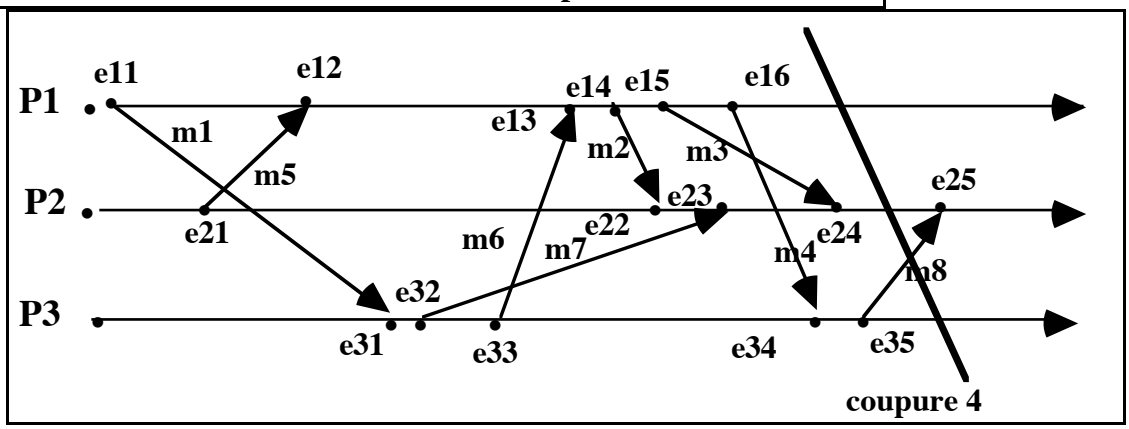
ek,3 □ ei,5 est-ce vrai ou faux?

COHÉRENCE DES COUPURES



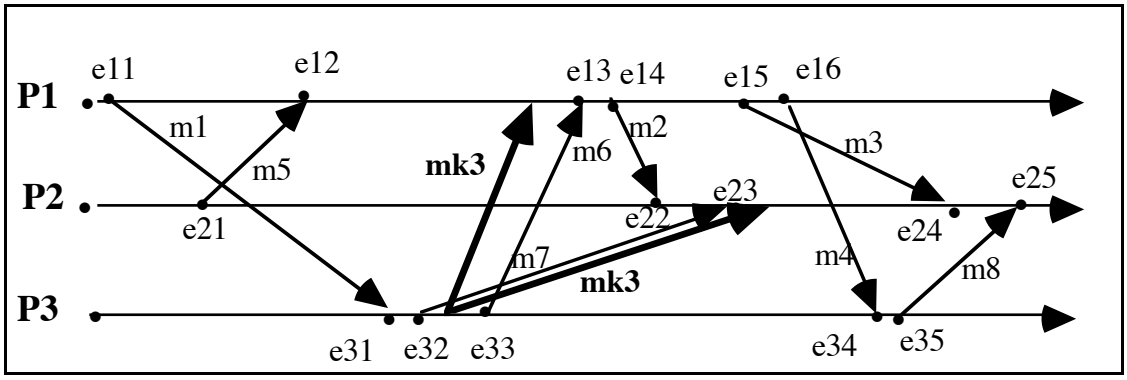






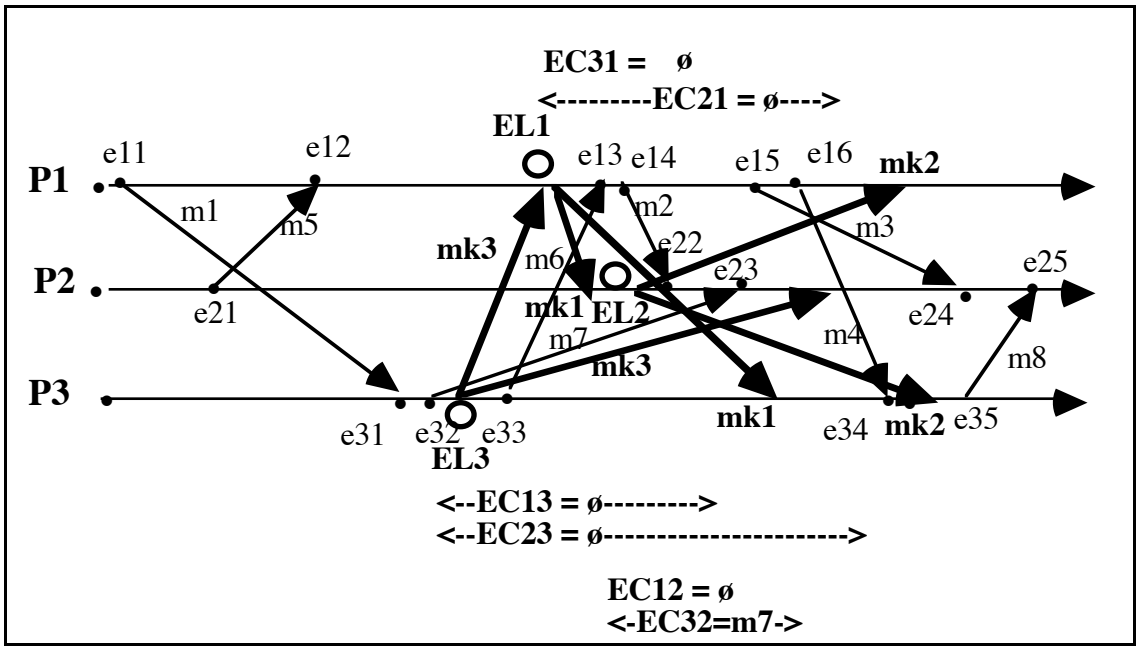
exercice 5: quelles sont les coupures cohérentes
solution de l'exercice 5 : les coupures 2 et 4

CALCUL D'ÉTAT GLOBAL COHÉRENT

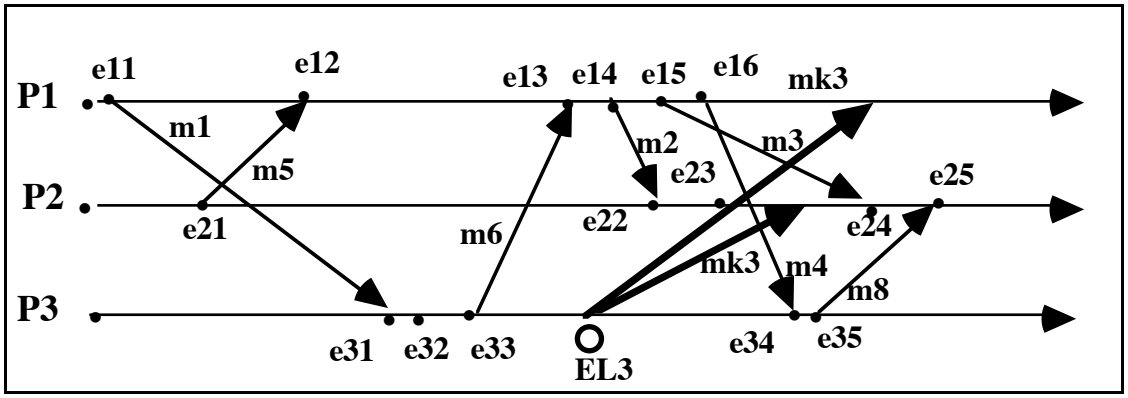


exercice 6:
P3 lance un calcul d'état cohérent après e32 selon Chandy Lamport compléter

solution de l'exercice 6

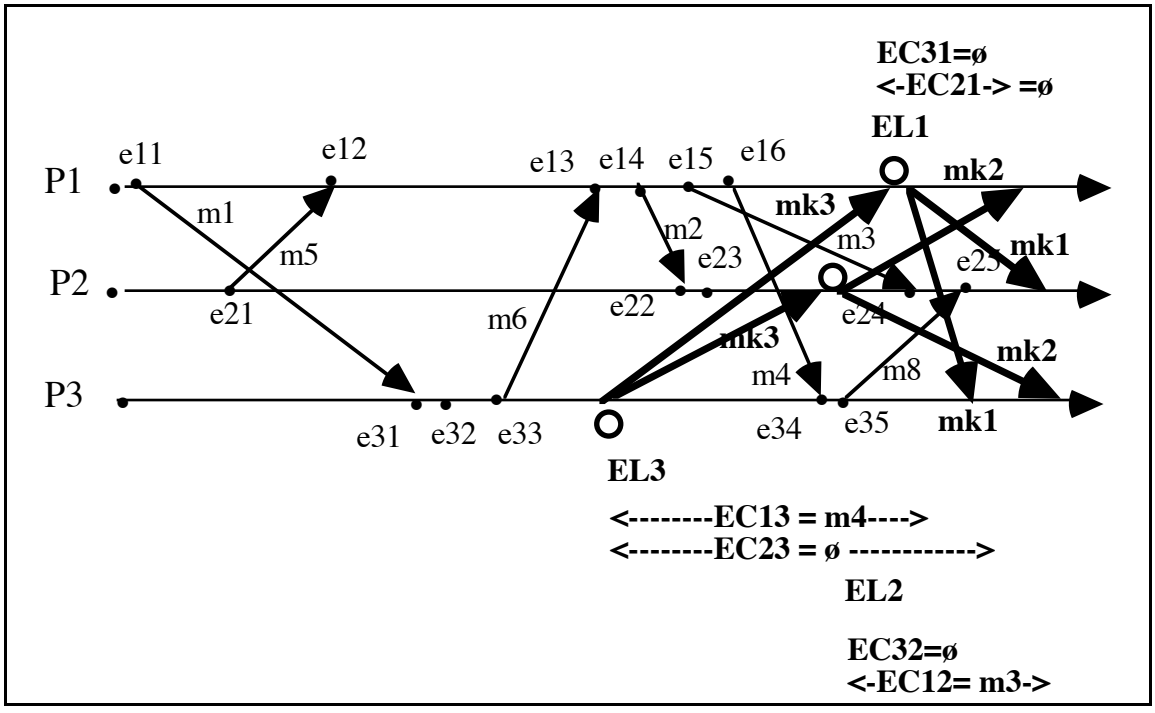


CALCUL D'ÉTAT GLOBAL COHÉRENT

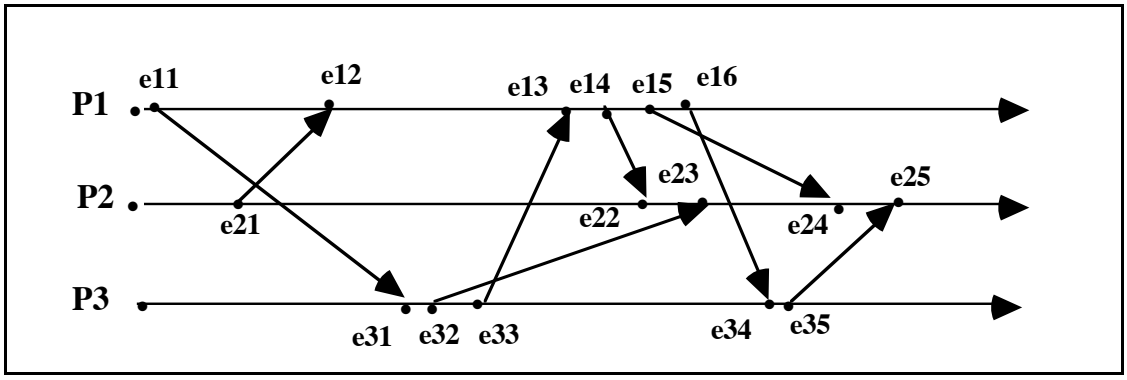


exercice 7 :
P3 lance un calcul d'état cohérent après e33 selon Chandy Lamport compléter

solution de l'exercice 7

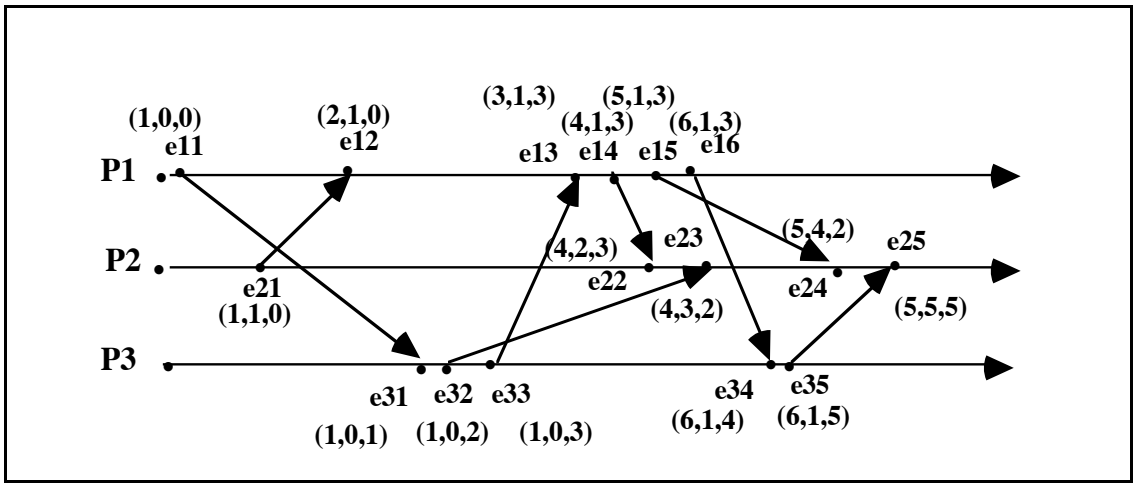


HORLOGES VECTORIELLES

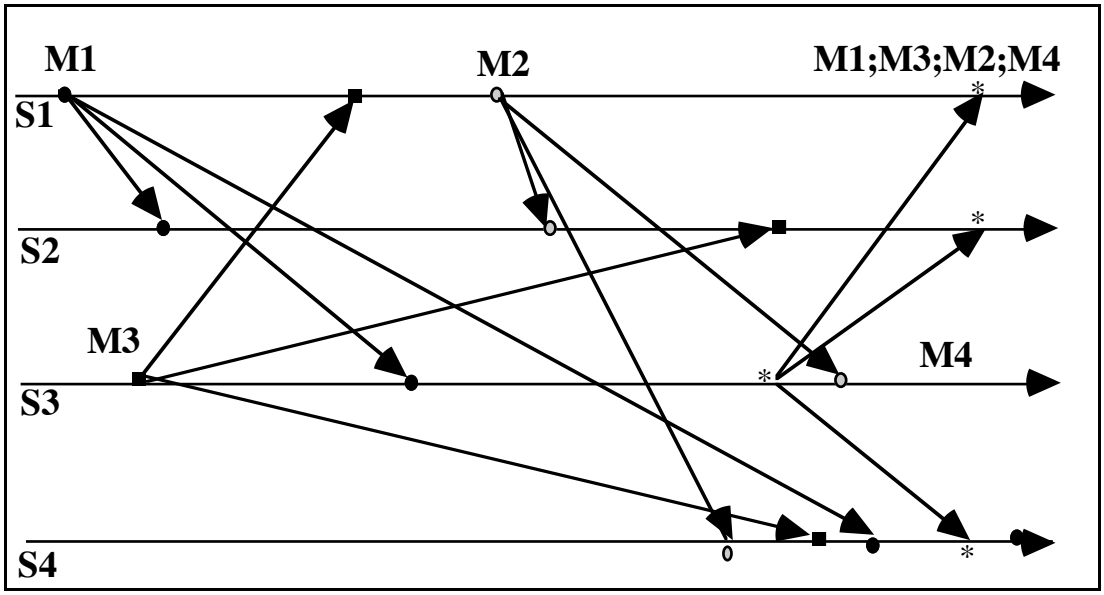


exercice 8 : estampiller avec les horloges vectorielles

solution de l'exercice 8

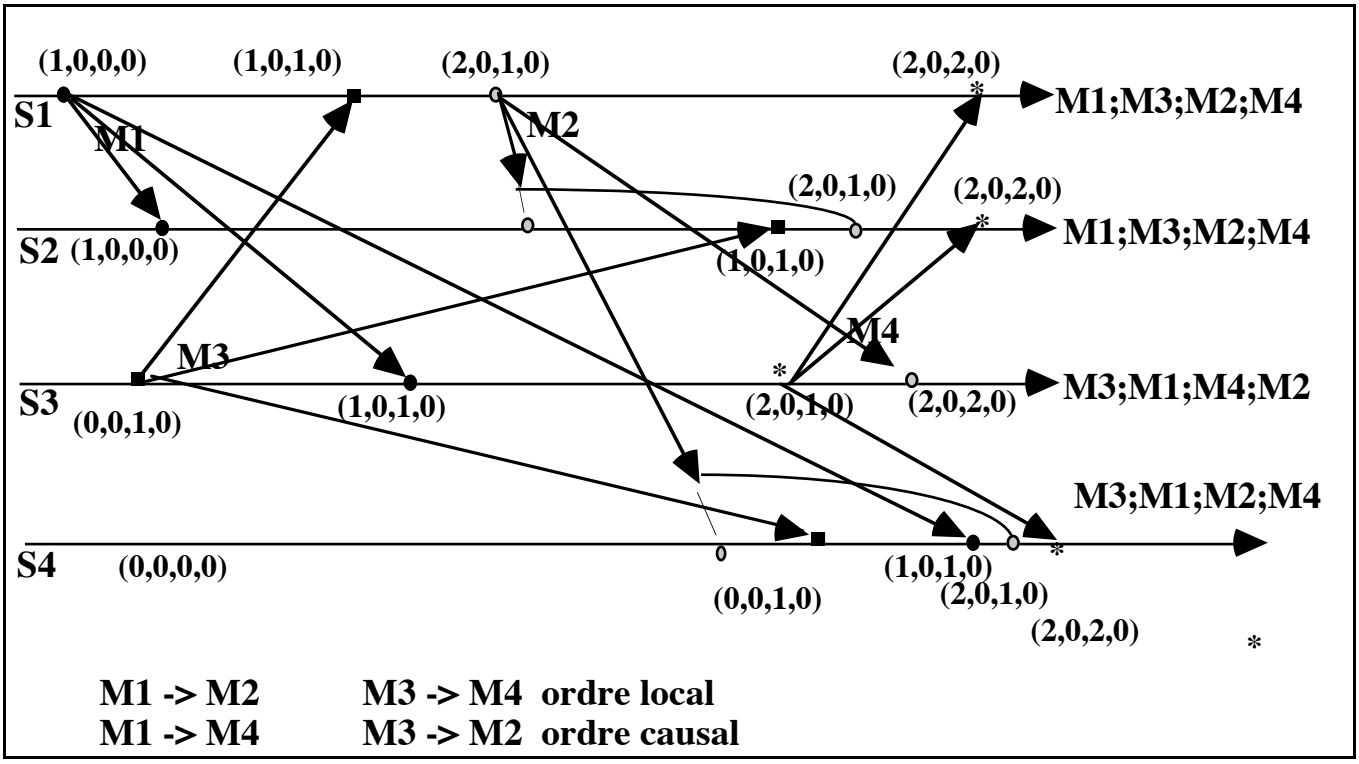


DIFFUSION AVEC ORDRE CAUSAL



exercice 9
délivrer les messages reçus selon l'ordre causal en mettant en oeuvre des horloges vectorielles

solution de l'exercice 9



ORDRE TOTAL ET ESTAMPILLES DE LAMPORT

exercice 10

On veut analyser un programme réparti sur 4 sites. Pour cela, on enregistre sur chaque site les envois et arrivées de messages, ce qui permet de reconstituer la trace d'une exécution.

