Principes des systèmes d'exploitation

Support de transparents

Partie 2: Systèmes distribués

IUP MIAGE

Faculté de sciences -UNSA

N. Le Thanh

février 1995

Plan du cours

II- Partie 2 : Systèmes distribués

II.1- Introduction

- **objectifs**
- **concepts matériels**
- **concepts logiciels**
- **bases de la conception des systèmes distribués**

II.2- Communication dans les systèmes distribués

- **couches de protocoles**
- **modèle client-serveur**
- **appels de procédures à distance**
- **communication de groupe**
- **Quelques approches pratiques**

Plan du cours

II.3- Synchronisation dans les systèmes distribués

- **synchronisation d'horloge**
- **exclusion-mutuelle Algorithmes d'élection**
- **transactions atomiques**
- **interblocage dans les systèmes distribués**

II.4- Etude de cas (TD/TP)

- **Service web et programmation Client/serveur sur web**
- **Modèle CORBA et systèmes à objets distribués**
- **Modèles client/serveur et à objets de Java : rmi, jdbc, servlet, applet**

II- Partie II : systèmes distribués
II.1- Introduction

II.1.1- Objectifs Avantages/Inconvénients

SYSTÈME DISTRIBUÉ = SYSTÈME POSSÈDANT PLUSIEURS PROCESSEURS COOPÉRANTS

OBJECTIFS

- ***Coût*** **: plusieurs processeurs à bas prix**
- ***Puissance de calcul*** **: aucune machine centralisée peut réaliser**
- ***Performance*** **: calcul parallèle**
- ***Adaptation*** **: à des classes d'applications réelles naturellement distribuées**
- ***Fiabilité*** **: résistance aux pannes logicielles ou matérielles**
- ***Extensibilité*** **: croissance progressive selon le besoin**

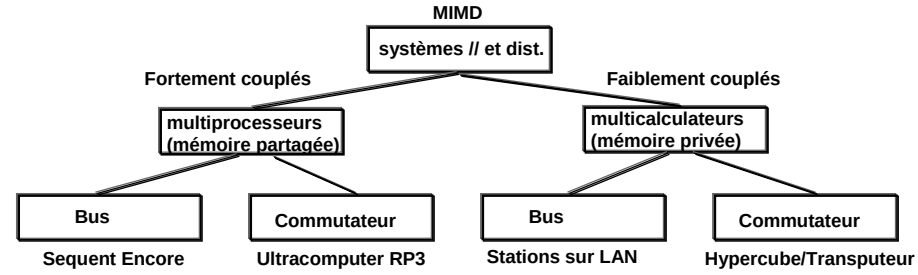
Avantages	Inconvénients
- partage de données - partage de périphériques - communication - souplesse (politiques de placements)	- logiciels : peu de logiciels disponibles - réseaux : la saturation et délais - sécurité : piratage

II- Partie II : systèmes distribués
II.1- Introduction

II.1.2- Concepts matériels

Taxonomie de Flynn (1972)
- nombre des flux d'instructions
- nombre des flux de données

SISD :	un seul flux d'instruction et un seul flux de données (ordinateurs centralisé)
SIMD :	un seul flux d'instruction et multiples flux de données (machines // vectorielles)
MISD :	multiples flux d'instruction et un seul flux de données (pas de machine réelle)
MIMD :	multiples flux d'instruction et multiples flux de données

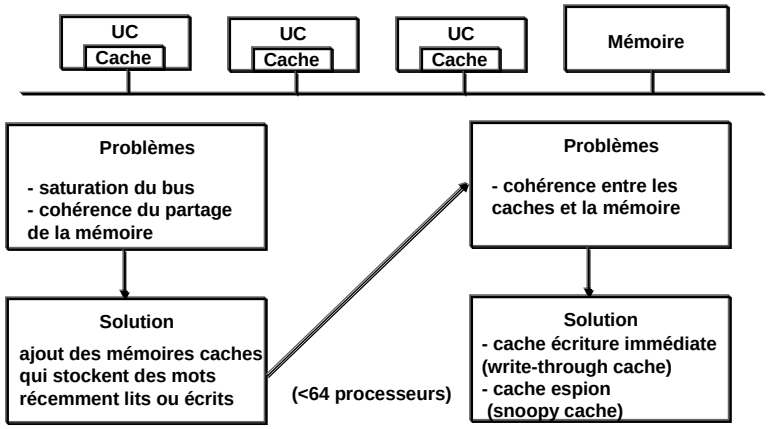


Page 5

II- Partie II : systèmes distribués
II.1- Introduction

II.1.2- Concepts matériels

II.1.2.1- Multiprocesseurs à bus

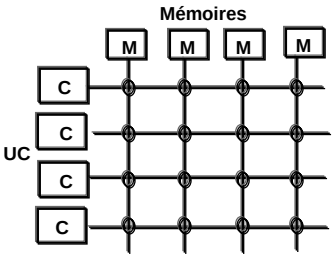


Page 6

II- Partie II : systèmes distribués
II.1- Introduction

II.1.2- Concepts matériels

II.1.2.2- Multiprocesseurs commutés



Technique de matrice de commutation (Crossbar Switch)

- les UC et mémoires sont reliées par une matrice de commutation
- à chaque l'intersection, le noeud de commutation (crosspoint switch) peut-être ouvert ou fermé
- quand une UC veut accéder à un module de mémoire elle ferme temporairement le noeud de commutation corresondant
- si plusieurs UC veulent accéder au même module, une file d'attente est nécessaire

Inconvénient

le nombre des noeuds de commutation nécessaires : il faut n^2 de noeuds de commutation pour rélier n UC aux n module de mémoire

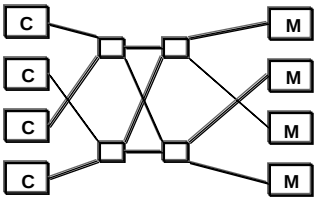
⇒ réseaux d'interconnexion minimisant le nombre de noeuds ?

Page 7

II- Partie II : systèmes distribués
II.1- Introduction

II.1.2- Concepts matériels

II.1.2.2- Multiprocesseurs commutés



Principe du réseau oméga (multiétage)

- utilisation des commutateurs 2x2 : 2 entrées et deux sorties
- chaque commutateur peut relier à n'importe quelle entrée et à n'importe quelle sortie
- pour relier n UC à n mémoires, il nécessite n étages dont chacun contient $\log_2 n$ commutateurs 2x2
- le nombre nécessaire de commutateurs est : $n \times \log_2 n$

Inconvénient : Temps de propagation

- Si n= 1024, on a besoin 10 étages
- Avec les UC de 50Mhz, le cycle de calcul est de 20ns
- une requête mémoire traverse 20 étages (allé/retour) en 20 ns si le temps de commutation est de 1ns
- On doit avoir 10 240 commutateurs à 1ns !!!

⇒

CONCLUSION

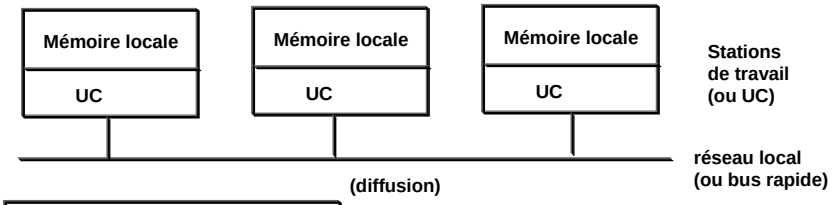
construire un gros multiprocesseurs fortement couplé à mémoire partagée est difficile et coûteux

Page 8

II- Partie II : systèmes distribués
II.1- Introduction

II.1.2- Concepts matériels

II.1.2.3- Multicalculateurs à bus



- Facile à construire
- Bas coût
- Le trafic est moins important (pas d'échange UC-mémoire)
- souvent utilisé dans les réseaux locaux avec une vitesse de 10 à 100Mbits/s

Remarque

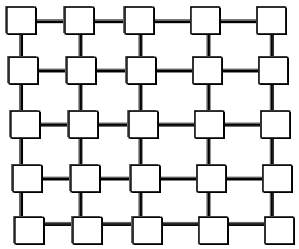
Le bus rapide peut être construit par les différentes techniques (exemple : réseau bynet de teradata BDC1024

Page 9

II- Partie II : systèmes distribués
II.1- Introduction

II.1.2- Concepts matériels

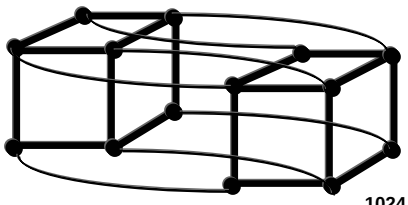
II.1.2.4- Multicalculateurs commutés



Treillis

- câblage simple
- le chemin plus long croit en racine carré du nombre d'UC

Remarque
Utilisation de crossbar et multiétage



Hypercubes
(4 dimensions)

1024 UC
16 384 UC

- Un hypercube est un cube à n dimensions
- 4 dimensions : 2 cubes de 3 dimensions avec les sommets homologues reliés
- 5 dimensions : 2 hypercubes de 4 dimensions avec les sommets homologues reliés
- la complexité du câblage croit en $\log_2$ du nombre UC
- le chemin plus long croit en $\log_2$ du nombre UC

Page 10

II- Partie II : systèmes distribués
II.1- Introduction

II.1.3- Concepts logiciels

II.1.3.1- Classification

Le matériel est important, mais le logiciel l'est plus encore

La classification est, par nature, imprécise et floue :

systèmes d'exploitation faiblement couplés	systèmes d'exploitation fortement couplés
- réseaux des ordinateurs indépendants (et alors la coopération des ordinateurs indépendants sans réseau ?)	- multiprocesseurs exécutant un seul programme en // (jeu d'échecs par exemple) - machine des bases de données - machine // à processeurs symétriques ...

multiprocesseurs SE faiblement couplé	multiprocesseurs SE fortement couplé	Systèmes d'exploitation réseaux
multicalculateurs SE faiblement couplé	multicalculateurs SE fortement couplé	Systèmes d'exploitation distribués
		Systèmes multiprocesseurs à temps partagé

Page 11

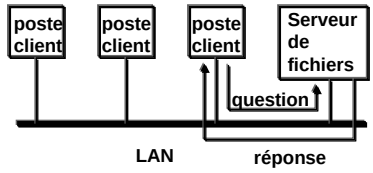
II- Partie II : systèmes distribués
II.1- Introduction

II.1.3- Concepts logiciels

II.1.3.2- Systèmes d'exploitation réseaux et NFS

La combinaison la plus fréquente : matériels et logiciels faiblement coupés

Connexion point-à-point		système d'exploitation réseaux	
- postes avec SE indépendant - partage des imprimantes - accès à un autre poste en mode terminal		- postes indépendants + serveurs de fichiers - système de fichiers partagés - gestion central des utilisateurs	
Réseaux locaux	Internet	Réseaux locaux	Internet
- stations NT - windows workgroups	FTP, TELNET, ...	Novell, Lanmanager Unix, NTserver services	HTTP, NFS (Network File Manager)



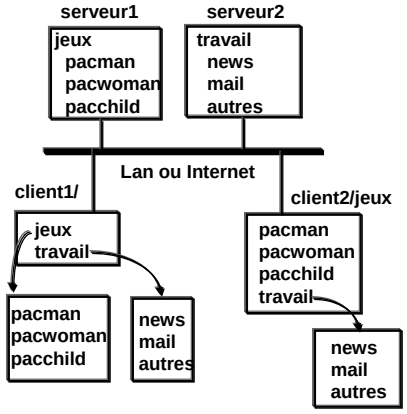
Page 12

II- Partie II : systèmes distribués
II.1- Introduction

II.1.3- Concepts logiciels

II.1.3.2- Systèmes d'exploitation réseaux et NFS

Architecture et protocoles NFS



Serveurs

- exporter des répertoires (/etc/exports)
- gérer les accès aux répertoires exportés pour les clients (locaux ou à distance)
- protection des fichiers
- transmettre les données aux clients (réponses)
- les clients et les serveurs peuvent partager un même système de fichiers

Clients

- monter des répertoires exportés sur sa station
- utiliser ces répertoires comme (ou presque) ceux locaux
- plusieurs clients peuvent partager les mêmes répertoires (fichiers partagés)
- toute machine peut à la fois client et serveur
- systèmes d'exploitation locaux peuvent être <>

Page 13

II- Partie II : systèmes distribués
II.1- Introduction

II.1.3- Concepts logiciels

II.1.3.2- Systèmes d'exploitation réseaux et NFS

Architecture et protocoles NFS

Un protocole est un ensemble de requêtes envoyées par les clients aux serveurs auxquelles correspondent les réponses des serveurs aux clients

1er protocole : Protocole de montage

1- Le client envoie une requête contenant le chemin d'accès à un répertoire du serveur et une demande de l'autorisation de le monter dans son arborescence.

