TRANSACTIONNEL

RÉPARTI

G. Florin

INTRODUCTION

Système d'objets partagés répartis

Ensemble d'objets à accéder à distance (ou localement) par plusieurs applications.

Enregistrements de fichiers répartis.
Données d'une base de données répartie.
Variables d'instance d'objets répartis.

Actions élémentaires

Opérations sur les objets considérées ici

lire (x , val)
écrire (x , val)

Autres opérations possibles

création , destruction des objets,
opérations complexes.

Contraintes d'intégrité

L'ensemble des relations que doivent vérifier les objets.

État cohérent (du système d'objets)

Un état du système ou les contraintes d'intégrité sont satisfaites.

Transaction

Application qui exécute des actions (notées tlire, técrire) sur les objets partagés

Exemple : Transaction T

```
t_début
    tlire(x);
    técrire (x, val1);
    técrire (y, val2);
t_fin;
```

Propriétés ACID (Atomicité, Consistance, Isolation, Durabilité)

Deux présentations

- Une formulation très générale définie pour un ensemble quelconque d'opérations.
- Une formulation spécifique dans le cas des transactions.

Atomicité ("atomicity")

Forme générale

T**outes les opérations** associées à une **action "atomique"** sont réalisées **complètement** ou **aucune d'entr'elles** n'est exécutée.

Cas du transactionnel

. Ou bien l'exécution d'une transaction est totale.

Toutes les actions lire et écrire d'une transaction sont effectuées et amènent la base jusqu'à un nouvel état.

. Ou bien l'exécution n'a aucun effet sur les objets.

Toutes les modifications partielles engagées sont annulées et l'on reste sur l'ancienne version.

Z Origine des difficultés:

- les accès concurrents
- les pannes.

Consistance ("consistency")

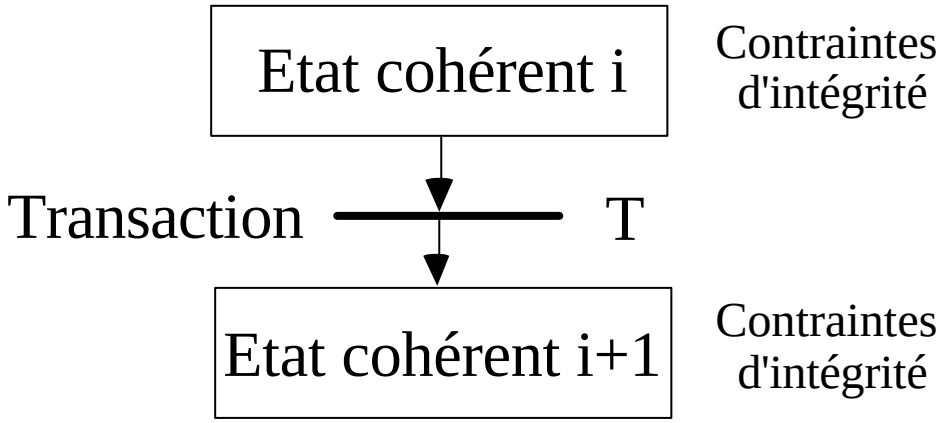
Forme générale

Un ensemble d'opérations d'accès vérifient certaines propriétés (exemple consistance séquentielle d'une mémoire).

Cas du transactionnel

Une transaction est consistante lorsque exécutée seule elle amène un ensemble d'objets d'un état cohérent à un autre état cohérent (consistante avec la sémantique de cohérence d'un ensemble d'objets).

L'objectif primordial est de maintenir la cohérence de la base d'objets.



Isolation ("isolation")

Forme générale

Un utilisateur ne peut avoir accès à aucun résultat intermédiaire d'une action atomique avant la terminaison de la totalité de la transaction.

Cas du transactionnel

L'isolation est la propriété qui permet à un ensemble de transactions s'exécutant simultanément (en concurrence d'accès aux données partagées) d'apparaître comme s'exécutant de façon indépendante les unes des autres (sérialisable).

On ne peut accéder à des valeurs intermédiaires d'une transaction.

C'est le travail de contrôle de concurrence proprement dit.

Durabilité ("durability")

Forme générale

C'est la propriété d'un système de garantir que les effets d'un ensemble d'opérations ne sont altérés par aucune sorte de pannes.

Cas du transactionnel

Les effets des opérations décidées définitivement doivent être durables même en cas de défaillances (de processeurs ou de communications).

Autre terminologie très fréquente:

La persistance

(les objets accédés de façon concurrente sont persistants).

Les trois fonctions principales du transactionnel distribué

Contrôle de concurrence en réparti ("concurrency control")

L'ensemble des **techniques** qui permettent de **synchroniser** les activités parallèles de plusieurs **transactions**,

qui accèdent à **des objets partagés sur des sites différents** et par suite interfèrent les unes avec les autres.

Ils assurent la propriété **d'isolation**.

Reprises arrières en réparti ("backward recovery")

Les mécanismes systèmes de sauvegarde et de reprise sur pannes qui assurent que les pannes ne peuvent corrompre les données

Ils assurent l'**atomicité** et la **durabilité** (**la persistance)** des données.

La gestion des performances

Fournir les mécanismes systèmes qui permettent d'accepter le plus grand nombre de transactions possible par unité de temps sans que le temps de réponse de chaque requête soit trop dégradé:

- ordonnancer les transactions
  (priorités, ...)

- créer, administrer les processus serveurs de transactions (équilibrage de charge, partage de charge)

En résumé

Le moniteur transactionnel réparti est la partie **du système d'exploitation réseau** pour le traitement d'unités de travail de type transactionnel.

Il réalise:
- des fonctions de synchronisation
- des fonction de gestion de ressources
- des fonctions de tolérance aux pannes.

Plan

Chapitre I

Problèmes du transactionnel réparti

I.1 Le contrôle de concurrence

Difficultés liées à l'entrelacement des lectures et des écritures

Exemple

Deux objets x et y avec contrainte y = 2x

Écritures incohérentes

Transaction T1

Action 1 : écrire(x,10) {x=10}
Action 4 : écrire(y,20) {y=20}

Transaction T2

Action 2 : écrire(x,30) {x=30}
Action 3 : écrire(y,60) {y=60}

A la fin l'état des données est incohérent {x=30} {y=20}

Z Autre dénomination : écritures perdues

Lectures incohérentes

Transaction T1

Action 1 : lire(x) {x=10}

Action 4 : lire(y) {y=60}

Transaction T2

Action 2 : écrire(x,30) {x=30}

Action 3 : écrire(y,60) {y=60}

Les valeurs lues par T1 sont incohérentes:

{x=10} {y=60}

Remarque:

. Même pour une seule variable: possibilité de non reproductibilité des lectures (si une transaction en parallèle écrit la variable) => pas d'isolation.

. Problèmes entre les lectures et les écritures.

Principe de sérialisabilité

Rappel : principe de consistance des transactions (propriété C)

Une transaction exécutée seule maintient la cohérence des données
=> Toute suite de transactions exécutées dans un ordre quelconque conduit à un état cohérent des objets.

Exécutions parallèles entrelacées.

. Une exécution parallèle avec entrelacement des opérations de lecture et d'écriture est indispensable pour des questions de performance.

Exécutions sérialisées

. Une exécution parallèle est **sérialisable** si son résultat est le même que celui obtenu par une **exécution séquentielle**.
(L'ordre d'une exécution concurrente est aléatoire tous les états corrects résultants d'une exécution séquentielle en ordre quelconque des transactions sont valides).

Modèle de transaction "horizontale" avec contrôle de concurrence

- Réaliser les lectures et les écritures en espace de travail pour détecter des conflits.
- Les traiter par le contrôle de concurrence.

action tlire(x) : lecture à partir de la base ou de l'espace de travail.

action préécrire(x,val) : annonce de l'intention d'écrire x (espace de travail).

```
/* Création d'espace de travail temporaire/
/* Mise en oeuvre d'un contrôle de */
/* concurrence pour éviter les lectures ou  */
/* les écritures incohérentes. */

     t_début
          tlire(x);
          préécrire (x, val1);
          préécrire (y, val2);
     t_fin;
/*   -------- Point de validation ---------      */
/*       Écriture dans la base                    */
          écrire ( x , val1 ) ;
          écrire ( y , val2 ) ;
```

Relations entre transactions

Transactions vivantes et validées

- Une transaction en cours (vivante) est notée T

- Une transaction validée est notée T*

Conflit entre deux transactions

Une transaction accède à un objet ayant déjà été lu ou écrit par une autre transaction.

L'une des opérations est une préécriture sinon il n'y a pas de problème.

Trois conflits

Lecture-Préécriture	**LP**
Préécriture-Lecture	**PL**
Préécriture-Préécriture	**PP**

Relation d'ordre entre transactions (relation de précédence)

L'existence d'un conflit entre deux transactions entraîne une contrainte sur l'ordre des transactions pour que leur exécution soit sérialisable.

Notion de dépendance effective

$\mathbf{T_1 \text{-------}> T_2}$

On fixe des le premier conflit l'ordre entre les deux transactions (la plupart des cas).

On indique par la notation précédente que T_1 doit s'exécuter avant $\mathbf{T_2}$ (tout doit se passer comme si T_1 est exécutée avant $\mathbf{T_2}$).

Notion de dépendance potentielle

$\mathbf{T_1 - - - - T_2}$

$\mathbf{T_1}$ et $\mathbf{T_2}$ doivent être traitées dans un certain ordre qu'il n'est pas indispensable de fixer immédiatement (il n'est fixé qu'au moment de la validation).

Règles de création des dépendances entre les transactions

1 Dépendances entre une transaction vivante T_2 et une transaction validée T_1^*

Règle naturelle $T_1^* \rightarrow T_2$

2 Dépendances entre deux transactions vivantes T_1 et T_2

Conflit PL (préécriture , lecture)

T_1	préécrire(x)
Ensuite T_2	lire(x).

Solution a) Fournir à T_2 la valeur actuelle de l'objet (celle qui précède les deux transactions T_1 et T_2) et forcer la dépendance : $T_2 \rightarrow T_1$

Solution b) Mettre en attente T_2 jusqu'à la validation de T_1 et ensuite fournir à T_2 la nouvelle valeur résultant de l'exécution de T_1 soit forcer la dépendance : $T_1^* \rightarrow T_2$

Conflit LP (lecture, préécriture)

T_1	lire(x)
Ensuite T_2	préécrire(x).

L'utilisation par T_1 d'un objet qui sera ensuite modifié par T_2 entraîne la dépendance

$$T_1 \rightarrow T_2$$

Conflit PP (préécriture, préécriture)

T_1	préécrire(x)
Ensuite T_2	préécrire(x).

On ne sait pas encore quelle transaction va être validée la première donc on note l'existence d'une dépendance potentielle:

$$T_1 \text{ - - - - } T_2$$

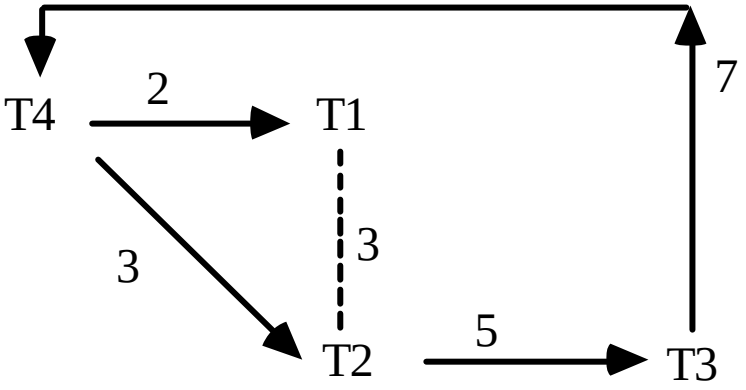
Dès que l'une des transactions est validée la dépendance est fixée et devient selon l'ordre de validation:

$$T_1^* \rightarrow T_2$$

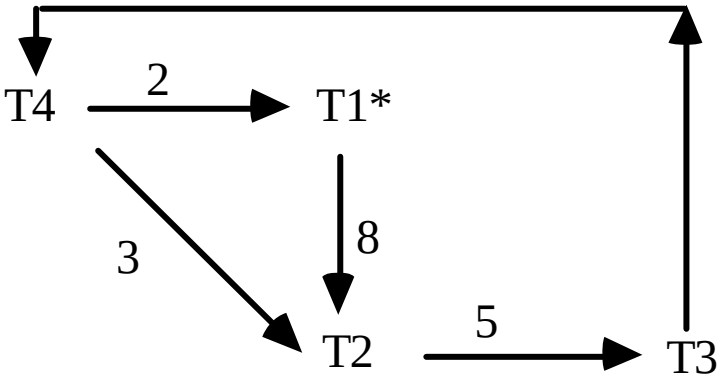
$$T_2^* \rightarrow T_1$$

Exemple de graphe de précédence

T1	T2	T3	T4
2: préécrire(x)	3:préécrire(x)	5:préécrire(y)	1:lire(x)
8:valider	4:lire(y)	6:lire(z)	7:préécrire(z)



Graphe de dépendance jusqu'à l'instant 7



A l'instant 8 : validation de T1

Résultat Principal (Papadimitriou)

- Si **le graphe de dépendance est sans circuit** alors il est possible **de sérialiser**

=> Par suite de conduire dans tous les cas à un état de cohérence final.

- **L'existence** dans le graphe de dépendance **d'un circuit** constitué uniquement de dépendances effectives entraîne **l'impossibilité de sérialiser.**

=> Risque très grand de perte de cohérence.

Remarques: principe de sérialisation

- L'absence de circuits dans le graphe est une condition suffisante de maintien de la cohérence.

Rejet d'une transaction

. Pour éviter que le graphe de dépendance ne possède des circuits **il faut abandonner des transactions qui appartiennent à des circuits**.

. On peut rejeter une transaction vivante **dès que l'une des relations conduit à l'établissement d'un circuit**.

. Compte tenu du caractère irréversible de la validation **il suffit que le graphe des transactions validées soit sans circuit** pour assurer la sérialisabilité des transactions

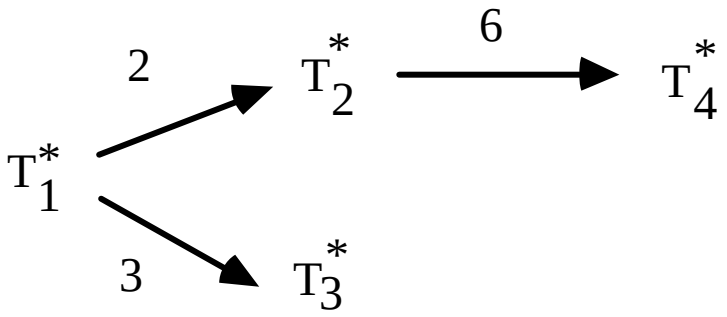
=> Rejet de transactions uniquement au moment de la validation.

Unicité de l'ordre de sérialisation

- L'ordre de sérialisation n'est pas unique:
le graphe G définit un ordre partiel pour lequel plusieurs ordres totaux sont possibles

T1	T2	T3	T4
1: lire(x)	2:préécrire(x)	4:préécrire(y)	6:préécrire(z)
3:lire(y)	5:lire(z)		
7:valider	8:valider	9:valider	10:valider

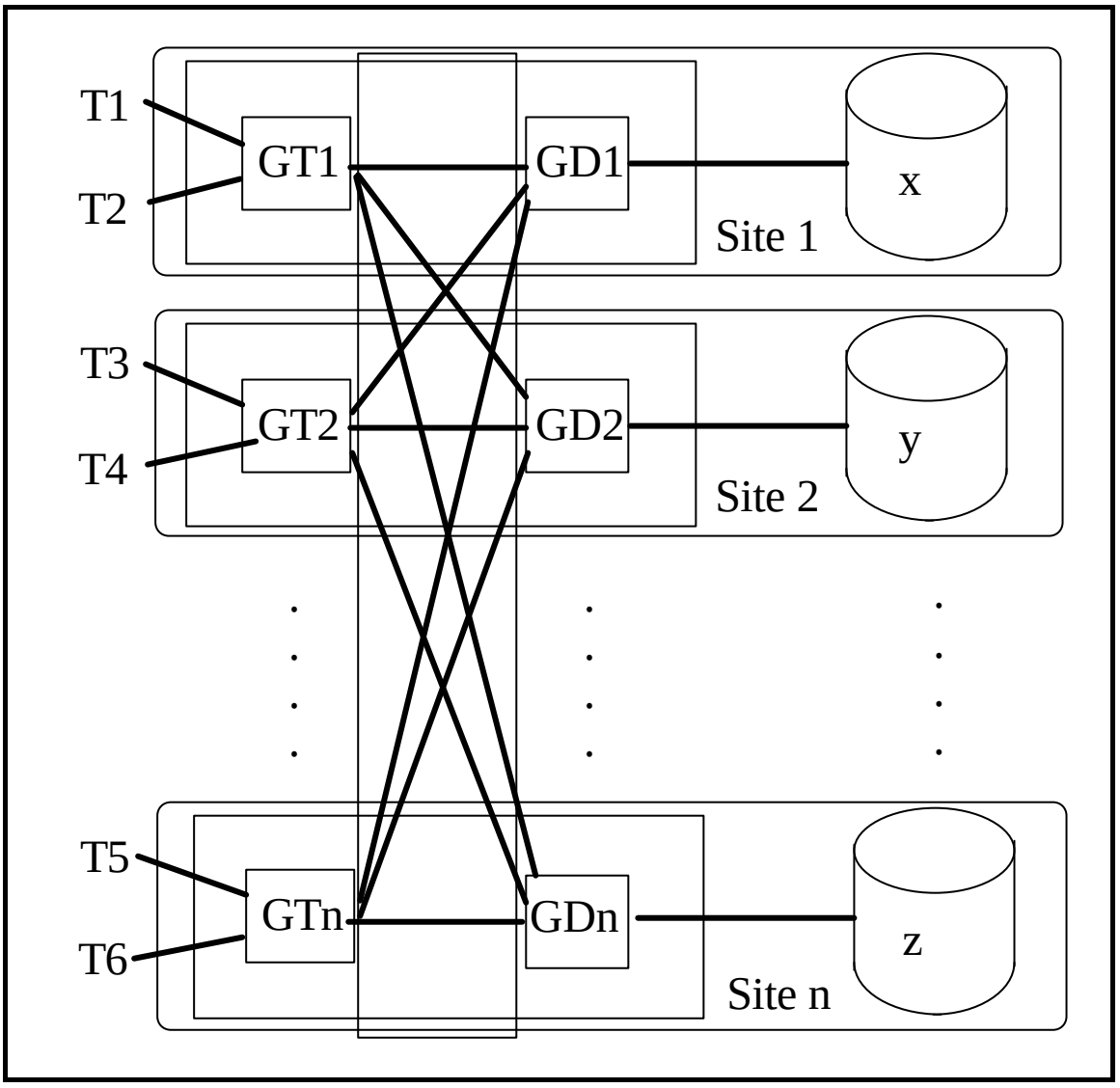
Graphe de dépendance avant la validation



Les transactions sont sérialisables dans les ordres suivants:

T1 T2 T3 T4
T1 T2 T4 T3
T1 T3 T2 T4

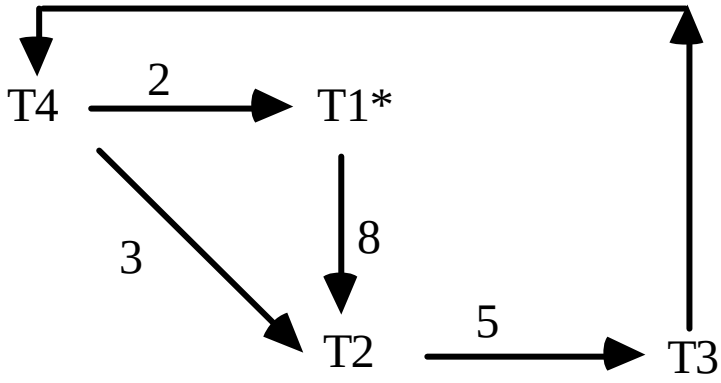
Architecture de système transactionnel réparti



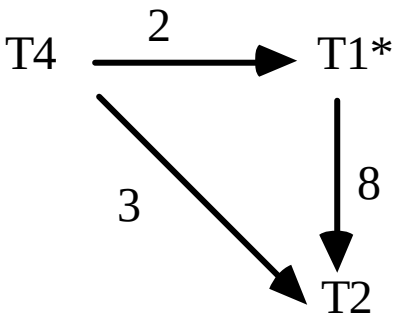
. GT gestionnaire de transaction
. GD gestionnaire de données
. Une transaction s'adresse à un seul GT
. Une donnée peut-être en copie unique.
. Une donnée peut-être en copie multiple.

Dépendances entre transactions en environnement réparti

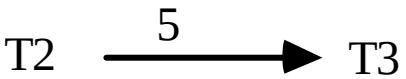
T1	T2	T3	T4
2: préécrire(x)	3:préécrire(x)	5:préécrire(y)	1:lire(x)
8:valider	4:lire(y)	6:lire(z)	7:préécrire(z)



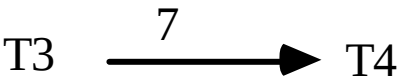
Graphe complet : univers centralisé



Connaissance sur le site de x



Connaissance sur le site de y



Connaissance sur le site de z

Graphe réparti : univers réparti, localisation différente des objets

Solutions au problème du contrôle de concurrence

Les deux grandes catégories

Le contrôle continu

Pour toute opération d'accès on vérifie que la sérialisation reste possible.

. Pour **toute transaction vivante T** qui effectue une nouvelle **action**, **le graphe de dépendance $G^*(T)$** des transactions validées augmenté de la transaction vivante T doit rester **sans circuit**.

=> Une transaction T pourra donc valider à tout instant sans créer de circuits

Remarque

C'est une **approche pessimiste** car on vérifie la sérialisabilité a priori comme si elle avait une forte probabilité.

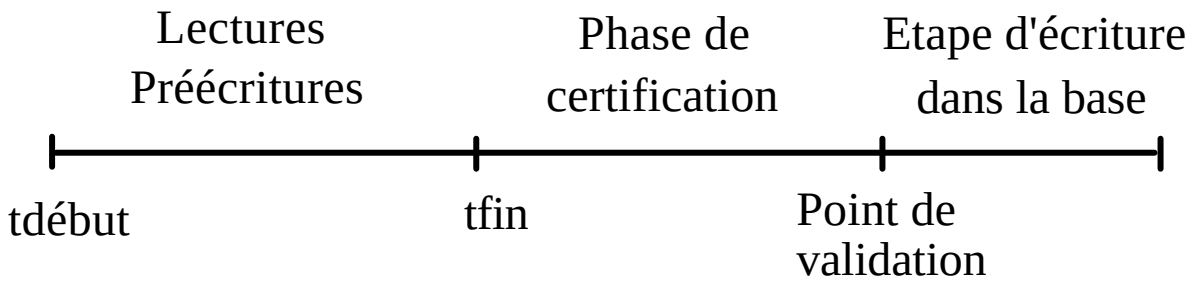
Le contrôle par certification

. On ne recherche **l'existence de circuits**

. **que dans G*** le graphe de dépendance des transactions validées

. **au moment de la validation** d'une nouvelle transaction.

S'il existe un circuit la transaction qui valide est rejetée (redémarrée).



Remarque

C'est une **approche optimiste** car on vérifie la sérialisabilité après coup comme si elle avait peu de chances de se produire.

Une méthode de contrôle continu utilisable en univers réparti
Le verrouillage à deux phases (2PL , "Two phase locking")

1 Le verrouillage

. **Avant d'effectuer une opération** d'accès, u**ne transaction doit poser un verrou** en lecture (resp en écriture) sur l'objet considéré

. **Ce verrou doit être maintenu jusqu'à la fin** des opérations d'écriture dans la base.

. **Deux transactions ne peuvent détenir** en même temps **des verrous conflictuels**: ils portent sur le même objet et **si l'un au moins est un verrou en écriture**.

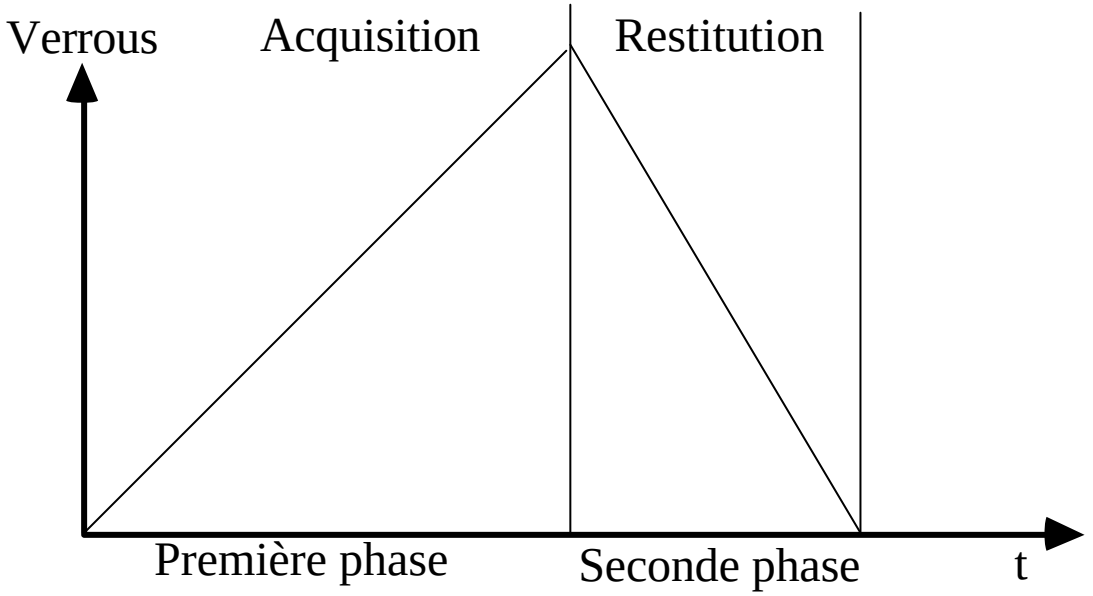
	Mode d'accès en cours		
	Non verrouillé	Verrouillé en lecture	Verrouillé en écriture
Demande d'accès en lecture	OUI	OUI	NON
Demande d'accès en écriture	OUI	NON	NON

2 Le verrouillage à deux phases

Après avoir relâché un verrou une transaction ne peut plus en obtenir un autre => **deux phases.**

Phase 1: du début jusqu'à la validation la transaction acquiert des verrous

Phase 2: du point de validation jusqu'à la fin la transaction réalise des écritures dans la base et relâche des verrous.