Une première exécution donne la première trace.

Une deuxième exécution du même programme donne une deuxième trace.

On note mij_k, le k^e message envoyé par le site i vers le site j. On utilise les horloges logiques et l'estampillage de Lamport pour donner un ordre total aux événements observés (envoi et arrivée de messages).

1. Que peut-on dire des événements :

A : arrivée sur le site 1 du message m41_1

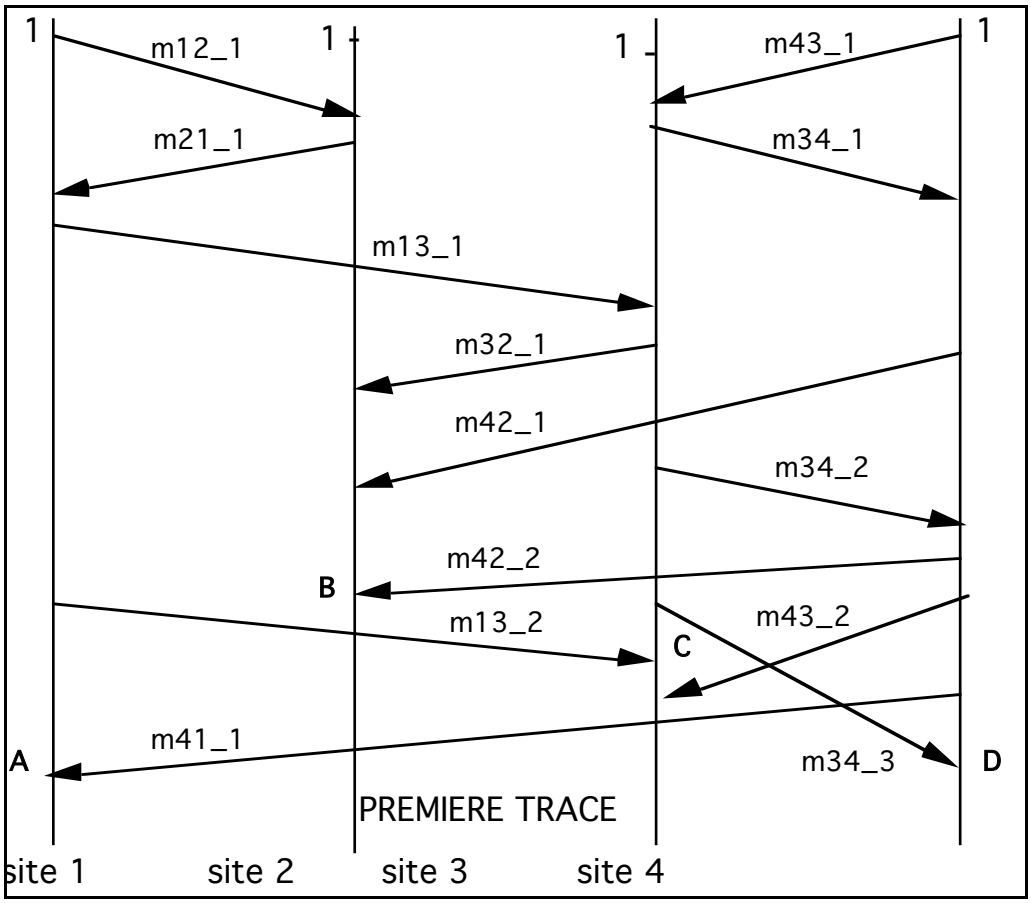
B : arrivée sur le site 2 du message m42_2

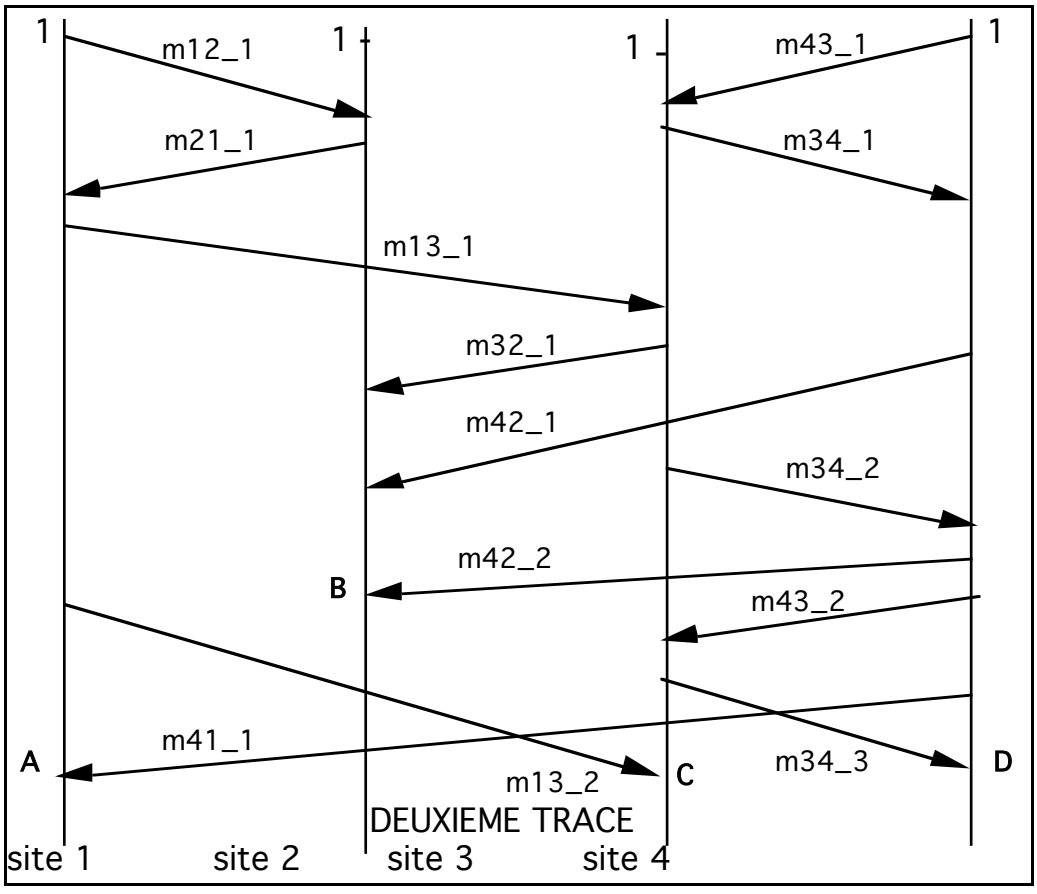
C : arrivée sur le site 3 du message m13_2

D : arrivée sur le site 4 du message m34_3

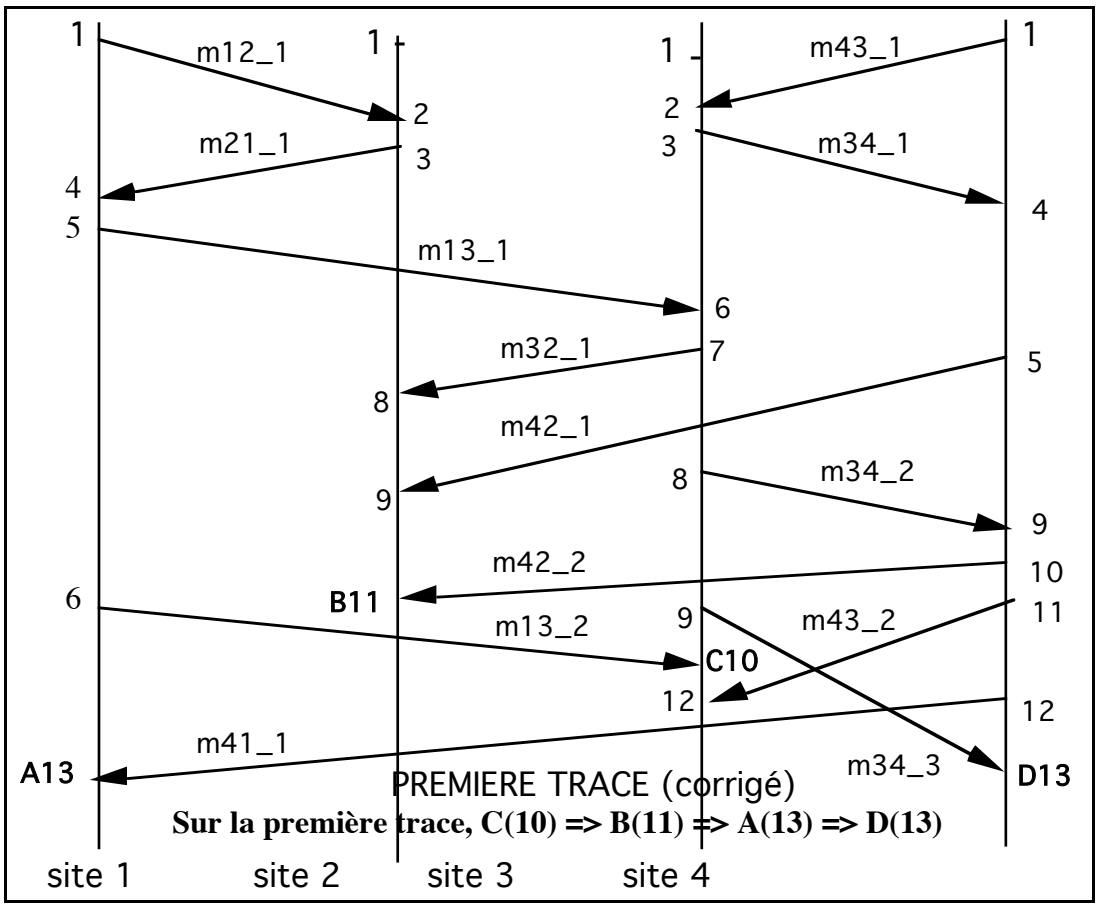
2. Dans chacune des traces, implanter des horloges vectorielles et retrouver la précédence entre A, B, C, D

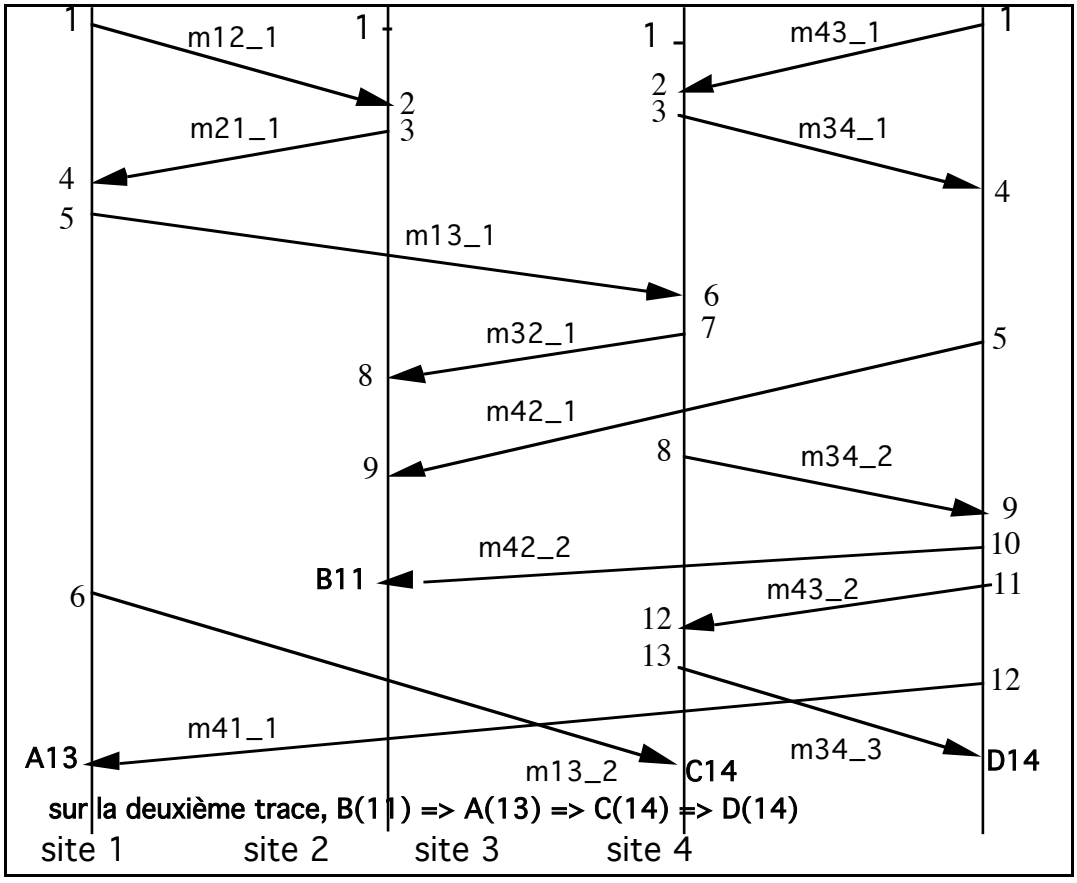
3. Dans chacune des traces, implanter la date logique à la Lamport et classer ces événements selon l'ordre total, => , fourni par les estampilles :


1
1
1
1
m12_1
m43_1
m21_1
m34_1
m13_1
m32_1
m42_1
m34_2
m42_2
B
m13_2
m43_2
C
m41_1
A
m34_3
D
PREMIERE TRACE
site 1
site 2
site 3
site 4


1
1
1
1
m12_1
m43_1
m21_1
m34_1
m13_1
m32_1
m42_1
m34_2
m42_2
B
m43_2
m41_1
A
m13_2
C
m34_3
D
DEUXIEME TRACE
site 1
site 2
site 3
site 4

solution de l'exercice 10 : UN CORRIGE POUR LES TRACES





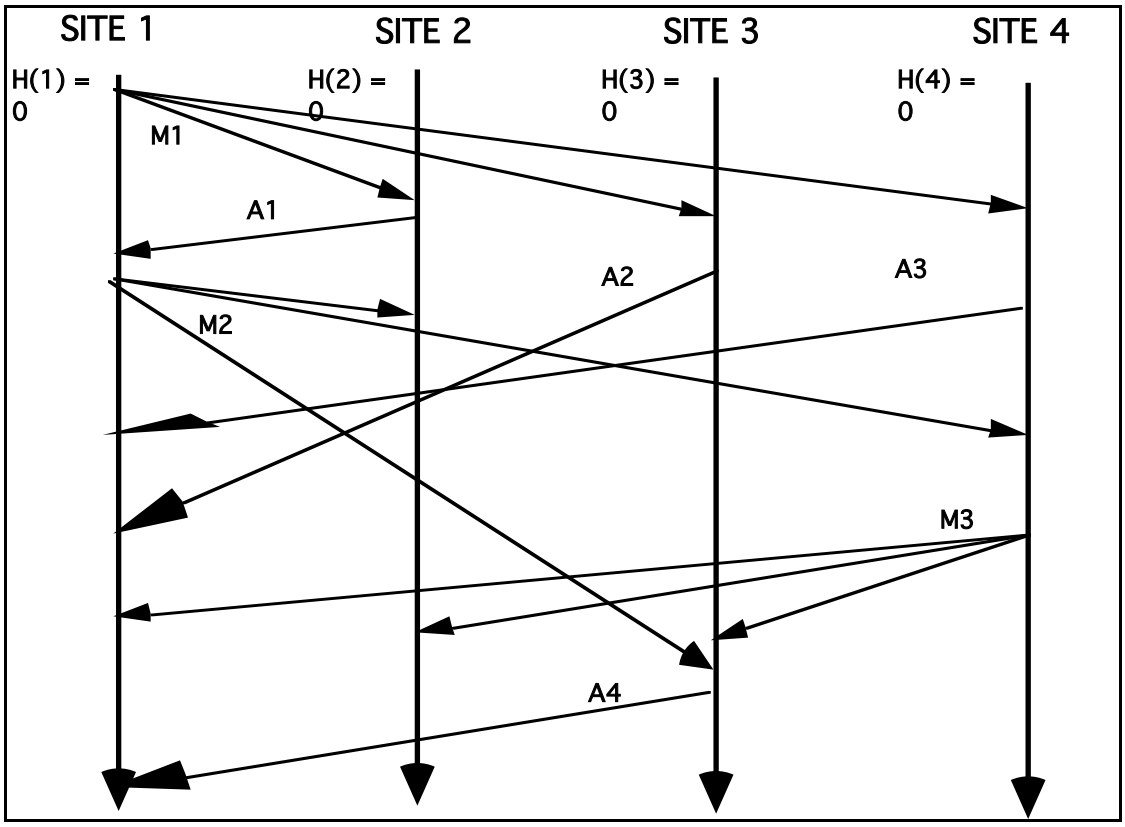
HORLOGES LOGIQUES A LA LAMPORT

exercice 11

Soit quatre sites qui diffusent des messages M1, M2, M3, ... ou qui répondent par des acquits A1, A2, A3, A4, ...Les acquits sont renvoyés à l'émetteur.