2- Si le chemin est correct et le répertoire est exportable, le serveur revoie au client une clé de fichier (file handle) concernant ce répertoire. Cette clé comporte des champs qui identifient uniquement le type de Système de Fichiers, le disque, le n° de i-noeud du répertoire et les informations sur les protections.

3- Classiquement, le client a un fichier Shell script /etc/rc contenant les commandes de montage. Ce Shell script sera exécuté automatiquement au lancement du système

4- Automontage (Unix de Sun) : le montage se fait automatiquement à la 1ere ouverture d'un fichier à distance, le client contacte les serveurs et monter le répertoire contenant ce fichier

Page 14

II- Partie II : systèmes distribués
II.1- Introduction

II.1.3- Concepts logiciels

II.1.3.2- Systèmes d'exploitation réseaux et NFS

Architecture et protocoles NFS

2ème protocole : Protocole d'accès

1- Le client envoie aux serveurs des requêtes de manipulation des répertoires ou des fichiers

2- La plupart des appels système UNIX sont supportés par NFS, à exception OPEN et CLOSE. L'omission de ces opérations permet aux serveurs de ne pas gérer les fichiers ouverts (serveur sans état (stateless). Chaque requête d'accès se suffit à elle-même. L'appel READ contient la clé du fichier, la position courante et le nombre d'octet à lire. On utilise LOOKUP à la place de OPEN. L'inconvénient principal est l'absence de contrôle de cohérence. On ne peut pas verrouiller le fichier (cas il est toujours à état fermé), donc le partage d'accès peut introduire des incohérences

3- Il existe sous UNIX système V, le mécanisme Remote File System (RFS) qui demande le fichier soit ouvert avant la lecture. Le serveur doit gérer dans ce cas un descripteur de fichier pour tout fichier ouvert à distance

4- Chaque requête doit s'identifier par la cryptographie à clés publiques. Il possède une clé cryptée à envoyer au serveur . Le serveur stocke des couples (clé, valeur) dans NIS (Network Information Service). Quand on lui envoie une clé, il retourne une valeur.

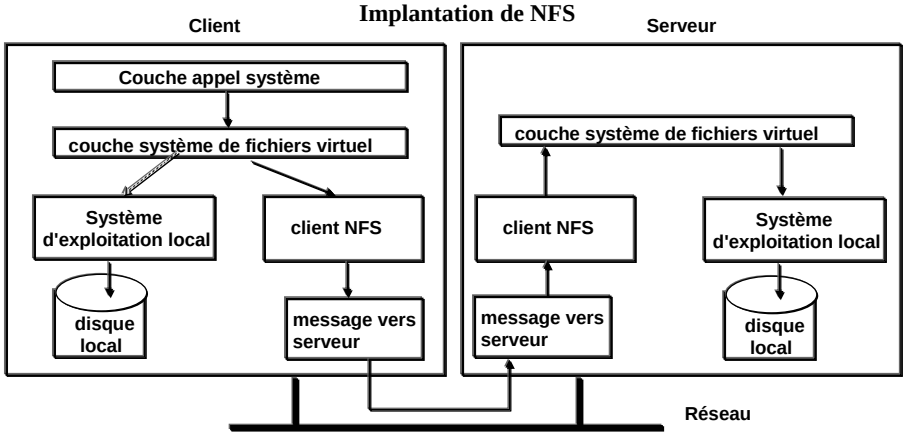
Page 15

II- Partie II : systèmes distribués
II.1- Introduction

II.1.3- Concepts logiciels

II.1.3.2- Systèmes d'exploitation réseaux et NFS

Implantation de NFS



Page 16

II- Partie II : systèmes distribués
II.1- Introduction

II.1.3- Concepts logiciels

II.1.3.3- Systèmes distribués

Un système distribué est un système qui s'exécute sur un ensemble de machines sans mémoire partagée, mais que pourtant l'utilisateur voir comme une seule et unique machine

Caractéristiques

- possédant un mécanisme de communication interprocessus unique et global permettant la dialogue entre deux processus quelconques

- possédant un système de protection unique et global

- possédant un mécanisme de gestion de processus unique

- possédant un ensemble des appels système unique disponible sur toutes les machines

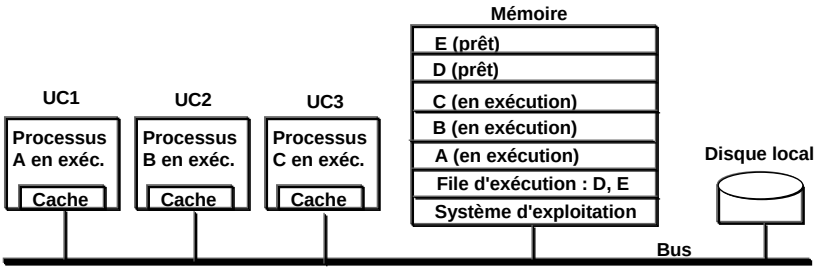
- possédant un noyau de système d'exploitation identique implanté sur toutes les machines

- possédant un mécanisme de gestion de mémoire identique

II- Partie II : systèmes distribués
II.1- Introduction

II.1.3- Concepts logiciels

II.1.3.4- Systèmes multiprocesseurs en temps partagé



- Un ensemble des processeurs "symétriques"
- Une mémoire commune et un espace disque commun
- Tous les programmes sont stockés dans la mémoire partagée
- Une file d'attente unique des processus exécutable se trouvent dans la mémoire partagée
- L'ordonnanceur travaille en section critique pour éviter que deux UC ne choisissent le même processus à exécuter
- L'exclusion mutuelle peut être réalisée en moyennant des sémaphores, moniteurs, ...

II- Partie II : systèmes distribués
II.1- Introduction

II.1.3- Concepts logiciels

II.1.3.5- Tableau de synthèse

Questions	S.E.R.	S.E.D.	S.E.M.
Ressemble-t-il à un monoprecesseur virtuel ?	Non	Oui	Oui
Doit-on avoir le même système d'exploitation ?	Non	Oui	Oui
Combien y a-t-il de copies du système d'exploitation ?	N	N	1
Comment la communication a-elle assurée ?	Fichiers partagés	Messages	Mémoire partagée
Faut-il utiliser des protocoles réseau unifiés ?	Oui	Oui	Non
Existe-il une seule file d'attente des exécutables ?	Non	Non	Oui
Le partage des fichiers a-t-il une sémantique bien définie	Non	Oui	Oui

II- Partie II : systèmes distribués
II.1- Introduction

II.1.4- Critères d'un système distribué

II.1.4.1- Transparence

Même mode d'utilisation que celui d'un système centralisé en temps partagé : niveau d'utilisateur et niveau de programme

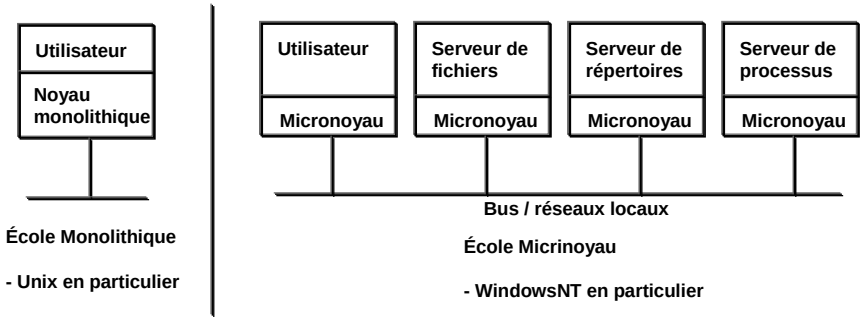
Types de transparence	Signification
Transparence à l'emplacement	L'utilisateur ne connaît pas où sont situées les ressources
Transparence à la migration	Les ressources peuvent être déplacées sans modification de leur nom
Transparence à la duplication	L'utilisateur ne connaît pas le nombre de copies existantes
Transparence à la concurrence	Plusieurs utilisateurs peuvent partagées en même temps les mêmes ressources
Transparence au parallélisme	Des tâches peuvent exécutées en parallèle sans que les utilisateurs le sachent

II- Partie II : systèmes distribués
II.1- Introduction

II.1.4- Critères d'un système distribué

II.1.4.2- Souplesse

La facilité de modification de configuration et d'extension



École Monolithique

- Unix en particulier

École Micrinoyau

- WindowsNT en particulier

II- Partie II : systèmes distribués
II.1- Introduction

II.1.4- Critères d'un système distribué

II.1.4.3- Fiabilité

Disponibilité

La disponibilité est la fraction de temps pendant laquelle le système est utilisable :

- **limiter de nombre des composants critiques**
- **dupliquer les parties clés des composants logiciels et matériels (redondance)**
- **maintenir la cohérence des copies (contradiction!!!)**

Sécurité

Les ressources doivent être protégées contre des utilisations abusives et malveillantes. En particulier le problème de piratage des données sur le réseau de communication

Tolérance aux pannes

Le système doit être conçu pour masquer les pannes aux utilisateurs. La panne de certains serveurs (ou leur réintégration dans le système après la réparation) ne doit pas perturber l'utilisation du système en terme de fonctionnalité (NFS sans état par exemple)

II- Partie II : systèmes distribués
II.1- Introduction

II.1.4- Critères d'un système distribué

II.1.4.4- Performances

Critères

- temps de réponse
- taux d'utilisation du système
- débit (nombre de travaux par heure)
- pourcentage utilisé de la bande de passante du réseau

Problèmes

- La communication est en général assez lente dans les systèmes distribués
- Le mécanisme de reprise sur panne consomme beaucoup de temps

Solutions

- minimiser les échanges de message
- maximiser des granules grosses (gros grains) et éviter des grains fins pour les calcul à distance
- réduire le champ d'application des mécanismes de reprises sur pannes

II- Partie II : systèmes distribués
II.1- Introduction

II.1.4- Critères d'un système distribué

II.1.4.5- Dimensionnement

Goulots d'étranglement potentiels

Concepts	Exemple
Composants centralisés	Un seul serveur de courrier pour tous les utilisateurs
Tables centralisées	Un seul annuaire en ligne
Algorithmes centralisés	Avoir un routage qui nécessite une connaissance totale du réseau

- Caractéristique des algorithmes distribués :

  - les algorithmes exécutables sur les postes clients (Java)
  - aucune machine n'a une information complète sur l'état du système
  - les machines prennent des décisions à partir des seules informations locales disponibles
  - la panne d'une machine n'empêche pas l'algorithme de fonctionner
  - on ne fait pas l'hypothèse de l'existence d'une horloge globale

II- Partie II : systèmes distribués
II.1- Introduction

II.1.5- Exercices

1) Donner deux avantages des systèmes distribués sur les systèmes centralisés

2) Un multiprocesseur à bus utilise un cache espion pour gérer une mémoire cohérente. Peut-on alors utiliser des sémaphores ?

3) Un multicalculateur comprenant 256 UC est organisé en treillis de 16x16. Quel est, dans le pire des cas, le délais de transmission d'un message (exprimer en nombre de passages d'un UC à la suivante) ?

4) Considérons maintenant un hypercube de 256 UC. Quel est alors le délais de transmission dans le pire de cas ?

5) Pourquoi certains serveurs NFS sont-ils conçu pour être sans état ?

6) NFS permet à une machine d'être à la fois client et serveur. Est ce utile ? Pourquoi?

7) Quelles sont les tâches essentielles d'un micronoyau ? Quels sont les avantages des micronoyaux sur les noyaux monolithiques ?

8) Un serveur de fichiers expérimental est opérationnel les 3/4 du temps et en débogage le reste du temps. Combien de fois ce serveur doit-il être dupliqué pour que sa disponibilité soit au moins de 99% ?