Remarque: difficultés en univers réparti si des données sont en copies multiples.

Remarques concernant le verrouillage à deux phases

- La méthode de verrouillage à deux phases consiste à forcer en cas de conflit la dépendance : T_1 ------> T_2 si l'action de T_1 dans le conflit précède celle de T_2.

T_2 est donc bloquée en attente de la validation de T_1.

T_1 peut donc poursuivre à sa guise et sa validation peut intervenir à tout instant.

T_2 reste bloquée jusqu'à ce que T_1 valide puis relâche son verrou.

- L'ordre de sérialisation correspond à l'ordre fixé lors de conflits qui est aussi l'ordre chronologique des validations.

- Les conflits qui existent entre les transactions vivantes sont ignorés tant que ces transactions restent bloquées.

Le verrouillage à deux phases produit des ordonnancements de transactions sérialisables

Supposons qu'il existe un circuit dans le graphe de dépendance en verrouillage à deux phase.

a) Puisque $Ti \rightarrow Tj$ alors Ti a détenu un verrou demandé par Tj (Ti a relâché ce verrou avant que Tj ne l'obtienne)

Ti a validé avant Tj .

b) Puisqu'il existe un circuit on a aussi :

$$Tj \rightarrow Tu \rightarrow .....Tl \rightarrow Tk \rightarrow Ti$$

Tj a relâché un verrou avant que Tu ne l'obtienne Tj a donc validé avant Tu

Et ainsi de suite.

Tj a validé avant Ti.

Contradiction: L'existence de circuit est contraire au principe du verrouillage à deux phases.

Problème du verrouillage à deux phases: l'interblocage

Principe du verrouillage => blocage

Demande d'accès conflictuel d'une transaction à un objet verrouillé
=> Mise en attente.

=> Interblocage de ressources:

T_1 détient x et demande y
T_2 détient y et demande x

Les deux transactions ne peuvent plus avancer .

Solution : détecter l'interblocage et détruire une transaction

Détection d'interblocage = détecter l'existence de circuits dans le graphe des attentes de ressources (qui est encore le graphe de dépendance réduit aux transactions vivantes).

Remarque: Problème de concurrence et problème d'interblocage

Problème de l'accès concurrent aux objets

Prévention ou détection de circuits dans le graphe de dépendance des transactions validées et des transactions vivantes.

Problème de détection d'interblocage

Prévention ou détection de la formation de circuits dans le graphe des attentes.

Problèmes très similaires

Solution au problème de l'interblocage en réparti

Utilisation de délais de garde

Si une transaction reste en attente trop longtemps elle est détruite.

Correct si elle est interbloquée

Mauvais si elle est en attente de transactions longues.

Difficulté: choix du délai de garde.

Utilisation d'enquêtes pour détecter des circuits dans le graphe des attentes

Solution au problème de détection : suivre les différents chemins du graphe des attentes au moyen de messages d'enquête.

Si un message d'enquête revient à l'émetteur c'est qu'il y a circuit.

Solution coûteuse en univers réparti:

Exemple de solution: algorithme d'Obermarkt

Utilisation d'estampilles (algorithmes de Rosenkrantz, Stearn, Lewis)

Toutes les transactions reçoivent des estampilles (numéros de séquence).

Les priorités à la destruction sont déterminées par les estampilles.

Algorithme Attend ou Disparaît ("WAIT-DIE")

/* Tj détient un verrou que Ti demande */
si (num (Ti) < num (Tj)) **alors**
    Ti attend ;
sinon
    Ti est détruit ;
finsi

Remarque:

La solution est équitable si les estampilles sont gérées à l'ancienneté.

I.2
Le problème d'atomicité

Le protocole de validation atomique
(ACP, "Atomic Commitment Protocol")

Les participants

Un coordinateur: C ("coordinator")

. Supporte un gérant de transaction TM ("transaction manager").
. Dirige de façon centralisée la validation.

Les exécutants: S1, ... , Sn supportent

. Un gestionnaire de données DM ("data manager") pour réaliser des accès.
. Un journal de transactions distribuées DTlog ("Distributed Transaction Log") ou chaque exécutant enregistre en mémoire stable les informations les plus importantes de façon à survivre aux pannes.

Remarque:

Les exécutants peuvent aussi réaliser une validation atomique "coopérative" sans l'aide d'un coordinateur.

Les principes généraux du protocole de validation atomique transactionnel

a) Le gestionnaire de données travaille sur des copies des données (en espace de travail) lors de la phase de préparation.

- Tout objet est recopié dans l'espace de travail de la transaction lors de sa première lecture.

- Tout objet est écrit dans l'espace de travail.

b) Lorsque l'on est prêt à valider la transaction, chaque objet est recopié dans le journal en mémoire stable (non affecté par les pannes).

c) Il existe un point de validation ("**commit point**") à partir duquel la transaction sera obligatoirement terminée Soit les valeurs en mémoire stable sont intégrées de façon définitive (**"commit"**)

Soit la transaction abandonnée: tout le travail temporaire est défait (**"rollback"**).

Modèle de transaction avec validation atomique

/* Préparation d'un exécutant */
t_début
    tlire(x);
    préécrire (x, val1);
    préécrire (y, val2);
t_fin;

/* Pas de problèmes pour ce site*/
/* Début de phase de validation */

    ecrire_stable (x , val1) ;
    écrire_stable (y , val2) ;

/* ----- Point de validation ----- */
/* Pas de problème pour les exécutants*/
/* Écriture mémoire stable dans la base*/

    écrire (x , val1) ;
    écrire (y , val2) ;

/* Destruction des informations temporaires */
Remarque : modification possible préécrire
    - annonce de l'intention de modifier

- écriture directe en mémoire stable.

Propriétés d'un protocole de validation atomique

C'est un protocole de **consensus** sur l'exécution d'une transaction entre des participants qui émettent un vote:

. oui, validation ("commit")

. non, abandon ("abort").

Il faut atteindre un consensus unanime selon les règles suivantes.

AC1 : **Tout participant qui atteint une décision atteint la même décision que les autres (atomicité)**.
Cette hypothèse implique que tous terminent de façon cohérente.

AC2 : **Un participant ne peut plus revenir sur sa décision s'il l'a émise.**
La décision d'un site est irrévocable.

AC3 : **La décision générale de validation est prise si tous les participants ont voté oui (validation).**
La validation n'est atteinte que si tous ont répondu oui.

Propriétés d'un protocole de validation atomique (suite)

AC4 Si la décision de tous est validation et s'il n'y a pas de pannes la validation est réalisée.

Pour éviter de toujours décider abandon.

AC5 Si la décision générale est validation et qu'il n'y a que des pannes que l'algorithme peut tolérer => lorsque les pannes sont réparées et s'il n'y a pas de nouvelles pannes la validation est réalisée.

Pour éviter qu'une indécision perpétuelle ne soit possible.

La validation à deux phases (2PC "Two Phase Commit")

Principes d'une validation atomique (syntaxe de l'API XA de l'X/open)

a) Le coordinateur fait réaliser par les exécutants des actions sur des copies d'objets.

b) Le coordinateur initialise la phase terminale d'une transaction en demandant un vote aux exécutants (message **XA_PREPARE)**.

c) Les exécutants **OK** votent oui (non si **KO)**.

d) Le coordinateur ayant collecté toutes les réponses décide de valider ou d'abandonner par **XA_COMMIT** ou **XA_ROLLBACK**.

e) S'il a reçu **XA_COMMIT**, un exécutant recopie ses données dans la base de données et peut envoyer **XA_END** au coordinateur.

Différentes pannes possibles

. Panne du coordinateur
. Panne des exécutants
. Coupure de réseau

Problèmes liés aux pannes

Pour détecter les situations de pannes on arme différents délais de garde.

Terminaison de la validation en cas de pannes

Délai de garde dépassé chez le coordinateur

Cas 1: Coordinateur en attente de validation exécutant => décision d'abandon

Cas 2 : Coordinateur en attente d'acquittement final => répétition (sur demande) du message de validation ou d'abandon

Délai de garde dépassé chez un exécutant

Cas 1 : En phase de préparation non réponse du coordinateur

=> décision d'abandon de l'exécutant.

Cas 2 : Après avoir voté :

=> situation d'incertitude chez un exécutant qui peut durer très longtemps.

Notion de phase d'incertitude d'un exécutant

Le site Sk ayant notifié sa réponse de vote oui **ne reçoit pas de réponse** du coordinateur en raison de pannes.

=> Il ne sait pas s'il doit valider ou abandonner.

Exemples

. Coupure de Sk du reste du réseau
. Relation de Sk avec des sites en état d'incertitude
. Panne de Sk en état incertain et réparation
. Panne du coordinateur entre réponse oui et le message XA_COMMIT.

=> **Blocage d'une transaction** en cours pour une durée arbitrairement grande (en raison des pannes).

Solutions pour lever l'incertitude d'un exécutant

1 Basée uniquement sur la reconnexion avec le coordinateur opérationnel (après réparation).

2 Basée sur un dialogue entre les exécutants : certains sites peuvent savoir quel est le statut de la transaction.

. L'un des exécutants est encore dans la phase préparatoire => il va abandonner.

. L'un des exécutants a déjà abandonné.

. L'un des exécutants connaît le statut validé de la transaction.

. Tous les exécutants que peut contacter le demandeur du statut sont dans le même état d'incertitude.

Traitements de reprise après panne

Reprise après panne du coordinateur

Reprise dans **la phase préparatoire** avant diffusion du statut de la transaction:

=>Abandonner la transaction, (même s'il serait possible de continuer).

Reprise **après décision prise**:

=> Ne rien faire.

Reprise après panne chez un exécutant

Reprise **dans la phase préparatoire**

=> Abandon de la transaction.

Reprise **après abandon** (message envoyé ou non) => Ne rien faire.

Reprise **après validation** (message envoyé ou non) => traitement identique à celui concernant la phase d'incertitude.

Reprise **en état de décision connue**

(transaction validée ou abandonnée)

=> Ne rien faire.

Programmation de la validation à deux phases

Algorithme du coordinateur

début

/* Préparer la transaction */

envoyer message XA_PREPARE à tous ;
écrire_stable début_valid (id_transaction);
armer_délai_réponse;
attendre (événement) ;

si (retombée_garde ou réponse non) **alors**
/* Un/plusieurs sites ne sont pas OK */
écrire_stable abandon(id_transaction);
envoyer XA_ROLLBACK à tous;
finsi;

si (tous les exécutants votent oui) **alors**
écrire_stable validation(id_transaction);
envoyer C_COMMIT à tous les exécutants;
finsi;

fin coordinateur;

Algorithme d'un exécutant

début
/* Réaliser la préparation de la transaction */

armer (délai_message_XA_prepare) ;
attendre (événement) ;

si (retombée_garde) **alors**

écrire_stable (abandon (id_transaction)) ;
fin exécutant;
finsi ;

si (XA_PREPARE et vote oui) **alors**
écrire_stable (oui (id_transaction)) ;
envoyer C_READY au coordinateur ;
armer (délai_message_valid_abandon) ;
attendre(événement) ;
si (retombée_garde) **alors**
/* Cas d'incertitude trop longue */
exécuter programme de terminaison ;
finsi ;

si (message XA_COMMIT) **alors**
écrire_stable(validation(id_transaction));
écrire définitivement les modifications ;
finsi ;

si (message XA_ROLLBACK) **alors**
écrire_stable (abandon(id_transaction)) ;
détruire les informations mémoire stable;
finsi ;

finsi ;

si (XA_PREPARE et vote non) **alors**
écrire_stable (abandon (id_transaction)) ;
envoyer NON au coordinateur ;
détruire les informations mémoire stable ;
fin exécutant ;

finsi ;

fin exécutant **;**

Phase de terminaison par dialogue entre les participants

Algorithme du demandeur de statut de la transaction

début

envoyer DEM_STATUT(transaction) à tous;
armer (délai_réponse) ;
attendre (événement) ;

si (message réponse XA_COMMIT) **alors**
  écrire_stable (validation(id_transaction));
  écrire définitivement les modifications;
finsi ;

si (message réponse XA_ROLLBACK) **alors**
  écrire_stable (abandon(ident_transaction));
 détruire les modifications en mémoire stable;
finsi ;

si (retombée_garde) **alors**
 état d'incertitude non levé **recommencer**;
finsi ;
fin terminaison_demandeur**;**

Algorithme du répondeur (aux demandes de statut de la transaction)

```
début

attendre (événement ) ;

si (message DEM_STATUT) alors

 si (le site a voté non ou s'il a noté
          abandon(ident_transaction) alors
  envoyer (XA_ROLLBACK);
  sinon
     si (le site a noté
          validation(ident_transaction) alors
          envoyer (XA_COMMIT);
     sinon

     /* Le répondeur est aussi incertain */
     /* problème non résolu                */
     finsi ;
  finsi ;
finsi ;

fin terminaison_répondeur ;
```

Chapitre II

Exemples des protocoles normalisés

X/open DTP, OSI TP

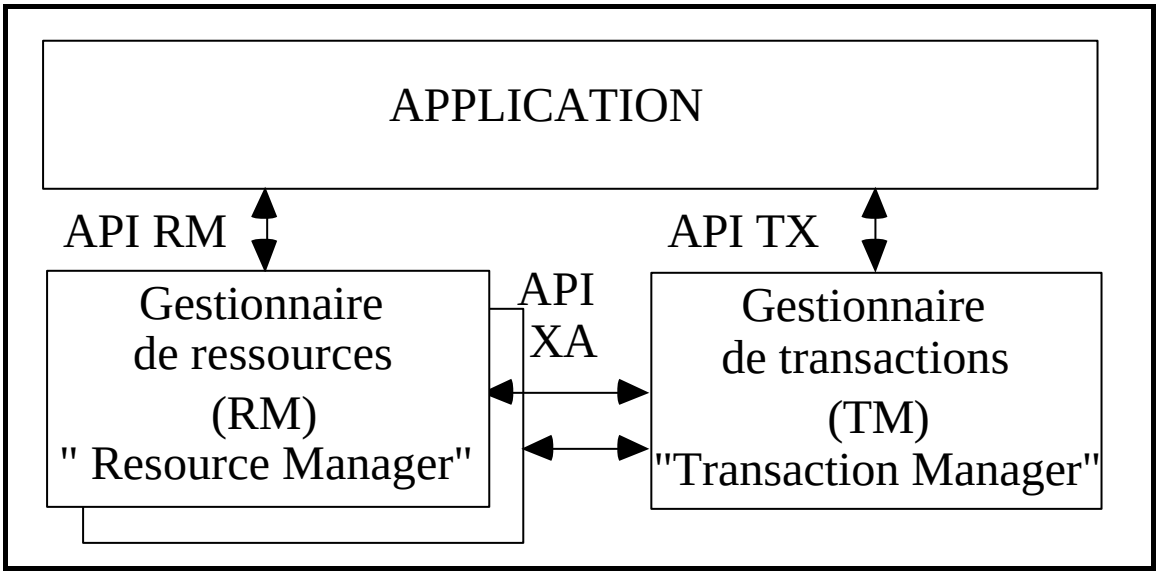
"Distributed Transaction Processing"

II.1 Modèle de référence du traitement transactionnel de l'X/open

Vision X/open centralisée (1991) ("TPRM Transaction Processing Reference Model")

Objectif: donner une interface transactionnelle indépendante d'un moniteur transactionnel.

Les programmes ne dialoguent qu'avec des gestionnaires de ressources locales (a priori plusieurs).



Gestionnaire de ressources
RM Resource Manager

C'est un gestionnaire de données partagées **persistantes**: base de données, système de fichiers...

- Il offre des possibilités de **coordination** au moyen d'un protocole de validation à deux phases.
- Son interface est accessible directement à l'usager et surtout au moyen du protocole XA.

L'interface XA

xa_start: mettre un gestionnaire de ressource en mode traitement d'une transaction.
xa_prepare, xa_commit, xa_rollback: réaliser une validation à deux phases.
xa_end : finir une transaction.
ax_reg : un gestionnaire de ressources s'enregistre auprès d'un gestionnaire de transactions.

Gestionnaire de transactions
TM Transaction Manager

C'est un gérant de transactions qui contrôle pour le compte de l'usager des gestionnaires de ressources.

L'interface TX

tx_begin: une application délimite le début d'une nouvelle transaction.

tx_commit, tx_rollback: pour valider ou pour reprendre une transaction.

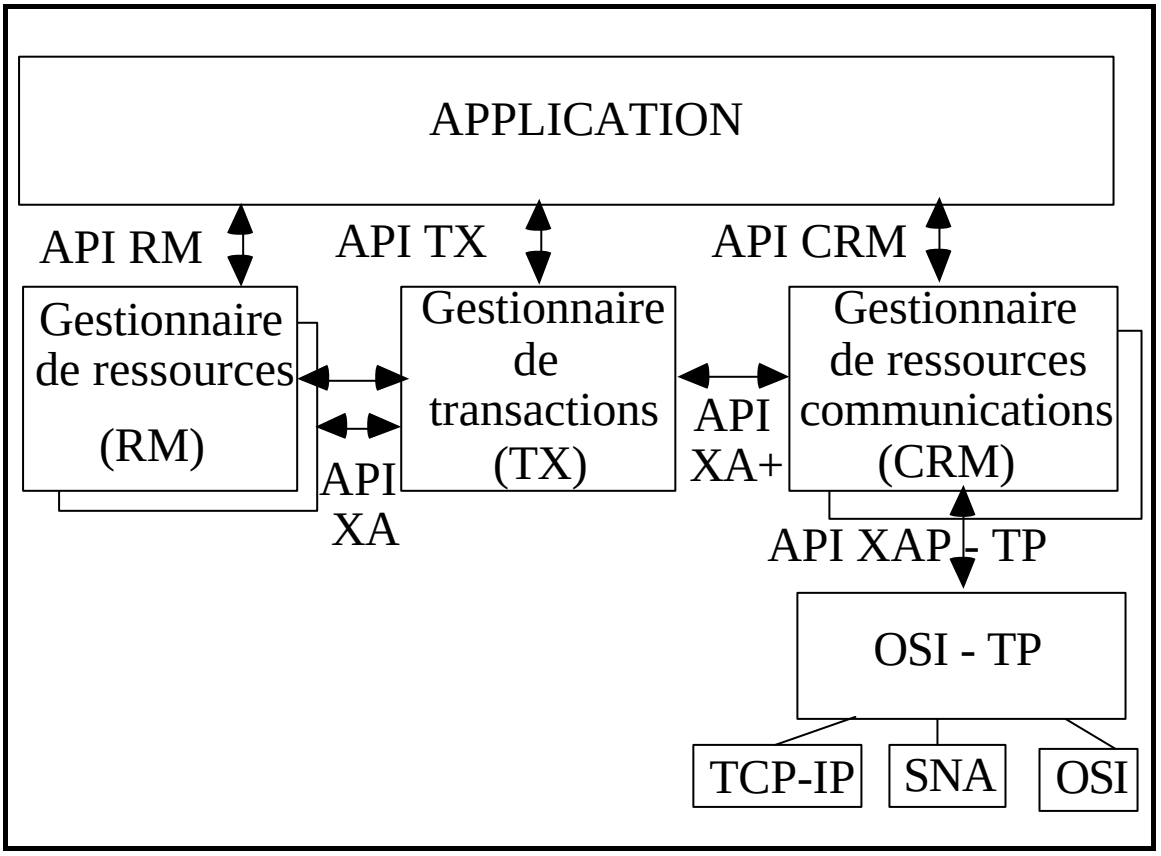
tx_info: informations sur une transaction.

tx_set_transaction_control: pour utiliser des transactions chaînées (avec des validations partielles pendant une transaction importante).

Vision X/open répartie (1994) ("DTP Distributed Transaction Processing")

Objectif: donner une interface de communication réseau entre des gestionnaires de transactions pour faire du transactionnel réparti.

S'appuyer pour cela sur la norme de transactionnel OSI-TP et les protocoles réseaux habituels.



Gestionnaire de ressources communications
CRM Communications Resource Manager

C'est un outil de communications réseaux pour acheminer à distance les directives transactionnelles.

- Trois types d'interface usager ont été normalisées (notées sur la figure API CRM):

XATMI - dérivée de l'interface **ATMI** du moniteur transactionnel **TUXEDO**.
CPI-C V2 - une interface de communication entre applications dans l'univers **IBM**
TxRPC - offre une version transactionnelle pour le **RPC** de **DCE**.

- L'interface entre les gestionnaires de transactions (TM) et le gestionnaire de communication est définie par une extension aux communications du protocole XA appelée XA+.

II.2 OSI TP
"Transaction Processing"

II.2.1 Principes généraux de conception de OSI TP

Définir un environnement pour le support de transactions réparties en validation à deux phases

- Fournir une coordination **"multi-partie"** des ressources.

- Permettre suite à une panne **la restauration d'un état cohérent.**

- **Détecter les pannes**
(communications, systèmes, ...).

- Permettre de **redémarrer une transaction** suite à une restauration.

- Indiquer la **bonne** ou la **mauvaise fin** d'une transaction.

Principes généraux de OSI TP

Fournir la possibilité de délimiter et de structurer un ensemble de transactions imbriquées.

- Organiser une transaction sous forme d'un **arbre de sous-transactions**.

- Permettant le **regroupement de transactions** à l'intérieur d'une application

Supporter différentes politiques de contrôle de concurrence.

- Le protocole ne choisit pas une technique de contrôle de concurrence.

- Chaque gérant de données implante la sienne (grande variété des méthodes, performances).

- S'il y a problème dans un gestionnaire de données => redémarrage.

Offrir des moyens de sécurité

- Supporter différentes **politiques de contrôle d'accès** (au moins celles définies dans la recommandation de sécurité OSI 7498-2).

- **Classifier les objets dans des groupes** afin de faciliter le contrôle d'accès.

- **Authentifier** les activités utilisatrices.

- Permettre **la non répudiation** à la participation à une transaction.

Notion de dialogue

- Un traitement transactionnel est constitué par un ensemble de **dialogues** entre **entités** utilisatrices **paires**.

- Un dialogue = une communication dans une relation **point à point.**

- Au moyen d'un "dialogue" deux entités utilisatrices ("TPSUI") peuvent :

. Échanger des données
. Notifier des erreurs
. Réaliser une transaction
(valider, invalider ...)
. Ordonner la fin du dialogue
. Se synchroniser
(en fonction des besoins)

Modes de contrôle des dialogues.

Mode polarisé
("Polarized control mode")

Un utilisateur contrôle le dialogue à un instant donné.

Il doit disposer d'un jeton (voir le service de session) pour réaliser des opérations autres qu'urgentes (notification d'erreurs, reprise (rollback), terminaison non négociée).

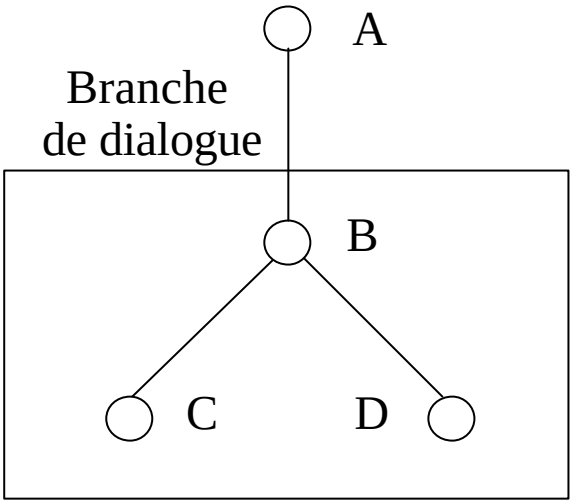
Mode Partagé
("Shared mode")

Les deux utilisateurs possèdent le contrôle du dialogue simultanément.

Les dialogues sont donc réalisés sans contrôle de jeton.

Arbre de dialogue

C'est un arbre qui a comme noeuds les entités utilisatrices et comme arcs les dialogues ("dialogue branch").



Sous-arbre d'une transaction
emboitée dans l'arbre global

Exemple:

Le site A (direction) interroge B (comptabilité) pour connaître le montant des ventes du mois de l'entreprise.

Les ventes sont réalisées dans deux divisions qui ont leurs propres systèmes de gestion C et D.

B créé une transaction emboîtée à l'intérieur de la transaction demandée par A pour collecter les chiffres nécessaires.

Niveau de coordination d'un dialogue

A) L'opération est un dialogue simple

. Exemple l'interrogation consiste en une simple lecture (comme celle de A vers B).

. Son niveau de coordination de dialogue est sans contrôle mot-clé **"NONE".**

. Les propriétés ACID ne sont pas nécessaires ou sont directement prises en charge par l'utilisateur.