1 : A quelle heures locales des horloges logiques construites selon la règle de Lamport, le message M3 est-il envoyé par le site S4 et reçu par les sites S1, S2, S3 dans le fonctionnement ci-dessous?

2. Mettre des horloges vectorielles et délivrer les messages dans le respect de l'ordre causal



MISE A L'HEURE DES HORLOGES LOGIQUES

exercice 12

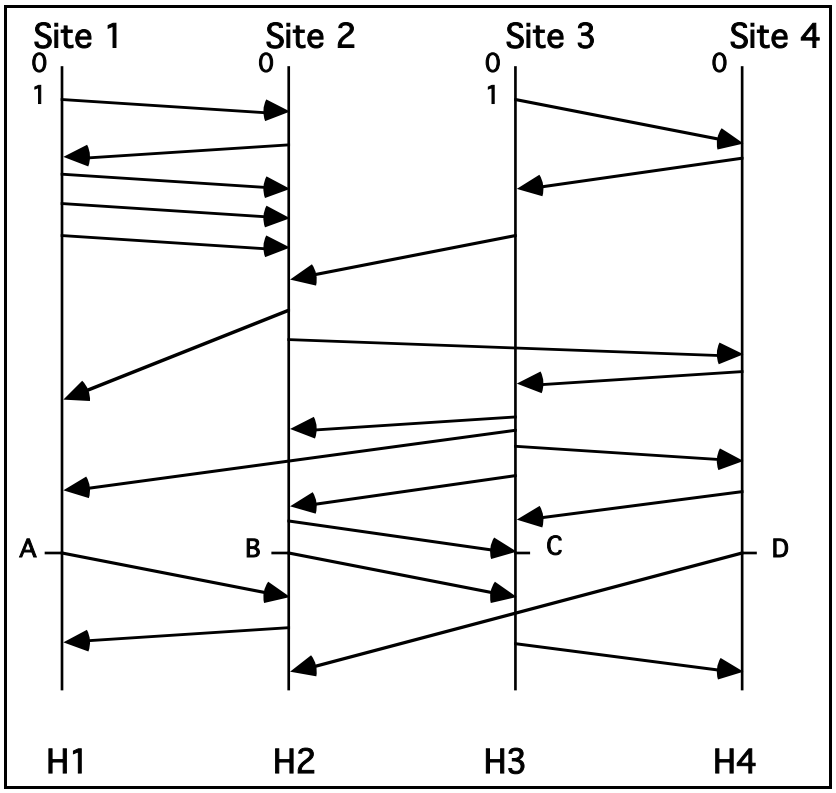
Un système qui comprend 4 sites a été l'objet des transferts de messages donnés dans la trace ci-dessous. On suppose que les horloges logiques de chaque site Hi étaient initialisées toutes à 0, que chaque message est estampillé selon la méthode de Lamport et que les horloges logiques ont été remises à l'heure pour respecter l'ordre total obtenu grâce à cet estampillage.

A). Dater tous les événements d'envoi et de réception de message qui apparaissent sur la trace.

B). Quelles sont les valeurs des horloges logiques de chacun des sites, à la fin de la trace.

C). En déduire une comparaison des événements A, B, C et D selon la relation d'ordre total => induit par l'estampillage à la Lamport.

D). Dater avec des horloges vectorielles. Quelle relation causale y a-t-il entre les événements A, B, C et D



solution de l'exercice 12. CORRIGE DES HORLOGES LOGIQUES

A la fin H1 = 24, H2 = 24, H3 = 23, H4 = 24

H1(A) = 18, H4(D) = 20, H2(B) = 21, H3(C) = 21 donc A => D => B => C

EXCLUSION MUTUELLE RÉPARTIE

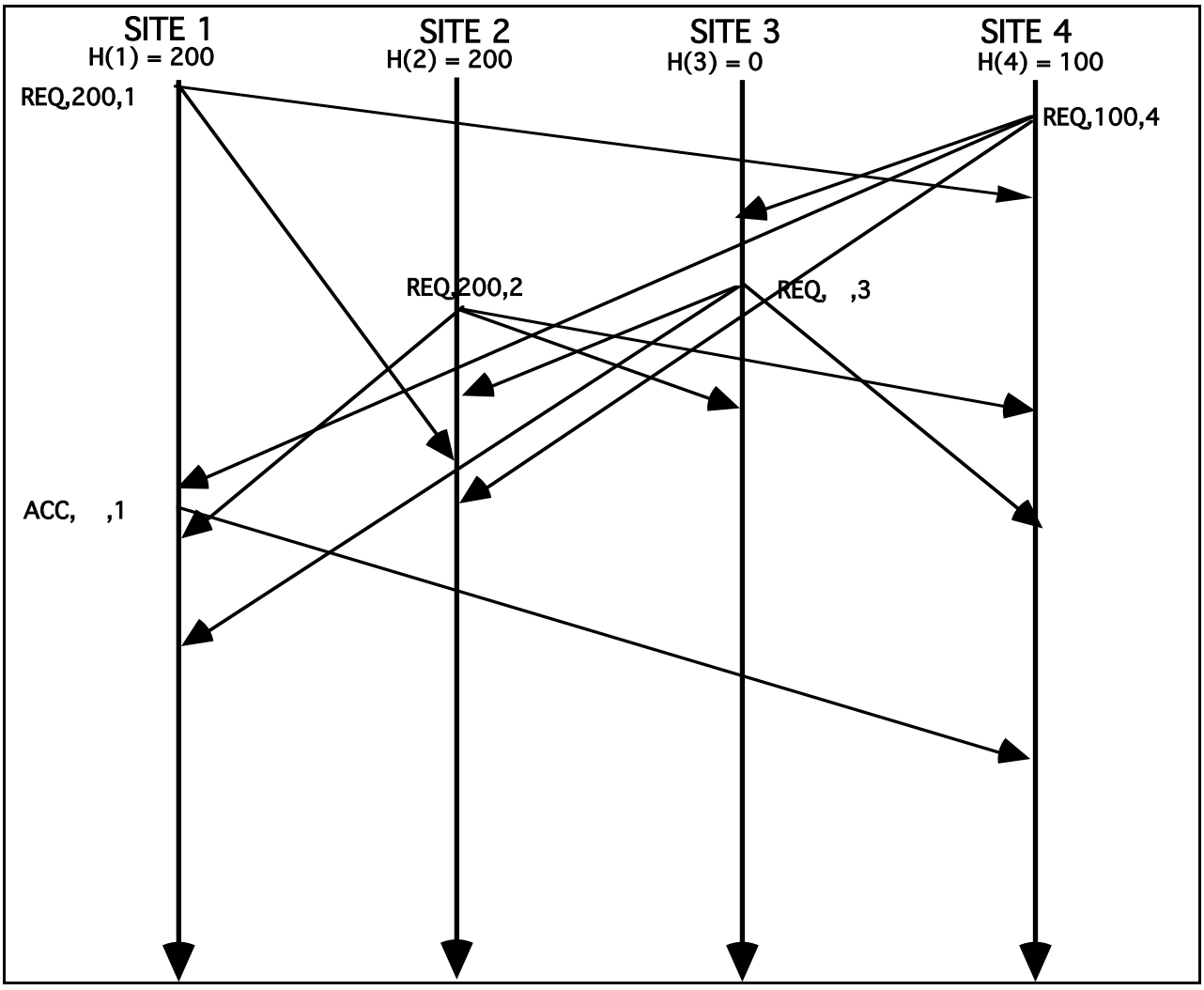
exercice 13

Les 4 sites envoient des demandes d'entrée en section critique comme indiqué ci-dessous. On vous demande de gérer cette situation :

1) avec l'algorithme de Lamport,

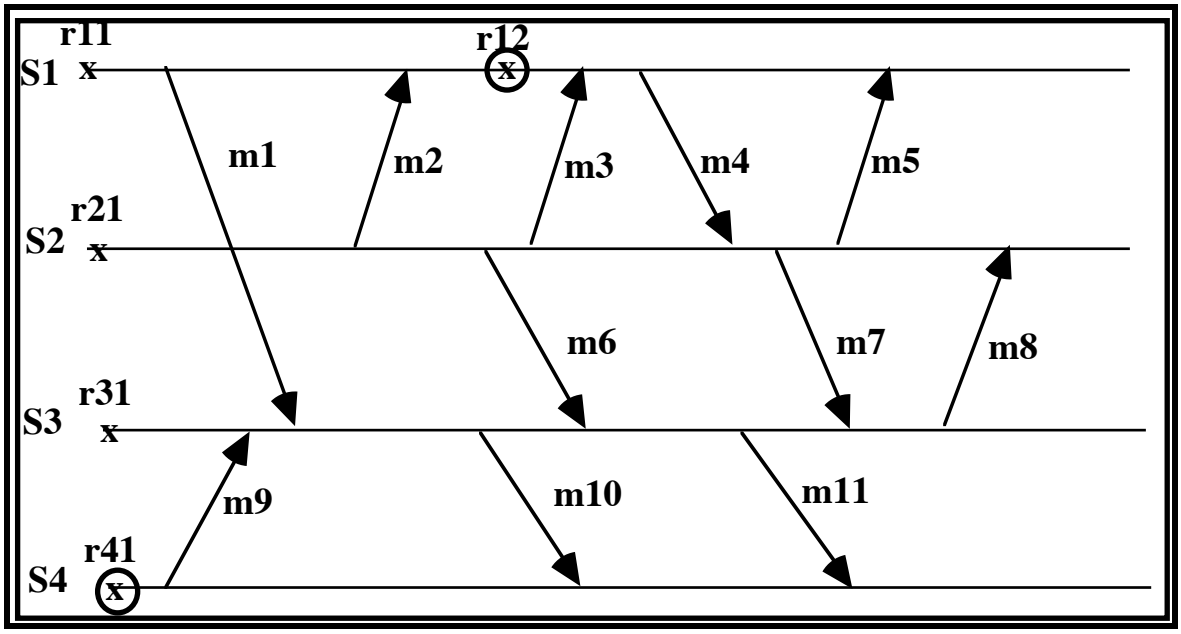
2) avec l'algorithme de Ricart et Agrawala

On suppose que les messages complémentaires pour chaque algorithme sont envoyés dès que c'est logiquement possible (Il n'y a pas d'attente locale et l'ordre local respecte l'ordre des demandes reçues. On rappelle que les canaux doivent être FIFO pour Lamport, mais non pour Ricart-Agrawala. On supposera pour ce dernier algorithme seulement que le canal C34 n'est pas FIFO et que l'ACC vers le site 4 est doublé par le message de demande REQ, ,3)

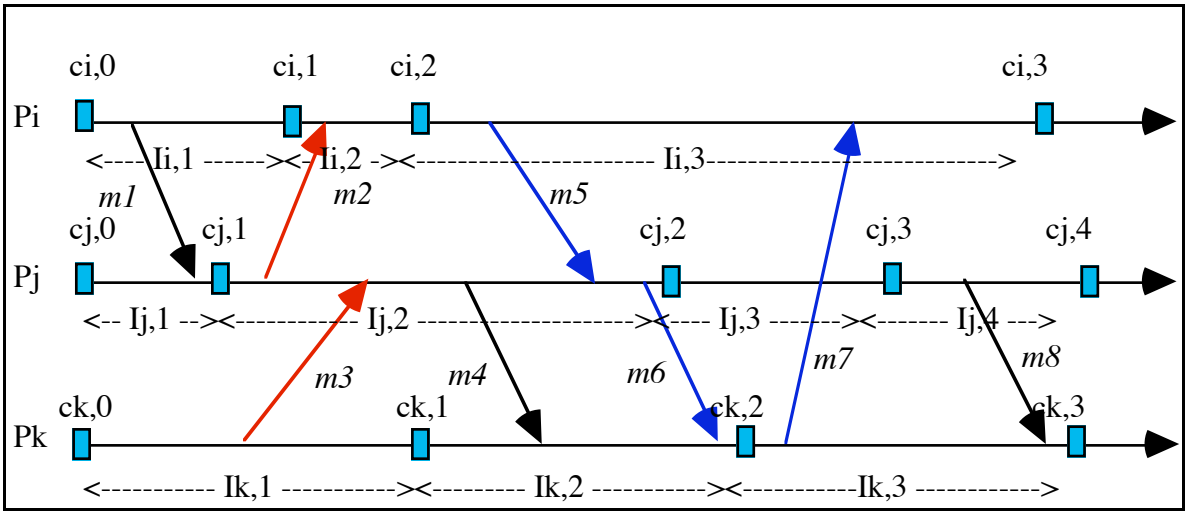


CHEMIN EN ZIGZAG

exercice 14.1 : montrer le chemin en zigzag de r41 à r12



exercice 14.2 rechercher les chemins et cycles en zigzag



solution de l'exercice 14.1 : m9, m8, m2

solution de l'exercice 14.2

ck,0 vers ci,2 : m3, m2
ci,2 vers ck,2 : m5, m4 ou encore m5, m6
ck,2 vers ck,2 : m7, m5, m6
ci,2 vers ci,2 : m5,m2

9. POINTS DE CONTRÔLE COHÉRENTS DANS LES SYSTÈMES RÉPARTIS ASYNCHRONES

• POINT DE CONTRÔLE LOCAL : état local de site

• POINT DE CONTRÔLE GLOBAL : ensemble de points de contrôle locaux, un par site, ligne de contrôle

INTÉRÊT

rétablissement après une défaillance : point de reprise, ligne de reprise

détection de propriétés d'une exécution répartie : état

NOTION DE COHÉRENCE

un point de contrôle global est cohérent s'il ne contient pas de point de contrôle local faisant état de la réception d'un message alors qu'aucun de ses autres points de contrôle locaux ne fait état de l'émission correspondante. (Chandy et Lamport 1985)

DÉTERMINATION DE POINTS DE CONTRÔLE COHÉRENTS

CONDITION NÉCESSAIRE : pas de liaison causale entre deux points de contrôles locaux
pas suffisant : existence de dépendances cachées (non captable par estampillage)

CONDITION NÉCESSAIRE ET SUFFISANTE : pas de chemin en zigzag (Z-chemin) entre deux points de contrôles locaux

THÉORÈME FONDAMENTAL : un ensemble quelconque de points de contrôle locaux peut être étendu pour former un point de contrôle global cohérent si, et seulement si, il n'y a pas de Z-chemin connectant deux points de contrôle de cet ensemble. (Netzer et Xu 1995)

ABSTRACTION D'UNE EXÉCUTION RÉPARTIE

• On ignore les états locaux qui ne sont pas des points de contrôle et on considère comme abstraction les points de contrôle locaux et leurs relations de dépendance.

CETTE ABSTRACTION EST-ELLE COHÉRENTE?

COHÉRENCE SELON NETZER ET XU

• L'abstraction est cohérente si tout point de contrôle local appartient à au moins un point de contrôle global cohérent.

Tout point de contrôle local est utile pour construire une ligne de contrôle (cela élimine l'effet domino).
Tout point de contrôle local est utile si et seulement aucun Z-chemin n'est un cycle (Netzer et Xu 1995)

En plus des points de contrôle spontanés pris à l'initiative de chaque site, il faut forcer des points de contrôle pour atteindre cette cohérence
??? => détermination de points de contrôle forcés pour garantir cette propriété de cohérence

COHÉRENCE SELON WANG

• L'abstraction est cohérente s'il n'y a aucune relation de dépendance cachée entre les points de contrôles locaux qui constituent cette abstraction. (propriété RDT : *Rollback dependency Trackability,* Wang 1997)

Cette propriété permet de calculer au vol le point de contrôle global minimal cohérent auquel appartient un point de contrôle local donné (utile pour la recherche d'erreurs logicielles, la validation des entrées-sorties, le redémarrage après interblocage, le calcul de points d'arrêts causaux distribués)

???=> détermination de points de contrôle forcés pour garantir cette propriété de cohérence

MODÈLE DE COMMUNICATION ASYNCHRONE
SYSTÈMES RÉPARTIS ASYNCHRONES

CHAQUE SITE

- processeur, mémoire locale
- simplification : à tout processeur est associé un et un seul processus

RÉSEAU DE COMMUNICATION

- canaux bi-points
- connexe : tout processus peut communiquer avec tous les autres
- communication et synchronisation entre processus par messages
  via le réseau de communication

PROPRIÉTÉS CARACTÉRISTIQUES

- Pas de mémoire commune
- Pas d'horloge physique globale partagée par 2 processus ou plus
- Absence d'hypothèse sur les vitesses relatives des processeurs et sur les durées de transfert des messages
  - Pas de majorant connu sur le temps de transfert des messages
  - Pas de minorant connu sur les vitesses des processeurs

Modélise le cas de réseaux à charge variable et qui ne peuvent être bornés a priori (Internet) et des réseaux longue distance (WAN)
Cette absence d'hypothèse permet d'obtenir des résultats généraux (qui s'appliquent aussi aux systèmes répartis synchrones)

EXÉCUTION RÉPARTIE : ORDRE PARTIEL D'ÉVÉNEMENTS

• La séquence d'instructions exécutée par un processeur est modélisée par une séquence d'événements ; celle-ci définit un processus.