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.1- Introduction

Problème : communication inter-processus

- dans un système centralisé : communiquer par la mémoire partagée
- dans un système distribué : en général, l'absence de la mémoire partagée, on doit communiquer par messages avec des protocoles

Protocole

- ensemble de règles qui gère les communication inter-processus dans un système distribué

- il détermine un l'accord sur la façon dont les communications doivent d'effectuer

Plan

- les protocoles de communication (modèle OSI)

- le modèle Client / Serveur

- les Appels de procédure à distance (RPC=Remote Procedure Call)

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.2- Rappels des protocoles OSI

II.2.1.1- Introduction

O.S.I. = Open Systems Interconnection reference model

Système ouvert = un système qui prêt à communiquer avec un autre système ouvert en utilisant des règles standards (protocoles) quant au format au contenu et à la signification des messages qui sont envoyés ou reçus

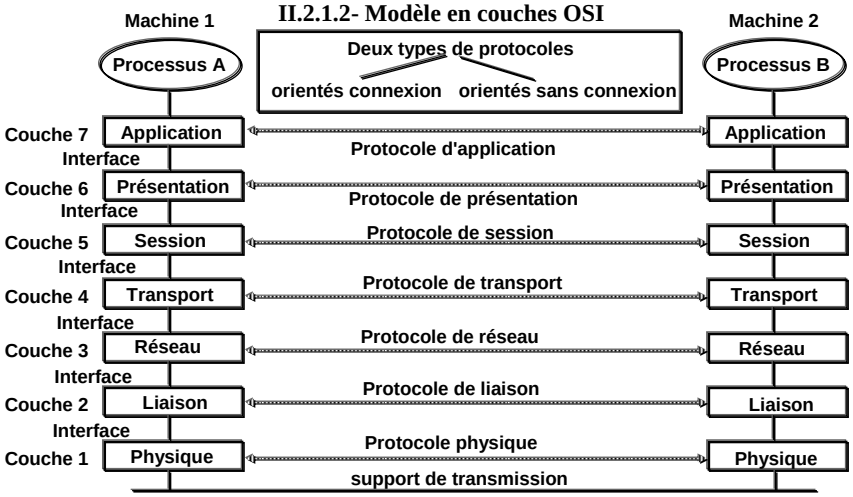
Objectif

Définition d'un ensemble de protocoles permettant un accord à tout niveau, depuis les détails de bas niveau de la transmission (voltage indiquant la valeur d'un bit, longueur du bit, ...) jusqu'à ceux de plus haut niveau de la présentation de données (longueur d'un nombre, d'une chaîne de caractères, format de représentation des données, ...)

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.2- Rappels des protocoles OSI

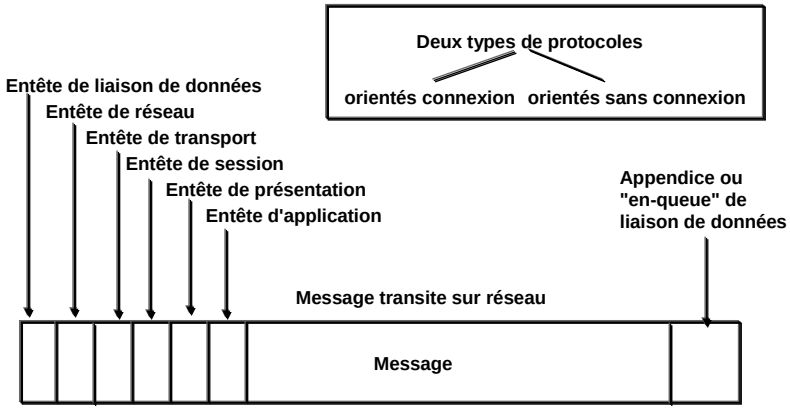
II.2.1.2- Modèle en couches OSI



II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.2- Rappels des protocoles OSI

II.2.1.2- Modèle en couches OSI



Page 29

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.2- Rappels des protocoles OSI

II.2.1.2- Modèle en couches OSI

- Couche physique : assurer la transmission : connecteurs physique, codages physiques adaptation au mode de transmission, ...

- Couche de liaison : assurer la transmission sans erreur entre deux extrémistes d'une liaison directe

- Couche réseau : assurer le routage des messages sur les réseaux d'interconnexion (gestion de congestion, choix de chemins, ...

- Couche transport : assurer une transmission fiable entre l'expéditeur et le destinataire contrôle de perte des paquets, l'ordre des paquets, ...

- Couche de session : extension de la couche transport : mécanismes de synchronisation de reprise sur panes, ... très peu utilisée en pratique !!!

- Couche de présentation : format de données, adaptation au terminal

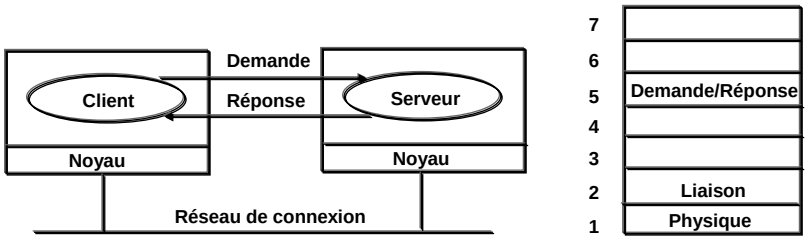
- Couche d'application: protocoles applicatifs : courrier électronique (X400), ftp, telnet,...

Page 30

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.2- Modèle Client/Serveur

II.2.2.1- Description



- Protocole simple sans connexion de type Question / Réponse

- Avantage : simplicité - efficacité

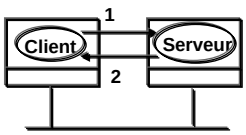
- Noyau : assurer l'envoi et réception des messages avec 2 primitives :
  - send (dest, &ptrm)
  - receive (adr, &prtm) (adr : l'adresse d'écoute du récepteur)

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.2- Modèle Client/Serveur

II.2.2.2- Adressage

Localisation connue

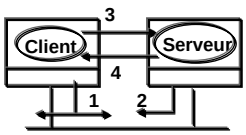


1 : demande vers S.p1
2 : réponse vers C.p2

- tout client doit connaître la localisation et le nom interne de tous les services. Il demande donc explicitement sous format :

machine.processus

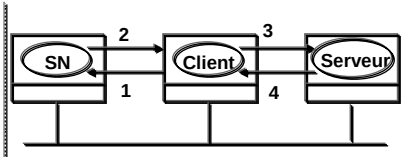
Localisation inconnue



1 : recherche par diffusion
2 : je suis là
3 : demande
4 : réponse

- chaque processus choisit un nom aléatoire telle sorte qu'il soit unique
- client diffuse une demande de localisation d'un process
- le serveur contenant ce processus donne son adresse machine

Localisation centralisée



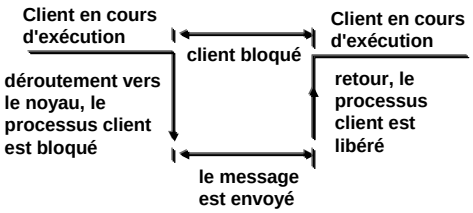
1 : demande à SN
2 : réponse de SN
3 : demande
4 : réponse

- les serveurs ont leur nom unique en code ascii
- un serveur gère la localisation et le nom interne des machines
- le client connaît seulement l'adresse du serveur de nom. Il lui demande de fournir l'adresse des serveurs à l'exécution

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.2- Modèle Client/Serveur

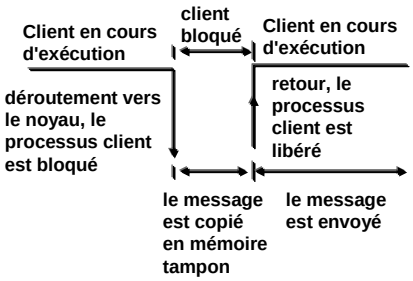
II.2.2.3- Primitives bloquantes et non bloquantes



- le processus appelant send ou receive sera bloqué pendant l'émission (ou réception) du message

- l'instruction qui suit send (ou receive) sera exécutée lorsque le message a été complètement émis (ou reçu)

- avantage : simple et claire, facile à la mise en oeuvre
- inconvénient : moins performant



- le processus appelant send sera libéré toute de suite, le message sera envoyé en // à l'exécution

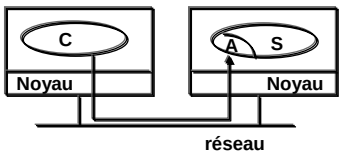
- avantage : performant
- inconvénient : perte de mémoire tampon, difficile à réaliser, risque de modif. du tampon

Page 33

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

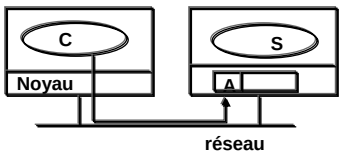
II.2.2- Modèle Client/Serveur

II.2.2.4- Primitives avec tampon ou sans tampon



- sans tampon : le message envoyé par le client sera copié par le noyau du serveur dans la zone ptrm et pour le processus A à l'adresse adr dans l'appel receive(adr,ptrm) déclenché par le processus A

- problème : si le processus client a appelé send avant que le processus serveur appelle receive alors, le noyau du serveur n'a ni adresse du processus serveur ni l'adresse de l'endroit où il doit copier le message reçu !!!



- avec tampon : les messages envoyés par les clients sont déposés dans une boite aux lettres. Le processus serveur A qui appelle un receive va déclencher une recherche par le noyau dans la boite aux lettres. Si le message existe, le noyau va le copier dans la zone de mémoire ptrm et libérer le processus A. Sinon l'appel receive sera bloqué pour l'attente de l'arrivée du message

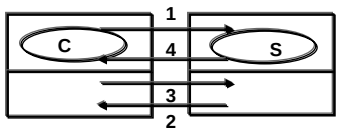
- problème : quand la boite aux lettres est saturée, on doit refuser tous les messages qui arrivent !!!

Page 34

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

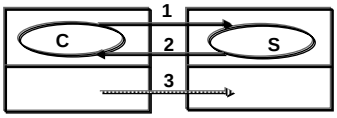
II.2.2- Modèle Client/Serveur

II.2.2.5- Primitives fiables et non fiables



Message avec accusés de réception individuels

1- Demande (client au serveur)
2- ACK (noyau à noyau)
3- Réponse (serveur à client)
4- ACK (noyau à noyau)



Message avec la réponse = 1 accusé de réception

1- Demande (client au serveur)
2- Réponse (serveur à client)
3- ACK dans certain cas (noyau à noyau)
(si le calcul est cher, on peut bloquer le résultat au serveur jusqu'à qu'il reçoit un ACK du client (3)

On peut avoir un compromis entre 2 solutions précédantes :

1- Demande (client au serveur)
2- ACK éventuellement si la réponse n'est pas revenue dans certain temps (noyau à noyau)
3- Réponse servant ACK si dans le lape de temps prévu (serveur à client)
4- ACK éventuellement (noyau à noyau)

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.2- Modèle Client/Serveur

II.2.2.6- Résumé

Objet	Option 1	Option 2	Option 3
Adressage	numéro de la machine	adresse choisie par le proc.	nom ASCII avec SN
Blocage	primitives bloquantes	primitives non-bloquantes avec copie vers le noyau	primitives non-bloquantes avec interruption
Mémorisation	pas de tampon	tampon temporaire	boite aux lettre
Fiabilité	non fiable	demande-ACK-réponse-ACK	demande--réponse-ACK

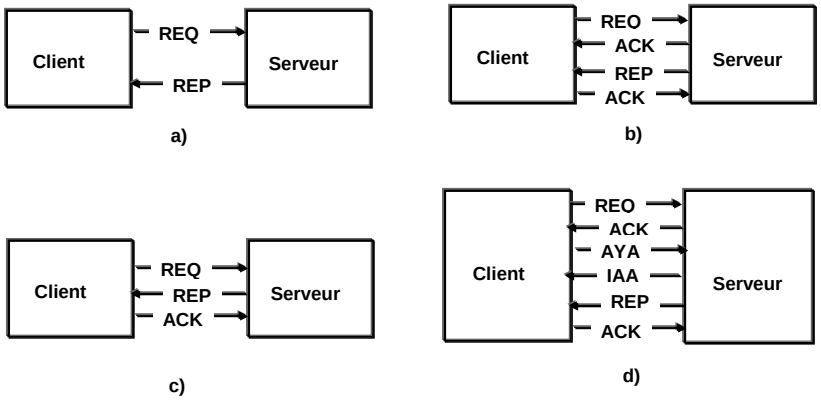
Code	Type de paquet	source-dest.	Description
REQ	demande	client-serveur	le client désire un serveur
REP	réponse	serveur-client	la réponse du serveur au client
ACK	accusé de réception	noyau-noyau	le paquet précédent est arrivé
AYA	es-tu vivant ?	client-serveur	le serveur fonctionne-t-il ?
IAA	Je suis vivant	serveur-client	le serveur n'est pas en panne
TA	essaye à nouveau	serveur-client	le serveur est saturé
AU	adresse inconnue	serveur-client	aucun processus sur serveur utilise cette adresse

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.2- Modèle Client/Serveur

II.2.2.6- Résumé

Exemples des échanges des paquets dans le protocole Client/Serveur



II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.3- Appels de procédures à distance

II.2.3.1- Introduction

Problème du modèle Client/Serveur

Concept basé sur l'échange explicite de données donc orienté Entrées/Sorties qui ne peut pas être le principe clé d'un système distribué