B) La branche de dialogue concerne une vraie transaction

. Son niveau de coordination est **"COMMITMENT".**

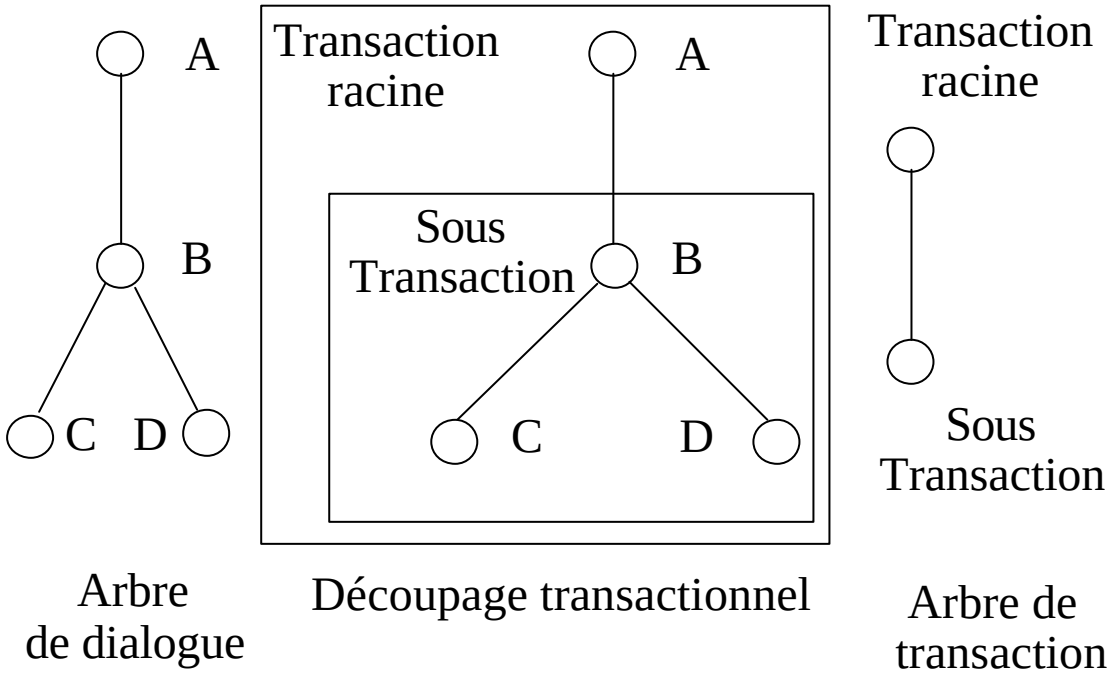
. Les propriétés ACID doivent être satisfaites par le prestataire du service transactionnel (sauf concurrence).

Arbre des transactions

C'est un arbre emboîté dans l'arbre des dialogues

Il a comme noeuds des entités utilisatrices qui définissent elles aussi des transactions (qui doivent satisfaire les propriétés ACID).

Exemple: (précédent modifié)
- Une transaction racine émise par A doit satisfaire les propriétés acid.
- Elle utilise une sous-transaction sur B qui elle aussi doit satisfaire ces propriétés.



II.2.2 Le service OSI TP

Les unités fonctionnelles du service

Gestion des dialogues

Fournir les services de base nécessaires à la gestion des dialogues.

Contrôle Partagé, polarisé

Utilisation simultanée ou à l'alternat du service. Demande ou relâche du contrôle.

Synchronisation

Permettre la synchronisation forte entre deux activités (activités au même point).

Validation

Fournir les services de validation ou d'invalidation de transactions.

Transactions chaînées

Gestion de suites de transactions dont certaines sont non validées (de niveau de coordination "none").

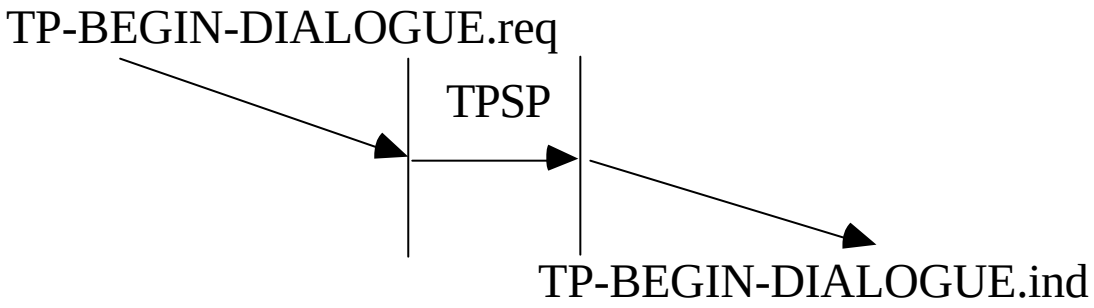
L'unité fonctionnelle de gestion de dialogue

1 TP-BEGIN-DIALOGUE req/ind

Demande d'ouverture d'un dialogue**.**

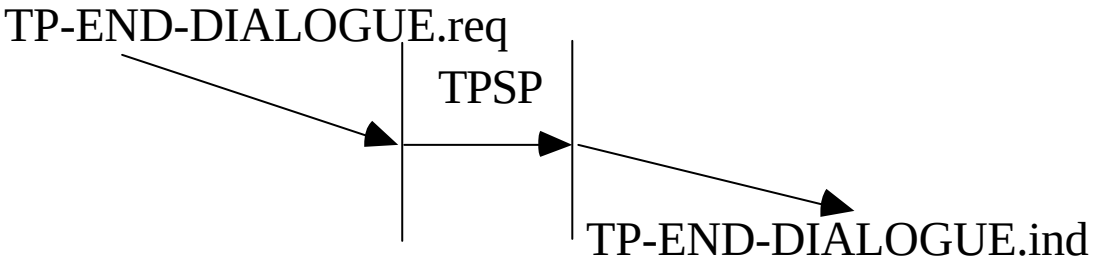
Paramètres

- Identificateurs émetteur et destinataire.
- Unités fonctionnelles désirées.
- Qualité de service.
- Niveau de coordination ("null" "commit").



2 TP-END-DIALOGUE req/ind

Demande de fin d'un dialogue**.**

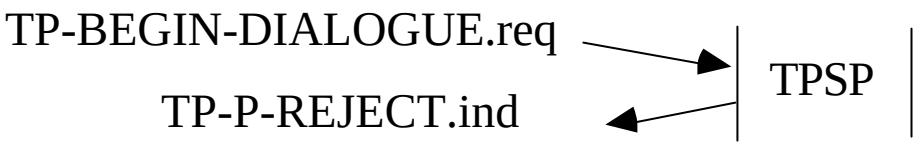


3 TP-P-REJECT indication

Le prestataire ne peut satisfaire la requête.

Paramètres : motif du rejet

- Identificateur d'usager inconnu
- Unités fonctionnelles non supportées
- Raison non spécifiée

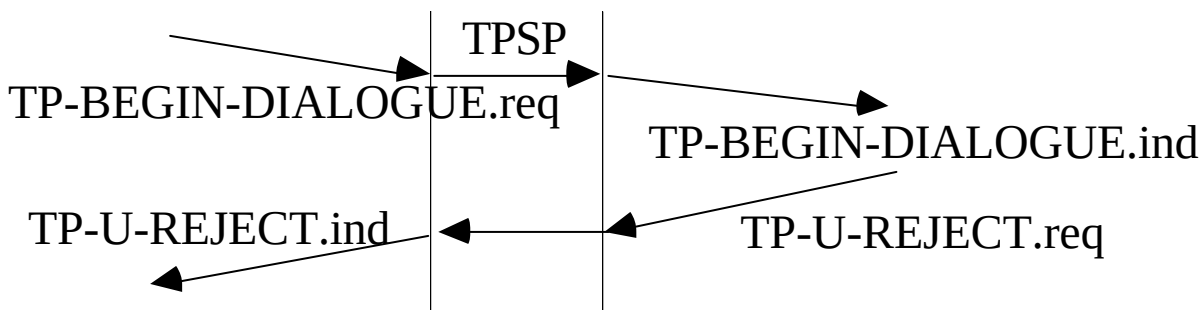


4 TP-U-REJECT request/indication

L'usager distant ne peut satisfaire la requête

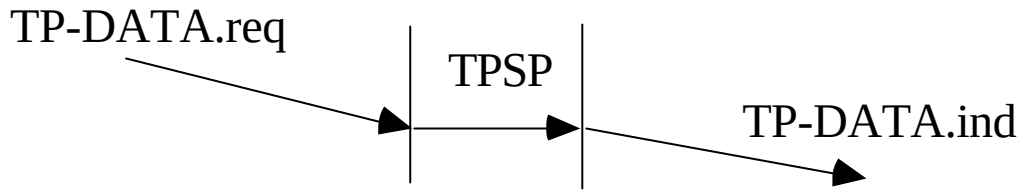
Paramètres

- ROLLBACK s'il faut une reprise.
- Motifs usagers précisant de rejet.



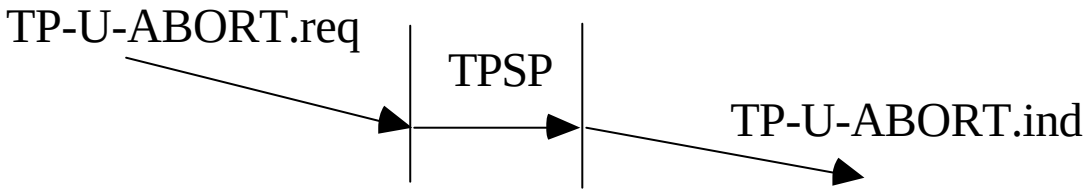
5 TP-DATA request/indication

Échange de données utilisateur**.**



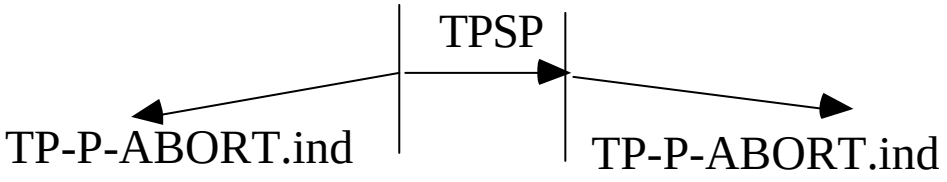
6 TP-U-ABORT request/indication

Abandon immédiat de transaction sur requête utilisateur (entraîne la reprise ROLLBACK si le mode commitment est en cours)**.**



7 TP-P-ABORT indication

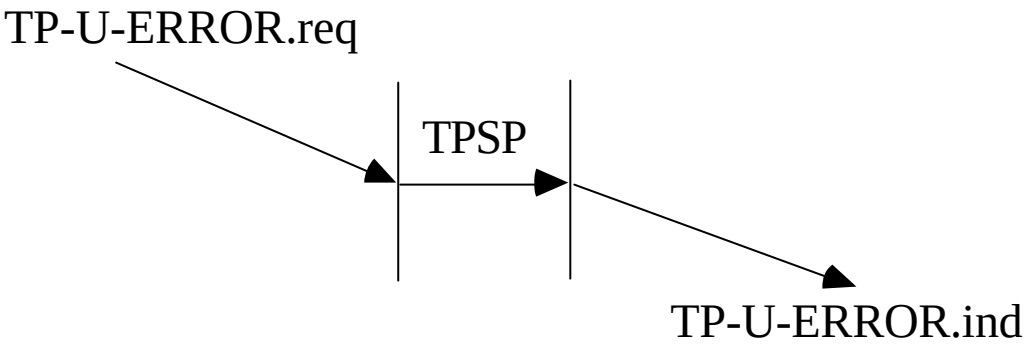
Erreur entraînant l'abandon de la transaction signalée par le prestataire de service (avec notification de la cause de l'erreur).



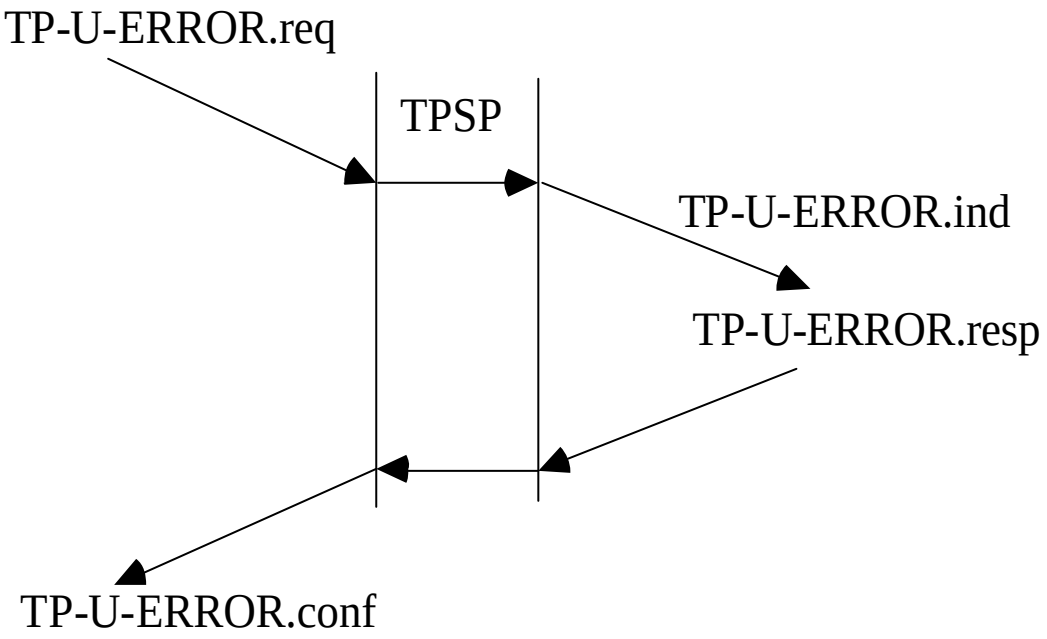
8 TP-U-ERROR request/indication response/confirmation

Signalement d'erreur utilisateur**.**

Cas du mode polarisé : échange non confirmé



Cas du mode partagé: échange confirmé



Les unités de gestion de contrôle polarisé et partagé

1 Contrôle partagé: pas de service.

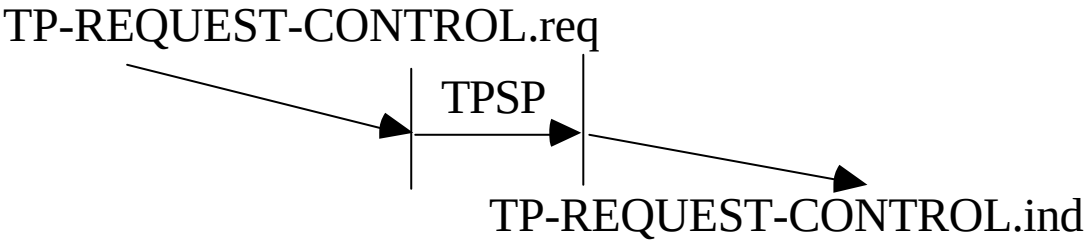
Dans cette unité les usagers peuvent émettre des requêtes simultanément (elles sont simplement soumises à la séquence autorisée de ces primitives).

2 Contrôle polarisé

Seul le possesseur du contrôle peut agir**.** Deux primitives permettent de demander le contrôle ou de le céder.

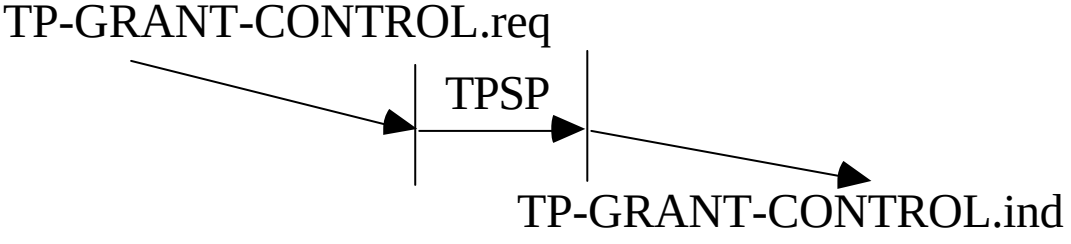
TP-REQUEST-CONTROL req/ind

Demander le contrôle au site distant**.**



TP-GRANT-CONTROL req/ind

Céder le contrôle au site distant**.**



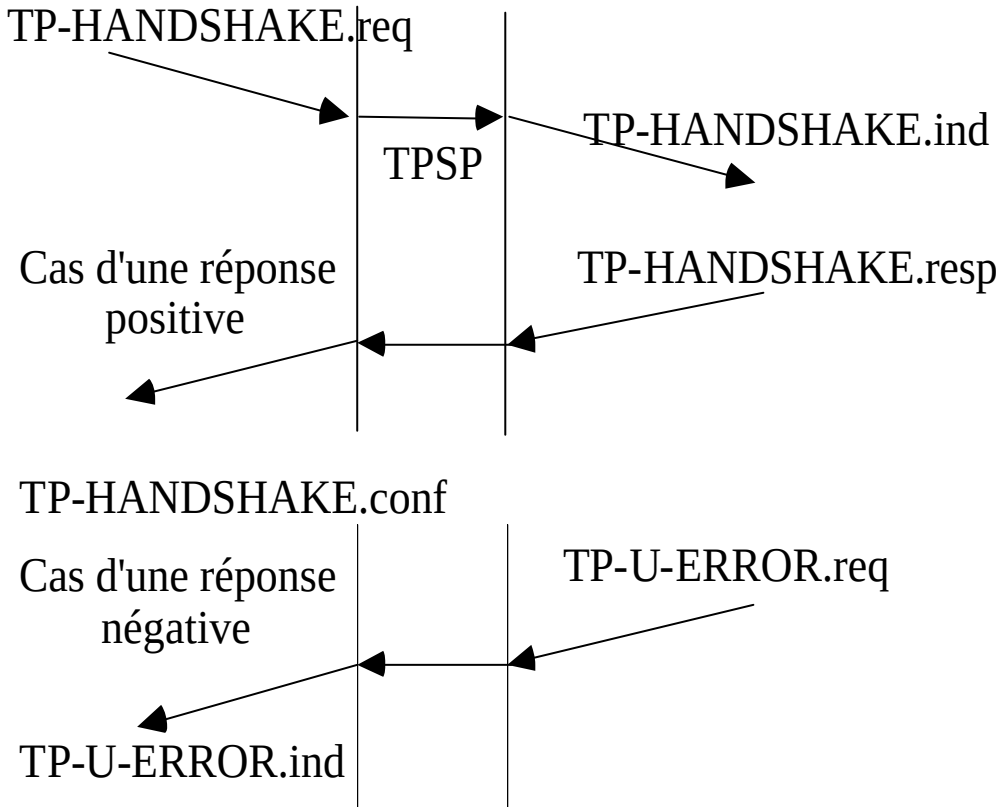
L'unité fonctionnelle de synchronisation

1 TP-HANDSHAKE req/ind resp/confirm

Les deux entités d'un dialogue se synchronisent sur l'état d'avancement des traitements.

La confirmation requise concerne l'avancement du traitement du destinataire jusqu'à un point précis.

Si la confirmation est impossible alors il s'agit d'une erreur utilisateur.



2 TP-HANDSHAKE-AND-END req/ind resp/confirm

Les deux entités d'un dialogue se synchronisent sur l'état d'avancement des traitements et terminent le dialogue.

Exactement le même comportement que le "handshake" de base.

3 TP-HANDSHAKE-AND-GRANT-CONTROL req/ind resp/confirm

Élément de service utilisable en mode polarisé.

Les deux entités d'un dialogue se synchronisent sur l'état d'avancement des traitements et l'unité demande la synchronisation cède le contrôle en mode polarisé

Exactement le même comportement que le "handshake" de base.

L'unité fonctionnelle de validation

Utilisation d'une validation à deux phases sur l'arbre des transactions

- La validation (ou l'invalidation) est demandée par le noeud racine.
- La requête se propage récursivement à l'intérieur de l'arbre des transactions.

Originalité principale :
l'arbre des transactions.

Première phase

Toutes les unités transactionnelle sont amenées à se trouver dans un état prêt à valider (READY).

Une décision d'abandon peut-être prise par un seul intervenant pendant la première phase:
elle sera notifiée à tous.

Seconde phase

L'état prêt ayant été collecté la transaction est validée récursivement dans l'arbre des transactions.

Éléments de services en Phase 1
Optimisation d'une transaction longue

Mise en état READY pour validation d'un sous-arbre.

- Optionnel pour améliorer les performances
- Plus aucun échange n'est autorisé

TP-PREPARE.req --> TP-COMMIT.ind

TP-PREPARE.req demande à un sous-arbre complet de transactions de se mettre en état prêt (indication **TP-COMMIT.ind**).

TP-CONTINUE-COMMIT.req
-------> TP-READY.ind

Si le subordonné est OK il répond **TP-CONTINUE-COMMIT.req.** Si tous les subordonnés ont accepté la validation le demandeur reçoit un **TP-READY.ind**.

TP-ROLLBACK.req
----------> TP-ROOLBACK.ind

Si un site n'est pas en état de valider il génère **TP-ROLLBACK.req** qui se traduit par **TP-ROOLBACK.ind** chez le demandeur (et tous les autres sites).

Éléments de services en Phase 1
Préparation à valider proprement dite

Mise en état READY de tous les sites subordonnés: obligatoire pour la validation.

TP-COMMIT.req -----> TP-COMMIT.ind

TP-COMMIT.req transmis par la racine de la transaction génère chez tous les sites de l'arbre un **TP-COMMIT.ind**

Ceci initialise la phase de terminaison de la transaction (la racine comme les subordonnés ne peuvent plus émettre de données).

TP-CONTINUE-COMMIT.req ------->
TP-COMMIT-RESULT.ind

Cas des subordonnés OK **TP-CONTINUE-COMMIT.req** se traduit chez le demandeur finalement par **TP-COMMIT-RESULT.ind**:

TP-ROLLBACK.req ----->
TP-ROOLBACK.ind

Cas des subordonnés KO.

Éléments de services utilisés en Phase 2
Cas d'une transaction validée

Si tous ont répondu **TP-COMMIT.req** le prestataire de service TP génère un **TP-COMMIT-RESULT.ind** pour tous (aussi bien le coordinateur que les subordonnés).

La validation étant réalisable chaque subordonné l'effectue et signale la terminaison par un message **TP-DONE.req.**

Lorsque tous les subordonnés ont signalé TP-DONE.req une transaction le prestataire de service du site coordinateur génère une primitive de terminaison totale **TP-COMMIT-COMPLETE.ind** à tous les participants.

Éléments de services utilisés en Phase 2
Cas d'une transaction invalidée

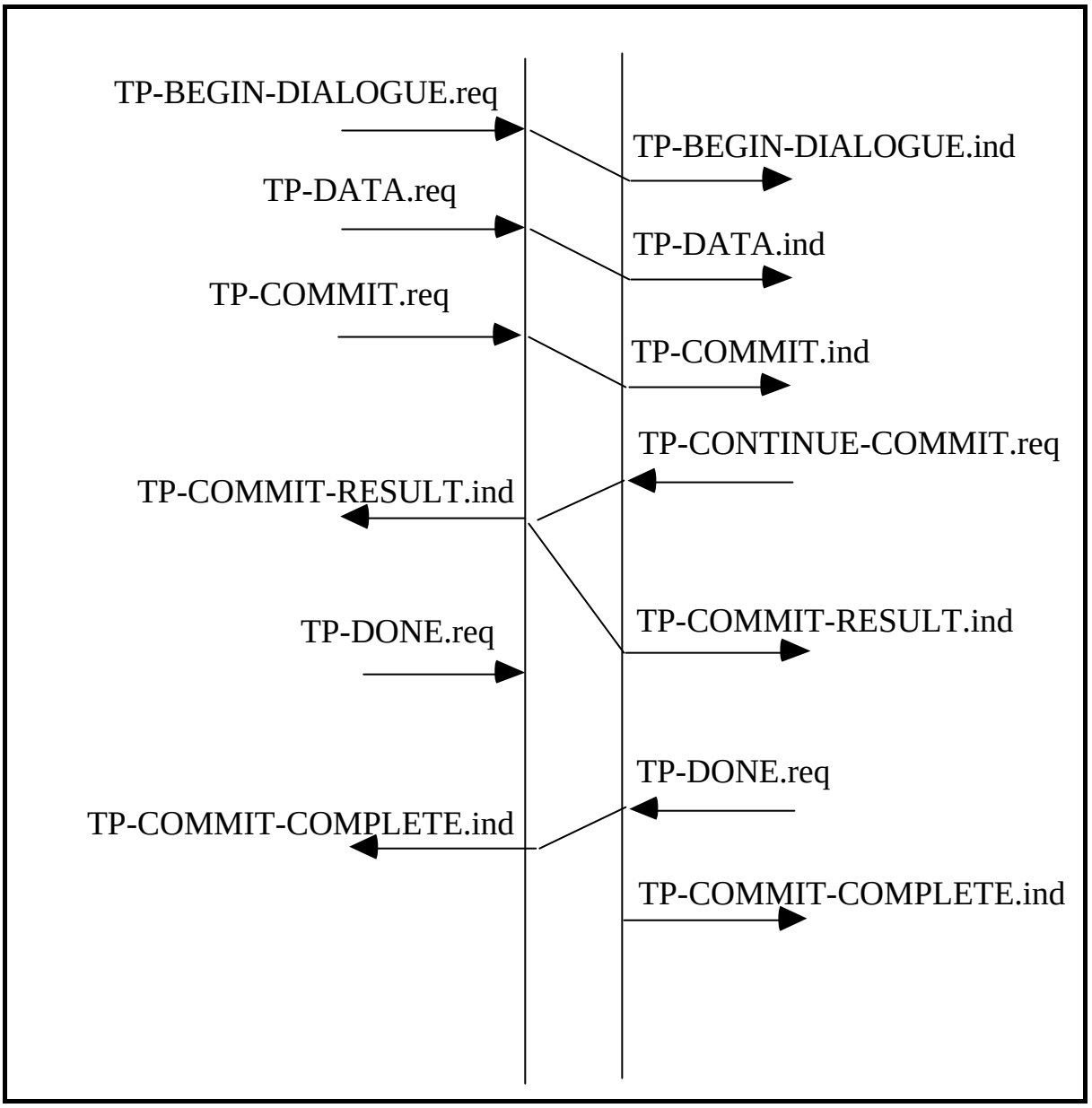
Si un seul a généré **TP-ROLLBACK.req** la transaction doit-être invalidée.