• L'exécution répartie est composée de n processus séquentiels (ou sites) P1, P2,..., Pn où Pi est la séquence $e_{i,1}$ ... $e_{i,x}$ ... telle que

$e_{i,x}$ désigne le x-ième événement du processus Pi

Trois types d'événements :

- **émissions de messages, notées *send(m)***
- **réceptions de messages, notées *receive(m)***
- **événements internes (sans communication)**

Toute interaction d'un site avec un autre site constitue un événement
E : ensemble de tous les événements du modèle

ORDRE PARTIEL hb (Lamport 1978)

hb = (E, -e->) où -e-> est défini par :

$$e_{i,x} \text{ -e-> } e_{j,y} \iff \begin{cases} i = j \text{ et } x \le y \\ \text{ou } \exists\, m : e_{i,x} = send(m) \text{ et } e_{j,y} = receive(m) \\ \text{ou } \exists\, e_{k,z} : e_{i,x} \text{ -e-> } e_{k,z} \text{ et } e_{k,z} \text{ -e-> } e_{j,y} \end{cases}$$

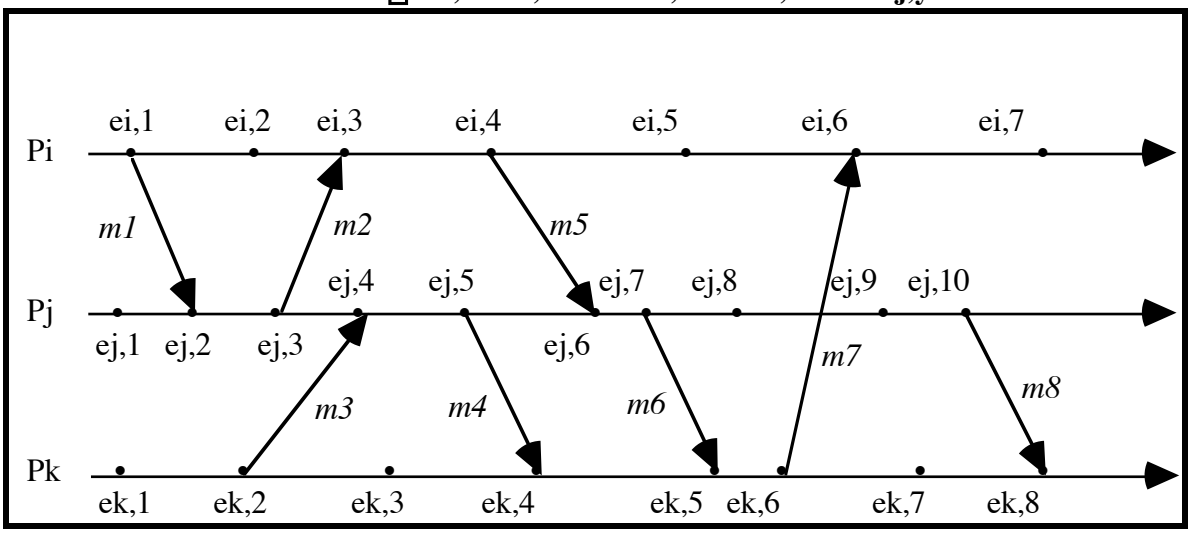


Fig.1. Exemple d'exécution répartie modélisée par des événements

hb : *happened before*, appelé aussi dépendance causale(potentielle)
nota : hb est antisymétrique ((a R b et b R a) => a = b), réflexive (a R a) et transitive (a R b et b R c => a R c). c'est donc une relation d'ordre

EXÉCUTION RÉPARTIE : RELATION ENTRE LES ÉTATS LOCAUX

Tout processus Pi a un état local initial $s_{i,0}$
L'état $s_{i,X}$ (x > 0) résulte de l'exécution de $e_{i,1}$ $e_{i,2}$ ... $e_{i,x}$, soit
$\{s_{i,0}\}$ $e_{i,1}$, $e_{i,2}$ ... ,$e_{i,x}$ $\{s_{i,x}\}$ ou encore $\{s_{i,x-1}\}$ $e_{i,x}$ $\{s_{i,x}\}$
Pi correspond à la séquence d'états $s_{i,0}$ $s_{i,1}$ ... $s_{i,x}$ ...

Par définition $e_{i,x}$ appartient à $s_{i,y}$ si x ≤ y
$e_{i,x}$ a été produit par Pi avant que ce processus n'atteigne $s_{i,y}$

STRUCTURE sb ENTRE LES ÉTATS LOCAUX

S : ensemble de tous les états locaux relatifs à une exécution répartie
sb = (S, -s->) où $s_{i,x}$ -s-> $s_{j,y}$ <=> $e_{i,x+1}$ -e-> $e_{j,y}$
la relation -s-> est antisymétrique, transitive, mais elle n'est pas réflexive ; ce n'est donc pas une relation d'ordre.

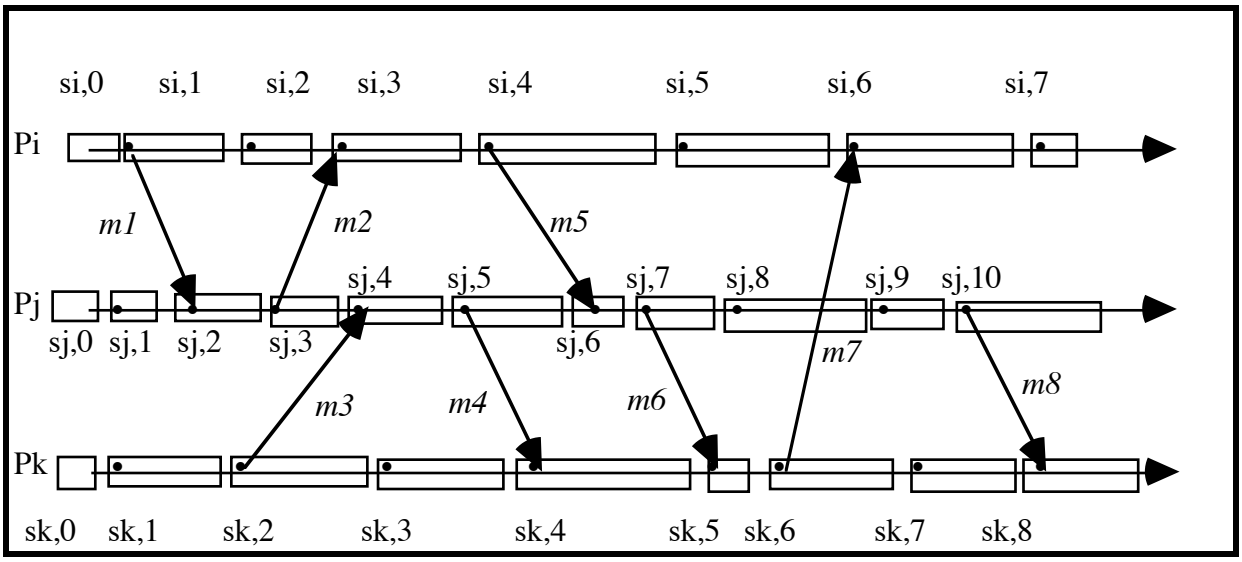


Fig.2. Exemple d'exécution répartie modélisée par des états locaux

État global cohérent : un état global, ensemble d'états locaux, un par processus, (s1, s2, ..., sn) est cohérent si ∀ i, j ¬(si -s-> sj)
(Chandy et Lamport 1985)
Un état cohérent ne contient aucun événement *receive(m)* sans contenir aussi l'événement *send(m)* correspondant.

($s_{i,2}$, $s_{j,2}$, $s_{k,2}$) est cohérent, mais ($s_{i,3}$, $s_{j,2}$, $s_{k,2}$) ne l'est pas

POINTS DE CONTRÔLE

Pour certains besoins (reprise pour recouvrement arrière,...), la modélisation distingue certains états locaux, qu'on appelle points de contrôle (ou points de reprise ou ligne de reprise). Les autres états locaux sont ignorés.

Le x-ième point de contrôle $c_{i,x}$ correspond à un état local $s_{i,y}$ où $x \leq y$
Par convention, $c_{i,0} = s_{i,0}$
Soit C l'ensemble des points de contrôle définis sur une exécution répartie S, on a $S \supseteq C$

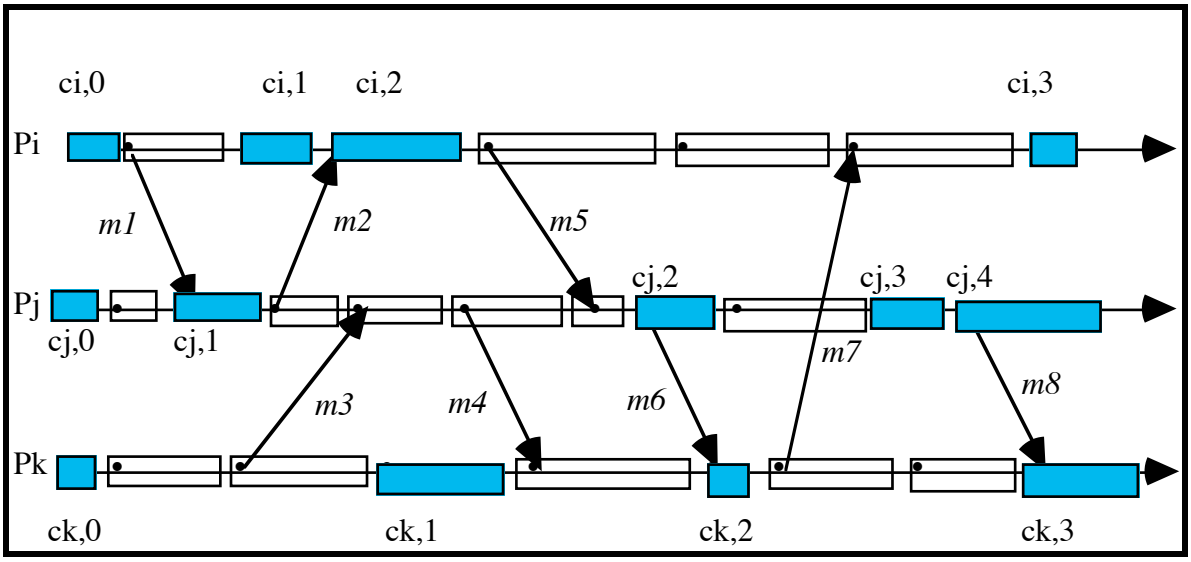


Fig.3. Exemple d'exécution répartie modélisée par des points de contrôle

Un point de contrôle global est un état global composé uniquement de points de contrôle locaux, un par processus.
Point de contrôle global cohérent : un point de contrôle global, ensemble de points de contrôle locaux, (c1, c2, ..., cn) est cohérent si c'est un état global cohérent donc si : ∀ i, j ¬(ci -s-> cj)

$(c_{i,1}, c_{j,1}, c_{k,1})$ est cohérent, mais $(c_{i,2}, c_{j,2}, c_{k,1})$ ne l'est pas.

ABSTRACTION zb D'UNE EXÉCUTION RÉPARTIE PAR INTERVALLES ET POINTS DE CONTRÔLE

INTERVALLE : séquence d'événements entre deux points de contrôle successifs $c_{i,x-1}$ et $c_{i,x}$; macroévénement : $\{c_{i,x-1}\}\ I_{i,x}\ \{c_{i,x}\}$

Comme -e-> sur les événements, on définit -i-> sur les intervalles.
soit I ensemble des intervalles d'une exécution répartie avec points de contrôle

$$I_{i,x} \text{ -i-> } I_{j,y} \iff \begin{cases} i = j \text{ et } x \le y \\ \text{ou } \exists m : \textit{send}(m) \in I_{i,x} \text{ et } \textit{receive}(m) \in I_{j,y} \\ \text{ou } \exists I_{k,z} : I_{i,x} \text{ -i-> } I_{k,z} \text{ et } I_{k,z} \text{ -i-> } I_{j,y} \end{cases}$$

cb = (I, -i->) constitue une abstraction de hb, relative à l'ensemble C des points de contrôle (ou des points de reprise).
C'est une relation d'ordre, car antisymétrique , réflexive et transitive

De même que -s-> a été définie à partir de -e->, pour les états locaux, on définit -z-> à partir de -i->, pour les points de contrôle (de reprise)
zb = (C, -z->) où $c_{i,x}$ -z-> $c_{j,y} \iff I_{i,x+1}$ -i-> $I_{j,y}$

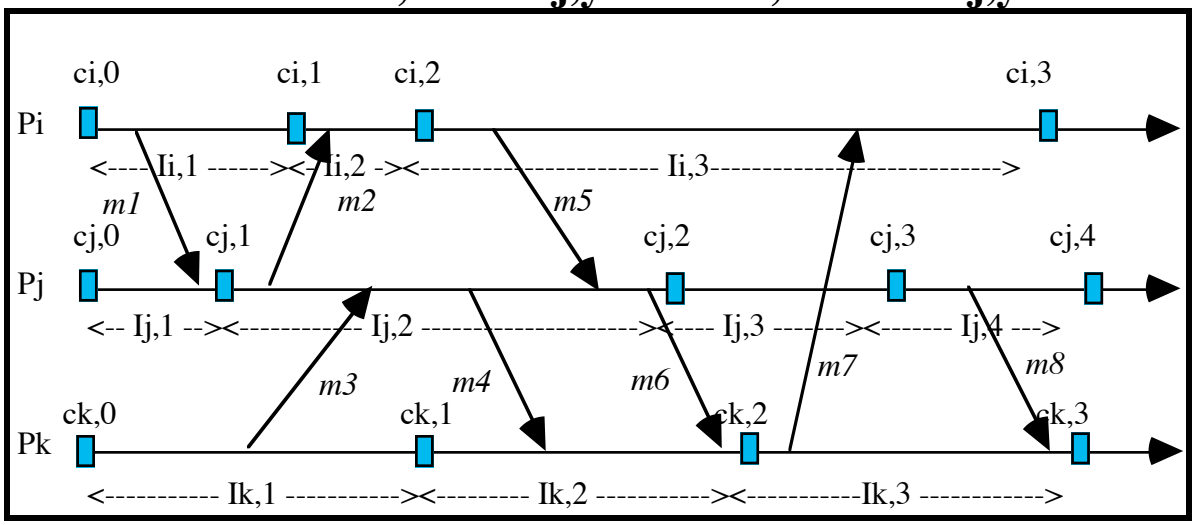


Fig.4. Exécution modélisée par des points de contrôle et des intervalles

zb n'est ni réflexive, ni antisymétrique, mais elle est transitive.
Par ailleurs $c_{i,x}$ -s-> $c_{j,y}$ => $c_{i,x}$ -z-> $c_{j,y}$

Ex : $I_{k,1}$ -i-> $I_{j,2}$ et $I_{j,2}$ -i-> $I_{i,2}$ donc $I_{k,1}$ -i-> $I_{i,2}$ donc $c_{k,0}$ -z-> $c_{i,2}$
mais on a $\neg(c_{k,0}$ -s-> $c_{i,2})$ **; pas de liaison causale entre** $c_{k,0}$ **et** $c_{i,2}$

Question fondamentale : quelle est la cohérence d'une telle abstraction?

Z-CHEMINS ET Z-CYCLES
(chemins en zigzag et cycles en zigzag)

(Netzer et Xu 1995) :

• Il y a un Z-chemin, ou chemin en zigzag, de $c_{i,x}$ à $c_{j,y}$ si

(1) i = j et x < y, les deux points sont sur le même processus

ou s'il existe une séquence de messages $m_1, m_2, ..., m_q$ ($q \geq 1$) telle que

(2.1) m_1 est envoyé par le site Pi après $c_{i,x}$

(2.2) si m_k ($1 \leq k < q$) est reçu par le site P_r, alors m_{k+1} est envoyé par P_r dans le même intervalle de reprise ou dans un intervalle postérieur (noter que m_{k+1} peut être envoyé avant ou après l'arrivée de m_k),

(2.3) m_q est reçu par le site Py avant $c_{j,y}$

• Il existe un Z-chemin de $c_{i,x}$ à $c_{j,y}$ si et seulement si $c_{i,x}$ -z-> $c_{j,y}$.

• Les Z-chemins donnent une vision opérationnelle de la relation -z->.

• Dans le cas particulier où, pour tout message m_k d'un Z-chemin, on a receive(m_k) -e-> send(m_{k+1}), le Z-chemin est causal.
Il existe un chemin causal de $c_{i,x}$ à $c_{j,y}$ si et seulement si $c_{i,x}$ -s-> $c_{j,y}$.

Il existe des Z-chemins qui ne sont pas causaux. Ceci est conforme au fait que la relation -z-> inclut la relation -s->

<u>Théorème</u> <u>fondamental</u> : Soit M un ensemble de points de contrôle locaux, M peut être complété pour former un point de contrôle global cohérent si, et seulement si, ∄ a, b ∈ M : ¬(a -z-> b).

• Cas particulier : M ={a}; le point a fait partie d'au moins un point de contrôle global cohérent si, et seulement si, ¬(a -z-> a).
Si a -z-> a, on dit que a appartient à un Z-cycle et a ne peut appartenir à aucun point de contrôle global cohérent ; il est donc inutile.

EXEMPLES DE Z-CHEMINS

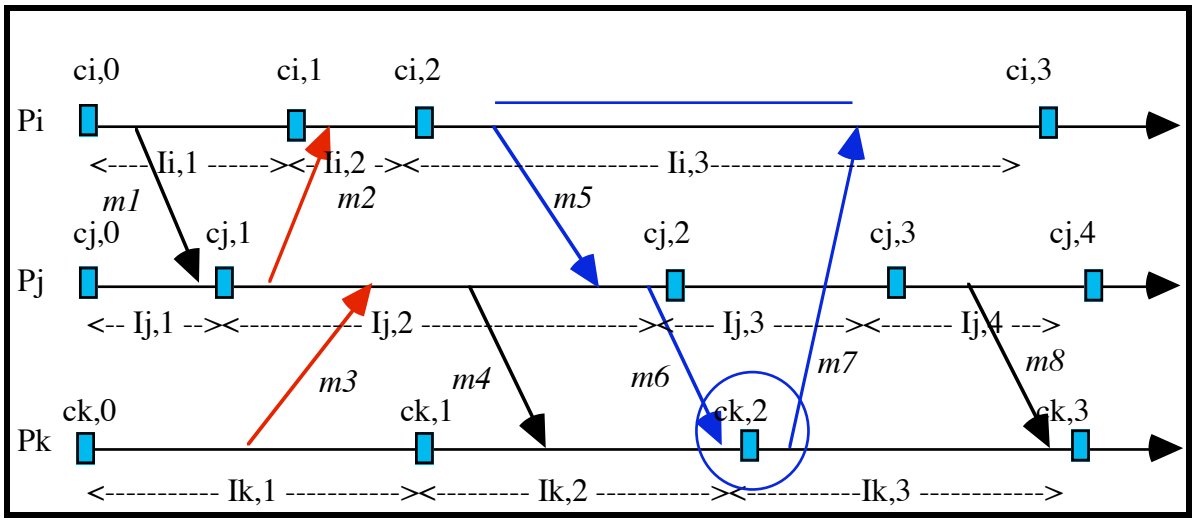


Fig.5. Z-chemins dans l'abstraction d'exécution

[m3,m2] est un Z-chemin de $c_{k,0}$ vers $c_{i,2}$;
[m5, m4] et [m5, m6] sont deux Z-chemins de $c_{i,2}$ vers $c_{k,2}$.