Appels de procédures à distance

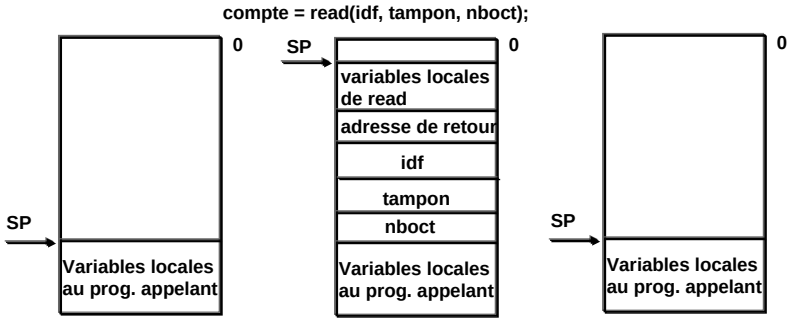
Concept introduit par Birrell et Nelson en 1984 : lorque un processus sur la machine A appelle une procédure sur une machine B, le processus sur A est suspendu pendant l'exécution de la procédure appelée se déroule sur B. Des informations peuvent être transmises de l'appelant vers l'appelé ou l'inverse par des paramètres

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.3- Appels de procédures à distance

II.2.3.2- Opération de base de RPC

Mécanisme d'appel de procédure dans un système centralisé



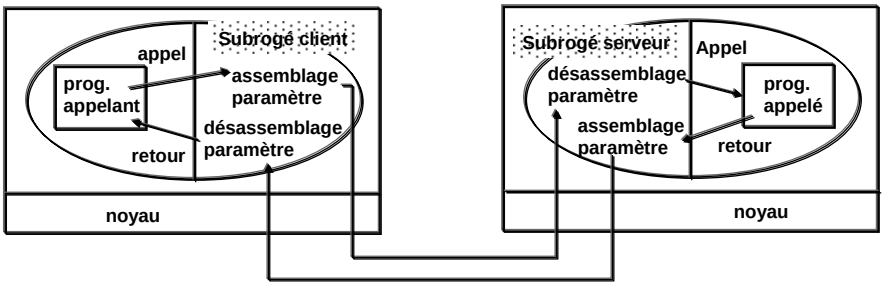
- modes de passage : par valeur ou par pointeur
- nature des paramètres : entré - sorti - entré / sorti

Page 39

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.3- Appels de procédures à distance

II.2.3.2- Opération de base de RPC

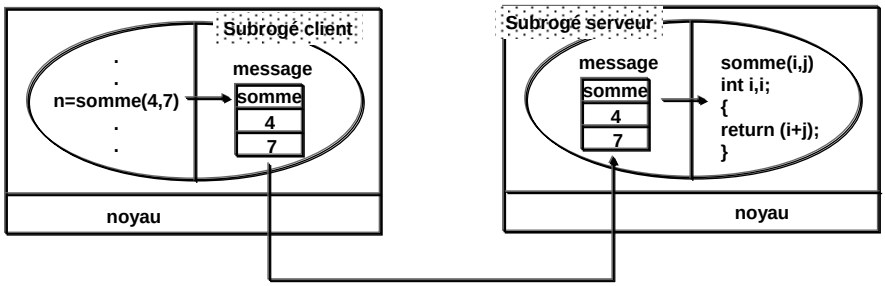


Page 40

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.3- Appels de procédures à distance

II.2.3.3- Passage de paramètres



Page 41

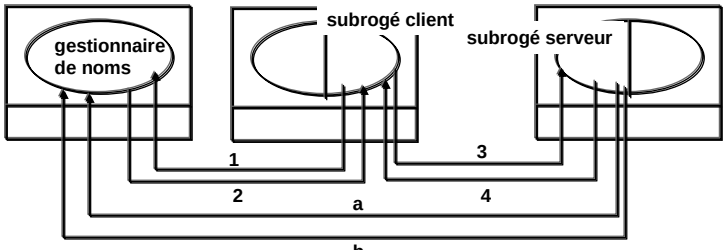
II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.3- Appels de procédures à distance

II.2.3.4- Nommage dynamique (dynamic binding)

Problème

la localisation du serveur par le client : écrire dans le code client l'adresse du serveur ou utiliser le mécanisme de nommage dynamique



a : enregistrer un nom (exportation de l'interface)
b : supprimer un nom

1- recherche un nom (importation)
2- retourne l'identificateur et descripteur
3 et 4- RPC

Page 42

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.3- Appels de procédures à distance

II.2.3.5- Sémantique de RPC en cas d ’échec

Cinq classes d ’erreurs

1. **Le client est incapable de localiser le serveur**
2. **La demande du client au serveur est perdue**
3. **La réponse du serveur au client est perdue**
4. **Le serveur tombe en panne après avoir reçu une demande**
5. **Le client tombe en panne après avoir envoyé une demande**

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

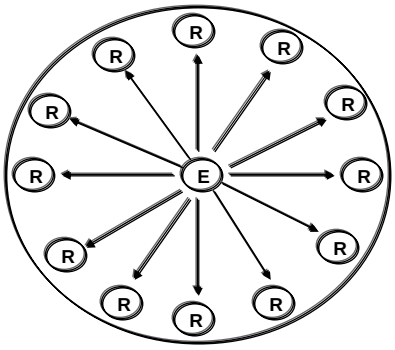
II.2.4- Communication de groupe (multipoints)

II.2.4.1- Introduction

Point-à-Point

Généralisation du RPC

Un émetteur doit communiquer en parallèle à n récepteurs regoupés dans un groupe sémantique



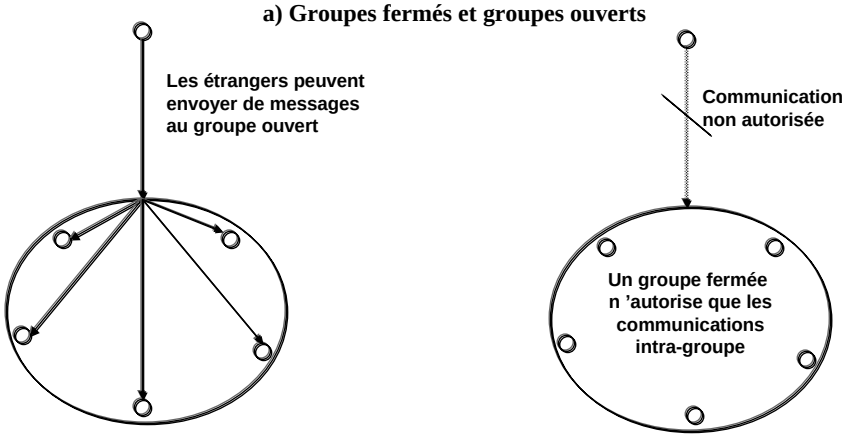
Un-à-Plusieurs

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.4- Communication de groupe (multipoints)

II.2.4.2- Classement

a) Groupes fermés et groupes ouverts



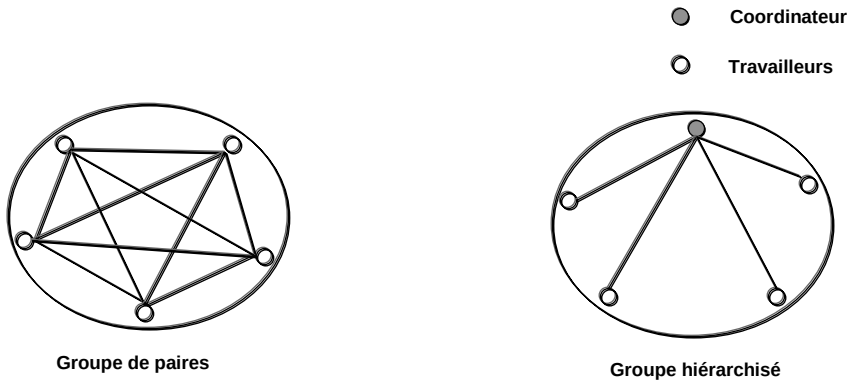
Page 45

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.4- Communication de groupe (multipoints)

II.2.4.2- Classements

b) Groupes de paires ou groupes hiérarchisés



Page 46

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.4- Communication de groupe (multipoints)

II.2.4.3- Gestion des groupes

- Création, dissolution d ’un groupe
- Adhésion à un groupe ou retrait d ’un groupe

Gestion centralisée

- Un serveur de groupes qui gère une base de données des groupes et une base de données des membres
- Avantages : efficace, claire et facile à implanter
- Inconvénients : fragile devant les pannes

Gestion distribuée

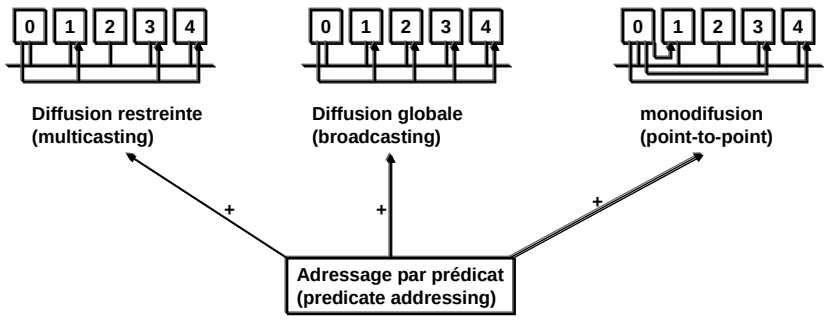
- Un groupe se crée s ’il y a au moins un adhérant
- pour adhérer à un groupe, le processus demandeur doit envoyer sa demande à tous les membres du groupe
- Avantage : Robustesse
- Inconvénient : complexe et coûteuse

Page 47

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.4- Communication de groupe (multipoints)

II.2.4.4- Adressage des groupes



Page 48

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.4- Communication de groupe (multipoints)

II.2.4.5- Atomicité

Propriété de diffusion atomique (atomic broadcast)

Un message envoyé à un groupe doit être reçu par tous ses membres ou par aucun

Algorithme de « relais intra-groupe » (Joseph et Birman 1989)

1- L 'expéditeur envoie un message à tous les membres d 'un groupe (des temporisateurs sont armés et des retransmissions faites si nécessaire)

2- Tout membre de ce groupe, qui a reçu ce message pour la première fois, doit l 'envoyer à tous les membres du groupe (des temporisateurs sont armés et des retransmissions faites si nécessaire)

3- Tout membre de ce groupe, qui a reçu ce message plus qu 'une fois, doit simplement le détruire

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

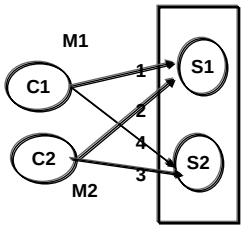
II.2.4- Communication de groupe (multipoints)

II.2.4.6- Séquencement

Propriété de séquencement

Soient A et B 2 messages à envoyer à un groupe de processus G. Si A est envoyé avant B sur le réseau, ils doivent être reçus par tous les membres du groupe G dans le même ordre : A puis B

MAJ Incohérente



Solutions

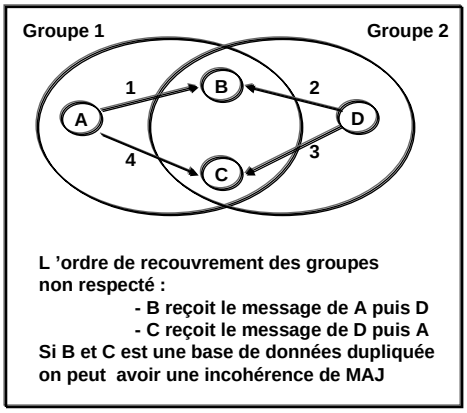
- Séquencement global (global time ordering) :
conserver, en réception, l 'ordre des messages à émission.
Si C1 envoie M1 avant que C2 envoie M2, alors le système doit assurer que tous les membres de G reçoivent M1 avant M2

- Séquencement cohérent (Consistent time ordering) :
Le système établir l 'ordre d 'envoie des messages et assurer que cet ordre soit respecté à la réception des messages.