Le prestataire de service TP génère pour tous les participants un **TP-ROLLBACK.ind**.

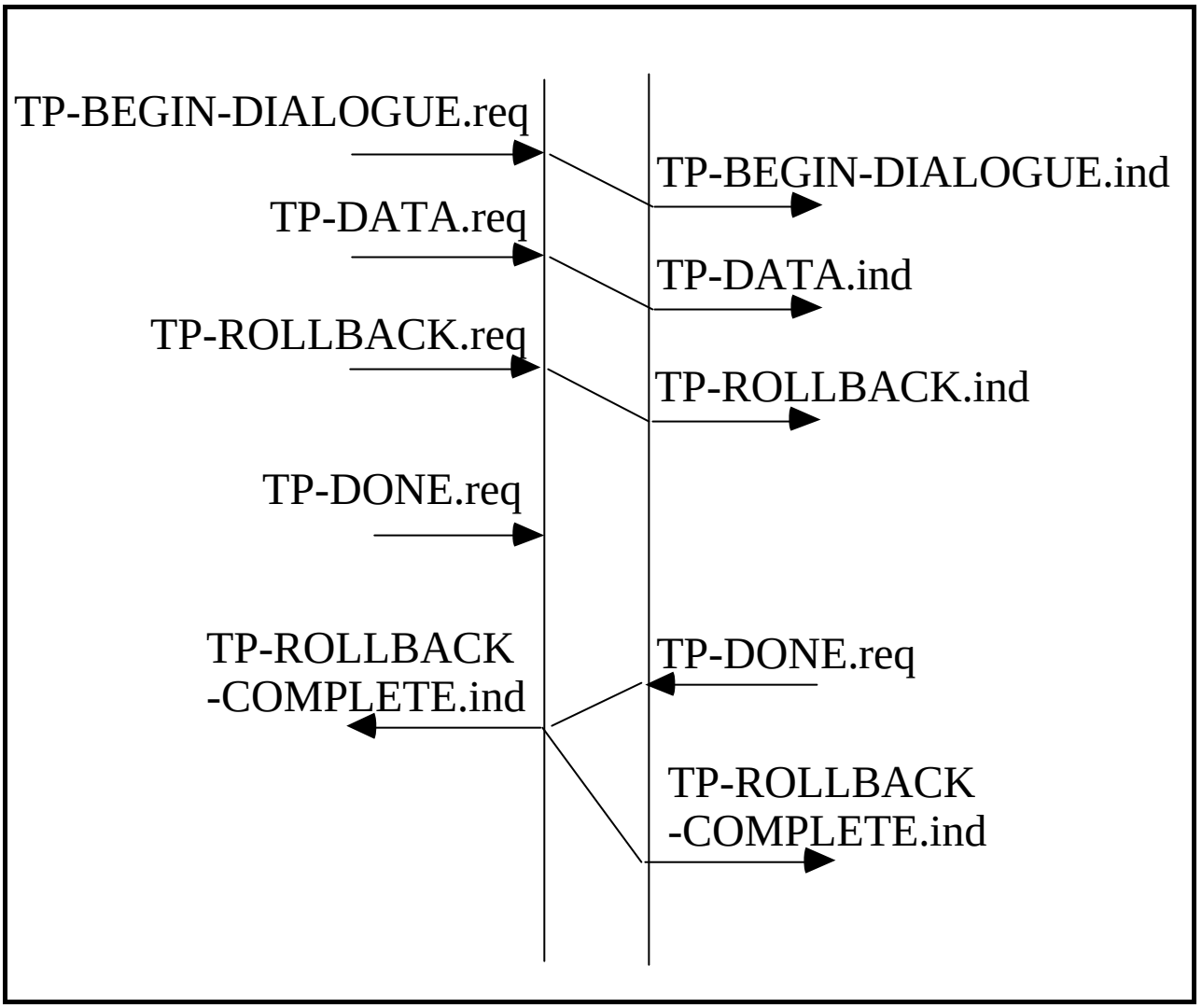
L'invalidation est réalisée par tous les sites qui détruisent leurs données et signalent **TP-DONE.req**.

Lorsque tous les subordonnés ont terminés le prestataire de service du site coordinateur génère une primitive de terminaison en invalidation **TP-ROLLBACK-COMPLETE.ind** à tous.

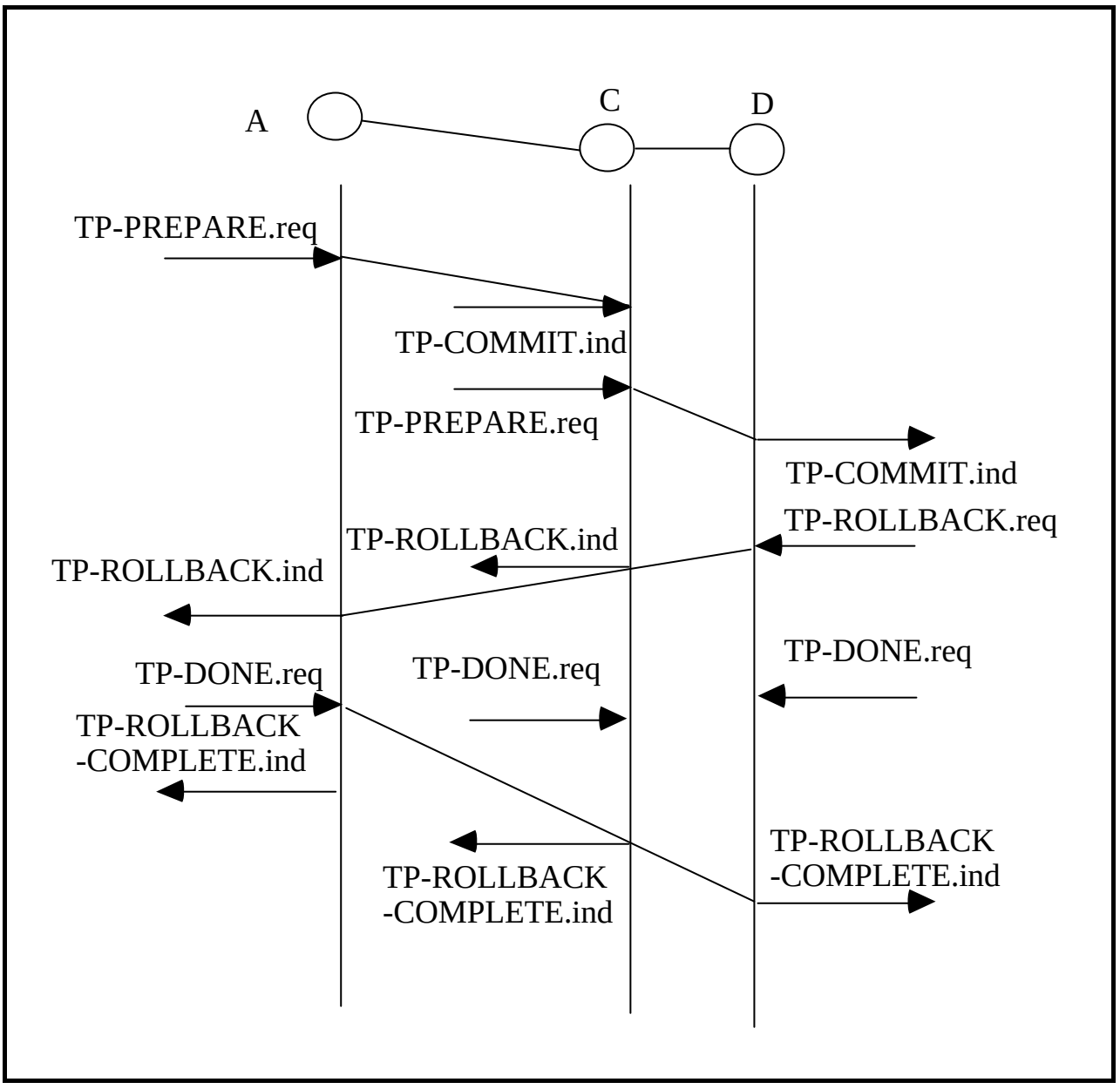
Exemple d'une transaction à deux participants avec validation



Exemple d'une transaction à deux participants avec invalidation



Exemple d'une transaction à trois participants avec invalidation



Conclusion transactionnel

Un outil très important dans les applications client serveur

Moniteurs transactionnels

Tuxedo de BEA systems

Initialement le transactionnel de référence sous UNIX améliré puis porté sur d'autres plates-formes (Windows NT).

Encina de Transarc IBM
Top End de ATT
Pathway de Tandem

Une évolution en cours avec l'introduction des client serveurs en approche objet réparti

L'utilisation des transactionnels comme outils de mise en oeuvre du contrôle de concurrence de la tolérance aux pannes, et des performances pour des approches:

CORBA : "Object Transaction Service"

OLE/DCOM : "OLE/Transactions"

Bibliographie

Georges Gardarin, Olivier Gardarin
"Le client-serveur" Eyrolles

Robert Orfali, Dan Harkey, Jeri Edwards,
"Client-serveur, guide de survie" Wiley.

SYSTEMES de FICHIERS REPARTIS

Un Exemple : NFS

Plan

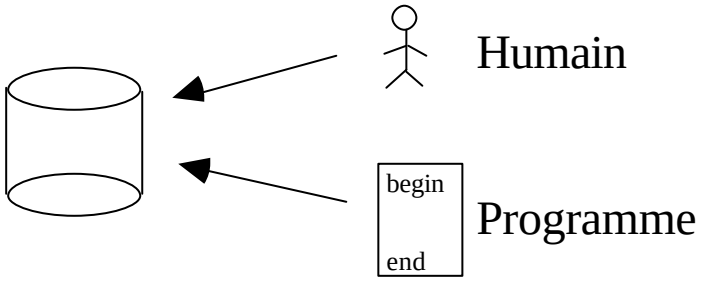
INTRODUCTION

Objectifs d'un système de fichiers (SGF)

* stockage permanent des informations sous forme de fichiers, **on parle de persistance**

Archive sur : disque, bande (dérouleur, cassette, hexabyte, DAT), disque amovible (optique, disquette) ... **mémoire stable**

Nature du contenu :
. données structurées
. données non structurées
. de natures différentes : exécutable, texte, image, son, vidéo, alphanumériques
. importance du volume des informations
* Utilisateurs :



. Contrôle d'accès,
. Partage de fichiers entre utilisateurs

Modèles d'architectures

La simplicité : micro-ordinateur de type PC/MS-DOS
Un utilisateur -> un seul programme à la fois

Le SGF résoud les 4 problèmes suivants :

- **désignation des fichiers** (par une arborescence souvent)
- **interface d'accès** pour les programmes
- **correspondance nom symbolique-adresse physique**
- **intégrité des données par rapport aux pannes** : alimentation, carte processeur, support disque (grace aux sauvegardes), logiciel

Un peu plus : OS/2 ou Macintosh Système 7

Un utilisateur -> plusieurs programmes (processus) à la fois

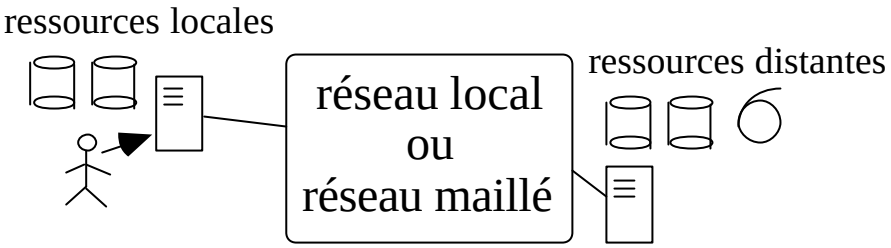
Les données peuvent être accédées de façon concurrente, le SGF prend en charge le **contrôle de concurrence**.

Encore un peu plus : Unix
Système temps partagé -> Multi-utilisateurs et Multi-processus

importance de la sécurité et de la protection

Objectifs d'un système de gestion de fichiers répartis (SGFR)

Extension des propriétés précédentes à un ensemble de machines reliées par un réseau de communication (réseau local, liaisons hertziennes, lignes spécialisées, liaisons modem, NUMERIS).



protocoles plus ou moins fiables :
OSI, Internet, Xerox-Novell, Applelink, ...

Les utilisateurs se répartissent sur les machines et ne travaillent pas toujours au même endroit (mobilité de son environnement)

Postes de travail de natures différentes :

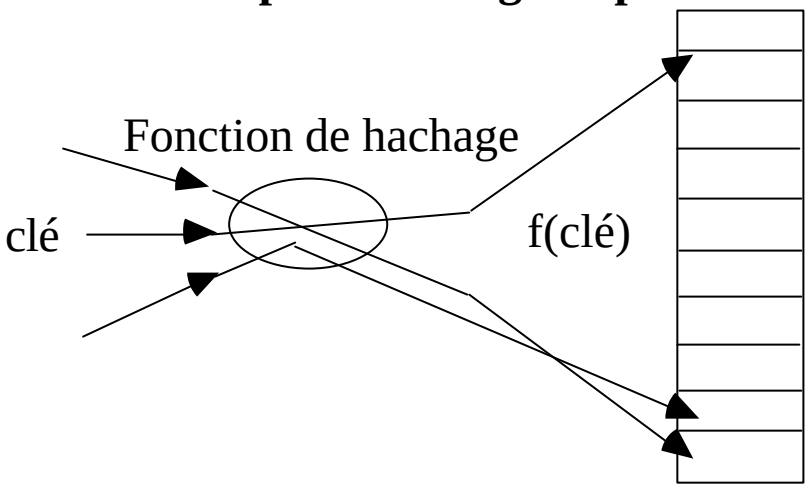
. micro-ordinateur, type PC ou Macintosh
. station de travail
. terminal X et calculateur

Rappels : Modèles de fichiers (1)

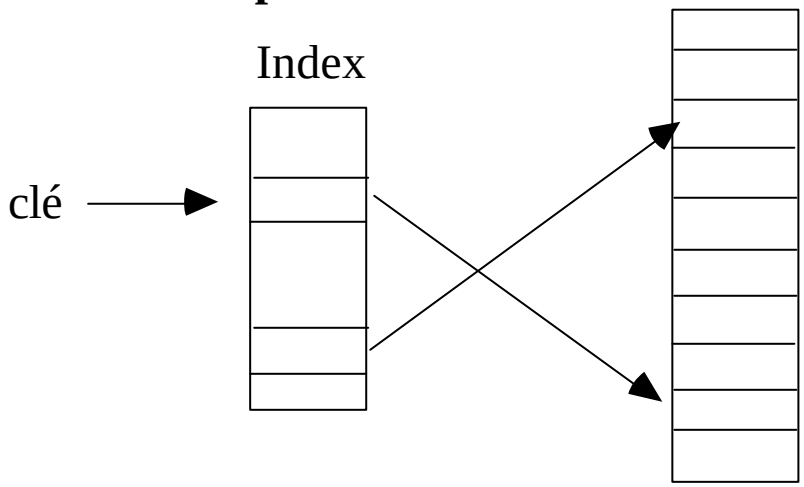
Ce qui distingue les fichiers, c'est leur mode d'accès :

Accès séquentiel : depuis le début

Accès direct par adressage dispersé fonction d'une clé



Accès direct par index sur une clé



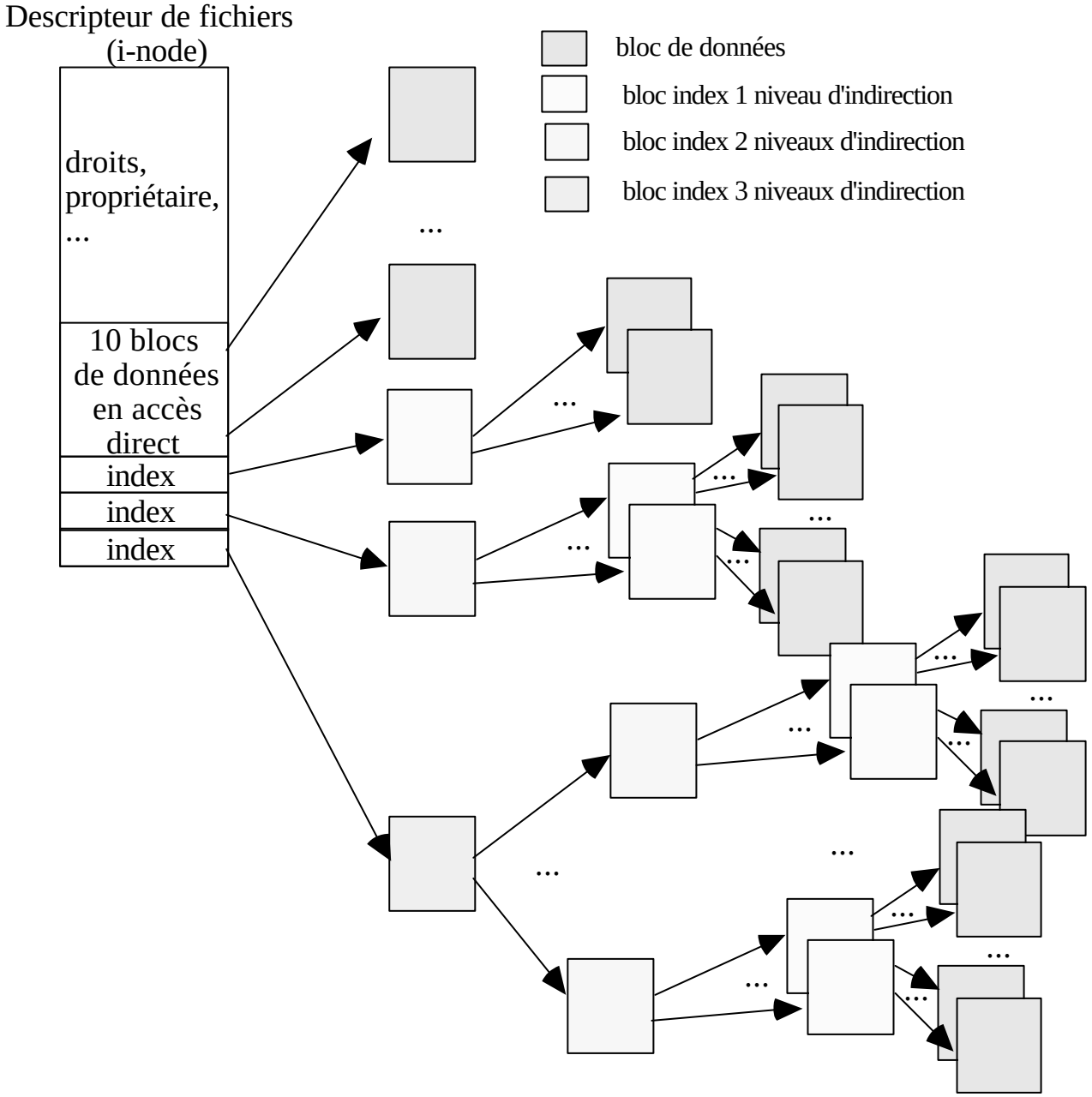
possibilité d'existence de plusieurs index : index primaire et des index secondaires

Rappels : Modèles de fichiers (2)

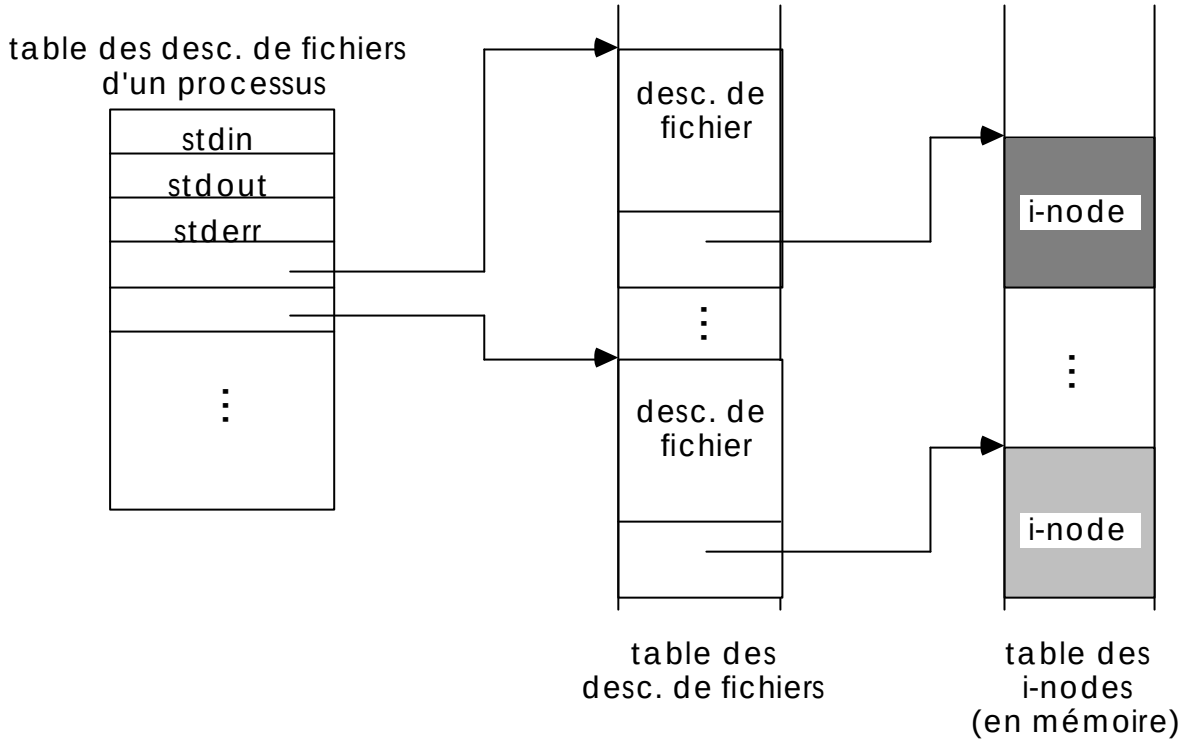
Exemple : Fichiers non structurés, de type Unix ou MS-DOS, les plus répandus

Vue utilisateur :

Vue Système :

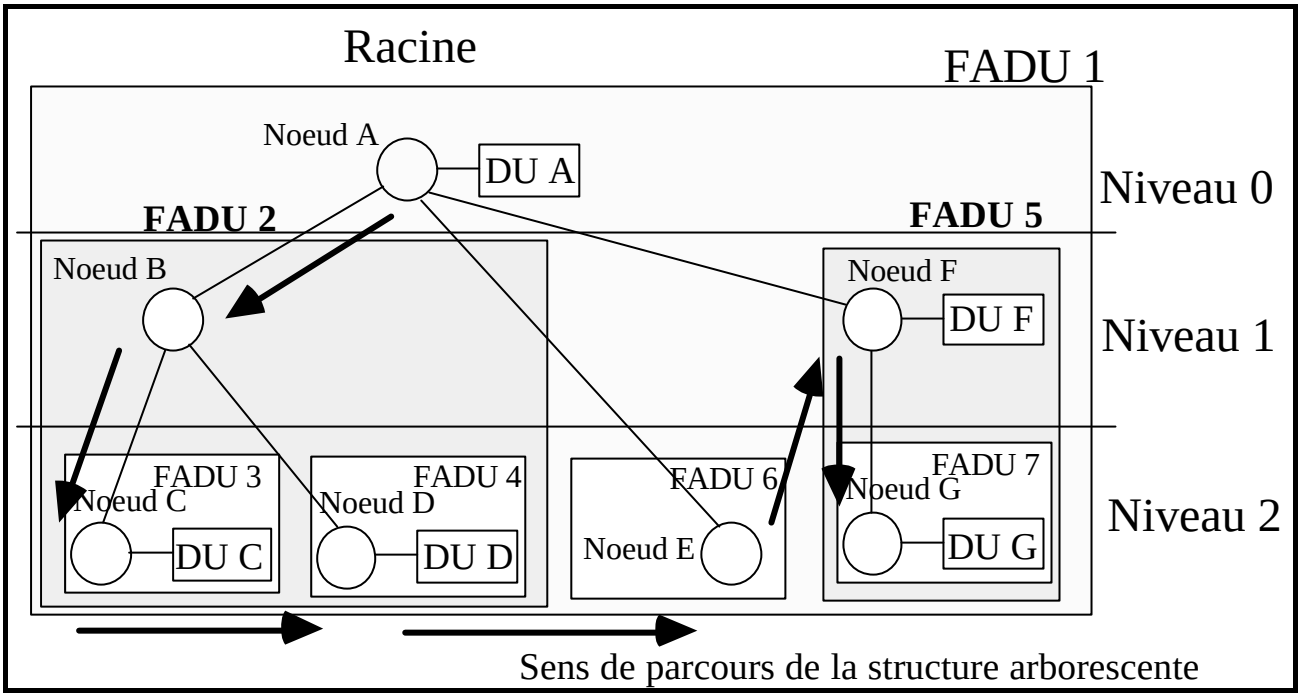


Vision des fichiers dans l'environnement d'un processus Unix :



Modèles de fichiers considérés par FTAM[1] pour le transfert de fichiers

Modèle Général d'un fichier virtuel :



Fichier à structure d'accès hiérarchique :

- un fichier est considéré comme une structure arborescente

- c'est un ensemble de noeuds et d'unités de données (DU) regroupés en unités d'accès au fichier (FADU)

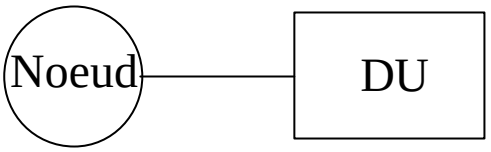
- une FADU est un sous arbre de l'arbre complet[2]