Le Z-chemin [m5, m6] est causal alors que [m5, m4] ne l'est pas.

application du théorème fondamental

- **Soit M = {$c_{k,0}$, $c_{i,2}$},**
  comme $c_{k,0}$ -z-> $c_{i,2}$, quel que soit le point de contrôle $c_{j,p}$ de Pj, {$c_{k,0}$, $c_{j,p}$, $c_{i,2}$} ne peut être cohérent.

  Par contre {$c_{i,1}$, $c_{j,1}$} peut être étendu par $c_{k,0}$ ou $c_{k,1}$.

- **Soit M = {$c_{k,2}$}.**

Le Z-chemin [m7, m5, m6] est un Z-cycle de $c_{k,2}$ sur lui-même.
Le point de contrôle local $c_{k,2}$ est donc inutile du point de vue de la détermination des points de contrôle globaux cohérents.

DÉFINITION DES POINTS DE CONTRÔLE

OBJECTIF

- **points de reprise permettant de relancer une exécution**
- **calcul de propriétés définies sur des états globaux**

CONTEXTE

• Tout processus a la possibilité de définir certains de ses états locaux comme des points de contrôle, pour des raisons qui lui sont propres :

occurrence d'une échéance éventuellement périodique,
réception d'un signal particulier,
passage à vrai d'un prédicat local

- **l'abstraction (C, -z->) doit satisfaire un critère de cohérence**

- **Si les processus définissent leurs points de contrôle indépendamment les uns des autres, il est possible qu'il n'y ait pas cohérence**

- **d'où une coordination est nécessaire entre processus :**

Les processus définissent des états locaux comme *points de contrôle spontanés*

Un mécanisme de coordination les oblige à définir d'autres états locaux comme *points de contrôle forcés*

• Nota : il reste des états locaux qui ne sont pas des points de contrôle

CLASSES DE PROTOCOLES

- ***protocoles à synchronisation explicite* : la coordination utilise des messages de contrôle qui sont ajoutés à ceux de l'application**

• *protocoles à synchronisation implicite* : la coordination utilise les messages de l'application pour véhiculer des informations de contrôle

PRÉVENTION DES Z-CYCLES

CRITÈRE DE COHÉRENCE

(P1) Tout point de contrôle local appartient à au moins un point de contrôle global cohérent

- **alors il n'existe pas de point de contrôle local inutile**

(P2) Tout point de contrôle local est associé, lors même de sa définition, à un des points de contrôle cohérents auxquels il appartient

- **on associe au vol une identité globale à chaque point local**

(P2) permet une reconstruction et une exploitation aisée, utile dans le cas d'une ligne de reprise

PROPRIÉTÉ FONDAMENTALE

- **Si on peut associer à tout point de contrôle $c_{i,x}$ une estampille notée $c_{i,x}.t$ (avec $c_{i,x}.t \in N$), telle que**

  $(\forall c_{i,x}, c_{j,y} : (c_{i,x} \text{ -z-> } c_{j,y} \Rightarrow c_{i,x}.t < c_{j,y}.t))$

alors, théorème : il n'y a pas de Z-cycle

preuve :

- **=> Supposons qu'il existe un Z-cycle de $c_{i,x}$ à $c_{i,x}$. On a alors $c_{i,x} \text{ -z-> } c_{i,x}$. On devrait avoir $c_{i,x}.t < c_{i,x}.t$, c'est impossible.**

- **<= Supposons qu'il n'y ait pas de Z-cycles et considérons le graphe orienté suivant :**

  les sommets : les points de contrôle locaux

  les arcs : les couples ($c_{i,x}$, $c_{j,y}$) tels que $c_{i,x} \text{ -z-> } c_{j,y}$

comme, il n'y a pas de Z-cycle, ce graphe est partiellement ordonné et il existe un tri topologique de ce graphe qui fournit la propriété d'estampillage requise.

conséquence :

- **Tous les points de contrôle locaux qui ont même estampille appartiennent au même point de contrôle global cohérent.**

PROTOCOLE À SYNCHRONISATION EXPLICITE

MARQUEUR

• On utilise des messages de contrôle appelés marqueurs
• Tout marqueur mk possède la propriété suivante : sur le canal où il est émis, tous les messages émis avant mk sont délivrés avant mk et tous les messages émis après mk sont délivrés après mk.

PROTOCOLE

soit lc_i l'estampille du dernier point de contrôle local de Pi

- (R1) Point de contrôle spontané : chaque fois que Pi prend un tel point de contrôle, il incrémente lc_i de 1 , attribue la valeur de lc_i à $c_{i,x}.t$ et envoie des marqueurs $mk(lc_i)$ à chacun des processus

- (R2) Point de contrôle forcé : lorsque Pi reçoit $mk(lc)$, si $lc_i < lc$ il prend un point de contrôle forcé $c_{i,x}$, recale lc_i ($lc_i := lc$) et attribue la valeur de lc à $c_{i,x}.t$; de plus il diffuse à son tour des marqueurs comme dans le cas R1

Ce protocole satisfait le théorème précédent, il n'y a pas de point de contrôle inutile (critère P1).
Tous les points de contrôle locaux de même estampille définissent un point de contrôle global cohérent (critère P2)

Le coût en nombre de messages est proportionnel au nombre de canaux.

Ce protocole dérive directement de celui de Chandy Lamport 1985

PROTOCOLE À SYNCHRONISATION IMPLICITE

PRINCIPE

faire transporter les estampilles par les messages de l'application

PROTOCOLE

- (R1) Point de contrôle spontané : chaque fois que Pi prend un tel point de contrôle, il incrémente lc_i et attribue la valeur de lc_i à $c_{i,x}.t$

- (R2) Point de contrôle forcé :

• Tout message *m* transporte la valeur courante lc_j de son émetteur Pj (soit *m.lc* cette valeur)

• lors de la réception d'un message *m* , Pi test si $lc_i < m.lc$. Si tel est le cas, Pi recale lc_i à la valeur de *m.lc* et définit l'état courant comme point de contrôle $c_{i,x}$ (avec $c_{i,x}.t := lc_i$)

P1 est satisfait : il n'y a pas de Z-cycle

P2 se règle comme suit, pour associer un point de contrôle global unique Ga = (c1, c2, ..., cn) à une valeur d'estampille a :

- s'il existe $c_{i,x}$ tel que $c_{i,x}.t = a$ alors ci = $c_{i,x}$

- dans le cas contraire ci est le point de contrôle local $c_{i,x}$ de Pi dont l'estampille $c_{i,x}.t$ est la plus grande parmi toutes celles qui sont inférieures à a

Ce protocole a été introduit par [Briatico 84], voir aussi [Hélary 97]

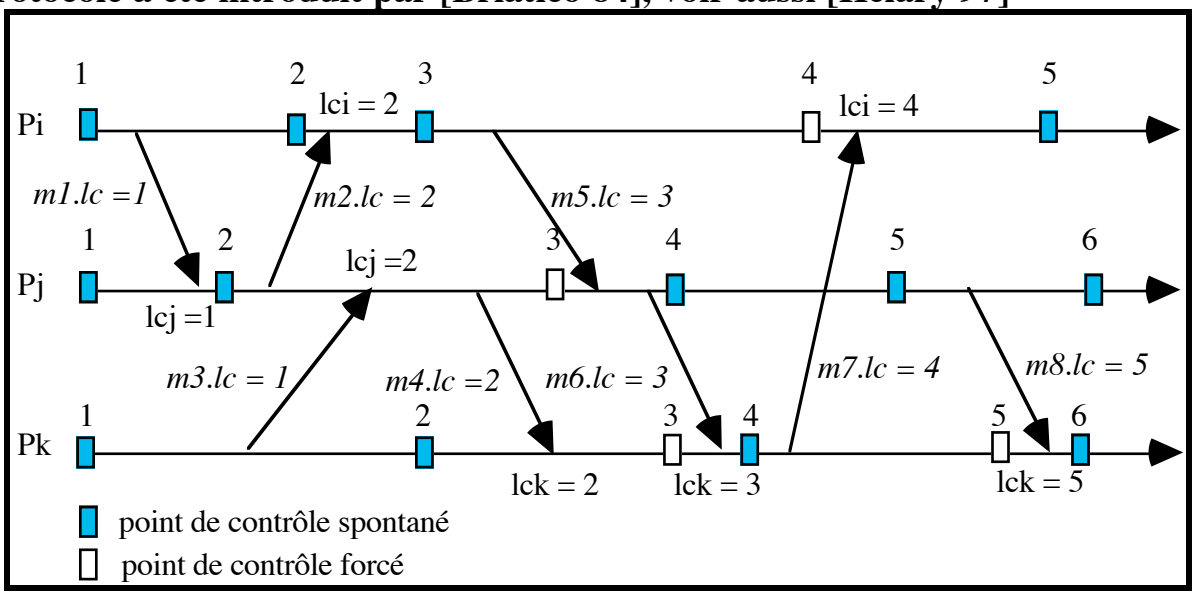


Fig.6. Points de contrôle forcés par synchronisation implicite simple

Problème : comment réduire le nombre des points de contrôle forcés?

PRÉVENTION DES Z-CHEMINS NON CAUSAUX

objectif

• calculer au vol le premier point de contrôle global cohérent

• impossible si dépendance cachée avec un Z-chemin non causal

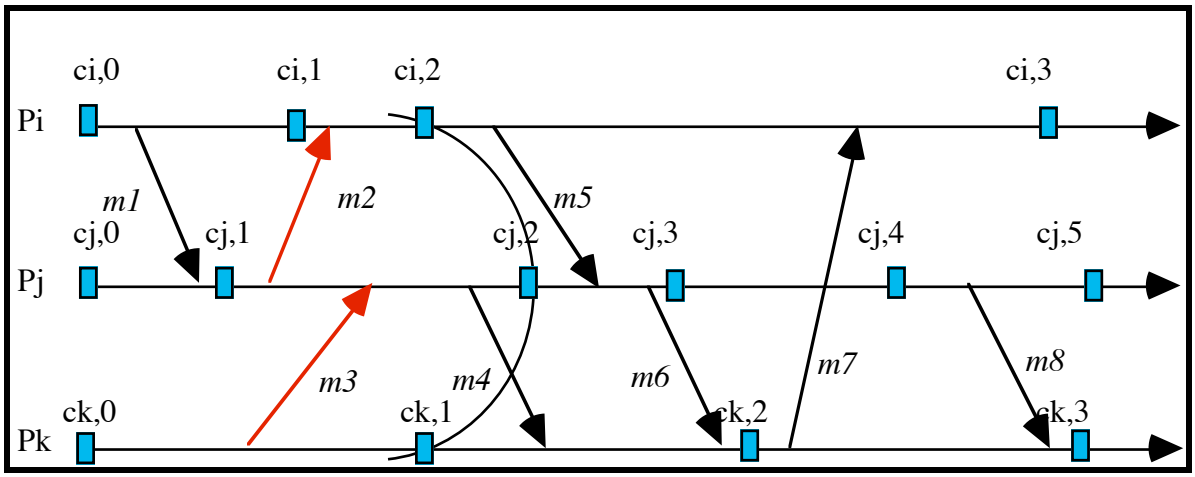


Fig.7. Point de contrôle global cohérent contenant $c_{i,2}$

Exemple :
Essayons de calculer au vol, à la définition de $c_{i,2}$, le premier point de contrôle global cohérent qui le contienne :
• ayant reçu *m2* (Z-chemin causal de longueur 1), Pi est informé que $c_{i,2}$ ne peut être cohérent avec $c_{j,1}$, mais seulement avec $c_{j,y}$ pour y > 1
• Pi n'a aucune relation causale avec Pk, et il ne peut être au courant du Z-chemin [*m3, m2*] de $c_{k,0}$ vers $c_{i,2}$ et ne sait pas qu'il doit exclure $c_{k,0}$ et considérer le point local qui le suit.

• Heuristique possible en doublant tout Z-chemin non causal par un Z-chemin causal (car alors la propriété RDT existe)

ABSTRACTION SATISFAISANT LA PROPRIÉTÉ RDT

Propriété RDT (*Rollback dependency Trackability*, Wang 1997)
(C, -z->) satisfait la propriété RDT si
$\forall$ $c_{i,x}$, $c_{j,y}$: $c_{i,x}$ -z-> $c_{j,y}$ => $c_{i,x}$ -s-> $c_{j,y}$

(C, -z->) satisfait la propriété RDT si chaque Z-chemin, soit est un chemin causal, soit est doublé par un chemin causal

Dans la figure 4, le Z-chemin [m5, m4] qui connecte $c_{i,2}$ à $c_{k,2}$ est doublé par le Z-chemin causal [m5, m6]

DÉTECTION DES DÉPENDANCES CAUSALES TRANSITIVES (horloge vectorielle)

• Chaque processus Pi est doté d'un vecteur VDT (vecteur des dépendances transitives ou horloge vectorielle)

- (V1) Pi, $VDT_i[i]$ initialisé à 1 ; les autres entrées à 0

- (V2) chaque fois que Pi définit un point de contrôle $c_{i,x}$, il exécute $VDT_i[i] := VDT_i[i] + 1$. La nouvelle valeur est l'estampille vectorielle de $c_{i,x}$

- (V3) tout message *m* transporte la valeur du vecteur de son émetteur soit m.VDT

- (V4) lorsque Pi reçoit un message *m*, il met à jour son vecteur de dépendances ; $VDT_i[k] := \max(VDT_i[k], m.VDT[k])$

• Et alors : $c_{i,x}$ -s-> $c_{j,y}$ <=> $VDT_{j,y}[i] \geq VDT_{i,x}[i]$

La propriété RDT devient : $c_{i,x}$ -z-> $c_{j,y}$ <=> $VDT_{j,y}[i] \geq VDT_{i,x}[i]$

RÉSULTAT

Dans ce contexte d'une abstraction satisfaisant la propriété RDT,
• le théorème fondamental devient : tout ensemble M de points de contrôle locaux qui ne sont pas causalement liés peut être étendu pour former un point de contrôle global cohérent
• le premier point de contrôle global cohérent auquel peut appartenir $c_{i,x}$ est défini par : $(c_{1,x1}, ..., c_{n,xn})$ tel que

$k \neq i$: $xk = VDT_{i,x}[k]+1$ et
$xi = VDT_{i,x}[i]$

STRATÉGIE DE RUSSELL

Une façon de garantir la propriété RDT consiste, par ajout de point de contrôle forcés, à éliminer tous les Z-chemins

Soit *ckpt* le méta-événement qui consiste à définir un état local comme point de contrôle local et faisons abstraction des événements autres que *send, receive et ckpt.*
Si chaque processus obéit au comportement

(*receive* send* ckpt*)*

il ne peut pas y avoir de Z-chemin non causal

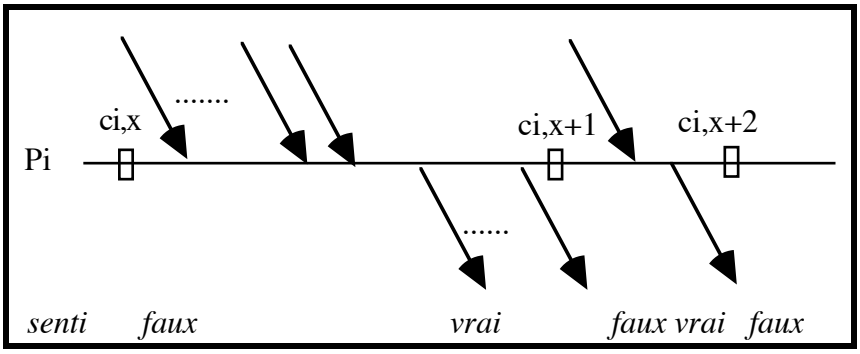


Fig.8. stratégie de Russell (1980)

PROTOCOLE DE RUSSELL

Chaque processus Pi est doté d'une variable booléenne $sent_i$ qui est initialisée à *faux* au début de chaque intervalle et qui est mise à *vrai* lorsque Pi envoie un message
-(R2) Point de contrôle forcé : à la réception d'un message , Pi prend un point de contrôle si la condition $sent_i$ est vraie (et ensuite remet $sent_i$ à *faux*)

-(R1) n'appelle pas de traitement particulier si on veut seulement garantir l'absence de Z-chemin

Si on veut associer à tout point de contrôle local, le premier point de contrôle global cohérent, il faut gérer les vecteurs VDTi et exécuter V2 dans R1 (lors de la prise de points de contrôle spontanés) et V2 et V4 dans R2 (lors de la prise de points de contrôle forcés)

STRATÉGIE FDAS

Pour obtenir la propriété RDT, il n'est pas nécessaire d'éliminer tous les Z-chemins non causaux, comme dans la stratégie de Russell, mais il suffit de prévenir les Z-chemins non causaux qui ne sont pas doublés par des chemins causaux.

STRATÉGIE FDAS
(*Fixed Dependency After Send* de Wang 1997)

Chaque processus est doté du booléen $sent_i$ et d'un vecteur VDTi

$sent_i$ est initialisée à *faux* après chaque prise de point de contrôle local, spontané ou forcé, et est mise à *vrai* lorsque Pi envoie un message

- (R1) Point de contrôle spontané : lorsque Pi définit $c_{i,x}$, il exécute au préalable l'énoncé V2 de gestion de VDT_i
(i.e. : il exécute $VDT_i[i] = VDT_i[i] + 1$ et prend la nouvelle valeur comme estampille vectorielle de $c_{i,x}$)

-(R2) Point de contrôle forcé : à la réception d'un message, Pi prend un point de contrôle si la condition suivante est vraie :

$$sent_i \text{ et } \exists k : m.VDT[k] > VDT_i[k]$$

Si $\neg sent_i$, alors le message *m* ne peut participer à un Z-chemin non causal (il n'y a pas de message parti de Pi dans le même intervalle et avant m).