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.4- Communication de groupe (multipoints)

II.2.4.7- Groupes non disjoints



L 'ordre de recouvrement des groupes non respecté :
- B reçoit le message de A puis D
- C reçoit le message de D puis A

Si B et C est une base de données dupliquée on peut avoir une incohérence de MAJ

Solution

Introduction des mécanisme de séquencement qui tient compte du recouvrement des groupes

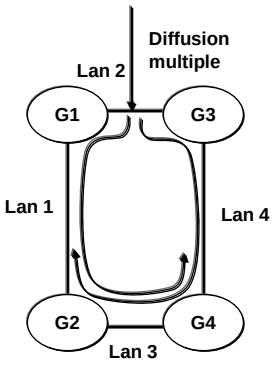
Problème

La mise en œuvre est compliquée et coûteuse

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.4- Communication de groupe (multipoints)

II.2.4.8- Elasticité



Propriété

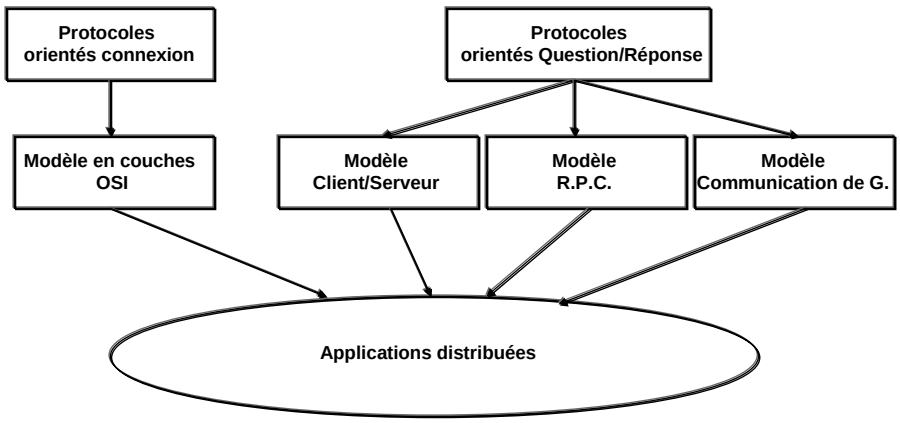
La communication de groupe doit être invariante par rapport à la taille du groupe : L'algorithme doit être efficace dans un petit groupe comme dans un grand groupe (nombre de membre et taille de l'espace)

Problèmes

- beaucoup d 'algorithmes fonctionnent très bien dans des petits groupes mais deviennent non fiable dans des grands groupe
- la taille de réseau avec la présence des passerelles rend plus compliqué le problème de communication de groupe

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.5- Résumé

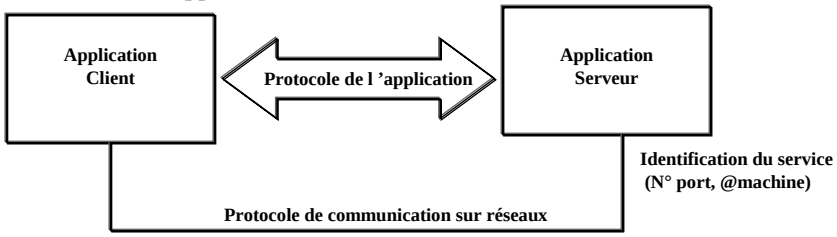


Page 53

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.6- Approches pratiques

Applications Clients / Serveur : Modèle de base



Exemple : Services Internet

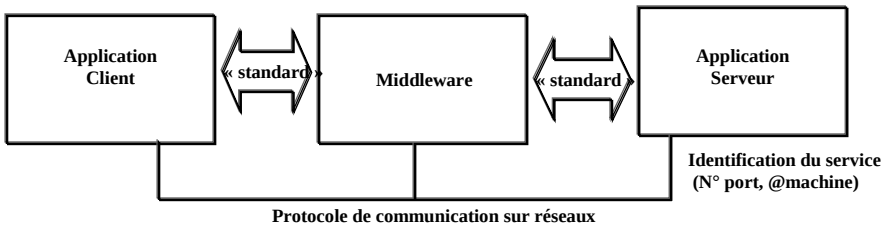
- **Telnet : Terminal Network protocol**
- **Ftp : File transport Protocol**
- **Courriers électroniques : SMTP, POP3, …**
- **Web : HTTP et Navigateur**
- **...**

Page 54

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.6- Approches pratiques

Applications Clients / Serveur : Modèle 3-Tiers



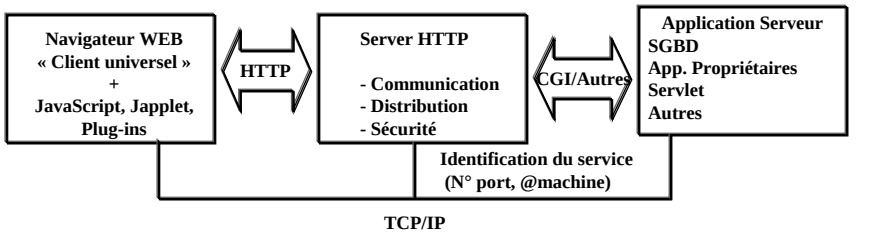
Exemples

- **ODBC : Object DataBase Connectivity**
- **JDBC : Java DataBase Connectivity**
- **Applications intra/internet**
- **Message Queues**
- **CORBA, DCOM / ACTIVEX, Java Rmi**

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.6- Approches pratiques

Applications Clients / Serveur sur web



Exemples

- **Commerces électroniques**
- **Applications Intranet / Internet / Extranet**
- **Télétravail**
- **Téléenseignement**
- **...**

II- Partie II : systèmes distribués
II.2- Communications dans les systèmes distribués

II.2.6- Exercices

1- Si les primitives de communication dans un système client/serveur sont non bloquantes, un appel *send* rendla main avant que le message ne soit complètement envoyé. Pour réduire les pertes de temps, certains systèmes ne copient pas les données dans le noyau, mais les transmettent directement à partir de l 'espace utilisateur. Pour un tel système, donnez 2 façons d 'avertir l 'appelant que la transmission est terminée et qu'il peut donc réutiliser le tampon d'émission

2- Pour chacune des applications suivantes, pensez vous qu'une sémantique du type « une fois au moins » soit meilleure qu'une sémantique du type « une fois au plus » ? Justifiez :

a) lecture et écriture de fichier sur un serveur de fichiers;

b) compilation d 'un programme;

c) opérateur bancaire à distance.

3- Supposons que le temps pris pour exécuter un RPC vide (sans données) soit de 1ms et qu 'il faille 1,5ms pour chaque Ko de données. Quel sera le temps mis pour lire 32Ko sur un serveur de fichiers en un seul RPC de 32Ko ? Et en 32 RPC de 1Ko ?

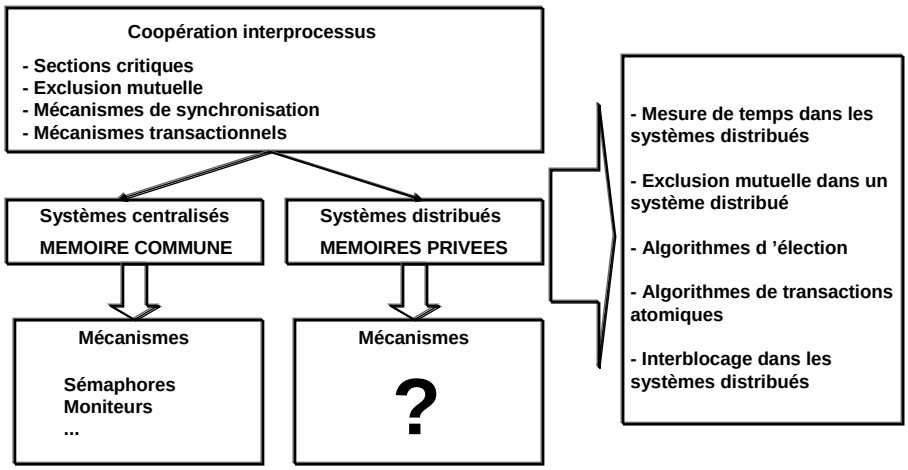
4- Proposer un mécanisme de gérer l 'appartenance à un groupe en supposant que le réseau permet la diffusion globale atomique

5- Considérons un système distribué dans lequel toutes les machines sont des multiprocesseurs redondants de telle sorte que la probabilité de panne soit quasi nulle. Proposez une méthode simple pour implanter la diffusion atomique avec séquencement globale un utilisant seulement la monodiffusion. (Indication : organisez ces machine en un anneau logique)

Page 57

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.1- Introduction



Page 58

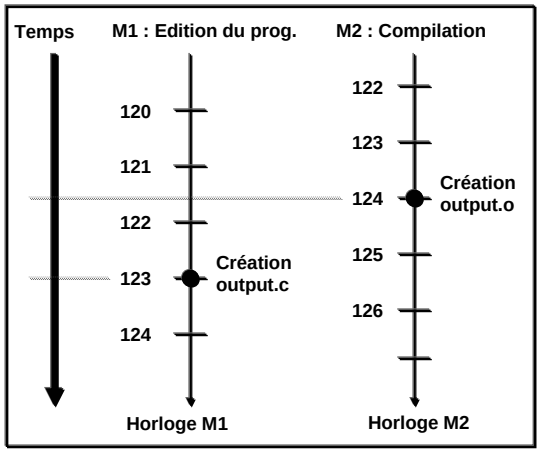
II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.2- Synchronisation d ’horloge

II.3.2.1- Introduction

Propriétés d’algorithmes distribués

1- les informations importantes sont disséminées sur plusieurs machines
2- les processus prennent des décisions fondées uniquement sur l ’information disponible localement
3- la garantie de la fiabilité doit être renforcée (talon d ’Achille à éviter)
4- il n ’existe pas d ’horloge commune ni de source de temps universel



II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.2- Synchronisation d ’horloge

II.3.2.2- Horloges logiques

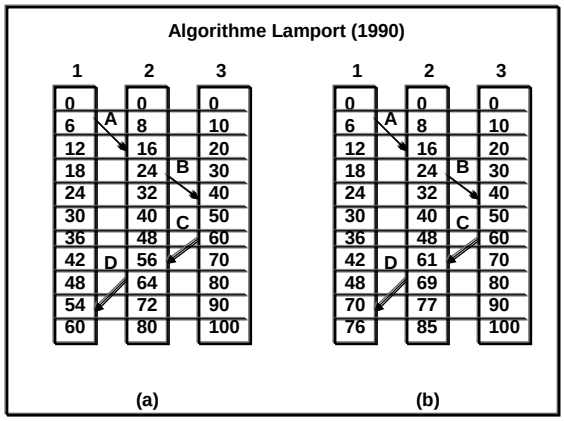
Horloge logique

Temporisateur associé à chaque processeur permettant générer un compteur de temps local (des impulsion d ’horloge)

Problème de dérive d ’horloge

Dans un systèmes multiprocsseurs au fil de temps, la synchronisation des horloges se dégrade

Introduction des erreurs



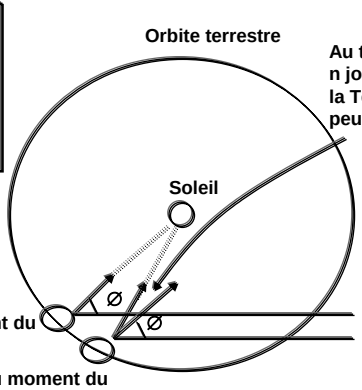
II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.2- Synchronisation d ’horloge

II.3.2.3- Horloges physiques

- Besoin pour les applications « temps réel »
- Nécessité de plusieurs horloges

- Problèmes :
  - synchronisation avec l’horloge universelle
  - synchronisation entre les horloges



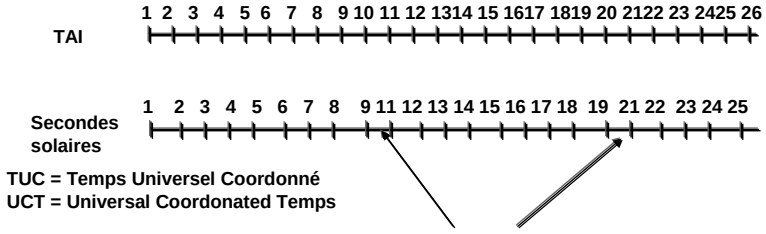
Page 61

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.2- Synchronisation d ’horloge

II.3.2.3- Horloges physiques

Réglage du décalage entre TAI et l ’horloge solaire : 1 seconde sautée quand le temps solaire devient supérieur à 800ms



TUC = Temps Universel Coordonné
UCT = Universal Coordonated Temps

Secondes sautées introduites quand il devient nécessaire de se synchroniser avec le soleil

Page 62

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.2- Synchronisation d 'horloge

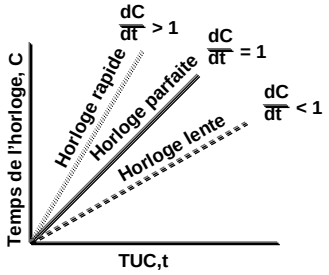
II.3.2.4- Algorithmes de synchronisation d'horloges physiques

Modèle commun de communication

- Chaque machine a 1 compteur de temps qui produit une interruption H fois /s. Quand ce compteur vient à épuisement, le gestionnaire d 'interruption ajoute 1 à l'horloge logique qui sont réglée à partir une date fixe en commun
- Soit C la valeur de cette horloge, soit t le temps TUC, la valeur de l 'horloge sur la machine p est Cp(t)
- Dans un monde parfait, on a Cp(t)=t pour tout couple (p,t). En d 'autres termes, dC/dt devrait idéalement être égal à 1
- En réalité, il existe une constante ρ, appelée *taux de dérive maximum*, telle que :

$$1 - \rho \leq dC/dt \leq 1 + \rho$$

- Si 2 horloges dérivent du TUC dans des directions opposées, leur décalage après un temps Δt sera au maximum 2ρΔt