[1] FTAM : File Transfert Access Management

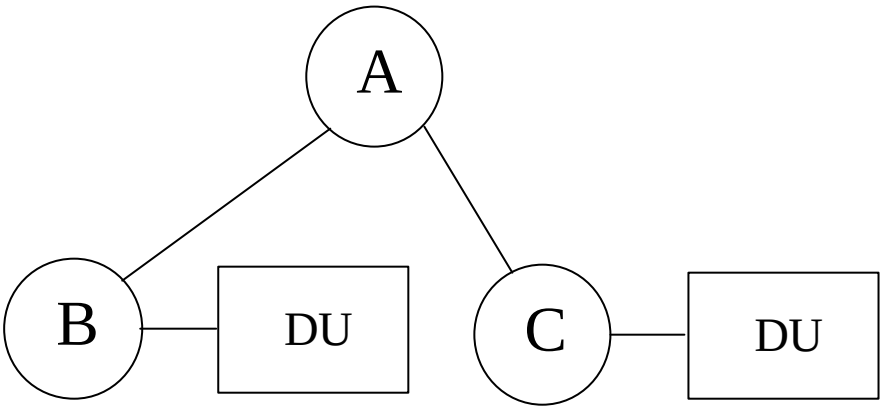
[2] les FADU sont transférées d'un site FTAM à l'autre par envoie des données et des informations de structure

Exemples :

Fichier virtuel non structuré (type Unix) :



Fichier virtuel séquentiel plat :



Approches de la répartition de fichiers (1)

1. **Transfert de fichiers :** Kermit, UUCP, ftp-Internet, ...
-> FTAM en est la version normalisée, un fichier est accessible par programme ou par utilisateur (commande)

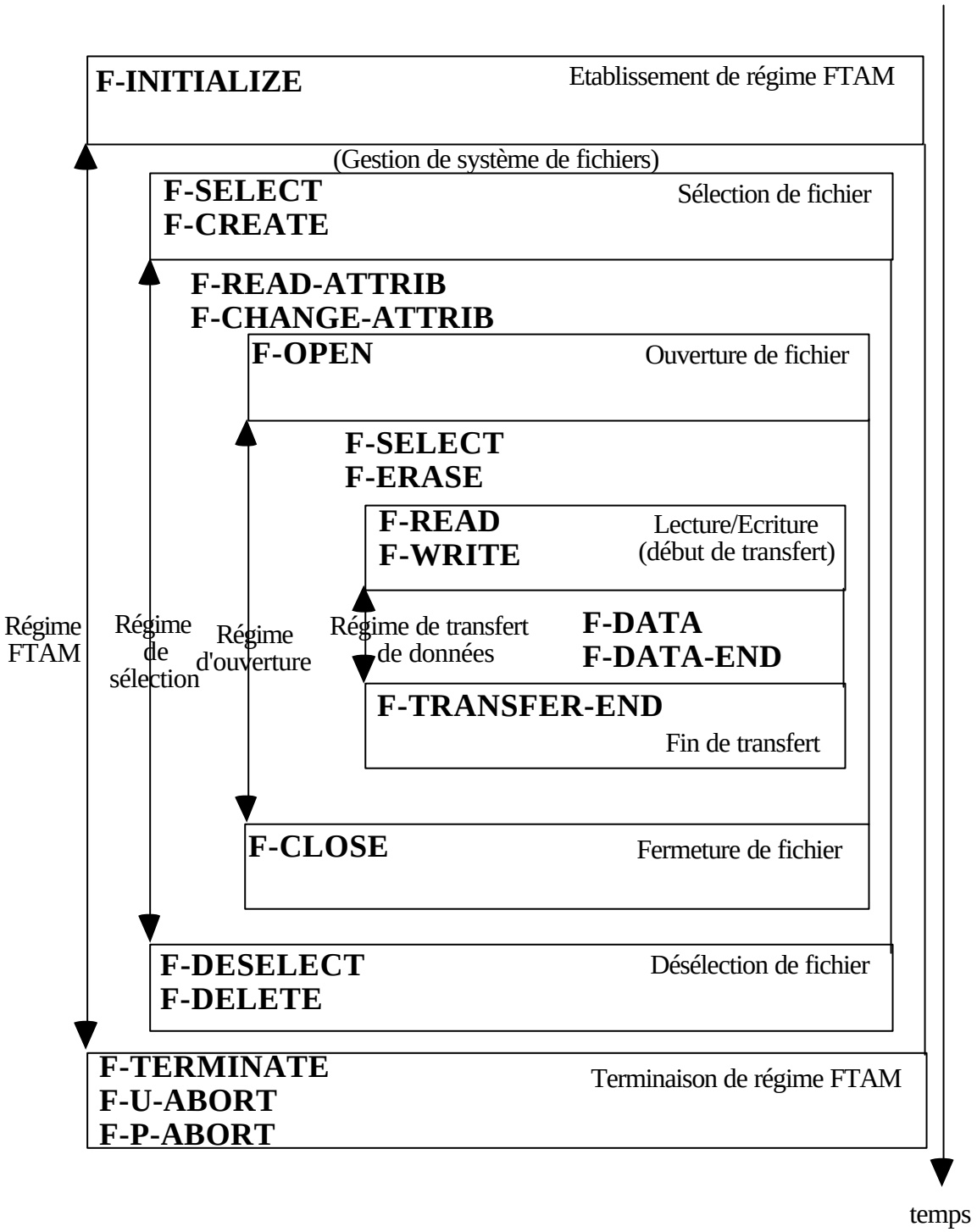
- permet la gestion de l'hétérogénéité : de machines, de systèmes d'exploitation, de structures de fichiers, de systèmes de fichiers
- ensemble de primitives de service dont l'enchainement est structuré par des régimes, chaque régime correspond à des opérations particulières.

Les opérations de lecture et d'écriture d'un fichier impliquent tout ou partie du contenu de celui-ci. Elles font référence à la FADU concernée par l'opération et indique en cas d'écriture la spécification de l'action demandée (insertion, remplacement, extension).

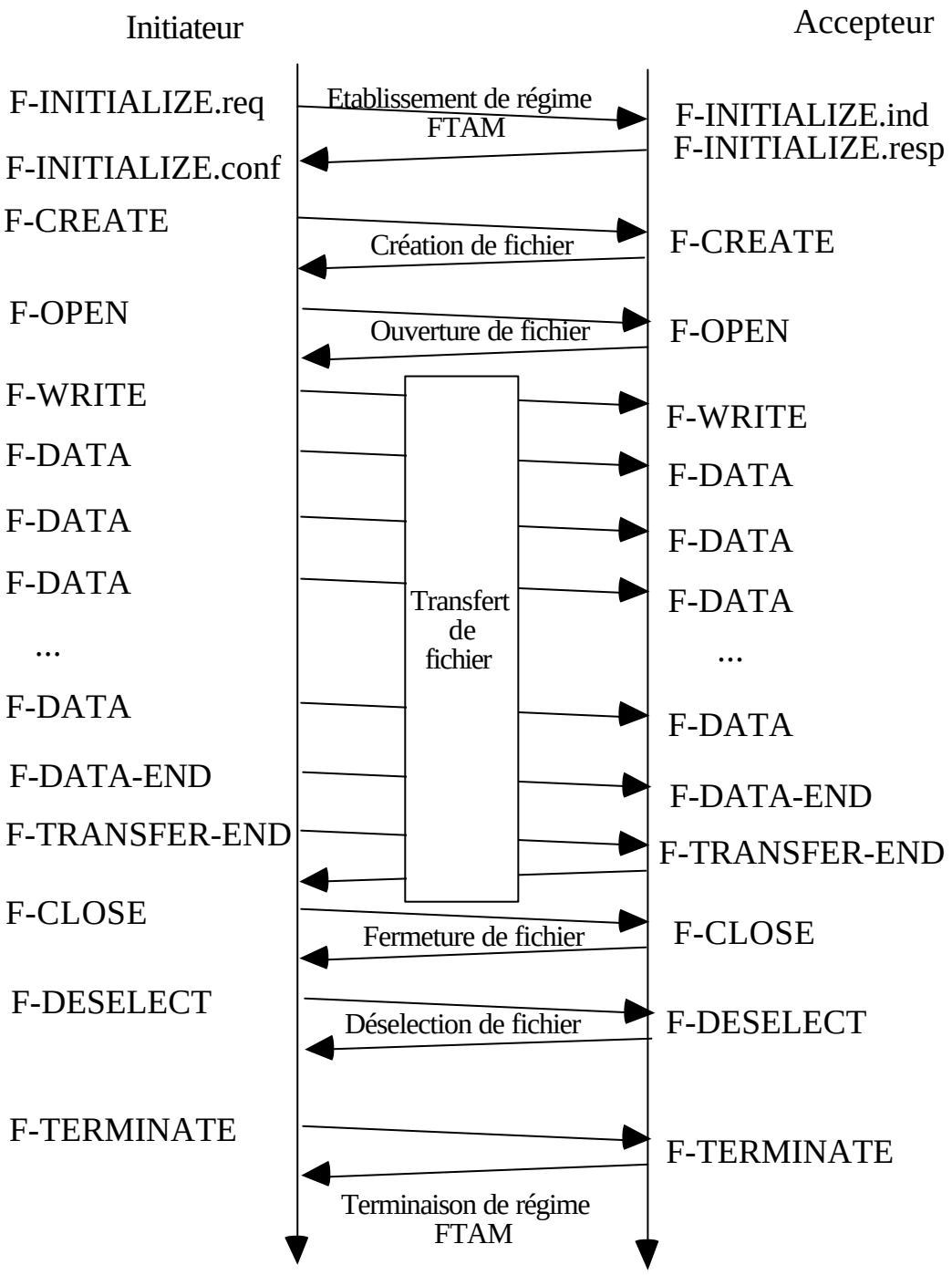
Le transfert proprement dit est marqué par une suite de primitives F-DATA, qui marquent à chaque fois le transfert d'un segment de données, un F-DATA-END indique la fin du transfert de données.

Il est possible de regrouper certains services en un méta-service, dans ce cas les primitives sont encadrées par F-BEGIN-GROUP et F-END-GROUP qui pourraient être comparées à une "(" et à une ")".

Régimes de service de fichiers FTAM



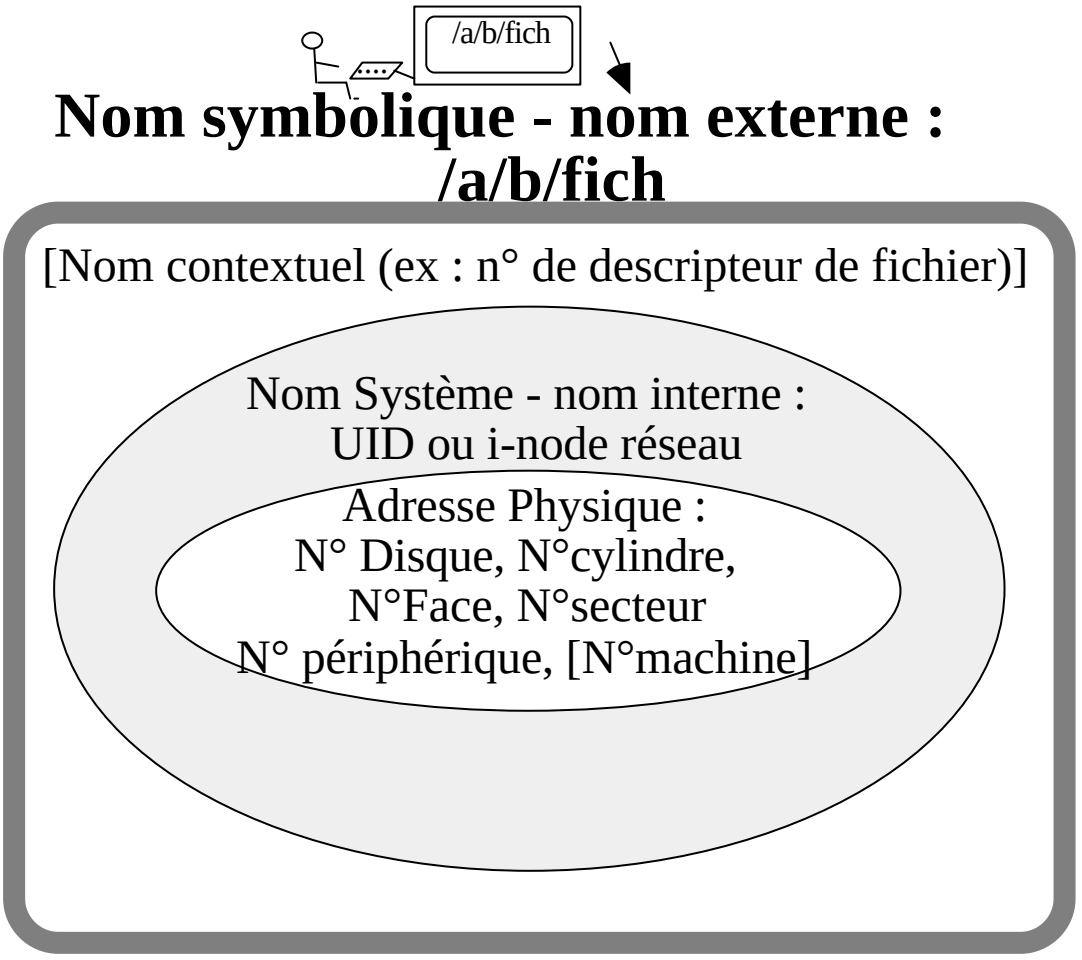
Exemple : Un transfert de fichier



Approches de la répartition de fichiers (2)

2. **Extension du modèle Unix à un réseau de machines :** NFS (Sun)
Apporte : **Performance**, hétérogénéité, modèle Internet, **"transparence"**

3. **Système Réparti :** Chorus (Chorus systèmes), OSF1-Mach (OSF), Amoeba (Tanenbaum)
possède des noms internes uniques et globaux au système (UID)



Apporte : **intégration du système d'exploitation et du système de fichiers**

Approches de la répartition de fichiers (3)

4. **Systèmes à objets répartis :** Commandos (projet Esprit), Guide (IMAG-Grenoble), SPRING (ex projet CLOUDS - futur produit de Sun ?)

La gestion de la persistance des objets masque l'utilisation d'un système de gestion de fichiers

Apporte : une meilleure adéquation entre la conception et l'implantation d'applications distribuées

5. **Bases de Données Réparties :** R* (IBM, produit jamais commercialisé)

l'accès aux données se fait par des prédicats qui portent sur des données réparties à travers la planète

Apporte : un modèle de données (relations, ...) adapté aux besoins des entreprises de gestion

Critères d'analyse d'un SGFR (1)

TRANSPARENCE :

- Transparence d'accès à travers le réseau :

Mêmes opérations d'accès aux fichiers en local qu'en distant, le transfert explicite des données est complètement invisible à l'utilisateur (humain ou programme)

- Transparence à la localisation des données :

Pas de contraintes dans le choix de la machine de travail, les données sont visibles de partout,

Pas besoin d'indiquer le nom de la machine où elles résident

PERFORMANCE :

Mêmes délais en accès fichier local qu'en accès fichier distant. Pour cet aspect, il faut évaluer :

la surcharge induite sur le réseau par le SGFR

le comportement du SGFR coté demandeur d'accès

le comportement du SGFR coté serveur de fichiers

MAINTENABILITE :

Amène une souplesse dans l'administration du SGFR

EXTENSIBILITE :

Evolution facile :

- Ajouts d'utilisateurs
- Ajouts de sites serveurs/ressources
- Ajouts de sites clients
- Interconnexion de réseaux

Critères d'analyse d'un SGFR (2)

TOLERANCE AUX PANNES :
Fonctionnement du SGFR malgré les pannes avec éventuellement une baisse de performances ou une baisse de fonctionnalité proportionnelle au type de panne subie :

- panne processeur (franche "fail-stop", omission, temporelle, byzantine)
- panne réseau (partitionnement, coupure, ...)
- panne du support (totale, corruption des données, ...)

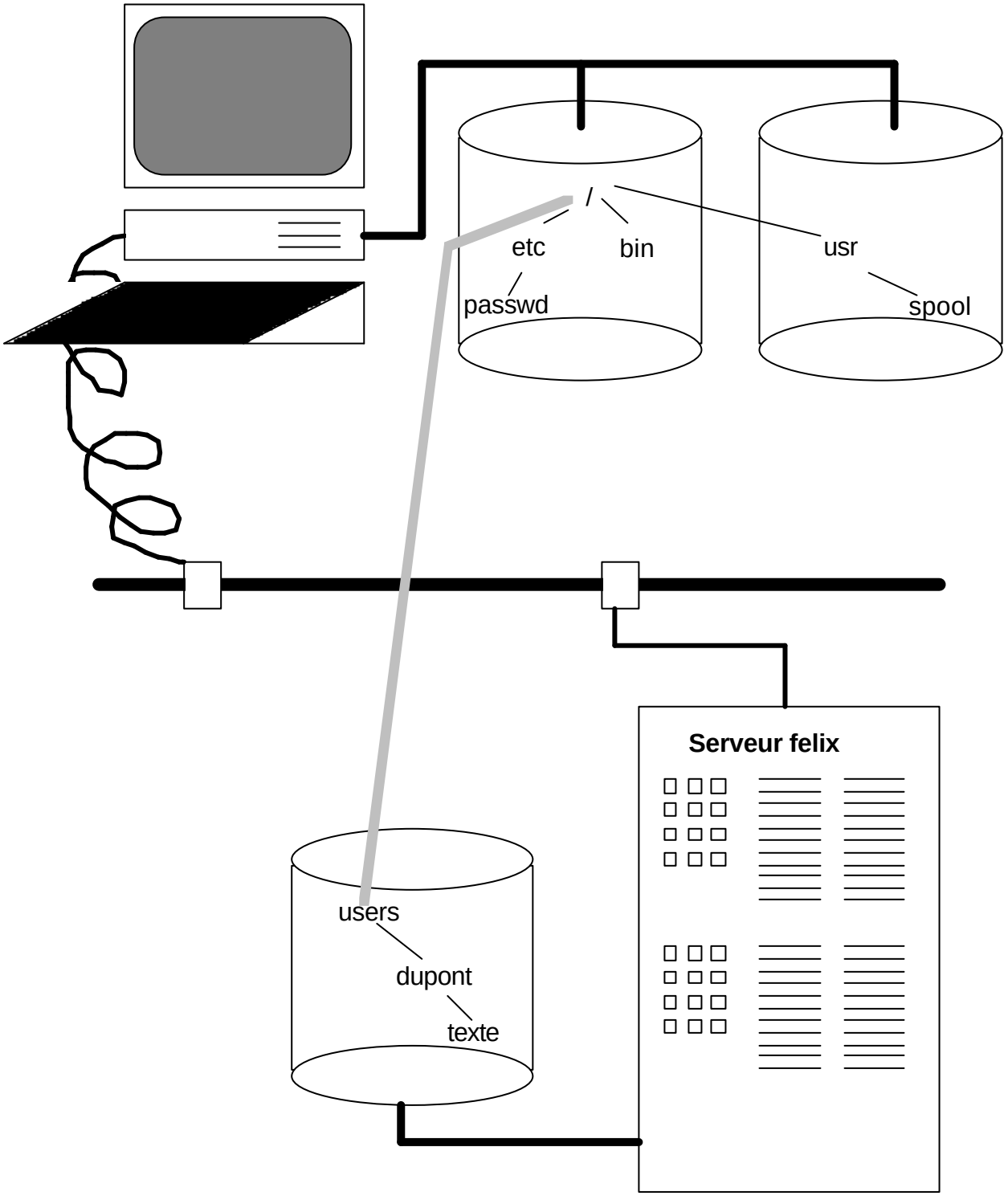
ADAPTABILITE :
Résistance à l'accroissement de la charge due aux utilisateurs et aux programmes qui augmentent.
Evaluer les facteurs d'échelles qui induisent la saturation.

ADAPTABILITE et TOLERANCE AUX PANNES sont liés :
une mauvaise réaction à la charge peut amener un processeur de serveur à avoir un comportement fautif (panne d'omission, panne temporelle)

NFS - Network File System

Objectifs d'NFS

Partage de fichiers :



Propriétés recherchées

- Partage de fichiers à travers un réseau de machines et de systèmes hétérogènes (VAX-VMS, PC-MSDOS, et de nombreux UNIX-NFS).

- Accès aux fichiers distants masqué aux utilisateurs et aux programmes

- Performances des accès voisines de celles d’un disque local

- Résistance aux pannes : serveur, client et réseau

- Extensibilité du système de fichiers simple

- Facilité d'administration (NIS, Network Information Service, ex YP)

Implantation d’NFS

Point de vue Réseau :

- Le protocole de Transport en **mode datagramme** UDP est utilisé

- Conçu sur le modèle Client/Serveur, à travers un **RPC de sémantique d'exécution au moins une fois**

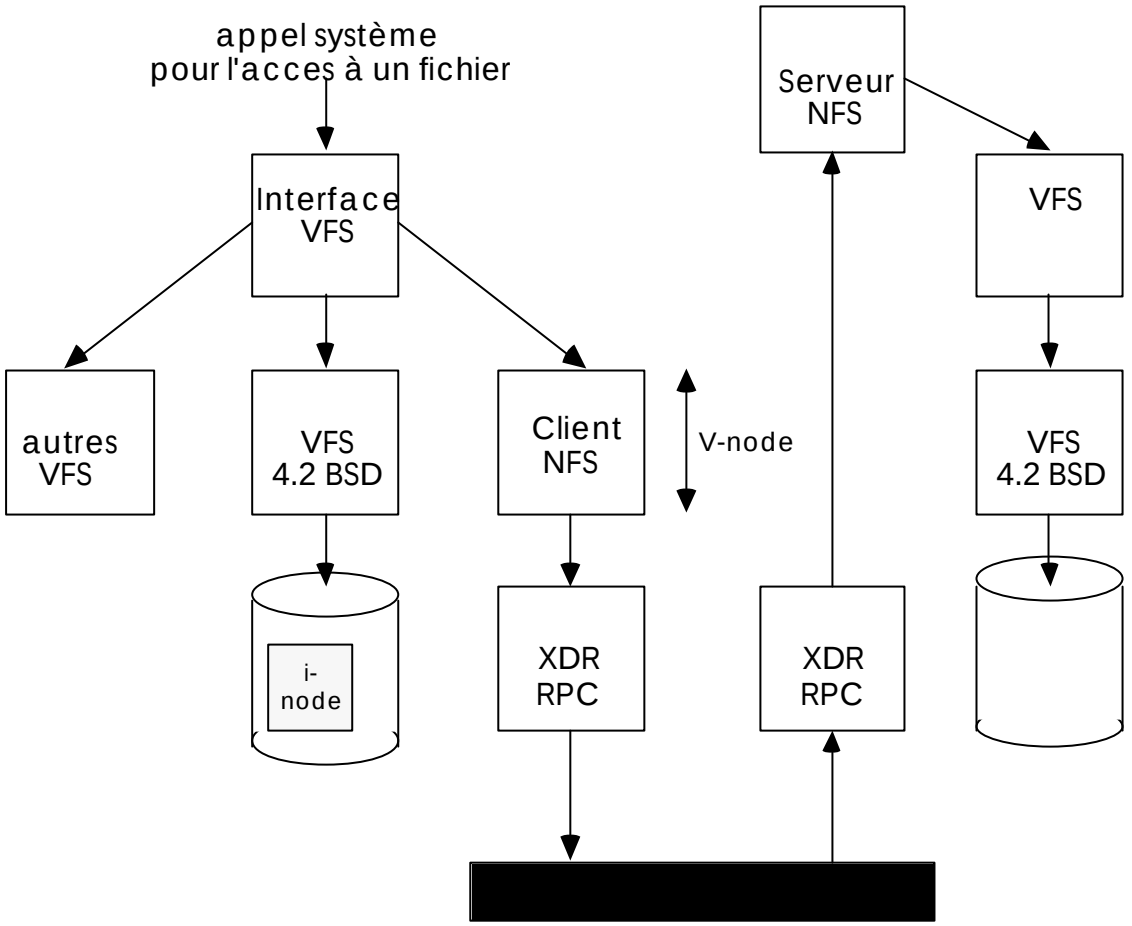
- Gestion de l'hétérogénéité par XDR

Point de vue Système :

- **Réécriture de l’interface d’E/S avec le noyau Unix** : Concept de **"Virtual File System VFS"**, et de **"virtual node V-node"**

- Protocole **NFS** et Serveur sans état

- Protocoles périphériques et Serveur avec état : **Mount**, Montage d'arborescences
  NIS, Gestion des paramètres
  Lock Manager, Gestion des verrous
  Network Status, Surveillance du réseau


appel système
pour l'acces à un fichier
Interface
VFS
autres
VFS
VFS
4.2 BSD
i-
node
Client
NFS
V-node
XDR
RPC
Serveur
NFS
XDR
RPC
VFS
VFS
4.2 BSD

VFS

VFS : Couche logique, interface qui masque l'accès à un sous-arbre local ou distant de l'arborescence des fichiers ou partition ("File System" au sens Unix)

- il est construit sur le concept de V-node
- il gère le montage et le démontage de partitions
- il permet l'accès aux fichiers lors de traversées de points d'attachement ou points de montage ("mount point")

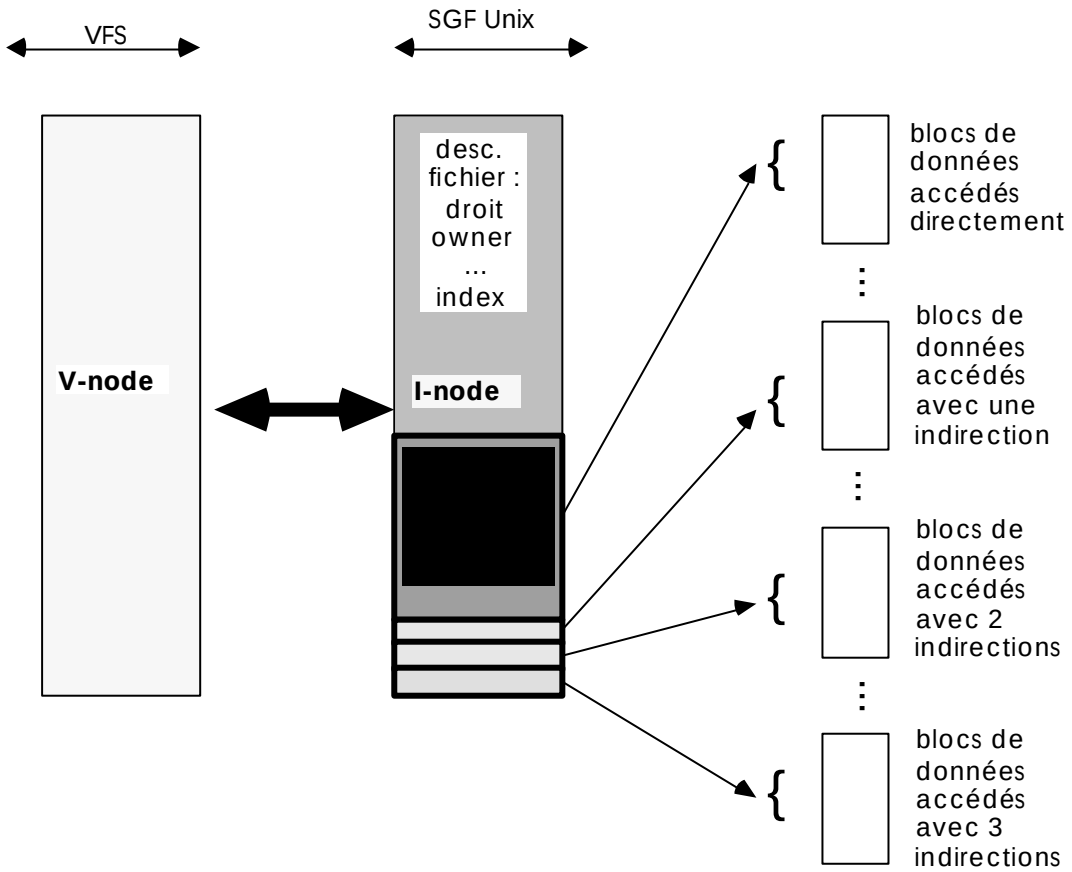
Notion de partition virtuelle

```
struct vfs {
        struct vfs      *vfs_next;              /* next vfs in vfs list */
        struct vfsops   *vfs_op;                /* operations on vfs */
        struct vnode    *vfs_vnodecovered;      /* vnode we mounted on */
        int             vfs_flag;               /* flags */
        int             vfs_bsize;              /* native block size */
        fsid_t          vfs_fsid;               /* file system id */
        u_short vfs_exroot;             /* exported fs uid 0 mapping */
        short           vfs_exflags;            /* exported fs flags */
        caddr_t         vfs_data;               /* private data */
};

/*¨vfs flags */

#define VFS_RDONLY      0x01            /* read only vfs */
#define VFS_MLOCK       0x02            /* lock vfs so that subtree is stable */
#define VFS_MWAIT 0x04          /* someone is waiting for lock */
#define VFS_NOSUID      0x08            /* someone is waiting for lock */
#define VFS_EXPORTED 0x10               /* file system is exported (NFS) */

struct vfsops {
        int     (*vfs_mount)();                 /* mount file system */
        int     (*vfs_unmount)();               /* unmount file system */
        int     (*vfs_root)();                  /* get root vnode */
        int     (*vfs_statfs)();        /* get fs statistics */
        int     (*vfs_sync)();          /* flush fs buffers */
        int     (*vfs_vget)();                  /* get vnode from fid */
};
```

V-Node

V-node : Couche logique, interface qui généralise la notion de descripteur de fichier (i-node Unix), et qui sépare les opérations d'accès à un fichier de leur implantation sur disque.