Si $\exists k : m.VDT[k] \leq VDT_i[k]$, tous les points de contrôle définis avant l'envoi du message *m* (ils sont révélés à Pi par m.VDT) sont connus de Pi (dans son vecteur VDTi). Le message *m* n'apporte à Pi aucune nouvelle information sur les dépendances et ne peut créer une dépendance qui serait cachée à Pi.

Cette stratégie est suffisante, mais n'est pas optimale. Elle peut obliger les processus à définir plus de points de contrôle forcés que nécessaire.

BIBLIOGRAPHIE

[Hélary 98] J.M.HÉLARY, A.MOSTÉFAOUI, et M.RAYNAL. Points de contrôle cohérents dans les systèmes répartis : concepts et protocoles. ***TSI (Technique et science informatique)*** **vol. 17(10), p.1223-1245, octobre 1998**

BIBLIOGRAPHIE COMPLÉMENTAIRE

[Briatico 84] D. BRIATICO, A.CIUFFOLETTI and L.A.SIMONCINI. Distributed Domino-Effect Free Recovery Algorithm. ***4th IEEE Symp. on Reliability in Distributed Software and data base Systems*****, p. 207-215, 1984**

[Chandy 85] K.M. CHANDY and L.LAMPORT, Distributed Snapshots : Determining Global States of Distributed Systems. ***ACM TOCS*****, Vol. 3,1, p. 63-75 1985**

[Hélary 97] J.M.HÉLARY, A.MOSTÉFAOUI, R.H.B.NETZER and M.RAYNAL. Communication-Based Prevention of Useless Checkpoints in Distributed Computations. ***16th IEEE Symp. on Reliable Distributed Systems*****, p. 183-190, 1997**

[Lamport 78] L.LAMPORT. Time, Clocks and the Ordering of Events in a Distributed System. Communications of the ACM, 21(7), p. 558-565, 1978

[Netzer 95] R.NETZER, J. XU. Necessary and Sufficient Conditions for Consistant Global Snapshot, ***IEEE Transactions on Parallel and Distributed Systems,*** **6(2), p.165-169, 1995**

[Russell 80] D.L.RUSSELL. State Restoration in Systems of Communicating Processes. ***IEEE Transactions on Software Engineering*****, p. 183-194, 1980**

[Wang 97] Y.M.WANG. Consistent Global Checkpoints that Contain a Given Set of Local Checkpoints. ***IEEE Transactions on Computers*****, 46(4), p.456-468, 1997**

10. DIFFUSION ORDONNÉE RESPECTANT UN ORDRE TOTAL DIFFUSION PAR ÉCHANGE DISTRIBUÉ

Exemple d'utilisation de la diffusion ordonnée

Cohérence de caches (technique de diffusion des écritures)

Cohérence de copies multiples.

Publish-subscribe systems (Gryphon)

Le fait que toutes les opérations de modifications d'un ensemble de données en copies multiples soient effectuées dans le même ordre sur toutes les copies suffit à assurer le maintien de la cohérence (faible) des copies.

HYPOTHÈSES CONCERNANT LA COUCHE RÉSEAU SOUS-JACENTE

• le réseau est un graphe G connexe non orienté avec service canal FIFO fiable (point à point) : ex. TCP
• service de diffusion fiable FIFO, respectant l'ordre local d'émission (pas ordonné totalement) : ex. SRM, RMTP
• pas de pannes de site et pas de perte de messages (avec suffisamment de retransmissions, tout message finit par arriver)

COMPORTEMENTS

• l'émetteur ne connaît pas l'identité des membres du groupe de diffusion : diffusion ordonnée anonyme (≠ "virtual synchrony" avec vue de groupe)
• l'ordre total concerne la délivrance des messages, on distingue donc :
•• recevoir un message = placer ce message dans un tampon du site
•• délivrer un message à une application s'exécutant sur le site

DIFFUSION PAR ÉCHANGE DISTRIBUÉ ("distributed swap")

• On crée un objet o d'échange distribué qui encapsule une valeur (initialement ‡ ou nil) et qui fournit une méthode swap(v) qui donne à l'objet la valeur v et reçoit en échange l'ancienne valeur de l'objet.
• Chaque groupe de diffusion crée un objet o d'échange distribué.

• Un site v qui veut diffuser un message m connu par son identificateur unique id(m) appelle o.swap(id(m)) et reçoit id(m'), l'identificateur du message m', prédécesseur immédiat de m. Le site v alors diffuse avec le service de diffusion non ordonnée le message <id(m'), m)>. Tout site qui reçoit ce message ne délivre m qu'après avoir délivré m'.
Notons que les identificateurs uniques id(m) ne sont pas ordonnés; chacun peut être obtenu par la concaténation d'un identificateur de site et d'un identificateur unique sur le site.

ÉCHANGE DISTRIBUÉ PAR ARBRE COUVRANT "arrow multicast"

L'objet réparti d'échange distribué est un arbre couvrant T (qui est un sous graphe de G) et on utilise une technique d'inversion de chemin. Chaque noeud u de l'arbre a deux attributs :
• lien(u) désigne un noeud, soit u lui-même, soit un voisin de u dans T
• valeur(u) qui contient un identificateur de message, ou nil ‡
• un noeud u est une racine quand lien(u) = u
L'arbre couvrant est initialisé de façon à ce que tous les chemins formés par les liens conduisent à une racine unique. (voir figure 1)

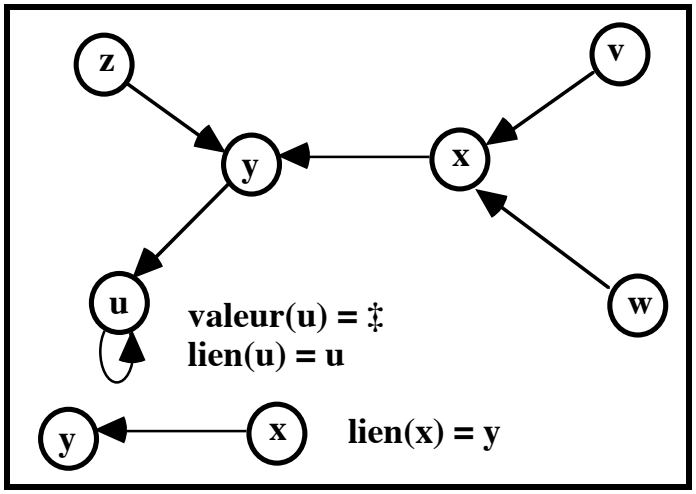


Figure 1. ***Arbre couvrant initial avec u comme racine.***

Quand un noeud v veut diffuser un message m1 connu par son identificateur unique id(m1), il initialise l'exécution répartie de swap(id(m1)) en envoyant < m1, swap(id(m1)) > à son voisin lien(v). Il se note comme racine future par lien(v) = v et il note valeur(v) = id(m1), nouvelle valeur de l'objet d'échange distribué. Pour retrouver l'ancienne valeur de l'objet d'échange distribué, le message < m1, swap(id(m1)) > doit être propagé jusqu'à l'ancienne racine.

Quand s un noeud qui n'est pas racine, reçoit le message < m1, swap(id(m1)) > envoyé par r, il fait suivre le message vers lien(s) = t et modifie lien(s) en lien(s) = r, pour le faire indiquer le chemin vers la nouvelle racine.
Quand s est une racine, donc lien(s) = s, valeur(s) fournit l'ancienne valeur de l'objet d'échange distribué, à savoir id(m'), l'identificateur du message m', prédécesseur immédiat de m1. L'exécution répartie de l'opération swap(id(m1)) est terminée. Le site s, et non le site initiateur v (on gagne ainsi un message au moins), diffuse alors avec le service de diffusion non ordonnée le message <id(m'), m)>. Tout site qui reçoit ce message ne délivre m qu'après avoir délivré m'.

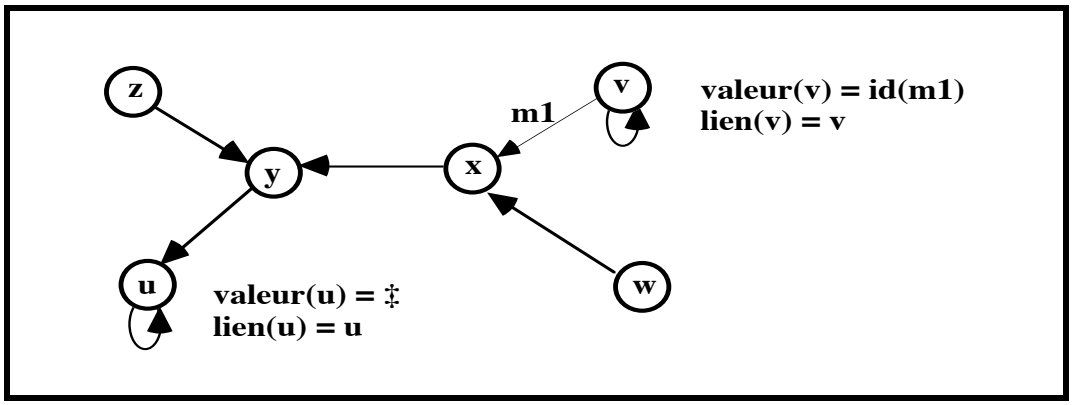


Figure 2. ***Le site v a envoyé un message m1 vers x.***

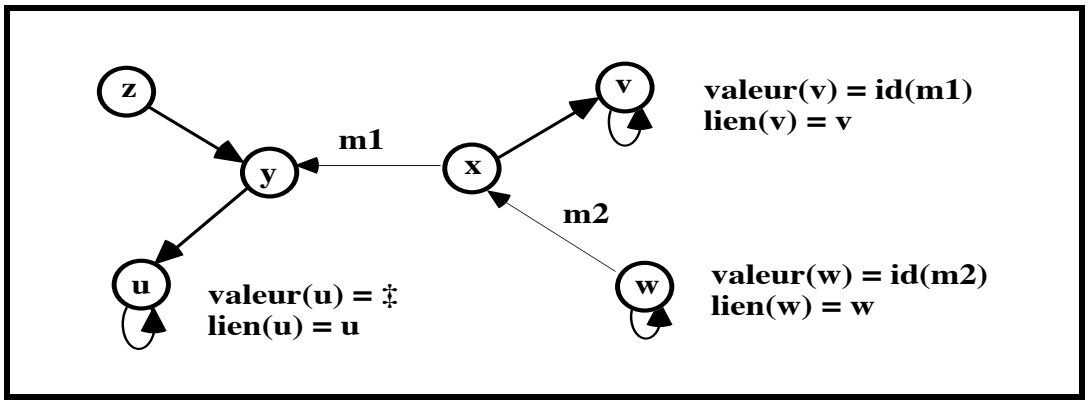


Figure 3. ***Le site w a envoyé un message m2 vers x.***

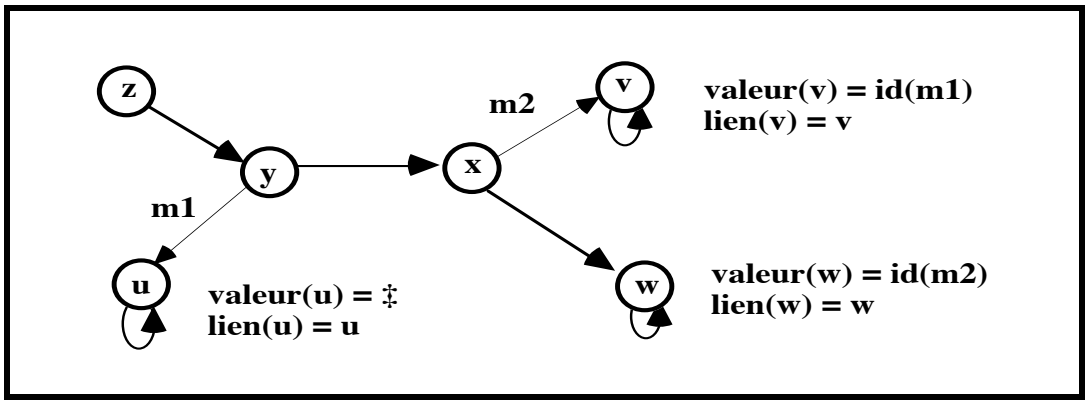


Figure 4. ***Les message m1 et m2 suivent les liens et les modifient au passage. Ainsi m2 est aiguillé vers v.***

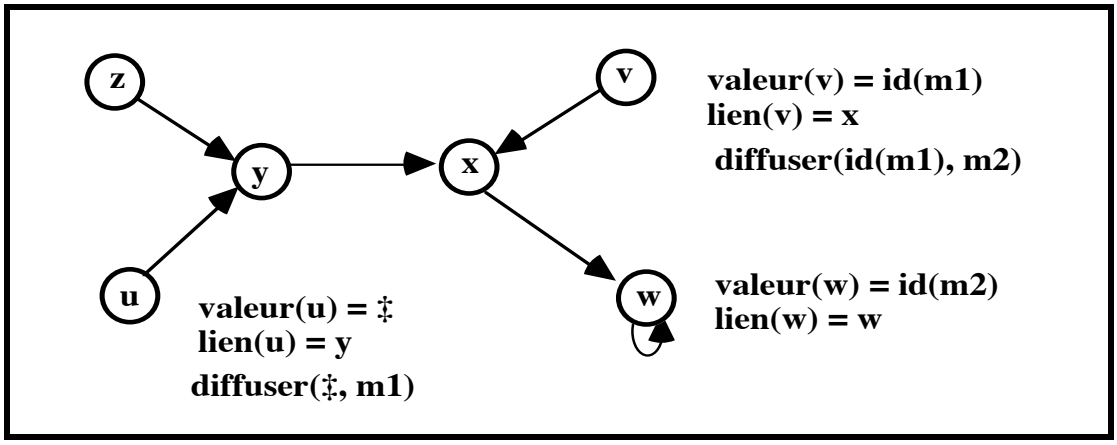


Figure 5. ***Les messages m1 et m2 ont atteint leur racine. Ils sont diffusés concurremment par u et v***

La dernière racine est w qui a envoyé le message classé en dernier et envoyé par v. Supposons que v souhaite diffuser un message m3. Si sa demande est faite avant l'arrivée de m2 sur v, alors on a l'ordre m1, m3, m2 et m3 est diffusé par v avant m2. Si sa demande a lieu après l'arrivée de m2 sur v, on a l'ordre m1, m2, m3 et m3 est diffusé par w

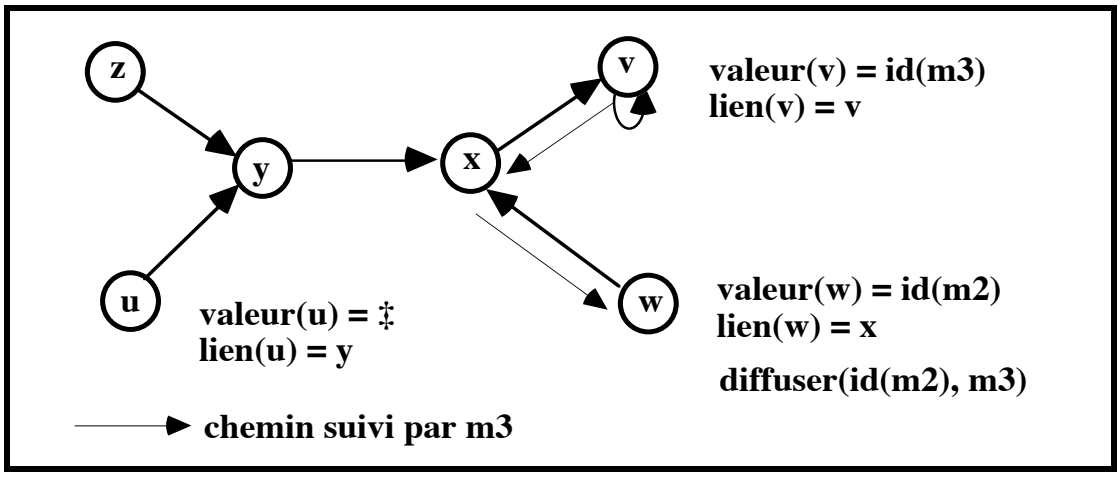


Figure 6. ***Le message m3 demandé par v est diffusé par w.***

Nota. Chaque message trouve une racine qui le diffuse. Le maximum de sites traversés de la source à la racine est le diamètre de l'arbre T.

Remarque : cet algorithme ordonne totalement les prises en compte des requêtes concurrentes en créant une chaîne répartie. Cette création d'un ordre total peut être utile pour traiter d'autres problèmes, comme la création d'un séquenceur réparti (incrémentation ou opérations séquentielles sur une variable répartie) ou l'exclusion mutuelle répartie.

Diffusion avec séquenceur distribué

Modifier cet algorithme pour en faire un séquenceur distribué.
Suggestion. Le compteur de numéro de séquence est dans la racine. Quand une requête lui parvient (la requête remplace le message m), la racine incrémente le compteur et l'envoie au site qui a déclenché la requête et qui est devenu la racine suivante. Ce site diffuse alors le message m avec le numéro de séquence reçu et stocke ce numéro de séquence. On conçoit que la requête peut être n'importe quelle opération sur la valeur stockée. Cette opération est exécutée en exclusion mutuelle (ou encore de manière atomique) par la racine avant de passer le tour à la requête suivante.

Exclusion mutuelle répartie

Modifier cet algorithme pour en faire un algorithme d'exclusion mutuelle répartie.
Suggestion. Faire comme pour le séquenceur distribué. Noter dans la racine courante u le site qui est la racine suivante dans une variable suivant(u) et après l'exécution de la section critique de u, répondre à la requête de suivant(u). Dans le

cas où le graphe est complet, l'arbre couvrant est une étoile de diamètre 2. Comparer à la solution proposée par Naïmi et Trehel.