Pour garantir que deux horloges ne dérivent pas plus que δ, alors les horloges doivent être resynchronisées au moins toute les δ/2ρ secondes

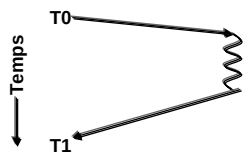
II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.2- Synchronisation d 'horloge

II.3.2.4- Algorithmes de synchronisation d 'horloges physiques

Algorithme de Cristian (1989)

Machine Client Serveur de temps



Chaque machine interroge périodiquement (< toutes les δ/2ρ s) le serveur de temps pour avoir l 'heure courante afin de pouvoir corriger d 'une manière adéquate son horloge

La correction d 'une horloge doit se faire graduellement. On retarde (ou accélère) une horloge rapide (ou lente) en diminuant (ou ajoutant) un certain nombre de ns de la (au) constante de temps utilisée pour avancer l 'horloge logique dans la routine d 'interruption. Cette correction a pour une durée nécessaire correspondant au décalage de l 'horloge locale par rapport le C_{TUC}

La machine Client doit tenir compte le temps de propagation des messages entre lui et le serveur de temps. La valeur de correction est

$$C_{cor} = (T1-T2)/2 + C_{TUC}$$

Si le temps de gestion I 'interruption I sur serveur est connu, on peut le tenir compte dans la formule :

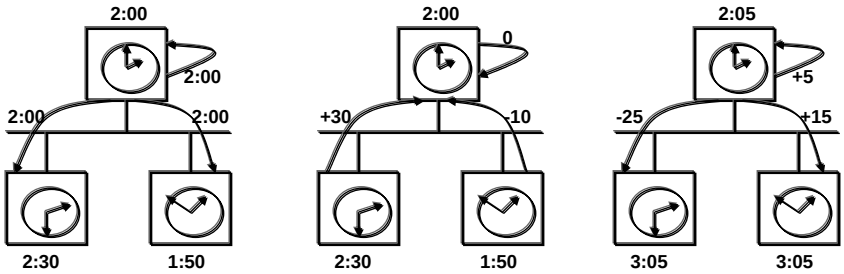
$$C_{cor} = (T1-T2-I)/2 + C_{TUC}$$

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.2- Synchronisation d'horloge

II.3.2.4- Algorithmes de synchronisation d'horloges

Algorithme de Berkeley



Dans UNIX de Berkeley (Gusella et Zatti, 1989), le serveur de temps est actif et scructe périodiquement chaque machine pour luis demande l 'heure. A partir de ses réponses, il calcule le temps moyen et demande à toutes les machines de régler son horloge

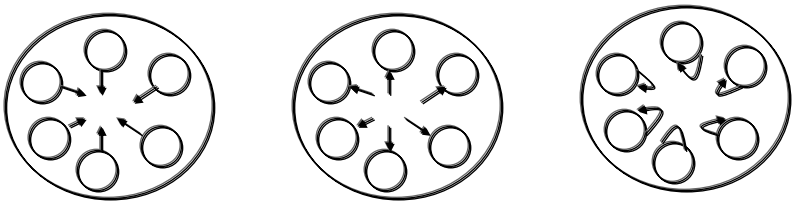
Page 65

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.2- Synchronisation d'horloge

II.3.2.4- Algorithmes de synchronisation d'horloges

Algorithmes basés sur la moyenne



Chaque machine diffuse son heure au début d 'une période de temps [T0+iR, T0+(i+1)R]

Chaque machine démarre un temporisateur et collecte tous les messages arrivés durant l 'intervalle S

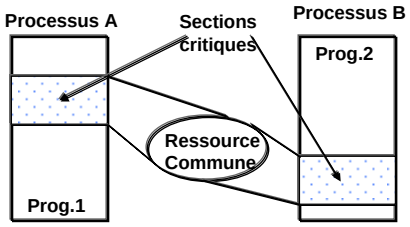
Chaque machine exécute un même algorithme à partir des données reçu pour l 'obtention d 'une heure moyenne

Page 66

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.3- Exclusion mutuelle

II.3.3.1- Rappels



Dans les systèmes centralisés

Sémaphores
Moniteurs
Compteurs d 'événements
Messages

Contrainte dans un système distribué

Les mécanismes basés sur une mémoire commune ne sont plus valables

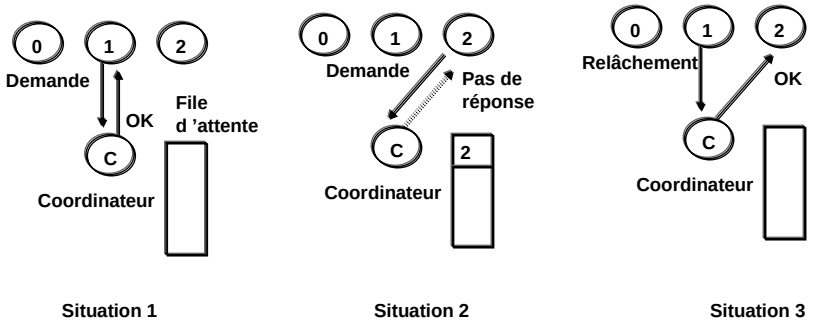
Exclusion mutuelle

A tout instant du système, il y a au maximum 1 processus qui peut entrer dans une section critique

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.3- Exclusion mutuelle

II.3.3.2- Un algorithme centralisé

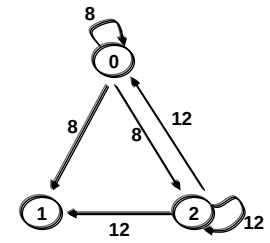


II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

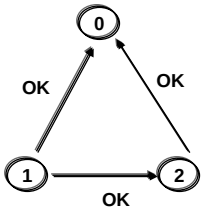
II.3.3- Exclusion mutuelle

II.3.3.3- Un algorithme distribué (Ricart & Agrawala-1981)

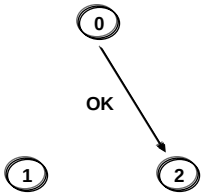
Hypothèse : il existe un ordre total sur les événements (ex. utilisation de l'algorithme Lamport)



P0 et P2 veulent entrer dans une même section critique



P0 est plus ancien, il gagne et entre en section critique

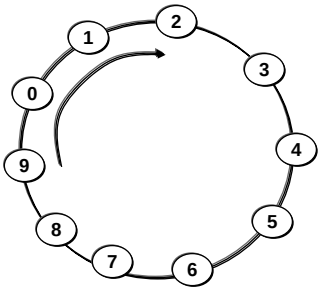


P1 entre en section critique

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.3- Exclusion mutuelle

II.3.3.4- Un algorithme de type anneau à jeton



- établir un anneau logique entre les processus
- introduire un jeton dans l 'anneau
- le processus possédant le jeton peut entrer en section critique. Il garde le jeton jusqu'à sa sortie de la section critique. Il passe ensuite le jeton au processus suivant dans l 'anneau logique
- sinon, il passe le jeton au processus suivant dans l 'anneau logique

Problèmes
- la perte de jeton -> algorithmes de détection et régénérer le jeton
- la panne d 'un processus -> régénérer l 'anneau ou l 'algorithme de détection de la mort du voisin

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.3- Exclusion mutuelle

II.3.3.5- Conclusion et comparaison

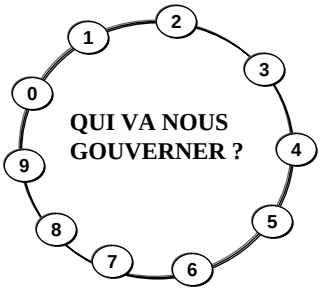
Type d'algorithme	Nombre de messages par entrée/sortie	Temps avant entrée	Problèmes
Centralisé	3	2	Plantage du coordinateur
Distribué	2(n-1)	2(n-1)	Plantage d 'un processus
Anneau à jeton	1 à ∞	0 à n-1	Perte de jeton, plantage de processus

L 'algorithme distribué est beaucoup plus sensible au plantage que l 'algorithme centralisé !

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.4- Algorithmes d'élection

II.3.4.1- Introduction



Hypothèses

- chaque processus a 1 numéro d 'identification unique
- il n 'existe qu'un processus par machine
- chaque processus connaît l 'identité de tous les autres processus, mais ne sait pas qui est actif et qui ne l 'est pas

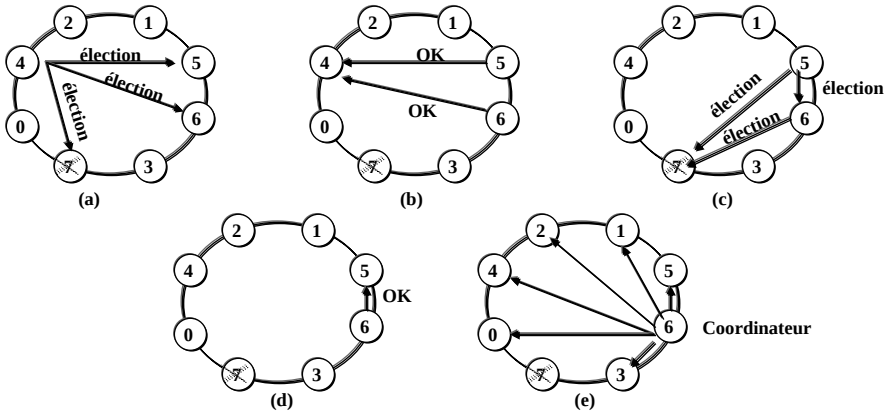
But

organiser une élection qui garantie que :
- le processus élu soit actif
- l 'élection soit terminée avec l 'accord de tous les processus sur l 'identité du processus élu

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.4- Algorithmes d'élection

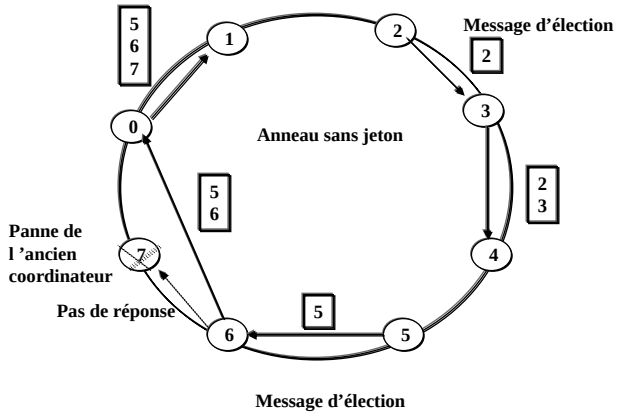
II.3.4.2- Algorithme du plus fort



II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.4- Algorithmes d'élection

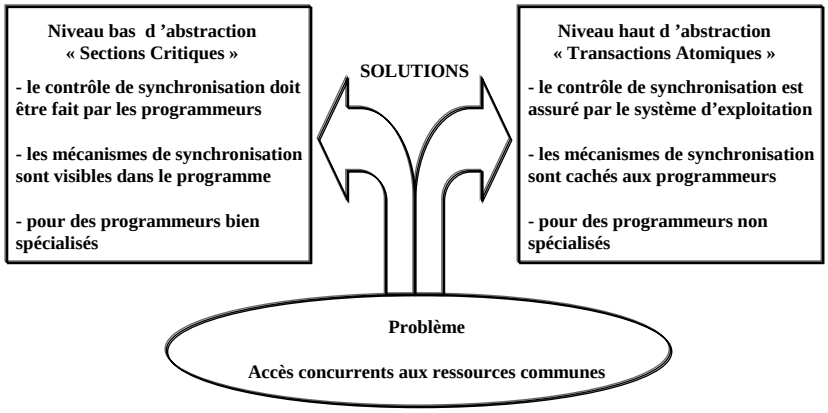
II.3.4.3- Algorithme pour anneau



II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.1- Introduction

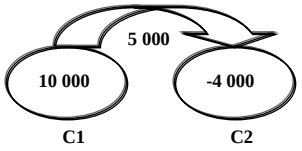


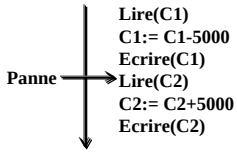
Page 75

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

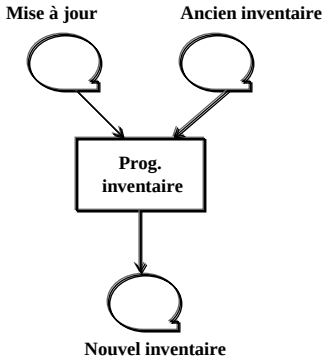
II.3.5.1- Introduction





Résultat : C1 = 5 000 et C2 = -4 000

LA PERTE D 'ARGENT



Un système tolérant aux pannes

Page 76

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.2- Modèle de transaction

TRANSACTION

On appelle transaction l'unité de traitement séquentiel, exécutée pour le compte d'un processus, appliquée à une mémoire stable cohérente, restitue une mémoire stable cohérente