Notion de descripteur virtuel de fichier



```
struct vnode {
        u_short         v_flag;         /* vnode flags (see below)*/
        u_short         v_count;                /* reference count */
        u_short         v_shlockc;              /* count of shared locks */
        u_short         v_exlockc;              /* count of exclusive locks */
        struct vfs              *v_vfsmountedhere;    /* ptr to vfs mounted here */
        struct vnodeops         *v_op;          /* vnode operations */
        union {
                struct socket   *v_Socket;              /* unix ipc */
                struct stdata   *v_Stream;              /* stream */
        } v_s;
        struct vfs              *v_vfsp;                /* ptr to vfs we are in */
        enum vtype              v_type;         /* vnode type */
        dev_t                   v_rdev;                 /* device (VCHR, VBLK) */
        caddr_t                 v_data;         /* private data for fs */
};

/* vnode operations */

struct vnodeops {
        int (*vn_open)();       int (*vn_close)();      int (*vn_rdwr)();
        int (*vn_ioctl)();      int (*vn_select)();     int (*vn_getattr)();
        int (*vn_setattr)();    int (*vn_access)();     int (*vn_lookup)();
        int (*vn_create)();     int (*vn_remove)();     int (*vn_link)();
        int (*vn_rename)();     int (*vn_mkdir)();      int (*vn_rmdir)();
        in (*vn_readdir)();     int (*vn_symlink)();    int (*vn_readlink)();
        in (*vn_fsync)();       int (*vn_inactive)();   int (*vn_bmap)();
        in (*vn_strategy)();    int (*vn_bread)();      int (*vn_brelse)();
        in (*vn_lockctl)();     int (*vn_fid)();
};

/ * Vnode attributes.  A field value of -1 represents a field whose value is unavailable (getattr) or which is not
to be changed (setattr) */

struct vattr {
        enum vtype      va_type;        /* vnode type (for create) */
        u_short va_mode;        /* files access mode and type */
        short           va_uid; /* owner user id */
        short           va_gid; /* owner group id */
        long            va_fsid;        /* file system id (dev for now) */
        long            va_nodeid;      /* node id */
        short           va_nlink;       /* number of references to file */
        u_long  va_size;        /* file size in bytes (quad?) */
        long            va_blocksize;   /* blocksize preferred for i/o */
        struct timeval  va_atime;       /* time of last access */
        struct timeval  va_mtime;       /* time of last modification */
        struct timeval  va_ctime;       /* time file ``created */
        dev_t           va_rdev;        /* device the file represents */
        long            va_blocks;      /* kbytes of disk space held by file */
};
```

Le Montage de fichiers distants (1)

Rappel du Montage d'un fichier sur un disque local

Notion de file system :

- Vue "administration système" :
Partition logique du disque physique qui contient une sous-arborescence des fichiers du système
La partition système est privilégiée parmi les autres. Elle contient les fichiers vitaux du système d'exploitation (souvent la première partition d'un disque, par exemple sd0a)
Une autre partition a un rôle important, la partition qui est liée à la pagination et au swapping (souvent la deuxième partition sur plusieurs disques , par exemple sd0b et sd1b)

- Vue implantation :
Un file system est désigné par un n° de périphérique logique et contient :

Boot bloc

Des informations de gestion, super-bloc

Une suite d' i-node[3], descripteurs de fichiers ou de répertoires

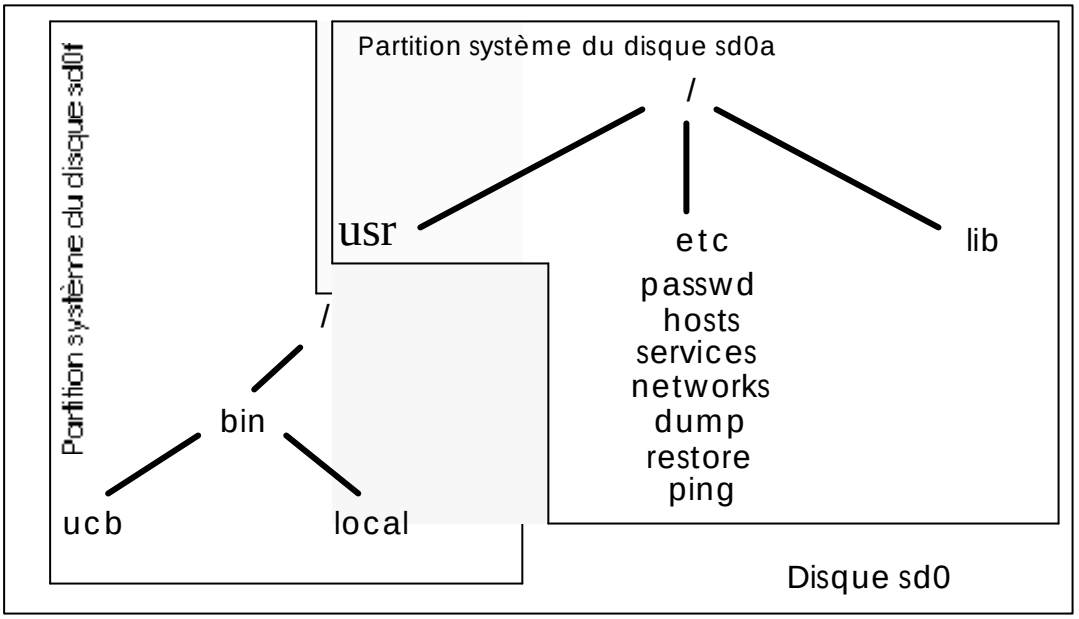
Une suite de blocs de données

Attachement d'une partition :
Association d'une partition logique d'un disque à un noeud de l'arborescence des fichiers d'une machine

Association d'un i-node (qui correspond au noeud d'attachement) de la *table des i-node résidente en mémoire* avec un super-bloc, cet i-node est marqué comme point de montage. Cette association ne se

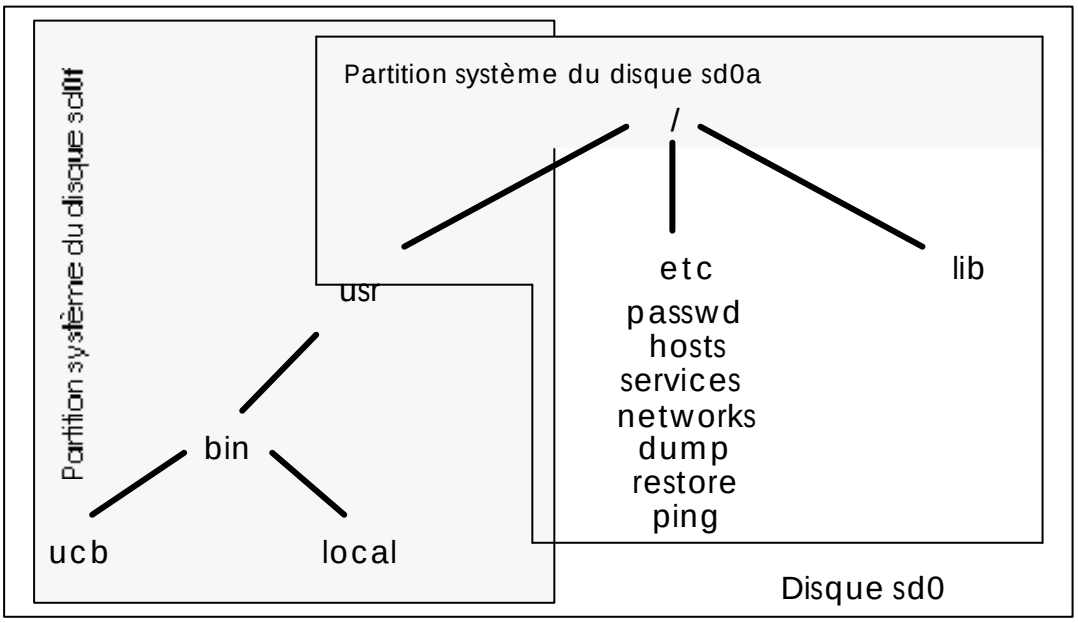
[3] Un i-node a un numéro unique dans un file system
L'inode # 2 correspond à la racine de la partie d'arborescence contenue dans la partition

fait qu’en mémoire centrale, jamais sur disque



Avant montage de la partition sd0f :

% cd /usr ; ls -l
%



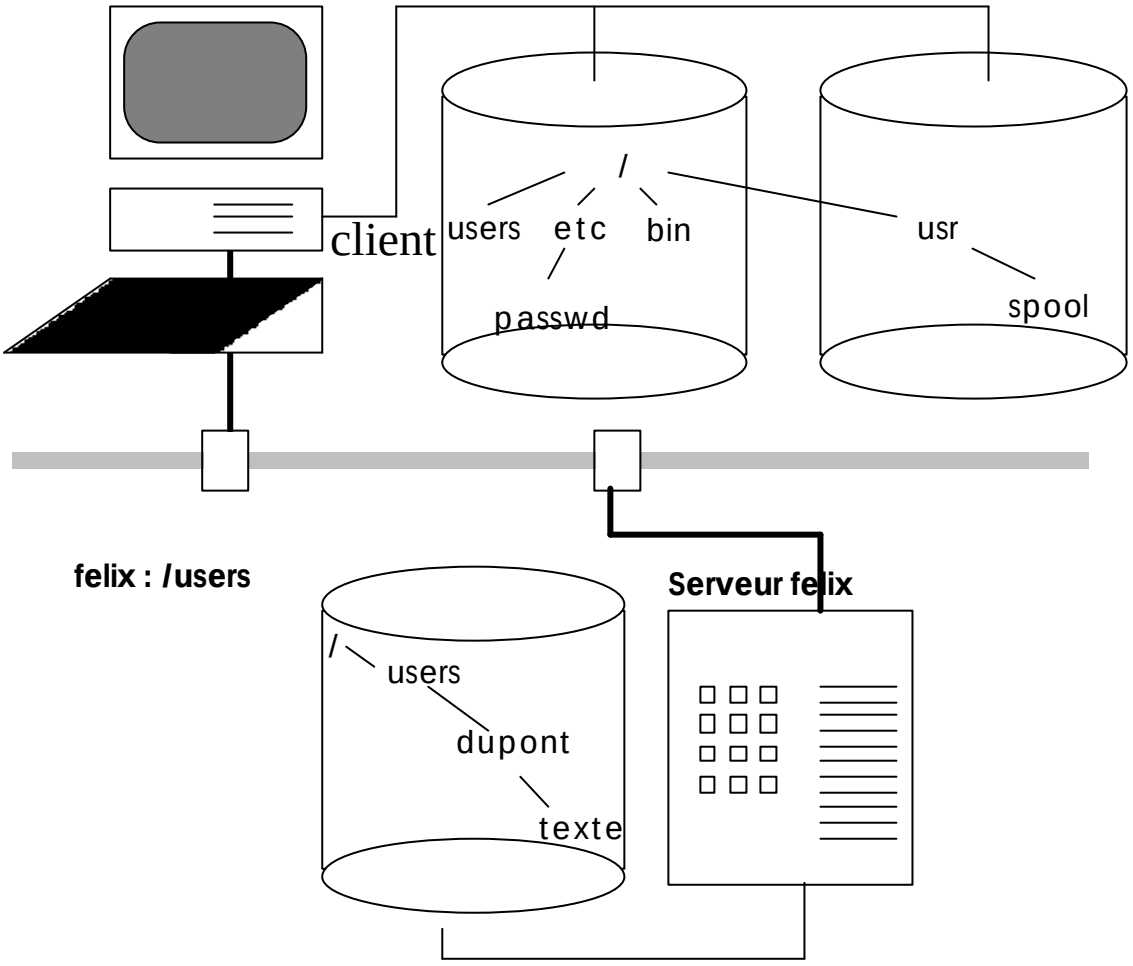
Après montage de la partition sd0f :

%mount /dev/rsd0f /usr
% cd /usr ; ls -l
drwxr-xr-x .... bin .....
%

Le Montage de fichiers distants (2)

Extension du mécanisme de montage à travers le réseau :

Vue de l'arborescence des fichiers avant le montage d'un répertoire distant sur le client :

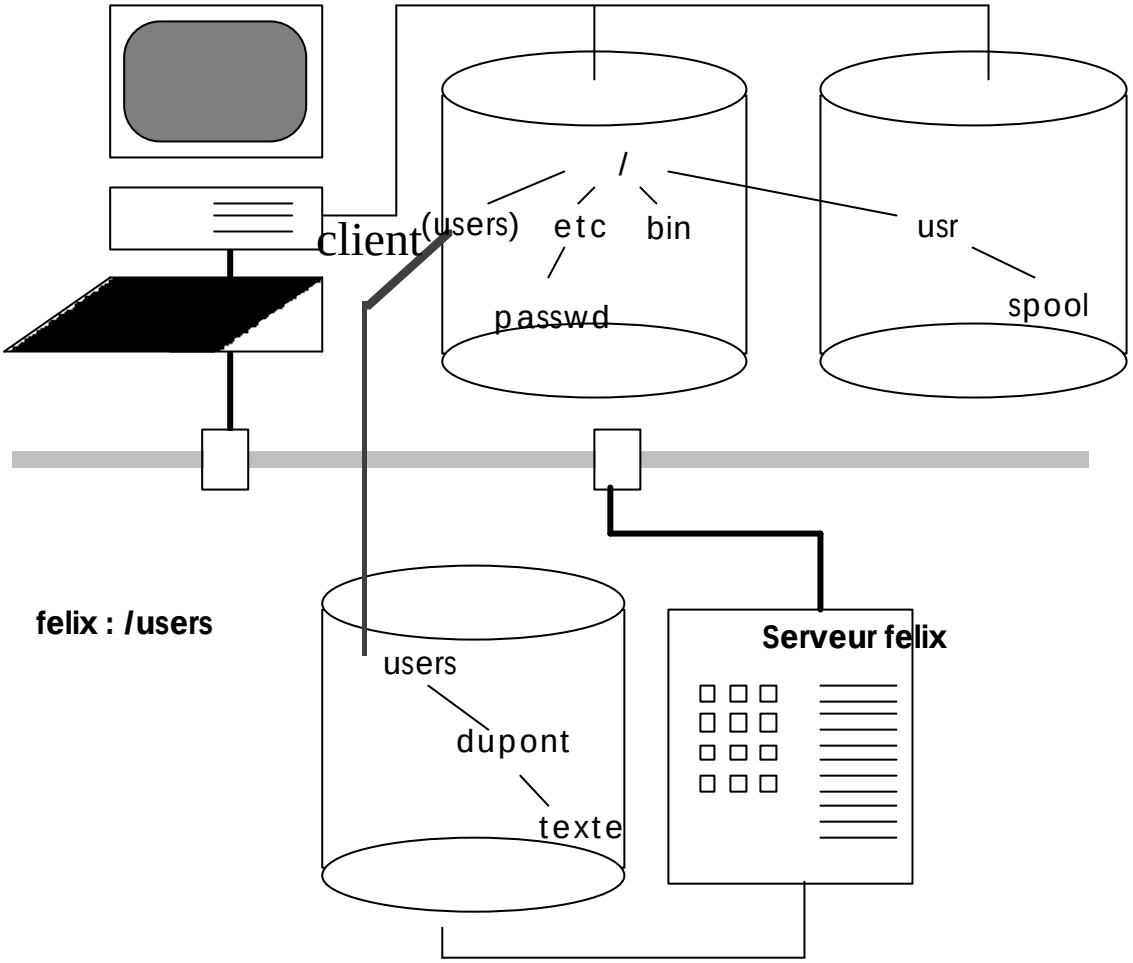


La liste des répertoires attachables exportés par le serveur est décrit dans le fichier /etc/exports : < nom de répertoire, liste d'accès des utilisateurs et machines autorisées >

```
/usr            -access=jo:dan:athena
/home/server    -access=iris
/users          -access=client
```

Remarque : Le nom d'un répertoire à attacher ne correspond pas nécessairement à une sous-arborescence complète et donc à une partition locale du serveur.

Vue de l’arborescence des fichiers après l'attachement d’un réperoire distant sur le client :commande mount felix:/users /users



Problème de désignation des fichiers :

Avec ce système d'attachement, les chemins d'accès aux fichiers peuvent différer d’un client à l’autre, les clients n'ont pas la même vue logique de l'arborescence complète, ceci amène à rechercher des noms uniques et globaux

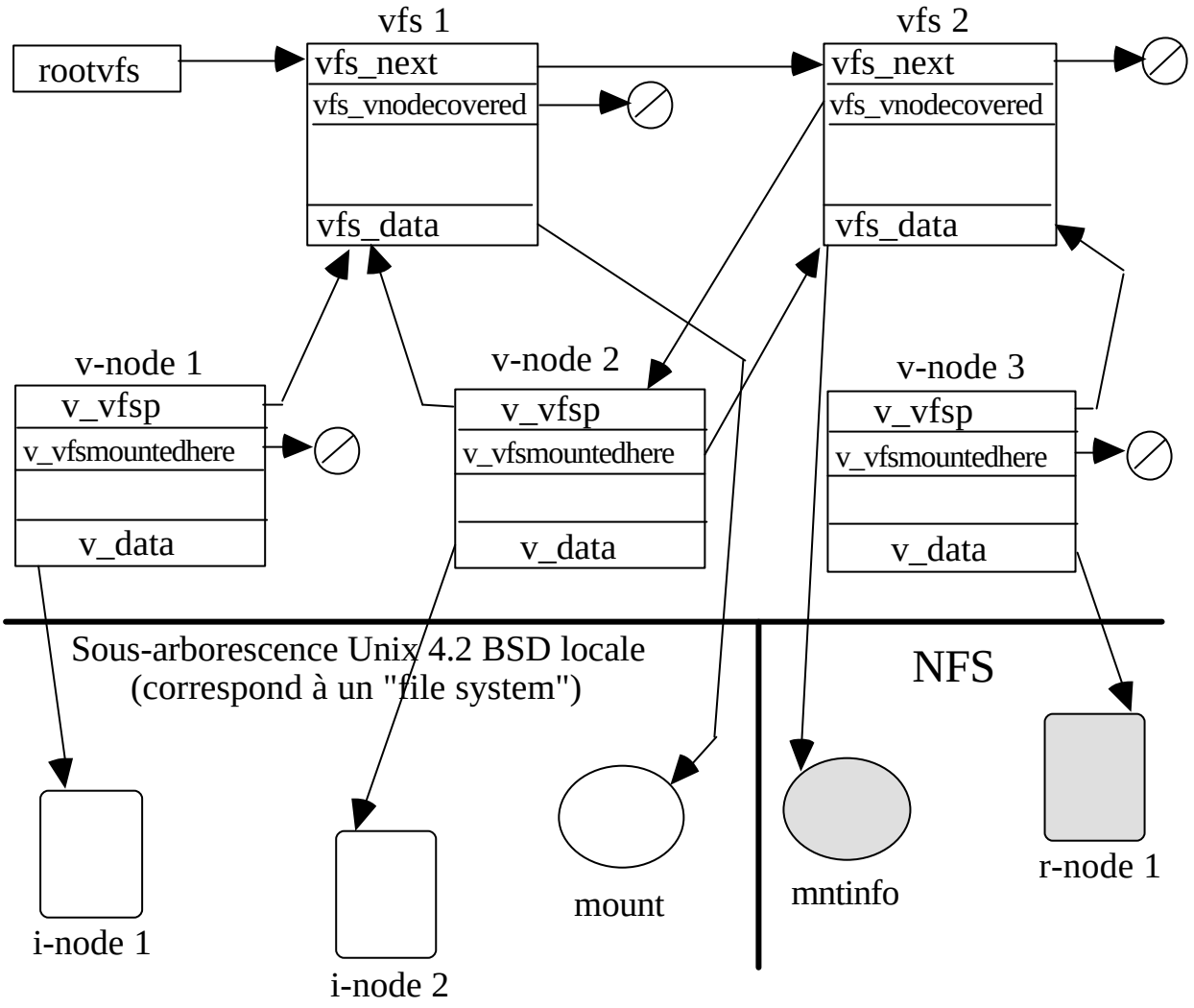
->

Les arborescences montées sont préfixées par un nom qui masque le serveur dans un répertoire dédié au montage de systèmes de fichiers distants.

=> **Noms administrativement globaux** dans NFS[4]

[4] Les noms administrativement globaux sont masqués à l’utilisateur par des **liens symboliques** qui rendent les noms des fichiers indépendants de leur localisation sur une machine du réseau.

Relation VFS-VNODE- SGF local



Poignée - File Handle

Identificateur de fichier sur le serveur:
- pour le client : 32 octets qu'il n'interprète pas (opaque XDR)
- pour le serveur : 32 octets dans lesquels il met les informations nécessaires à retrouver un fichier qu'il gère, ou le descripteur de celui-ci

Dans les premières versions de NFS, il était d'une longueur de 32 octets. Maintenant, il est de taille variable : <longueur,tableau>, dans NFS V3.

Pour un serveur Unix, le file handle est :

fs id	n° i-node	n° génération i-node	

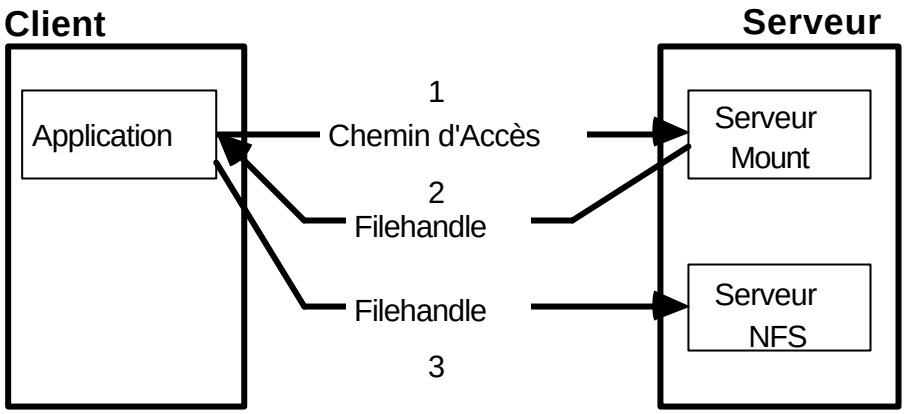
<n° de filesystem (8o), n° d'i-node (4o), n° de génération (4o), remplissage (18o)>

struct svcfh dans nfs.h

Le numéro de génération fonctionne comme un numéro d'époque. Si le fichier correspondant à un i-node a été détruit, puis qu'il a été réattribué lors de la création d'un fichier, le client risque de présenter un n° d'i-node obsolete et de recevoir des données du nouveau fichier. Le serveur détecte cette situation grace à ce numéro.