ENTRÉE ET SORTIE DU GROUPE DE DIFFUSION

L'entrée est immédiate. Un site u entre dans le groupe en se connectant comme une feuille de l'arbre couvrant à un site quelconque v déjà dans le groupe (donc dans l'arbre couvrant). Le site u note lien(u) = v et demande à v de d'accepter et transmettre désormais ses requêtes.
La sortie se passe comme suit. Le site u "verrouille" ses voisins dans l'arbre, après avoir attendu qu'il n'y ait plus de trafic entre u et ses voisins. Si u est racine, il doit choisir un voisin v comme nouvelle racine et lui demander de mettre à jour ses attributs de telle façon que valeur(v) = valeur(u) et lien(v) = v. De plus, que u soit ou non racine, chaque voisin w pointant sur u doit désormais pointer sur lien(u) (if lien(w)□□ then lien(w)□□ien(u) end if). Après ces modifications qui doivent être faites en exclusion mutuelle (d'où le "verrouillage"), le site u "déverrouille" ses voisins et quitte le groupe. La sortie du site u n'est possible que si cette sortie laisse le graphe connexe et que chaque voisin w peut bien pointer sur lien(u).
L'entrée et la sortie sont des opérations locales dont la complexité tient au degré de connexité du site et non à la taille du groupe de diffusion. Elles ne dépendent pas de la taille du groupe, et ne devraient donc pas être sensibles au facteur d'échelle ("scalability").

Références

Herlihy M., Tirthapura S., Wattenhofer R. Ordered Multicast and Distributed Swap. Operating Systems Review, 35,1. January, 2001. ACM Press.

11. ÉTUDE DE CAS DE COHÉRENCE CAUSALE
ORDRE PARTIEL CAUSAL D'ÉCRITURE

On se propose d'étudier l'utilisation de la cohérence causale pour la gestion des URL de serveurs dans les différents caches Web d'une application répartie multisites.

Par exemple, sur le web on a plusieurs serveurs équivalents, certains sont saturés, d'autres accessibles sporadiquement. Dans un groupe coopératif, on veut conserver, dans le cache local à chacun des N membres, l'URL du meilleur serveur connu et avertir les autres sites du groupe.

Soit donc un système réparti comprenant N sites. Il y a des écritures et des lectures locales dans chaque cache local.

Assurer la cohérence des N caches locaux peut se faire en diffusant les modifications à tous les sites et en attendant un acquittement de chacun des sites avant de passer à la modification suivante. Cette sérialisation des transactions, qui peut être obtenue par une exclusion mutuelle répartie ou par une diffusion respectant un ordre total, est coûteuse en message et se rythme sur le site le plus lent à envoyer son acquittement.

Lorsque l'utilisation du cache peut se satisfaire de cohérence causale, on peut avoir une solution plus simple et moins contrainte par le site le plus lent.

On simule ici l'enregistrement d'URL par le stockage d'une variable x qu'on fait évoluer comme suit :

```
function Evolution return Integer is
begin
        if x=0 then x := random(2,3) else x := x*x; end if; -- random donne 2 ou 3
        return x;
end Evolution
```

On a ainsi deux suites de valeurs possibles pour x

0, 2, 4, 16, 256, 65536, ...

0, 3, 9, 81, 729, 531441, ...

Les lectures respectent l'ordre causal si après avoir lu la valeur numéro n d'une de ces suites, la lecture suivante ne donne pas une valeur de numéro plus petite d'une des suites. Toute valeur de numéro supérieur à n est acceptable qu'elle soit de l'une ou l'autre suite.
Ainsi 0, 4, 3, 16, 256 n'est pas acceptable.
Et 0, 3, 4, 729 est acceptable

On verra dans la partie 3, comment traiter des valeurs de même numéro dans les deux suites.
Bien entendu cette notion de lecture causale se généralise pour plus de 2 suites de valeurs.

1. PREMIER SCÉNARIO ET COHÉRENCE CAUSALE DE REQUÊTES

On considère la première trace donnée figure 1. On constate que l'on a un ordre total entre les diffusions.

e11 -> e22 -> e13 -< e34

Dans une première réalisation, les messages contiennent des requêtes qui sont les méthodes à employer pour mettre à jour l'objet x. Il y a deux méthodes appelées set et square :

set(y) qui met à y l'objet x

square élève x au carré

Le message m1 diffusé en e11 contient set(2) ; les messages m2, m3, m4 diffusés en e22, e13 et e34 contiennent square.

Les arrivées de message demandent des mises à jour de l'objet x. Toutefois ces mises à jour doivent respecter l'ordre causal. Pour véhiculer cet ordre, on installe sur chaque site Si une horloge vectorielle Vi qui compte les diffusions faites sur chaque site et connues par Si . La valeur Vm = Vi de l'horloge du site émetteur Si accompagne chaque message émis et sert à la réception à imposer le respect de l'ordre causal et à mettre à jour l'horloge vectorielle Vj du site récepteur Sj. La suite des valeurs délivrées à l'application, quand il y a des demandes de lecture, ne doit pas être en contradiction avec l'ordre causal. Ainsi, la suite de lectures : 0, 4, 2, 16, 256, n'est pas acceptable ; par contre 0, 4, 16, 256 le serait.

QUESTION 1.1.
Donner les valeurs Vm des horloges vectorielles qui accompagnent les messages diffusés en e11, e22, e13 et e34. On peut noter m = {méthode, (Vm)}.

QUESTION 1.2.
Décrivez les calculs qui servent à réordonner causalement les messages reçus sur les sites S3 et S4. On note Vm l'estampille qui accompagne le message et V3 et V4 les horloges vectorielles des sites S3 et S4. Indiquez les messages qui doivent être réordonnés (on indiquera les événements de réception correspondants) et dans quel ordre.

QUESTION 1.3.
Quand une demande de lecture intervient sur S4 entre e41 et e42, il y a deux politiques de réponse compatibles avec le respect de l'ordre causal. Quelles sont-elles?
Mêmes questions pour une lecture entre e42 et e43.
Quelle est la réponse pour une lecture entre e43 et e44.
Donnez la suite des valeurs conservées après chaque événement sur les sites S3 et S4.

2. PREMIER SCÉNARIO ET COHÉRENCE CAUSALE D'ÉCRITURE

On considère toujours la première trace donnée figure 1. Mais maintenant on suppose que les messages contiennent les valeurs destinées à faire les mises à jour.
On doit distinguer les écritures primaires et les écritures de mise à jour.
Les écritures primaires causent les événements e11, e22, e13, e34 qui correspondent aux diffusions de messages . Le message m1 diffusé en e11 contient la valeur 2 ; le message m2 diffusé en e22 contient la valeur 4, le message m3 diffusé en e13 contient la valeur 16, le message m4 diffusé en e34 contient la valeur 256.

Les arrivées de message demandent des changement de valeur de l'objet x par des écritures de mise à jour. Toutefois ces changements doivent respecter l'ordre causal des

écritures primaires. Pour véhiculer cet ordre, on installe sur chaque site Si un compteur Hi , qui compte les écritures faites sur ce site (c'est une sorte de compteur de versions). L'état Cm du compteur Hi du site émetteur Si accompagne chaque message émis (Cm = Hi) et sert à la réception à imposer le respect de l'ordre causal et à mettre à jour le compteur Hj du site récepteur Sj. La suite des valeurs délivrées à l'application, quand il y a des demandes de lecture, ne doit pas être en contradiction avec l'ordre causal. Ainsi, la suite de lectures : 0, 4, 2, 16, 256, n'est pas acceptable ; par contre 0, 16, 256 le serait.

QUESTION 2.1.
Donner les états C des compteurs qui accompagnent les messages diffusés en e11, e22, e13 et e34. On peut noter m = {valeur, (Cm)}.

QUESTION 2.2.
Avec ce comptage, on peut accepter tout de suite une valeur plus récente, même si ce n'est pas la version suivant immédiatement celle qu'on change. On réalise ainsi une optimisation dans la mise à jour. Par contre, il ne faut pas oublier de respecter l'ordre causal. Ainsi en e41 on peut donner tout de suite à x la valeur 16 et délivrer cette valeur en cas de demande de lecture sur S4. mais cela n'est pas neutre pour les messages postérieurs qui arriveront avec des valeurs plus anciennes (donc dont le compteur a un numéro plus petit)
Décrivez l'algorithme qui, sur le site Sj récepteur d'un message, permet de prendre en compte cette optimisation et à mettre à jour la copie xj et le compteur Hj du site, tout en respectant l'ordre causal.
En appliquant cet algorithme, indiquez les calculs qui servent à ordonner causalement les valeurs reçues sur les sites S3 et S4. Donnez la suite des valeurs conservées après chaque événement de ces sites.

3. DEUXIÈME SCÉNARIO ET COHÉRENCE CAUSALE D'ÉCRITURE

On considère la deuxième trace donnée figure 2. On constate que l'on a un ordre partiel entre les diffusions.

e11 -> e12 -< e35
e21 -> e32 -> e35

On suppose à nouveau (comme dans la partie 2) que les messages contiennent les valeurs destinées à faire les mises à jour.

QUESTION 3.1.
Donner les états C des compteurs qui accompagnent les messages envoyés en e11, e12, e21, e32 et e35.

QUESTION 3.2.
Quand un message arrive avec un numéro de version C égal à celui de la version courante Hj sur le site récepteur Sj, on traite la version qui arrive comme si elle était plus ancienne (c'est à dire de plus petit numéro) que la version courante.
Décrivez l'algorithme qui, sur le site Sj récepteur d'un message, permet de prendre en compte cette politique, ainsi que l'optimisation vue en question 2.2 avant de mettre à jour la copie xj et le compteur Hj du site.
En appliquant cet algorithme, indiquez les calculs qui servent à ordonner causalement les valeurs reçues sur les 4 sites. Donnez la suite des valeurs conservées après chaque événement sur ces sites et donnez les valeurs envoyées dans chacun des messages..

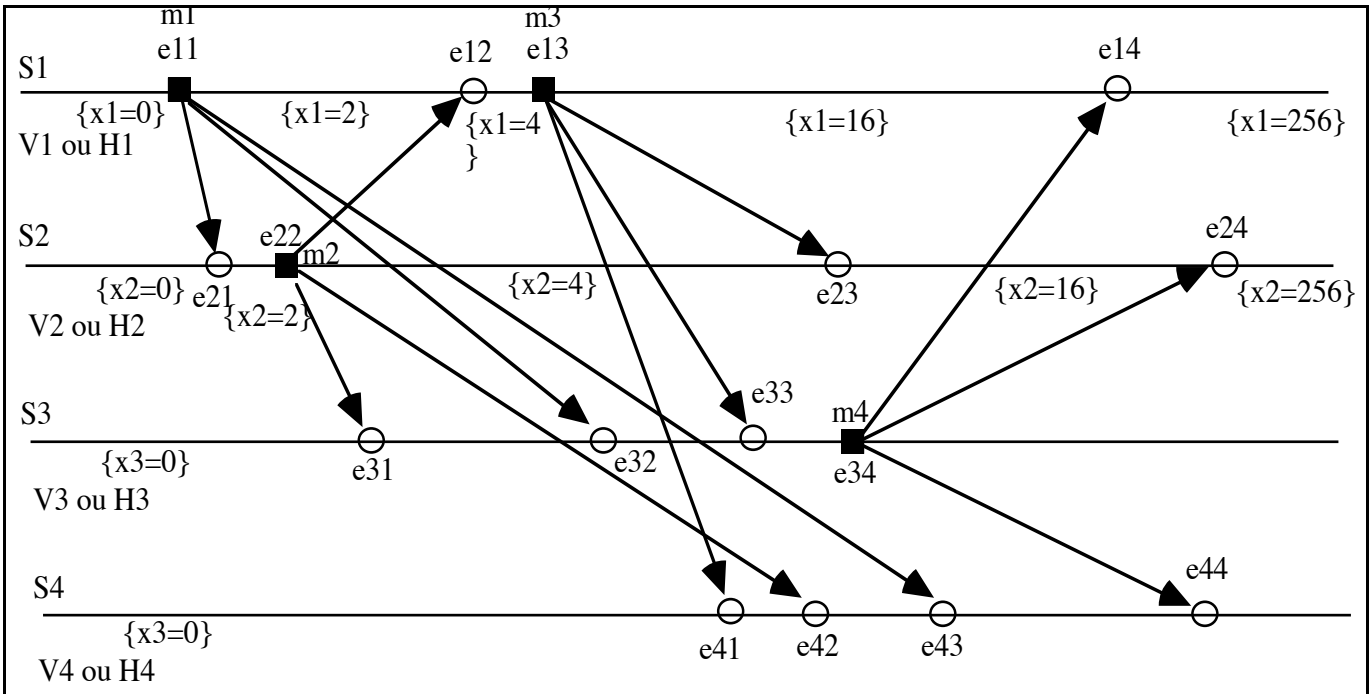
QUESTION 3.3.
Quand un message arrive avec un numéro de version C égal à celui de la version courante Hj sur le site récepteur Sj, on traite la version qui arrive comme si elle était plus récente (c'est à dire de plus grand numéro) que la version courante.
Décrivez l'algorithme qui, sur le site Sj récepteur d'un message, permet de prendre en compte cette politique, ainsi que l'optimisation vue en question 2.2 avant de mettre à jour la copie xj et le compteur Hj du site.
En appliquant cet algorithme, indiquez les calculs qui servent à ordonner causalement les valeurs reçues sur les 4 sites. Donnez la suite des valeurs conservées après chaque événement sur ces sites et donnez les valeurs envoyées dans chacun des messages..

QUESTION 3.4.
On constate des différences notables entre les résultats des questions 3.3 et 3.4. Est-ce gênant ?
D'autres politiques sont envisageables pour départager les valeurs qui ont le même numéro de version dans un message et sur le site destinataire. Imaginez en une qui soit compatible avec la causalité.

4. DEUXIÈME SCÉNARIO ET ORDRE TOTAL CAUSAL

On veut maintenant imposer un ordre total sur les versions, tout en gardant l'optimisation vue dans la question 2.2.

QUESTION 4.1.
Comment feriez vous pour avoir une solution correcte et combien de messages supplémentaires votre solution nécessiterait-elle ?