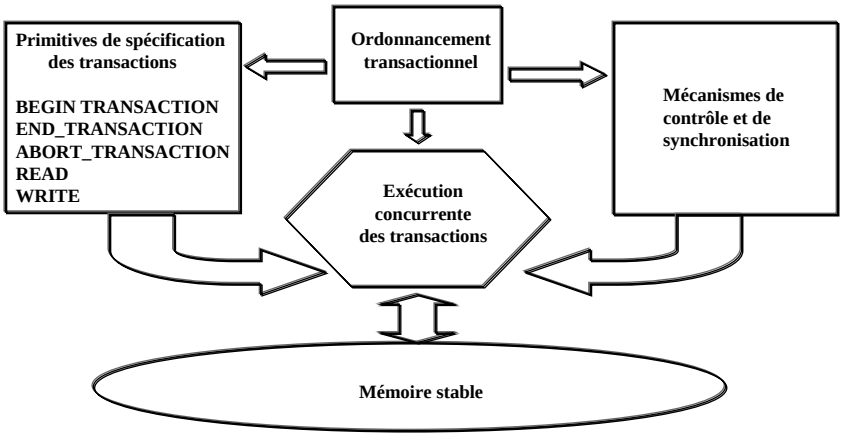
PROPRIÉTÉS

- **Atomicité :** **Une transaction est une unité logique indivisible de traitement qui est soit complètement exécutée soit complètement abandonnée**
- **Sérialité :** **Les transactions concurrentes n 'interfèrent pas entre elles**
- **Permanence :** **Si une transaction est validée, tous les changements par elle sur la mémoire stable sont devenus définitifs**

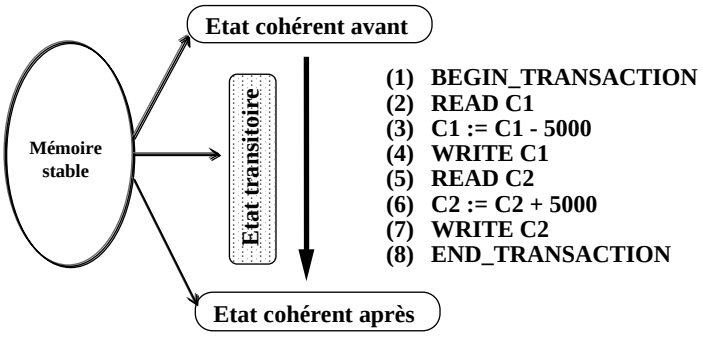
II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.2- Modèle de transaction

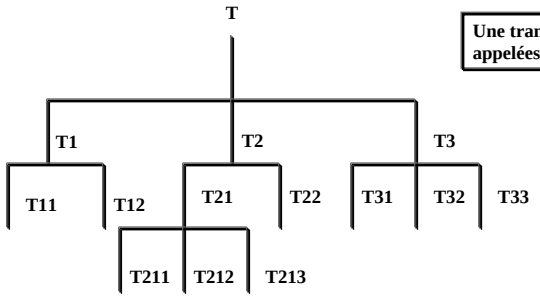


II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués
II.3.5- Transaction atomiques distribuées
II.3.5.2- Modèle de transaction



La mise-à-jour sûre par une transaction

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués
II.3.5- Transaction atomiques distribuées
II.3.5.2- Modèle de transaction
Transactions imbriquées



Une hiérarchie des transactions impliquées

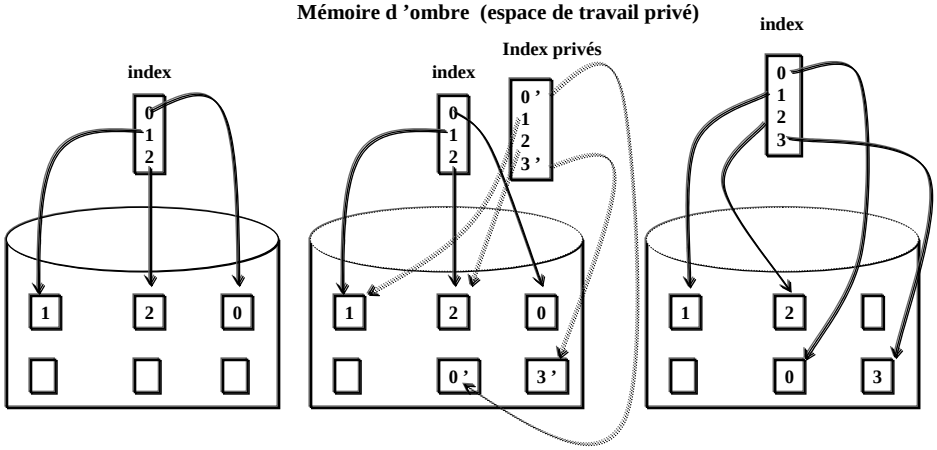
Une transaction peut contenir des sous-transactions appelées transactions imbriquées

- les filles d 'une transaction n e sont pas nécessairement exécutées sur le même processeur

- la profondeur d 'imbrication peut-être quelconque, ce qui engendre certains problèmes de gestion de cohérence et de performance

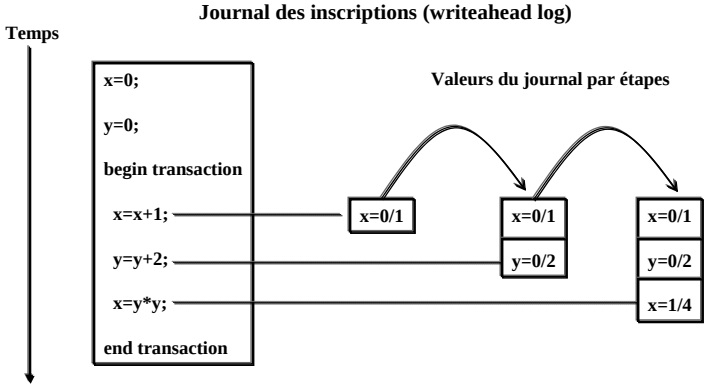
II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués
II.3.5- Transaction atomiques distribuées
II.3.5.3- Implantation de l'atomicité de transaction

Mémoire d 'ombre (espace de travail privé)



II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués
II.3.5- Transaction atomiques distribuées
II.3.5.3- Implantation de l'atomicité de transaction

Journal des inscriptions (writeahead log)

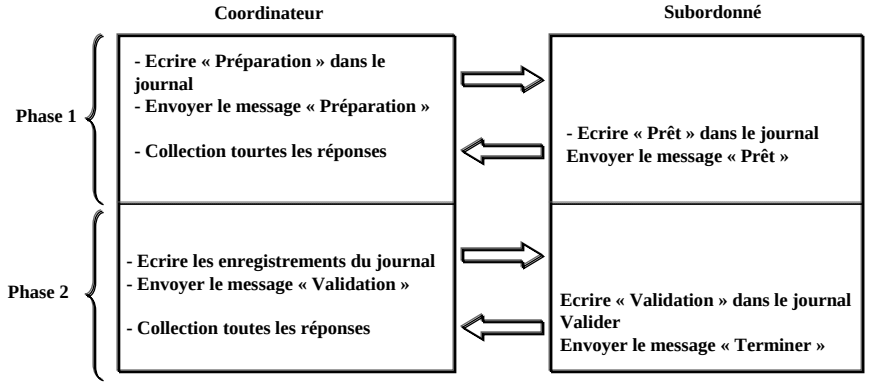


II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.3- Implantation de l'atomicité de transaction

Protocole de validation en deux phases (GRAY 78)



Page 83

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Implantation de sérialité de la transaction

Définitions de base

- **CONTRÔLEUR (SCHEDULER) : Un contrôleur transactionnel est un module système chargé de contrôler les accès concurrent aux données**
- **GRANULE : On appelle granule l'unité de données dont l'accès est contrôlé individuellement par un contrôleur**
- **OPÉRATION : Une opération est une suite d'actions élémentaires accomplissant une fonction sur un granule en respectant sa cohérence interne**
- **RÉSULTAT DE L'OPÉRATION : Le résultat d'une opération est l'état du granule concerné après l'application de l'opération considérée anssi que les effets de bords provoqués par l'opération**
- **ORDONNANCEMENT DE TRANSACTIONS (SCHEDULE - LOG) : Une ordonnancement des transactions (T1, T2, ..., Tn) est une séquence d'actions élémentaires obtenues en interclassant les diverses actions des transactions concernant**

Page 84

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Implantation de sérialité de la transaction

Définitions de base : Exemples

T1	T2
Lire A → a1	Lire A → a2
a1 + 1 → a1	a2 * 2 → a2
Ecrire a1 → A	Ecrire a2 → A
Lire B → b1	Lire B → b2
b1 + 1 → b1	b1 * 2 → b2
Ecrire b1 → B	Ecrire b2 → B

ORDONNANCEMENT 1		ORDONNANCEMENT 2
T1: Lire A → a1		T2: Lire A → a2
T1: a1 + 1 → a1	O11	T2: a2 * 2 → a2
T1: Ecrire a1 → A		T1: Lire A → a1
T2: Lire A → a2		T1: a1 + 1 → a1
T2: a2 * 2 → a2	O21	T2: Ecrire a2 → A
T2: Ecrire a2 → A		T2: Lire B → b2
T1: Lire B → b1		T2: b1 * 2 → b2
T1: b1 + 1 → b1	O12	T1: Ecrire a1 → A
T1: Ecrire b1 → B		T1: Lire B → b1
T2: Lire B → b2		T1: b1 + 1 → b1
T2: b1 * 2 → b2	O22	T1: Ecrire b1 → B
T2: Ecrire b2 → B		T2: Ecrire b2 → B

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

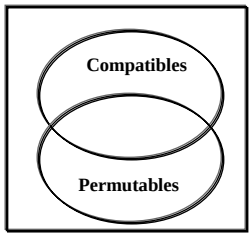
II.3.5- Transaction atomiques distribuées

II.3.5.4- Implantation de sérialité de la transaction

Propriétés des opérations

Opérations compatibles

Deux opérations Oi et Oj sont compatibles si et seulement si quel que soit l 'exécution simultanée de Oi et Oj le résultat de cette exécution est identique que celui d'une exécution séquentielle de ces opérations, soit de Oi suivie par Oj soit de Oj suivie par Oi



Opérations permutables

Deux opérations Oi et Oj sont permutables si et seulement si toute exécution séquentielle de Oi suivie par Oj donne le même résultat que celle de Oj suivie par Oi

O11:	O21:	O12:	O22:	O13:	O23:
Lire A → a1	Lire A → a2	Lire A → a1	Lire A → a2	Lire A → a1	Lire A → a2
a1 + 1 → a1	a2 * 2 → a2	a1 + 1 → a1	a2 * 2 → a2	a1 + 1 → a1	a2+10 → a2
Imprimer a1	Ecrire a2 → A	Imprimer a1	Imprimer a2	Ecrire a1 → A	Ecrire a2 → A

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Implantation de sérialité de la transaction

Ordonnancements sérialisables : définition

ORDONNANCEMENT SERIALISE - SUCCESSION (SERIAL SCHEDULE)

Un ordonnancement O de T1, .., Tn est dit sérialisé s'il est une exécution séquentiel des ces transactions (formellement s'il existe une permutation š de (1, 2, ..,n) telle que O = < Tš(1), Tš(2), .., Tš(n) >)

⇒

PROPRIÉTÉ

Une succession est une exécution sans perte d'opérations ni inconsistances

ORDONNANCEMENT SÉRIALISABLE (SERIALIZABLE SCHEDULE)

Un ordonnancement quelconque de T1, .., Tn est dit sérialisable si et seulement s'il donne pour chaque transaction participante le même résultat qu'une succession de T1, .., Tn

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Implantation de sérialité de la transaction

Ordonnancements sérialisables : caractérisation

Transformations de base

- Séparation des opération : remplacer les exécutions simultanées des couples d 'opérations compatibles E(Oi,Oj) par l 'exécution séquentielle équivalente (soit <Oi,Oj> soit <Oj,Oi>

- Permutation d 'opérations: remplacer l 'ordre d 'exécution d 'un couple d 'opérations permutables <Oi,Oj> par l 'ordre inverse <Oj,Oi>