Protocole Mount (1)

Le Protocole MOUNT, c’est un peu plus :



Le protocole "mount" résoud la correspondance entre un nom de répertoire : chemin d'accès sous forme de chaine de caractère qui représente un point d'attachement, et sa localisation sur le serveur cible.

=> rend NFS indépendant des conventions de désignation d’un répertoire (point d'attachement) sur un serveur

1. Le client envoie le nom du répertoire à attacher au serveur MOUNT

2. Le serveur "mount" lui retourne la **"poignée"** pour pouvoir entrer dans le répertoire d'attachement après avoir vérifié les droits d’accès. C'est une première poignée, et le client la conserve soigneusement sinon, il ne pourra plus accéder au sous-arbre correspondant.

3. Le client, lors d'une opération d'accès à un fichier, envoie au le serveur NFS le file handle qui lui a été retourné pour parcourir le chemin d’accès au fichier qui passe par le point d'attachement.

Protocole Mount (2)

Le protocole mount est réalisé par un ensemble de procédures RPC Sun :

0	NULL	ne fait rien
1	MOUNT[5]	retourne le file handle qui correspond au point d'attachement d'un sous-arbre
2	READMOUNT	retourne la liste des sous-arbres attachés
3	UNMOUNT	enlève un fichier de la liste des sous arbres attachés
4	UNMOUNTALL	vide la liste des sous-arbres attachés
5	READEXPORT	retourne la liste des sous-arbres exportés

[5] Rend le premier file handle pour un répertoire distant à monter

Comment se passe le montage de serveurs en cascade sur NFS :

cas 1 :

client :	mount serveur1:/usr/local	/usr/local
serveur1 :	mount serveur2:/usr/local/work	/usr/local/work

cas 2 :

client :	mount serveur1:/usr/local	/usr/local
client :	mount serveur2:/usr/local/work	/usr/local/work

Donnez le résultat.

Le protocole NFS (1)

1. Le **serveur NFS est sans état**. Il ne maintient aucune information sur les fichiers qu'il gère pour le compte d'un client. C'est le client qui conserve toutes les informations qui permettent au serveur de retrouver le fichier, elles sont dans le **filehandle**.

L'open() sur un fichier ne laisse donc aucune trace sur le serveur!

2. Résistance aux pannes : Quand un serveur tombe en panne, les clients restent bloqués jusqu'à la remise en route du serveur (mécanisme de RPC avec temporisation 1100s entre deux tentatives et 4 tentatives avant un message d'erreur).

"Server not responding. Still trying."

Puis tout reprend comme si rien ne s'était passé ...

Le protocole NFS (2)

3. idempotence du service : La sémantique du RPC[6] choisie par Sun (au moins une fois) implique l'idempotence de toutes les opérations d'accès aux fichiers, par exemple l'écriture séquentielle doit être transformée en écriture en accès direct ...

Toutes les opérations ne sont pas idempotentes[7] !!!

4. Cumul possible du rôle de serveur et du rôle de client

[6] Les appels de procédures RPC sont tous synchrones. C'est à dire bloquants pour ceux qui l'effectue

[7] Opérations de destruction par exemple : rm, ...

Quelques procédures du protocole NFS

Le protocole NFS est lui aussi réalisé à l'aide de procédure de type RPC Sun :

1 GETATTR (fhandle,attr)
rend les attributs d'un fichier

4 LOOKUP(dir,name,file,attr)
rend un fhandle file et les attributs attr du fichier name dans le répertoire de filehandle dir

6 READFILE(file,offset,count,attr,data)
rend count octets de données du fichier de filehandle file dans les données opaques data, à partir de la position offset dans ce fichier, les attributs attr du fichier sont retournés

8 WRITE(file,offset,data)
écrit dans le fichier de file handle file à partir de la position offset, les données opaques data transmises, l'opération write est atomique

9 CREATE(dir,name,satt,file,attr)
crée un fichier de nom name dans le répertoire de file handle dir avec les attributs satt, le file handle file, et les attributs du fichier créé sont retournés

parmi les paramètres, on peut trouver un "magic cookie" : indicateur sur de qui a déjà été lu dans un répertoire par le client, seul le serveur est capable d'interpréter cette information.

Précautions :

Les identificateurs d’utilisateurs et de groupes d’utilisateurs entre machines clientes et serveurs doivent être identiques sinon, il risque d’y avoir des problèmes de propriétés et de droits d’accès !!! L’utilisation de NIS est recommandée pour gérer cette homogénéité !!!

Les accès concurrents ne sont pas supportés par NFS, c’est le dernier qui écrit qui a gagné !!!

Synchronization des horloges sur les différentes machines pour que la commande make fonctionne correctement.

Mecanisme de cache - biod (1)

Les clients NFS peuvent utiliser une technique de cache pour améliorer les performances

-> Le **cache réside en mémoire centrale du client**, le disque local s'il existe n'est pas impliqué dans la gestion de cache

-> Les informations mises dans le cache sont :

- page contenu de fichier
- page contenu de répertoire
- descripteur de fichier

Mecanisme de cache - biod (2)

Problème de validation du cache en mémoire centrale :

Les pages sont estampillées avec leur date de dernière modification. Il y a validation par comparaison de cette date avec celle qui réside avec le fichier sur le serveur, s'il y a une différence, la page doit être rechargée.

Quand cette comparaison peut-elle s'effectuer ?

A l'initiative du client

Pour un fichier :
. A chaque ouverture de fichier
. A chaque défaut de page dans le cache
Pour un répertoire :
. A chaque ouverture de fichier, il y a contrôle de validité pour le répertoire qui le contient
. Les répertoires sont conservés dans le cache en lecture seulement, les modifications sont opérées sur le serveur directement

Validation garantie périodiquement sinon :

toutes les 3s pour un fichier
toutes les 30s pour un répertoire

Mecanisme de cache - biod (3)

Mise à jour du serveur :

Une page modifiée est marquée "sale" et devra être transférée vers le serveur.

Cette activité est réalisée de façon asynchrone par le noyau ... quand il a le temps !!!

Garantie :
Toutes les pages modifées seront recopiées sur le serveur au plus tard avant la fermeture du fichier (technique "write-on-close")

La gestion de la concurrence est sous la responsabilité de l'utilisateur ... à défaut, c'est le dernier qui écrit qui a gagné ... attention aux risques d'incohérence.

Transfert serveur-client :
. par blocs de 8 Ko
. tout le fichier pour les petits fichiers pas plus grands que 8 Ko

Réplication - "Automounter"

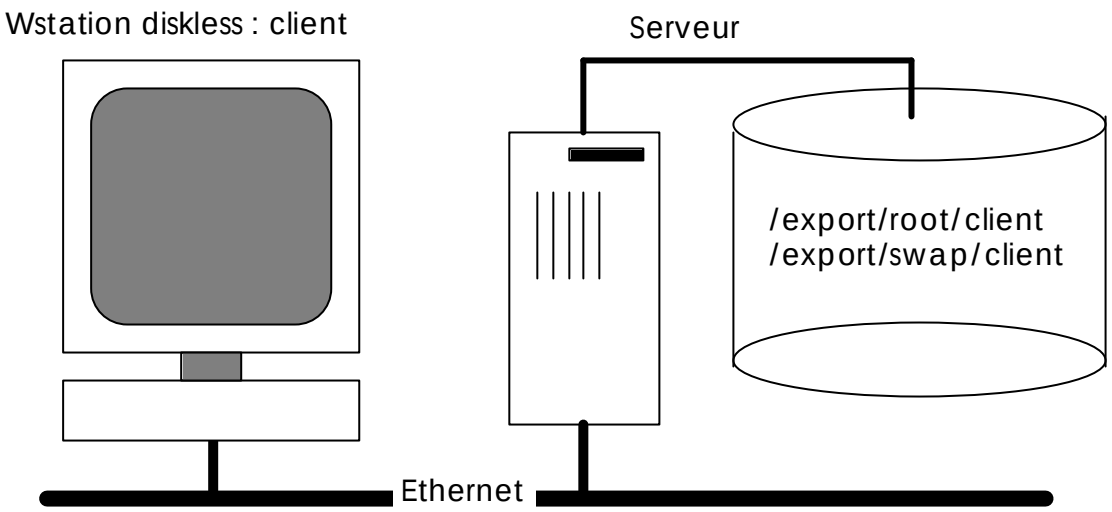
Pour un client, ce mécanisme permet d'avoir plusieurs serveurs pour réaliser un attachement de sous-arbre. **C'est une forme de réplication de serveurs**.

C'est le premier serveur qui répond au client demandeur qui est utilisé.

Pratique pour les fichiers exécutables et les fichiers de données accédés en lecture => TOLÉRANCE AUX PANNES pour ces fichiers.

Pour les fichiers modifiés, la propagation des écritures d'un serveur à un autre doit être faite "à la main".

Les Stations sans disque



Les stations sans disque utilisent NFS pour leur partition système (/vmunix, les fichiers de configuration, et les programmes du système fondamentaux), pour l'accès aux fichiers généraux (bibliothèques, compilateurs, commandes, espace utilisateur), et aussi pour leur partition de swap.

Rien ne s'oppose à ce que l'espace swap soit vu comme un large fichier en accès direct sur un disque distant.

Les Yellow Pages - NIS (1)

Principes :

- Les “Yellow Pages” forment une base de données dupliquée qui fait correspondre une “map” à un fichier[8] :

Fichier des utilisateurs	/etc/passwd
Fichier des sites connectés	/etc/hosts
Fichier des services	/etc/services
.....	

Elles permettent d’avoir une définition unique de certains paramètres de configuration du réseau de machines. Un réseau de machines qui utilisent la même copie de la base YP, définit un domaine d’administration YP.

- Deux types de machines :
Les serveurs qui possèdent une copie de la base de données (ypserv) :

. Le serveur maître sur lesquelles sont effectuées les mises à jour et qui effectue la propagation de ces mises à jour

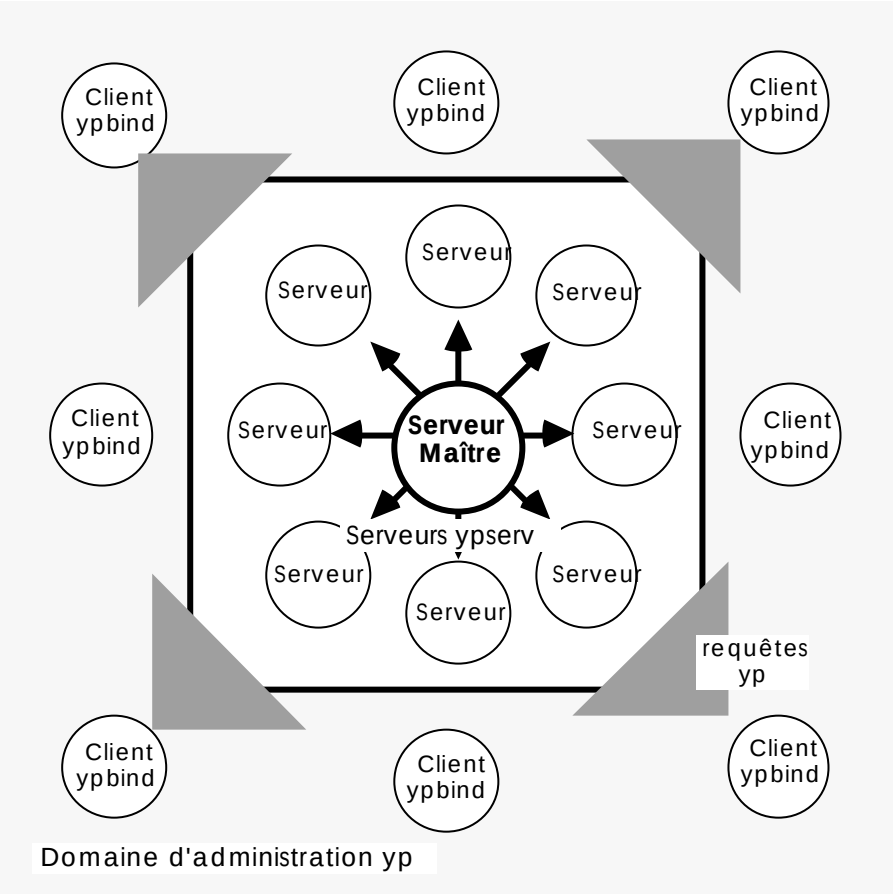
. Les serveurs esclaves => résistance aux pannes, récupération des mises à jours

Les clients qui interrogent un serveur pour accéder aux informations de la base de données (ypbind), pour connaître le serveur associé, utiliser la commande ypwhich.

- Les YP sont construites au-dessus de RPC/XDR

[8] Il est possible, pour certaines “map”, de consulter le fichier local avant de faire une requête à un serveur YP. C’est le cas, pour le fichier des utilisateurs.

Les Yellow Pages - NIS (2)



Les Yellow Pages - NIS (3)

- Cohérence faible des données entre serveur maître et serveurs esclaves, la mise à jour d'un mot de passe par exemple n'est prise en compte sur tous les serveurs qu'au moment de la prochaine mise à jour des "map" sur les serveurs esclaves.

- Lors de la panne d'un serveur , un client joint un nouveau serveur par une requête de diffusion, le premier qui a répondu devient son nouveau serveur.

Quand le serveur maître tombe en panne, plus aucune mise à jour n'est possible....

- Fonctionne bien entre machines homogènes, mal entre machines hétérogènes :

"tables" différentes, protocoles différents ???

Reste à voir en périphérie d'NFS:
. le Lock Manager => accès concurrent aux fichiers
. le Network Status Monitor => gestion de l'état des différentes machines du réseau[9]

[9] Serveur avec état qui complète tout ce que ne peut pas maintenir NFS pour rester sans états

Concepts Généraux

. Désignation et Transparence

. Sémantique du partage de fichiers

. Méthode d'accès à distance

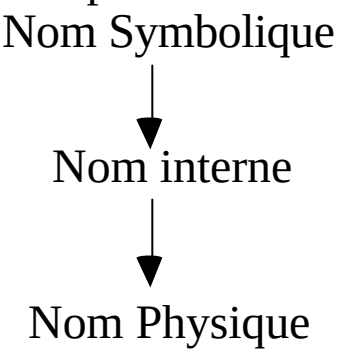
. Tolérance aux pannes

. Extensibilité

Désignation

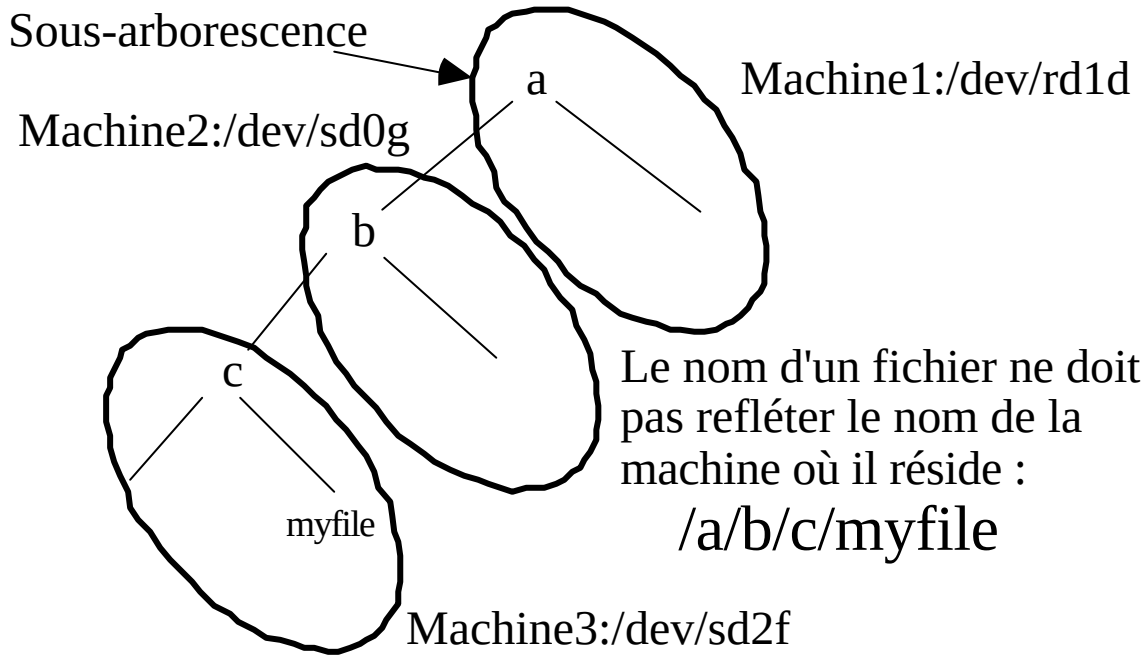
Comment passe-t-on d'un **nom symbolique** qui repère un fichier dans une **organisation logique** à l'**objet physique** qui le contient ?

La désignation gère la correspondance :



Modèle Unix est souvent retenu comme modèle de désignation symbolique:

ensemble de sous-arborescences connectées entre elles



Le nom d'un fichier ne doit pas refléter le nom de la machine où il réside :

/a/b/c/myfile

Localisation - transparence

La transparence, c'est pouvoir masquer l'endroit (la machine) où est archivé un fichier et le moyen d'y accéder (protocole d'accès).

Les fichiers distants sont désignés comme s'ils étaient locaux.

Cette propriété est particulièrement importante quand le SGFR s'appuie sur des copies multiples pour la TOLERANCE AUX PANNES.

Pour réaliser la transparence, on conserve dans une table une correspondance entre une sous-arborescence et la machine où elle réside. Cette correspondance est invisible aux utilisateurs humains comme programmes.

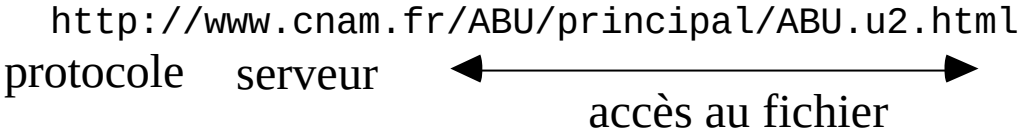
Tout le monde fournit cette propriété sauf Newcastle Connexion qui introduit une syntaxe d'exception :
- super-racine ou racine réseau : //
- le nom de serveur préfixe toute arborescence locale d'une machine

Problème : pas transparent, perte de portabilité
changement de serveur => changement de nom dans les programmes

Cette propriété souvent recherchée semble abandonnée dans certaines circonstances.

Le Web est un exemple de cette démarche à contre courant. C'est un environnement qui permet d'accéder à des informations maintenues sur les sites Web de l'Internet.

Les noms des fichiers accessibles par ce moyen sont appelés URL[10] :



Les facteurs qui ont présidés ce choix semblent être :

- le très grand nombre de serveurs implique une désignation uniforme même au prix d'une syntaxe d'exception qui provoque la perte de la transparence

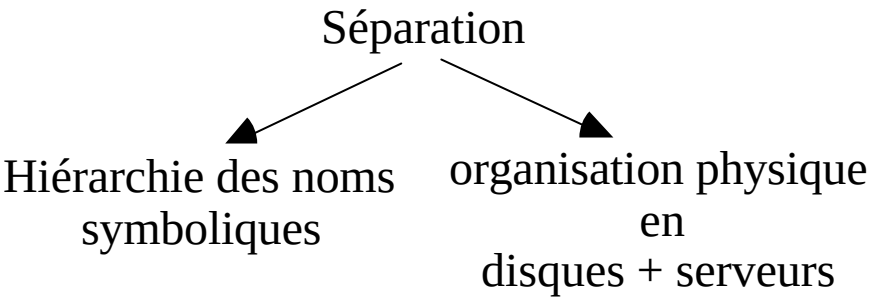
- la nécessité d'une uniformisation de l'accès aux données malgré la diversité des protocoles de communication pour retirer ces données

10 Uniform Resource Locator

Localisation - indépendance

L'indépendance est une propriété beaucoup plus forte que la transparence.

Indépendance par rappport à la localisation :



Elle permet la migration des fichiers car leur nom n'a pas besoin de changer.

Propriété utile pour faire de la répartition de charge en faisant migrer une sous-arborescence vers un serveur moins chargé.

Propriétés visées :

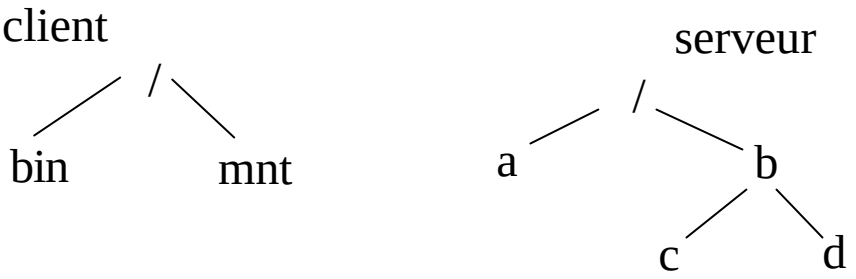
ADAPTABILITE et RECONFIGURABILITE

On trouve cette possibilité plutôt dans les systèmes répartis.

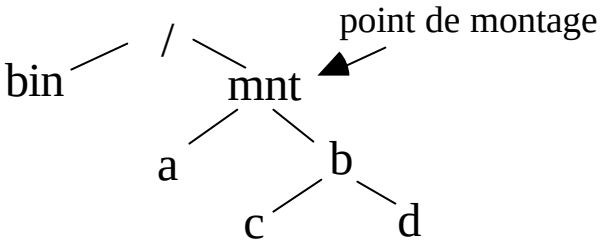
Techniques de Désignation (1)

1. **Mise en commun d'Arborescence** (Newcastle Connexion) ne fournit pas la transparence

2. **Attachement d'arborescences**



vue de l'arborescence après attachement



Transparence sur chaque site, mais il y a autant de vues de l'arborescence des fichiers qu'il y a de machines.

Les noms ne sont pas globaux.

En respectant un certain nombres de règles pour les opérations d'attachement (opérations effectuées par un administrateur d'habitude) on obtient des **noms administrativement globaux**.

Techniques de Désignation (2)

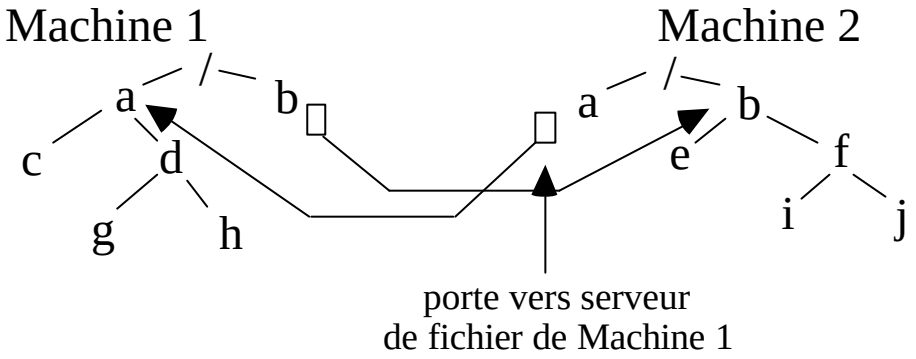
3. **Arborescence virtuellement centralisée** ou **arborescence de désignation unique**

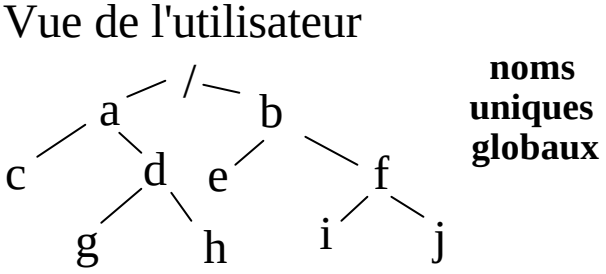
La vue logique des fichiers est la même sur toutes les machines. Les noms sont globaux. Les fichiers sont physiquement répartis sur les serveurs.

Cette solution s'appuie sur des noms internes uniques dans le temps et dans l'espace (UID).

UID Apollo :

site de création (20b)	date de création (36b)	type (8b)



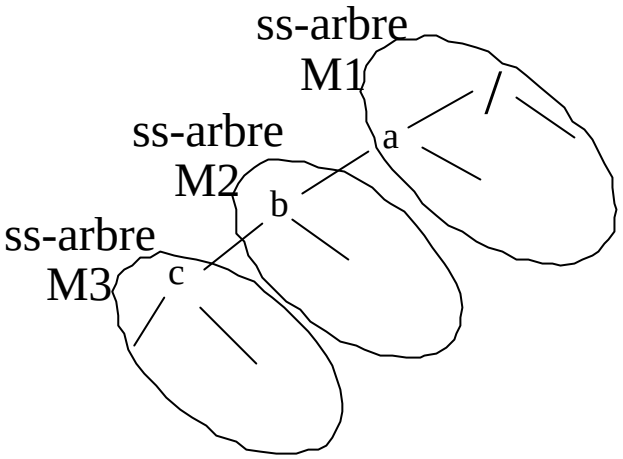


Solution qui n'est pas toujours commode pourtout ce qui est privé à une machine comme par exemples les executables.

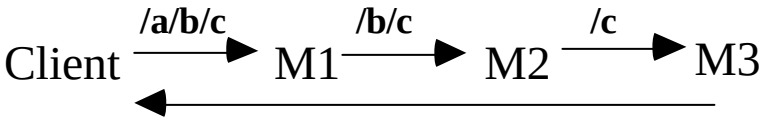
Techniques de Résolutions des noms (1)

1. **Transformation du chemin d'accès**

Principe : Chaque machine résoud le morceau de chemin qui la traverse.