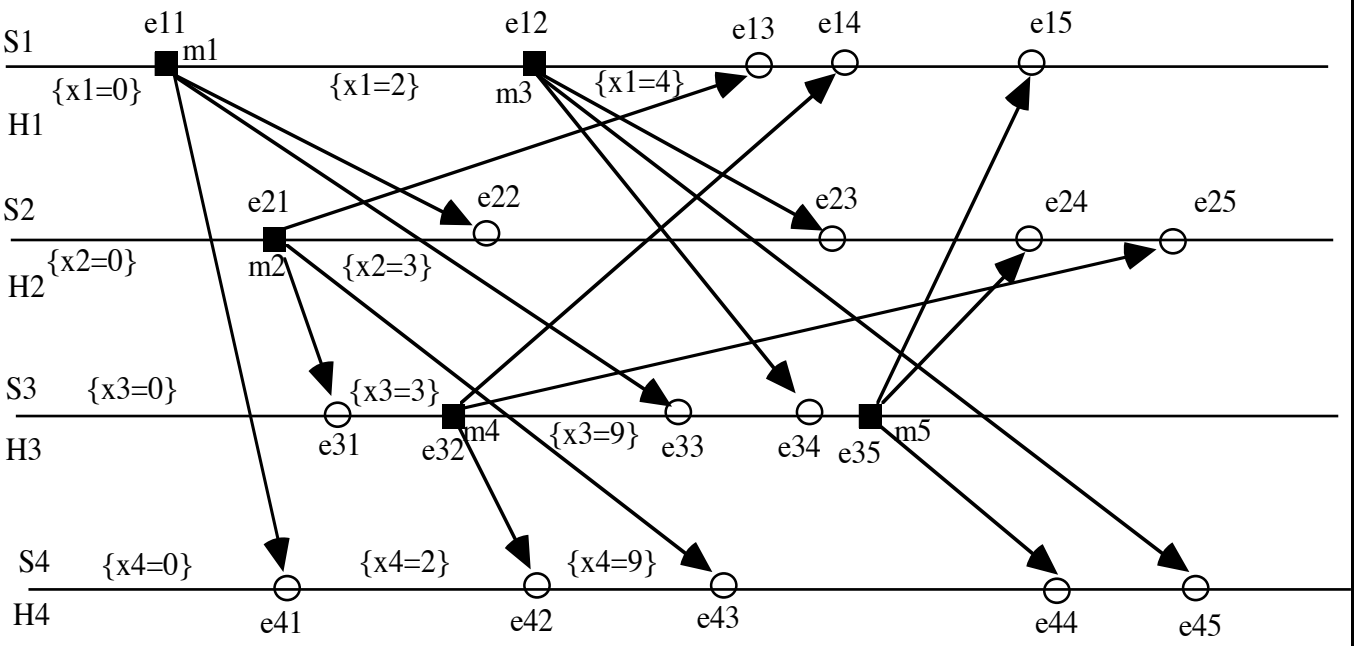


Légende : {xi=0} {xi=256} valeur délivrée quand il y a une lecture

■ diffusion de message

○ réception d'un message

FIGURE 1 : ORDRE TOTAL



Légende : {xi=0} {xi=9} valeur délivrée quand il y a une lecture

■ diffusion de message

○ réception d'un message

FIGURE 2 : ORDRE PARTIEL

SOLUTIONS

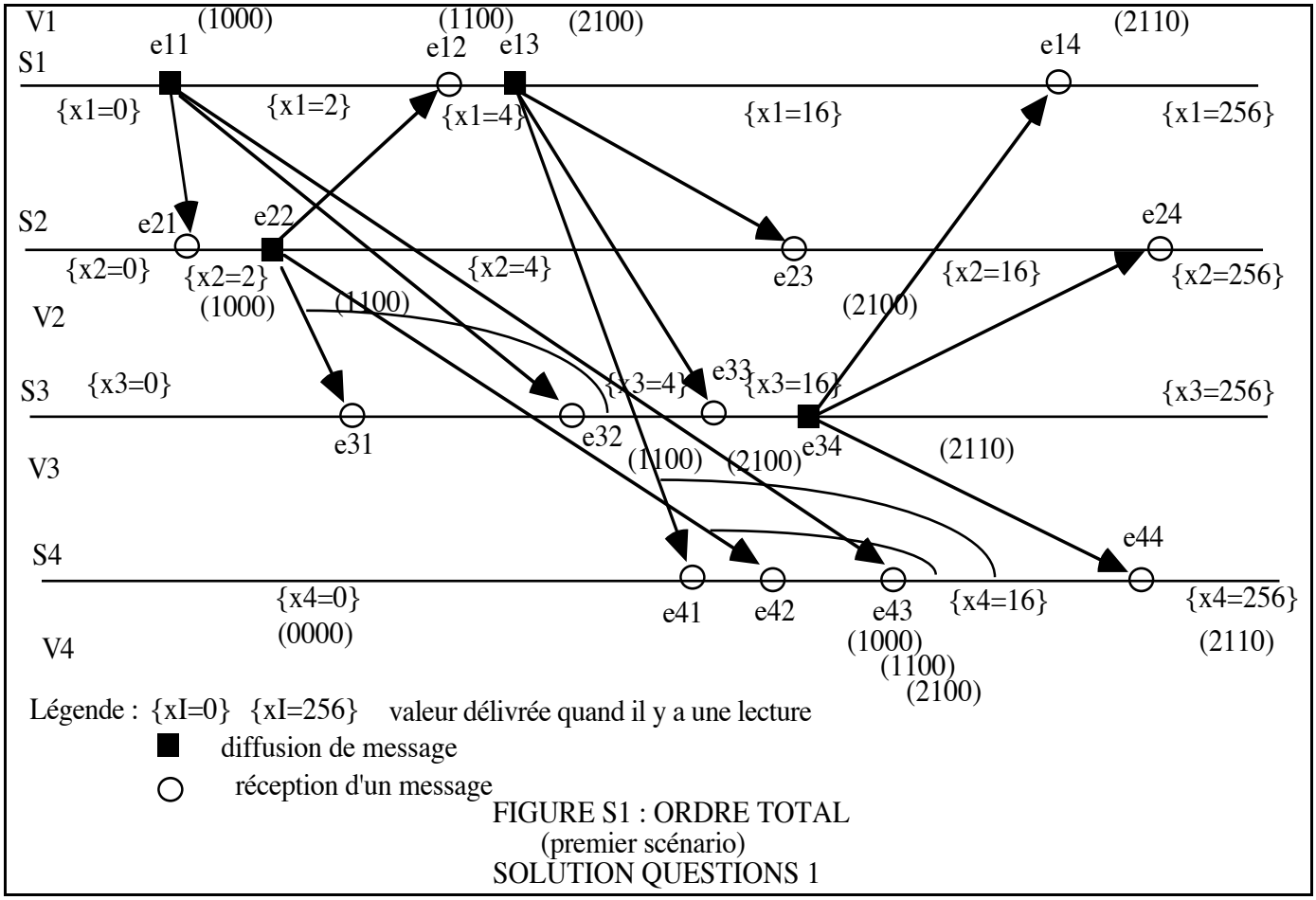
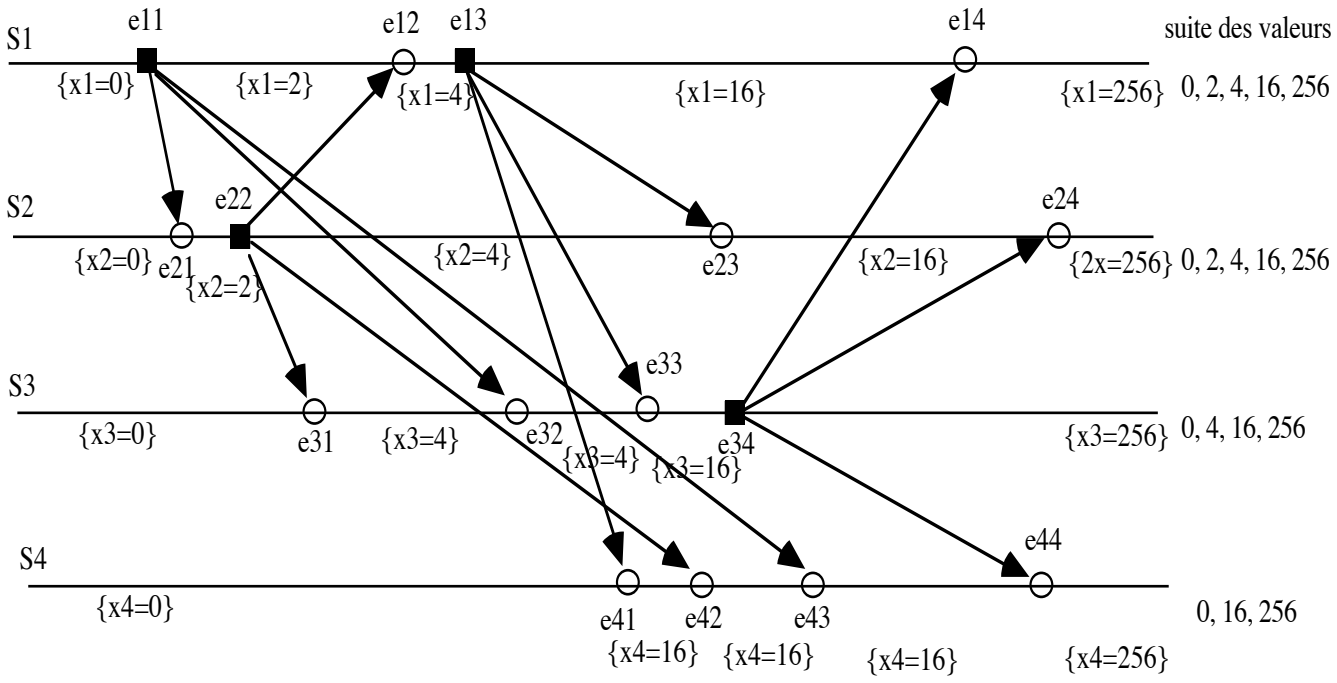


FIGURE S1 : ORDRE TOTAL
(premier scénario)
SOLUTION QUESTIONS 1

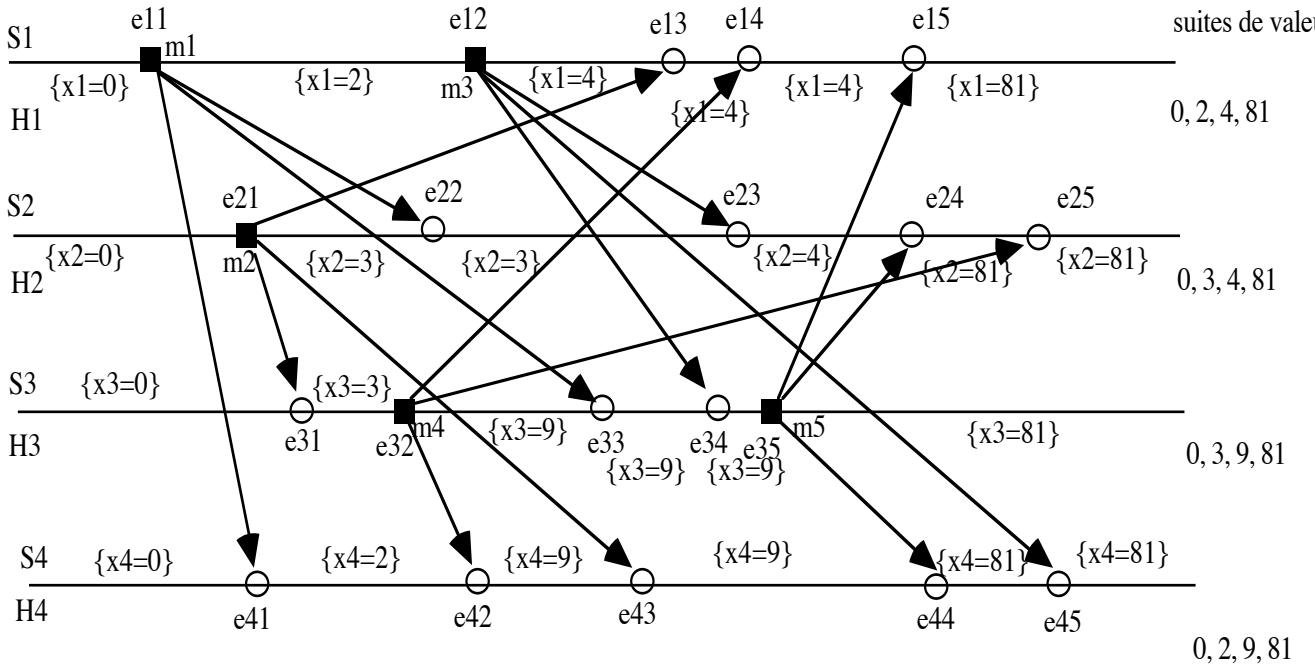


Légende : {xi=0} {xi=256} valeur délivrée quand il y a une lecture

■ diffusion de message

○ réception d'un message

FIGURE S2 : ORDRE TOTAL
(premier scénario)
SOLUTION QUESTIONS 2

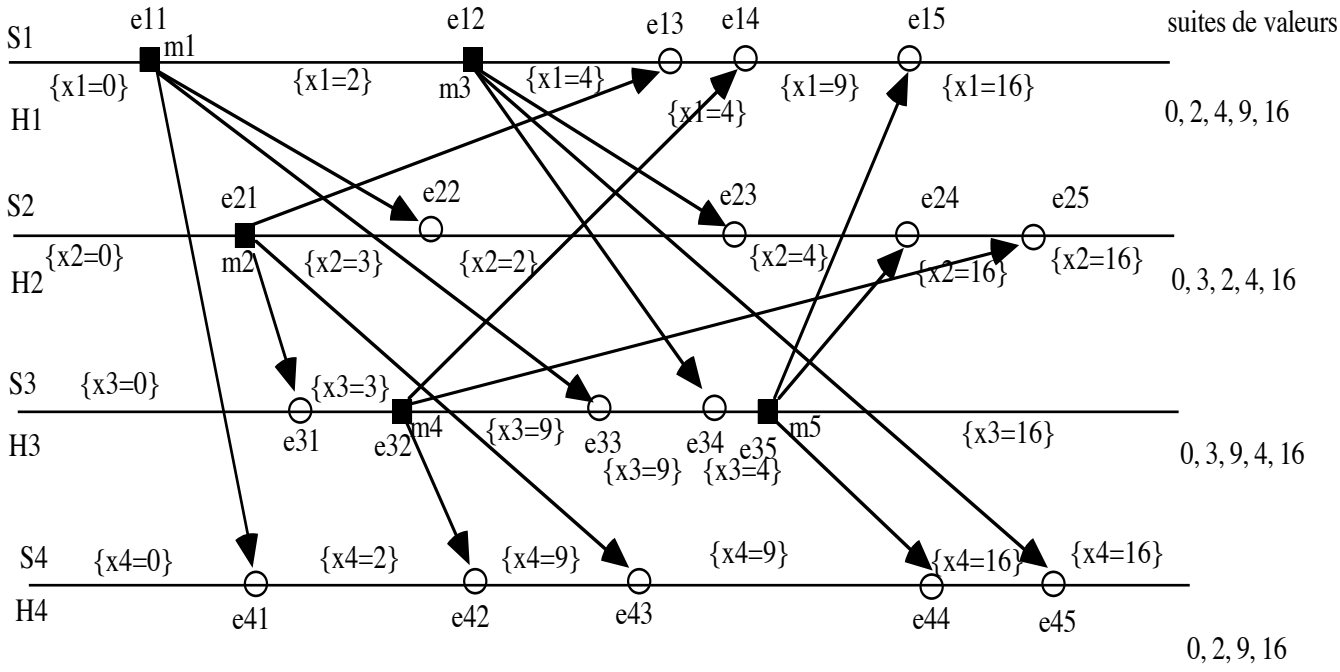


Légende : {xi=0} {xi=9} valeur délivrée quand il y a une lecture

■ diffusion de message

○ réception d'un message

FIGURE S3 : ORDRE PARTIEL
(deuxième scénario)
SOLUTION QUESTION 3.2



Légende : {xi=0} {xi=9} valeur délivrée quand il y a une lecture

■ diffusion de message

○ réception d'un message

FIGURE S4 : ORDRE PARTIEL
(deuxième scénario)
SOLUTION QUESTION 3.3

SOLUTIONS QUESTIONS 1

RÉPONSE 1.1.
e11 diffuse m1 = {set(2), (1000)}, e22 diffuse m2 = {square, (1100)}, e13 diffuse m3 = {square, (2100)}, e34 diffuse m4 = {square, (2110)}

RÉPONSE 1.2.
Sur S3, réception de m2 en **e31** avec Vm = (1100) venu de S2 alors que V3 = (0000).
Il faut attendre que le message m1 annoncé par Vm[1] = 1 soit arrivé. Cela se produit en e32. On doit réordonner comme suit délivrance du message reçu en e32 avant le message reçu en e31. Après cela on a V3 =(1100)
Sur S3 réception de m1 en **e32** avec Vm = (1000). Acceptable et alors V3 = (1000) puis on délivre le message m2 arrivé en e31 et alors V3 = (1100)
Sur S3 réception de m3 en **e33** avec Vm = (2100). Pas de problème car V3[1] = Vm[1] - 1.
Pas d'autres problèmes car partout ailleurs V3[k] ≥ Vm[k] pour k ≠ indice du site émetteur.
Sur S4 : réception de m3 en **e41** avec Vm = (2100) venu de S1 alors que V4 = (0000). On a : V4[1] < Vm[1] - 1 et : V4[2] < Vm[2]
Il faut attendre que :
le message m1 de S1 et annoncé par Vm[1] = 2 soit arrivé. Cela se produira en e43.
le message m2 de S2 et annoncé par Vm[2] = 1 soit arrivé. Cela se produira en e42.
Sur S4 : réception de m2 en **e42** avec Vm = (1100) venu de S2. On a V4[2] = Vm[2] - 1. C'est bon mais on a encore V4[1] < Vm[1].
Il faut attendre que :
le message m1 S1 et annoncé par Vm[1] = 1 soit arrivé. Cela se produira en e43.
Sur S4 : réception de m1 en **e43** avec Vm = (1000) venu de S1. On a V4[1] = Vm[1] - 1. C'est bon. Il en résulte V3 = (1000). On peut alors délivrer le message m2 reçu en e42. Il en résulte V3 = (1100). On peut alors délivrer le message m3 reçu en e41. Il en résulte V3 = (2100).
Pas d'autres problèmes car partout ailleurs V3[k] ≥ Vm[k] pour k ≠ indice du site émetteur.
Voir Figure S1 Solutions questions 1

RÉPONSE 1.3.
Deux politiques possibles entre e41 et e43 : 1) faire attendre la lecture ou 2) donner l'ancienne valeur qui est x = 0
Une seule politique entre e43 et e44 : délivrer la dernière valeur, c'est à dire x = 16

SOLUTIONS QUESTIONS 2

RÉPONSE 2.1.
e11 diffuse m1 = {2, (1)}, e22 diffuse m2 = {4, (2)}, e13 diffuse m3 = {16, (3)}, e34 diffuse m4 = {256, (4)}

RÉPONSE 2.2.
soit un message m = {valeur, (C)} reçu par le site SJ, l'algorithme du récepteur est :
if Hj > C **then** jeter le message m; **else** Hj := C ; xj := valeur; **end if;**
sur S3, on a la suite de valeurs : 0, 4, 16, 256
sur S4, on a la suite de valeurs : 0, 16, 256
Voir Figure S2 Solutions questions 2

SOLUTIONS QUESTIONS 3

RÉPONSE 3.1.
e11 diffuse m1 = {2, (1)}, e21 diffuse m2 = {3, (1)}, e12 diffuse m3 = {4, (2)}, e32 diffuse m4 = {9, (2)}, e35 diffuse m5 = { , (3)}

RÉPONSE 3.2.
soit un message m = {valeur, (C)} reçu par le site SJ, l'algorithme du récepteur est :
if Hj ≥ C **then** jeter le message m; **else** Hj := C ; xj := valeur; **end if;**
Voir Figure S3 Solutions questions 3.2
sur S1, on a la suite de valeurs : 0, 2, 4, 81
sur S2, on a la suite de valeurs : 0, 3, 4, 81
sur S3, on a la suite de valeurs : 0, 3, 9, 81
sur S4, on a la suite de valeurs : 0, 2, 9, 81

RÉPONSE 3.3.
soit un message m = {valeur, (C)} reçu par le site SJ, l'algorithme du récepteur est :
if Hj > C **then** jeter le message m; **else** Hj := C ; xj := valeur; **end if;**
Comme en 2.2.
Voir Figure S4 Solutions questions 3.3
sur S1, on a la suite de valeurs : 0, 2, 4, 9, 16
sur S2, on a la suite de valeurs : 0, 3, 2, 4, 16
sur S3, on a la suite de valeurs : 0, 3, 9, 4, 16
sur S4, on a la suite de valeurs : 0, 2, 9, 16

RÉPONSE 3.4.
Gênant ? C'est la conséquence de l'hypothèse qu'on peut se contenter d'une cohérence causale dans l'application concernée. Un cache qui contient des URL successives d'un serveur mobile ou d'un serveur parmi un groupe de serveur devrait pouvoir avoir une utilité, si ce n'est qu'un accélérateur de recherche et si on accepte que l'URL puisse ne plus être valable.
Autres politiques pour départager les valeurs de même numéro de version :
politique 1 : prendre la version du site de plus petit numéro (politique 2. de plus grand numéro). Politique 3 : tirer au sort à chaque arrivée posant problème. En fait toutes les politiques de choix sont acceptables car on accepte par principe la présence de l'ordre partiel. La causalité interdit seulement de prendre comme nouvelle valeur une version plus ancienne en causalité que la version actuelle.

RÉPONSE 4.

Voir le cours, partie Ordres, état global dans les systèmes répartis, chapitre 6 Etude de cas
Solutions possibles vues en cours
1. mettre un site séquenceur allouant des numéros de séquenc : coût : 2 messages par diffusion
2. mettre un site serveur primaire de la donnée x auquel on demande une action et qui diffuse les valeurs successives : coût 1 message par diffusion
3. utilser un anneau virtuel contenant un compteur : coût des messages de circulation du jeton de l'anneau virtuel
4. utiliser un algorithme d'exclusion mutuelle : coût de l'algorithme de Ricard Agrawala 2 N messages par diffusion