Théorème 1

Une condition suffisante pour qu ' un ordonnancement soit sérialisable est qu 'il puisse être transformé par l 'application des deux transformations de base en un ordonnancement sérialisé (succession) des transactions composantes

ORDONNANCEMENT 1		ORDONNANCEMENT 2	
T1: Lire A → a1	O11	T1: Lire A → a1	O11
T1: a1 + 1 → a1		T1: a1 + 1 → a1	
T1: Ecrire a1 → A		T1: Ecrire a1 → A	
T2: Lire A → a2	O21	T1: Lire B → b1	O12
T2: a2 * 2 → a2		T1: b1 + 1 → b1	
T2: Ecrire a2 → A		T1: Ecrire b1 → B	
T1: Lire B → b1	O12	T2: Lire A → a2	O21
T1: b1 + 1 → b1		T2: a2 * 2 → a2	
T1: Ecrire b1 → B		T2: Ecrire a2 → A	
T2: Lire B → b2	O22	T2: Lire B → b2	O22
T2: b1 * 2 → b2		T2: b1 * 2 → b2	
T2: Ecrire b2 → B		T2: Ecrire b2 → B	

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Implantation de sérialité de la transaction

Ordonnancements sérialisables : Graphe de précédence

Précédence

Soit S une ordonnancement sans opération simultanée. On dit que Ti précède Tj (noté Ti<Tj) dans S si et seulement s 'il existe deux opérations non permutables Oi et Oj telles que Oi est exécutée par Ti avant Oj par Tj

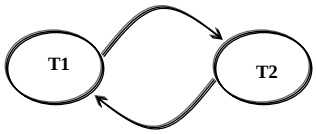
ORDONNANCEMENT 1

T1: A+1 → A
T2: A*2 → A
T1: B+1 → B
T2: B*2 → B

Théorème 2
Une condition suffisante pour qu'un ordonnancement soit sérialisable est que le graphe de précédence associé est sans circuit

ORDONNANCEMENT 2

T1: A+1 → A
T2: A*2 → A
T2: B*2 → B
T1: B+1 → B



Page 89

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Implantation de sérialité de la transaction

Verrouillage en deux phases

T1, T2, ...,Tn n transactions à exécuter

Matrice de compatibilité

	L	E
L	1	0
E	0	0

C

Opérations protocolaires
- LOCK(g,M) permet à une transaction demander de poser un verrou de type M sur le granule g
- UNLOCK(g) permet à une transaction demander de lever le verrou qu 'il a posé sur le granule g

Théorème
Les modes d 'opération demandés lors d 'une primitive LOCK(g,M) exécutée par une transaction Tp sont compatible avec les modes en cours d 'exécution sur g par les autres transaction si et seulement si :

$$M \subset \neg\,(\neg C * \bigcup_{i \neq p} A(g,i))$$

A(g,i) est le vecteur de mode actuellement posé par Ti sur le granule g

M est le vecteur de modes de verrou demandés par une transaction Tp sur le granule g

Page 90

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Implantation de sérialité de la transaction

Verrouillage en deux phases

Procédure LOCK (g,M)
Si $M \subset \neg (\neg C * \cup_{i \neq p} A(g,i))$
alors $A(g,p) := A(g,p) \cup M$
sinon « inserer '(p,M) dans Q[j] »;
« bloquer la transaction Tp »;
finsi;
fin LOCK;

Procédure UNLOCK (g)
$A(g,p) := (0)$;
Pour chaque (q,M ') de Q[g] faire
Si $M' \subset \neg (\neg C * \cup_{i \neq q} A(g,i))$
alors $A(g,q) := A(g,p) \cup M'$
«Extraire (q,M ') de Q[j] »;
« débloquer la transaction Tq »;
finsi; finpour;
fin UNLOCK;

Restriction « deux phases »

Une transaction est dit à deux phases si et seulement si elle n 'exécute pas de LOCK après avoir exécuté UNLOCK

Théorème
Tout ordonnancement complète d 'un ensemble de transactions « deux-phases » est sérialisable

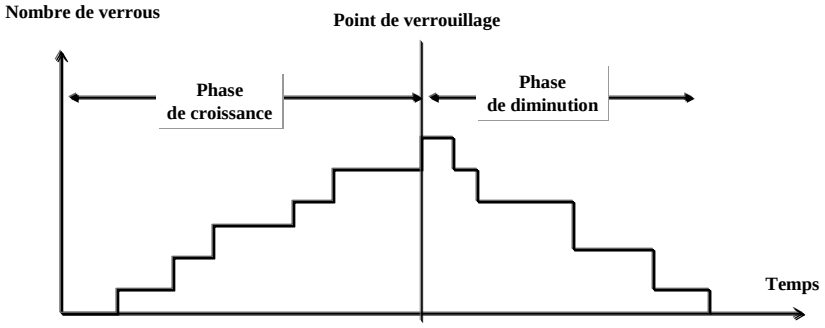
Page 91

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Implantation de sérialité de la transaction

Verrouillage en deux phases



Page 92

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Implantation de sérialité de la transaction

Contrôle de concurrence optimiste (Kung et Robinson, 1981)

Idée de base

1) Allez de l 'avant et faites tout ce que vous voulez, sans faire attention à ce que les autres font
2) S 'il arrive un problème, on s 'en occupera plus tard

Principes de fonctionnement

- considérer que chaque transaction se compose de deux étapes :
- une étape de lecture et calcul privé
- une étape d'écriture réelle dans la base (commitment)

- ordonner les transactions selon le moment où elles terminent la première étape de lecture et de calcul

- contrôler que les accès conflictuels aux granules s'effectuent bien dans l'ordre ainsi obtenu

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Implantation de sérialité de la transaction

Estampillage

Estampille de transaction

valeur numérique unique associée à une transaction au moment de son commencement

Estampille de granule

valeur numérique associée à un granule mémorisant l 'estampille de la dernière transaction ayant opéré sur ce granule

Principes

1- toute transaction a un estampille unique. Les estampilles forme un ordre total des transaction
2- une transaction ayant accédé à un granule doit mémoriser son estampille sur le granule
3- une transaction veut accéder à un granule doit comparer son estampille avec ce du granule, si il est plus jeune que celui dernière, il peut accéder au granule
4- dans le cas contraire, il devra se suicider

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Interblocage

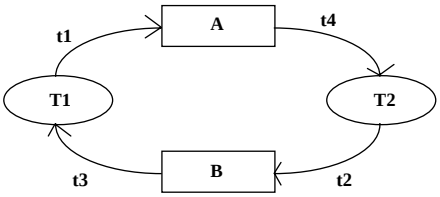
Quatre stratégies de gestion

1. La politique de l 'autruche (ignorer le problème)

2. La détection (permettre aux interblocages de se produire, les détecter, puis tenter de les éliminer)

3. La prévention (rendre de façon statique les interblocages structurelles impossibles)

4. L 'évitement (éviter les interblocages en distribuant les ressources avec précaution)



t1 > t2 > t3 > t4

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

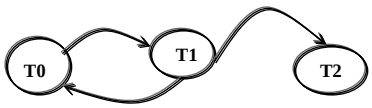
II.3.5.4- Interblocage

Graphe des attentes et graphes d 'allocation

Graphe des attentes (Murphy68)

Soient T l 'ensemble de n transactions concurrentes et R une relation binaire sur T, appelée « attend », définie comme suit : Ti attend Tj si et seulement si Ti attend le verrouillage d 'un granule g qui a déjà été verrouillé par Tj. G(T,r) est le graphe conçu sur T par cette relation R

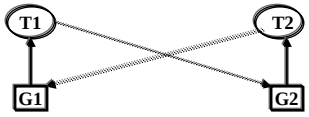
Théorème :
Il existe un interblocage si et seulement si le graphe des attentes possède un circuit



Graphe des allocations (Holt72)

Soient T l 'ensemble de n transactions concurrentes et G l 'ensembles des granules. Le graphe des allocations se compose des sommets dans T+G et des arcs définis comme suit : un arc relie Gi à Tp si et seulement si Tp a obtenu le verrouillage sur Gi; un arc relie Tq à Gj si et seulement si Tq attend un verrou sur Gj

Théorème :
Une condition nécessaire d 'existence d 'un interblocage est que le graphe des allocations possède un circuit

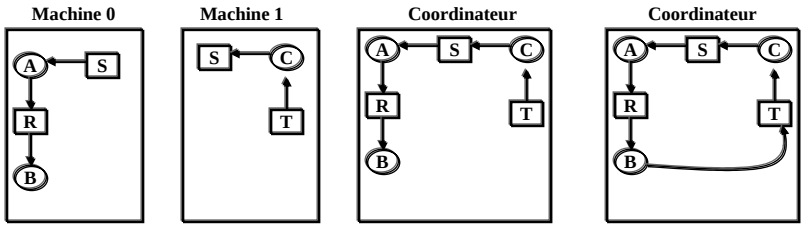


II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Interblocage

Détection des interblocages : algorithmes centralisés



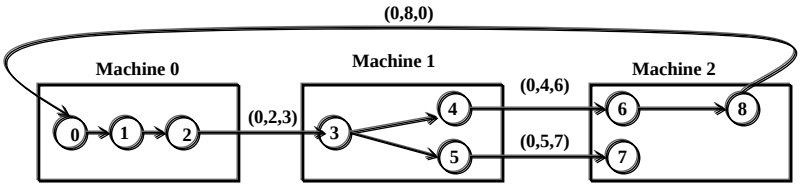
Problème de la « mort injuste »

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Interblocage

Détection des interblocages : algorithmes distribués



Algorithme de Chandy-Misra-Haas (1983)

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Interblocage

Prévention des interblocages : généralité

Principe
La prévention des interblocages consiste à construire prudemment le système de façon de rendre les interblocages structurellement impossibles. Autrement dit, elle consiste à supprimer l 'une des conditions qui rend possible l 'interblocage

Contraintes d 'allocation

- un processus ne détient qu'une ressource à la fois
- un processus doit déclarer toutes ces ressources qu'il a besoin
- un processus doit relâcher les ressources qu'il détient pour pouvoir allouer une nouvelle

Ordonnancement des ressources

- préordonner toutes les ressources du système

- un processus doit acquérir les ressources qu'il a besoin dans l 'ordre strictement croissant

Ordonnancement des transactions

- préordonner toutes les transactions par l 'estampille

- déterminer une politique d 'allocation des ressources adéquate permettant d 'éviter les interblocages

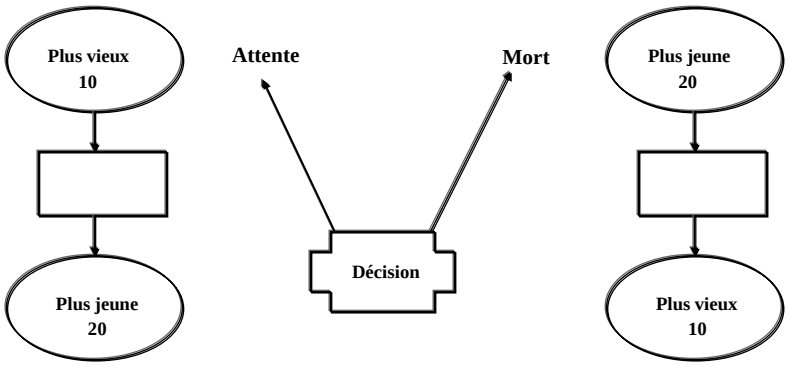
Page 99

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Interblocage

Prévention des interblocages : algorithme « attente-mort » (wait-die)



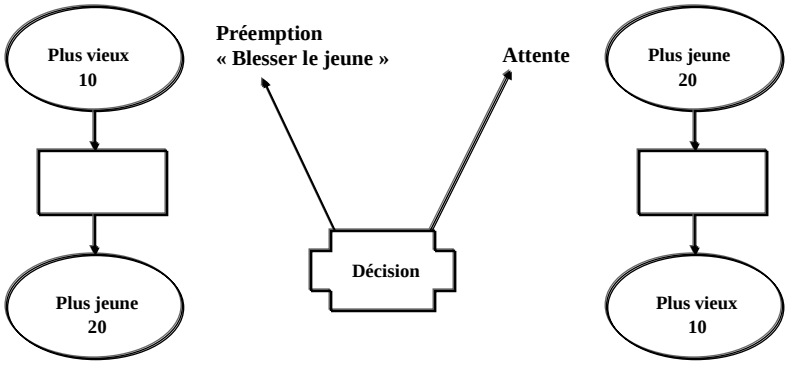
Page 100

II- Partie II : systèmes distribués
II.3- Synchronisation dans les systèmes distribués

II.3.5- Transaction atomiques distribuées

II.3.5.4- Interblocage

Prévention des interblocages : algorithme « blessure-attente »



Page 101