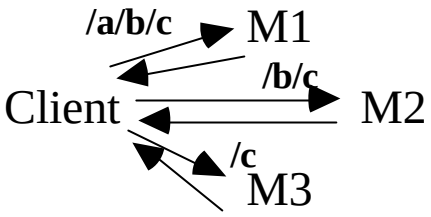


Un client veut accéder au fichier /a/b/c :
a. Client -> M1 qui reçoit tout le chemin
b. sur M1 :
- **/ évalué sur M1** : donne le **nom interne "racine"**, on va sur le disque
- **dans le répertoire racine** on trouve le **nom interne de "a"**, on va sur le disque
- **dans le répertoire "a"**, on trouve une indication telle **"b dans ss-arbre sur M2"**, on passe **/b/c à M2**
c. sur M2 : le reste du chemin **/b/c/myfile est évalué** de la même façon, **/c est passé à M3**
d. sur M3 la **fin du nom est résolu** et retour du résultat au client



Autre variante de la transformation de chemin d'accès :

. Résolution par chaque serveur et retour au client après chaque étape (image du polling).



L'effort de résolution est supporté surtout par le client.

Solution adoptée par NFS.

Techniques de Résolutions des noms (2)

2. **Méthode des identificateurs structurés**

Pour réaliser la correspondance entre le nom d'un fichier et sa localisation on utilise un identificateur structuré, celui-ci identifie une sous-arborescence qui contient le fichier cible.

identificateur structuré (is):

< identif de ss-arbre ; identif de fichier dans le ss-arbre>

Une table contient sur chaque site la correspondance :

nom symbolique <-> is

La table peut être gérée de différentes façons :
-> remplissage suivant le principe de résolution par transformation de chemin d'accès
-> gérée sur chaque site à l'image d'un cache ou accédée sur un serveur de nom

Exemple :

/a/b/c	<ss-arbre 3; 11>

ss-arbre 3	M3

Le nom est indépendant de la localisation, quand le fichier migre il suffit de mettre à jour la table.

Techniques de Résolutions des noms (3)

3. **Méthode des caches de suggestion ("hints")**

. intervient pour la localisation
. améliore la performance si le cache contient une information non périmée (valide)
. si l'information est invalide, on peut utiliser les techniques précédentes pour la recharger.

Solutions utilisées lors d'une résolution de chemin d'accès :

. accès cache de noms côté client sinon requête au serveur de fichiers
. accès cache sinon diffusion de la demande
. accès cache sinon emploie d'une heuristique
. accès cache sinon interrigation d'un serveur de noms

Techniques de Résolutions des noms (4)

4. **Points d'attachement**

Un point d'attachement est l'association d'une sous-arborescence à une feuille d'une autre sous arborescence.

Une table mémorise l'association

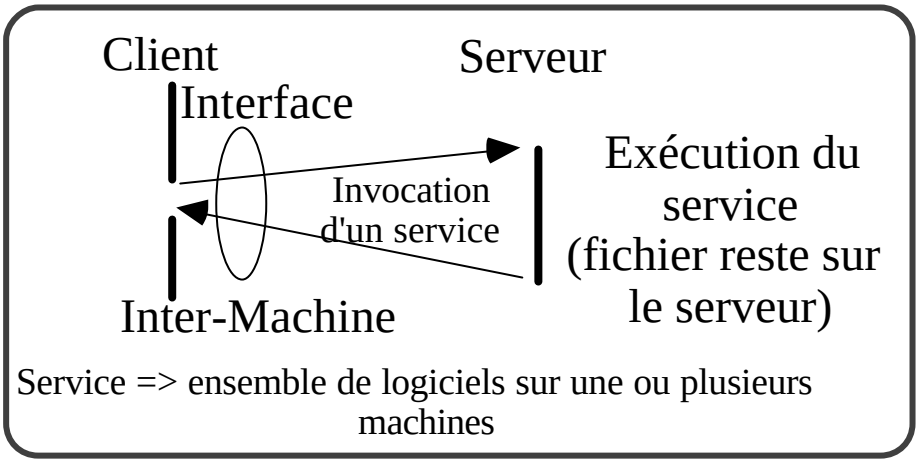
<feuille ; sous-arbre>

On parcours la table à chaque fois qu'un chemin d'accès traverse un point d'attachement.

Méthode d'accès aux fichiers distants (1)

Modèle Client-Serveur +
Interface qui masque l'implantation

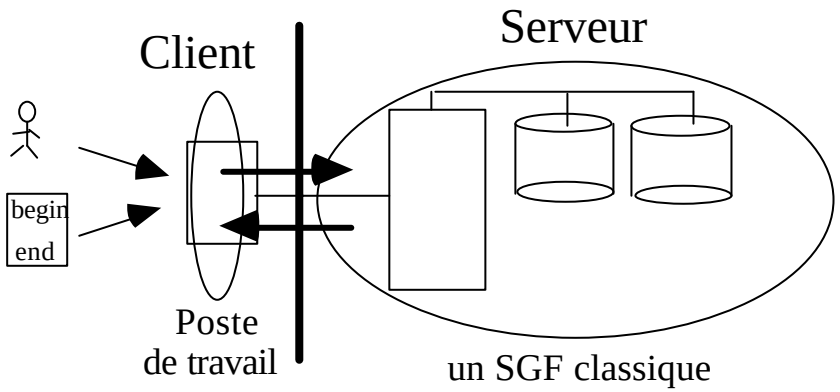
Accès à distance :



-> charge du serveur
-> charge du réseau

Importance de la sémantique d'exécution des demandes de service

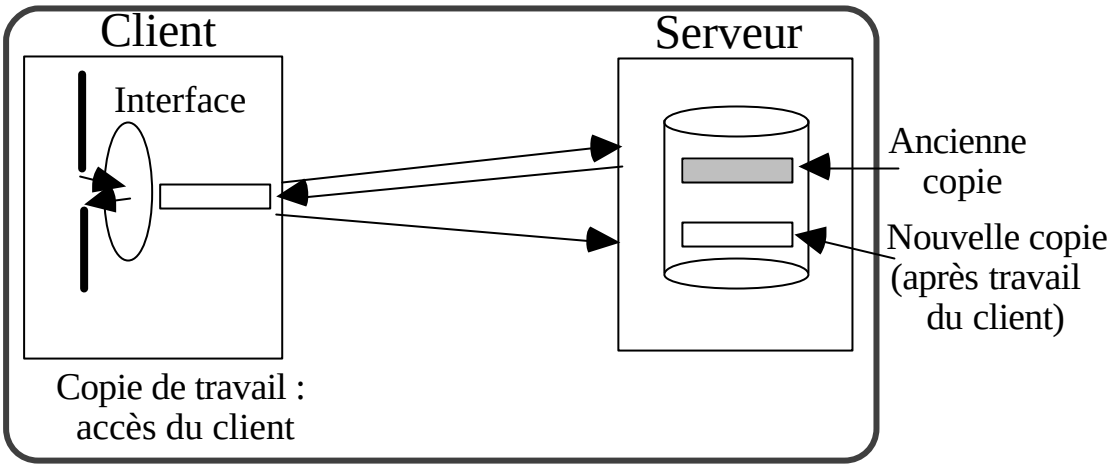
Application aux SGFR :



Interface :ensemble d'opérations sur les fichiers distants
créer, détruire, lire, écrire, changer les droits, ...

Méthode d'accès aux fichiers distants (2)

Téléchargement :



Importance de la politique de rafraichissement du serveur.

Le téléchargement d'un fichier peut être partiel, dans ce cas, c'est une **technique de cache** :
-> cache qui conserve une copie des données
-> transfert d'informations d'un fichier par groupe de blocs de données (repose sur le principe de localité des références - technique du "read ahead")
-> vidage du cache : LRU (adapté), FIFO (facile à implanter)
-> fixer la copie de référence, en général celle du serveur, il faut gérer la cohérence (toutes les copies sont identiques)

La sémantique du partage de fichiers a une importance sur la méthode d'accès.

Bibliographie

Nussbaumer 1991. "Téléinformatique IV : Messagerie électronique X400, Messagerie Industrielle MMS, Transfert de fichiers FTAM". Presses Polytechniques Romandes.

Tanenbaum 1992. "Modern Operating Systems". Prentice Hall.

Stern 1992. "Managing NFS and NIS". O'Reilly& Associate.

Levy, Silberschatz 1990. "Distributed File Systems : Concepts and Exemples". ACM computing Surveys. V22. N4.

Niveau Application

L'administration de systèmes et de réseaux

G Florin

Plan du chapitre administration

1 L'administration de systèmes et de réseaux: notions générales

1.1 Situation du problème.

1.2 Détails des enjeux et objectifs de l'administration.

1.3 Les niveaux politiques d'administration des systèmes et réseaux.

1.4 Les principales fonctions de la gestion des systèmes et réseaux.

2 Introduction à la gestion de réseaux OSI

2.1 Structure générale de la gestion OSI.

2.2 L'organisation d'une architecture de gestion OSI.

2.3 Les informations de gestion.

2.4 Les fonctions de communication CMIS/CMIP.

3 Introduction à la gestion de réseaux INTERNET SNMP

3.1 Introduction à la gestion SNMP.

3.2 La structure des informations de gestion.

3.3 le protocole de communication.

1

L'ADMINISTRATION DE SYSTÈMES ET DE RÉSEAUX: NOTIONS GÉNÉRALES

1.1 Situation du problème

Une définition

L'administration de réseaux c'est l'ensemble des processus de contrôle des matériels et des logiciels dans l'objectif d'en maximiser l'efficacité et la productivité.

Obtenir un service attendu

- **Maîtriser la mise en place**

Ex Installer un nouveau logiciel, un appareil.

- **Vérifier le bon fonctionnement d' infrastructures complexes.**

Ex Quelle est la charge supportée selon les heures par un équipement?

Améliorer l'efficacité de l'ensemble

- **Service rendu qualitativement**

Ex Rendre accessible un équipement dans des conditions de sécurité correctes.

- **Amélioration de la charge supportée par de meilleurs réglages.**

Ex Modifier la taille mémoire d'un logiciel.

Niveaux d'intervention "Network and system management"

Résoudre les problèmes de configuration, performance, facturation, anomalie, sécurité.

Niveau système

Administration classique:

Gestion des utilisateurs,

Gestion des équipements et systèmes installés sur chaque machine.

Niveau réseau

Gestion des matériels et logiciels réseaux.

Les deux problèmes sont de plus en plus difficilement distinguables.

Terminologie: administrer vs gérer

Essai de différenciation de deux significations administrer vs gérer

Administrer un système c'est définir une stratégie puis sélectionner un ensemble de critères permettant de valider des actions de gestion (politique de suivi des systèmes et de l'ensemble des moyens à mettre en oeuvre).

Aspects plutôt politiques

Gérer concerne plutôt les objets, les services et les protocoles mis en oeuvre au jour le jour pour le suivi des matériels et des logiciels.

Aspects plutôt opérationnels.

Les difficultés de l'administration

. **Existence d'un mouvement profond vers la répartition des systèmes**.

"Downsizing", "client-serveur" ...mettant en jeu des types très variés de systèmes informatiques:

Postes clients (PC windows, Mac)

Postes serveurs (Unix, Windows NT)

. **Contexte souvent très hétérogène**

Coexistence dans une même entreprise de solutions réseaux très diverses IBM/SNA, TCP/IP, Novell netware SPX/IPX, DECNET, Lan manager... (banques, multinationales mais aussi entreprises de taille moyenne).

. **Accroissement de la diversité et de la criticité des missions** confiées au système informatique:

Informatique classique, Communications interpersonnelles, Bureautique, Multimédia.

. Construction souvent non maîtrisée des architectures

Informatisation service par service.

Interconnexion des îlots au moyen de répéteurs, ponts, routeurs, commutateurs, passerelles.

=> Autant de problèmes potentiels de goulets d'étranglement, de sécurité

. **La facilité d'administrer n'est pas souvent l'un des critères principaux** dans la décision informatique.

. **Les îlots d'informatisation" sont vendus avec des moyens sommaires d'administration**.

Agents de gestion fournissant quelques paramètres spécifiques pour chaque équipement.

L'administration globale du réseau d'entreprise est difficile à organiser.

Problème essentiel de la gestion des réseaux: automatiser

. **Méthodes manuelles de supervision**

Gestion des configurations par
procédures manuelles,
Gestion des incidents par déplacement
utilisation du téléphone ...

Techniques possibles pour administrer un domaine **limité**, assez **homogène**, par **une seule équipe**.

.**Solutions automatisées** d'administration:

Ex Détection des pannes, téléchargement

Permettant de faire face à l'explosion exponentielle de la complexité des architectures.

Problème de l'administration des grands domaines

Composés de nombreux sous-domaines,
Administrés par plusieurs équipes,
Dans un environnement international,
=> Très nombreuses difficultés

Souvent dans un contexte de strict maintien ou de baisse des effectifs.

1.2 Détails des enjeux et objectifs de l'administration

Deux niveaux de perception:

- L'administration de systèmes/réseaux est un service pour l'ensemble de l'entreprise afin de faciliter son fonctionnement
- L'administration est un problème en soi avec ses techniques propres.

Enjeux architecturaux

. Gérer l'hétérogénéité des systèmes.
. Maîtriser des environnements centralisés aussi bien que distribués.
. Permettre une vision globale du réseau.

Enjeux organisationnels

. Garantir la flexibilité du système.
. Garantir la cohérence du système.
. Mise en oeuvre d'architectures reconfigurables en fonction des besoins organisationnels.
. Mettre en oeuvre des structures évolutives en fonction des besoins (modification, ajout de service sans remettre en cause l'ensemble).

Enjeux humains

. Améliorer la productivité du système.
. Améliorer la productivité des équipes (utiliser des équipes réduites d'administration)
. Capitaliser et transmettre les connaissances architecturales systèmes/réseaux.

Enjeux techniques

. Garantir la disponibilité (la pérennité): rendre le système de nouveau opérationnel après un incident.
. Garantir la qualité du service (sécurité, performances, ...).

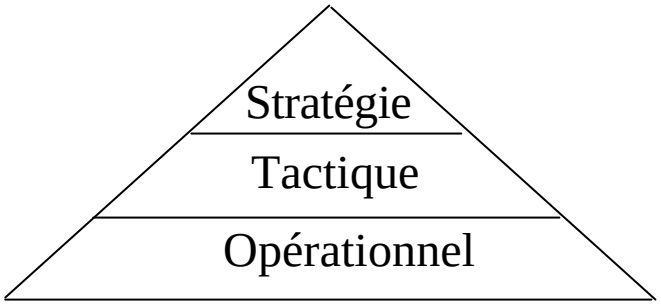
Maintenir uniforme a un niveau acceptable par l'utilisateur la perception qu'a celui-ci du fonctionnement du système et du réseau.

Enjeux économiques

. Minimiser les coûts d'investissement informatique et d'administration.
. Déterminer les coûts d'usage (facturation).
. Limiter les risques financiers de l'entreprise relatives aux défaillances de l'informatique.

1.3 Les niveaux politiques d'administration des systèmes et réseaux

Modèles de l'administration réseaux analogues à ceux des entreprises.
Exemple modèle d'Anthony 1965)



Les décisions prévisionnelles et stratégiques

Concerne la planification technique et financière du système informatique.

Garantit la pérennité des choix.

Suivi de l'utilisation, des événements survenus et analyse des besoins émergents

- Définition des critères de performance et de fiabilité à atteindre.
- Préparation des décisions concernant l'infrastructure du futur.

La gestion administrative

Liée à l'organisation de l'entreprise

Concerne les actions et objectifs à court terme définis sous les règles de contrôle budgétaire et la politique de taux d'équipement.

La gestion opérationnelle

Directement relative au fonctionnement du système, elle agit au niveau courant (réglage des paramètres de performance et suivi des incidents).

L'objectif des principales normes et logiciels de gestion de réseaux.

1.4 Les principales fonctions de la gestion des systèmes et réseaux

. Un cadre conceptuel proposé dans la norme OSI 7498-4 ("OSI Management Framework").

. Ce modèle énonce cinq domaines conceptuels fortement corrèlés: SMFA "System Management Functional Area"

. Découpage utilisable pour une approche modulaire de la gestion des systèmes et des réseaux.

Les cinq fonctions de la gestion

- La gestion de la configuration.

- La gestion des anomalies.

- La gestion de la comptabilité.

- La gestion de la sécurité.

- La gestion des performances.

Gestion de la configuration ("Configuration management")

But

Connaissance (exhaustive") de l'état du système à tout instant.

- Recueillir et diffuser les informations sur l'état du système et sa topologie
- Définir et gérer les objets d'administration.

Opérations à réaliser

- Surveiller les équipements
- Gérer les modifications
  (historique de l'évolution)
- Télécommander les systèmes
- Maintenance des objets de gestion représentant les caractéristiques intangibles des équipements.

Numéro de série, type de contrat d'entretien..

- Maintenance des objets de gestion représentant les caractéristiques instantanées des équipements.

Etat opérationnel, utilisateurs connectés, ...

- Téléchargement du logiciel.

Gestion des anomalies ("Fault management")

But

Offrir un service fiable et de qualité prédéfinie.

- Localiser les équipements défaillants puis déterminer les causes des défaillances.
(les défaillances peuvent être soit des pannes franches soit des fonctionnements dégradés).
- Types de problèmes
    Communication (perte de signal ...)
    Qualité de service (fort temps de réponse)
    Logiciel (surconsommation de ressources, incompatibilité, ...)
    Matériel (panne, dégradation des conditions de fonctionnement ex température)

Opérations à réaliser

- Détection de panne et diagnostic.
- Assertion sur l'étendue des dommages (effets induits par une panne).
- Tentative de correction.
- Mise à jour historique.

Gestion des informations comptables ("Accounting management")

But

Différencier les consommations individuelles en vue de la facturation.

- Analyse des coûts de consommation de ressources (réseaux, machines).
- Analyse des coûts de personnel.

Opérations à réaliser

- Surveillance des consommations (événements de réservation de ressource).
- Définition et gestion des tarifs.
- Gestion des profils d'utilisateur.
- Fabrication des documents comptables (facturation, comptabilité analytique ...).

Gestion de la sécurité ("Security management")

But

Détermination des failles potentielles du système afin de mettre en oeuvre la politique de sécurité (se prémunir contre les menaces).

- Analyse des coûts des attaques.
- Définition d'une politique de sécurité adaptée au coût d'une attaque réussie.

Opérations à réaliser

- Surveillance des intrusions.
- Gestion de l'historique des intrusions (audit de sécurité).
- Mise en oeuvre des techniques retenues de confidentialité, d'intégrité, d'authentification, de contrôle d'accès.
- Gestion et délivrance des identificateurs, des mots de passe, des clés utilisées dans les différents systèmes de cryptographie et les protocoles de sécurité.

Gestion des performances ("Performance management")

But

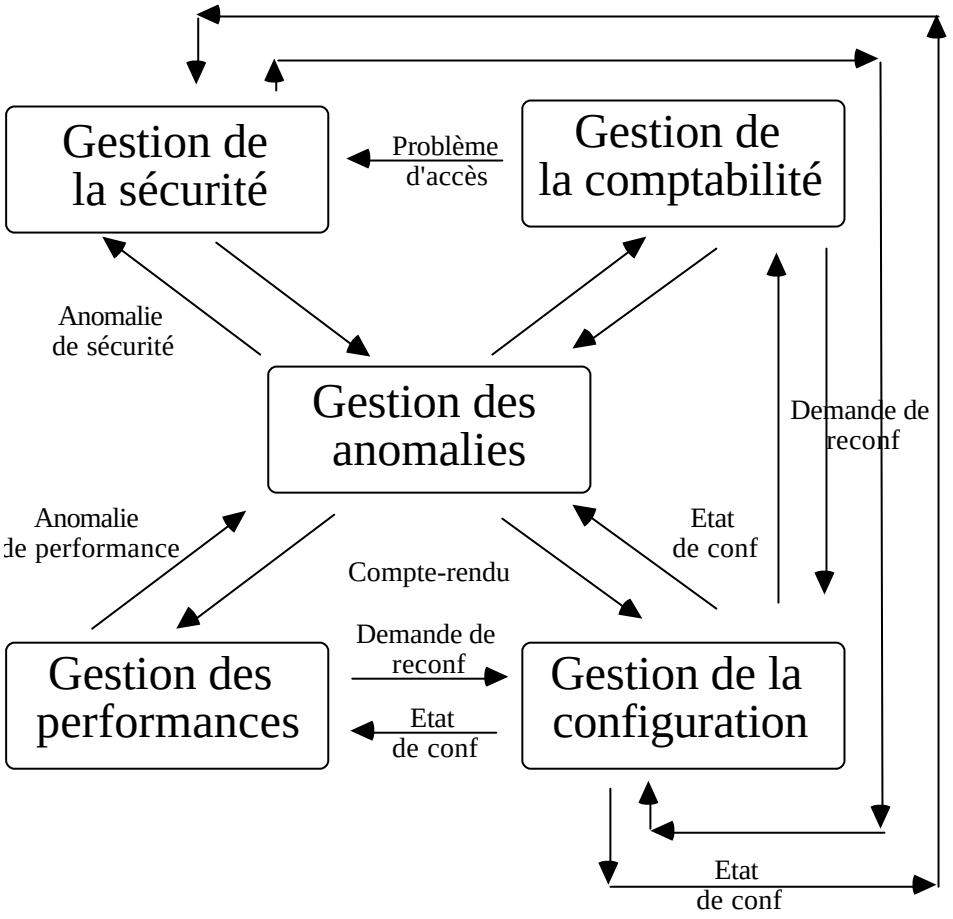
Suivre les principaux paramètres de comportement du système caractéristiques de son efficacité.

- Analyse des divers temps de réponse, débits.
- Analyse des situations de congestion.

Opérations à réaliser

- Surveillance des paramètres en temps réel (sur scrutation périodique, sur franchissement de seuil).
- Contrôle et analyse des données recueillies.
- Historisation des valeurs mesurées.
- Optimisation du systèmes, règlages des paramètres systèmes (nombre d'utilisateurs acceptés, ressources allouées, ...).
-Planification de l'exploitation en fonction des qualités de service attendues.

Exemples d'interactions entre les cinq domaines fonctionnels de l'OSI



1.4 Les principaux standards de la gestion des systèmes et réseaux

SNMP
"Simple Network Management Protocol"

- Produit par la communauté Internet pour fonctionner sur les réseaux TCP/IP.

- Deux versions successives SNMP V1 et SNMP V2.

- Une solution simplifiée et efficace pour définir de petits agents administration.

CMIP
"Common Management Information Protocol"

- Produit par la communauté de la normalisation internationale OSI pour fonctionner au dessus des piles OSI.

- Une solution plus complexe, offrant plus de fonctionnalités, en univers ouvert mais aussi plus lourde.

Autres standards

DME "Distributed Management Environnment"

- Produit par le consortium OSF ("Open Software Foundation") dans l'environnement DCE ("Distributed Computing Environnment")

Incorpore CMIP et SNMP et adapte leur usage au cadre DCE.

DMI "Desktop Management Interface"

- Produit par le consortium DMTF ("Desktop Management Task Force" IBM, DEC, HP ...) pour la gestion des parcs de PC et de réseaux locaux sous leurs systèmes habituels.

Bases de données d'information de gestion MIF ("Management Information Format") MIF PC, MIF carte réseau, MIF imprimante ...

TNM "Telecommunication Network Management"

- La vision par l'ITU-T ex CCITT reprenant et étendant les normes OSI CMIP (norme M30)

2

INTRODUCTION A LA GESTION

DE RÉSEAUX OSI

CMIP/CMIS

"Common Management Information Protocol and Service"

2.1 Structure générale de la gestion OSI

Au niveau 7 du modèle de référence

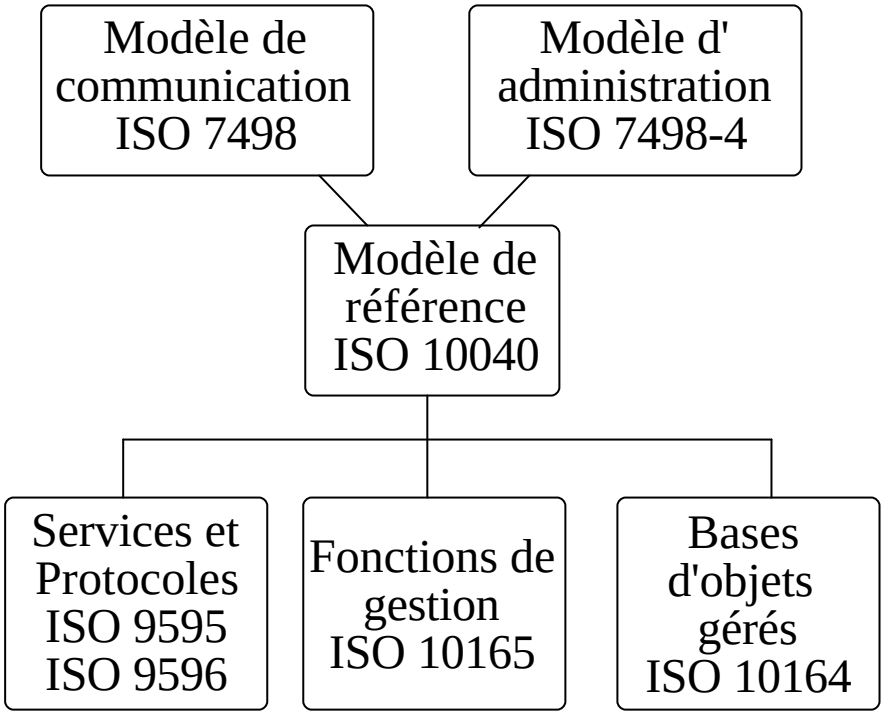
. Adopte les principes de structuration d'une application OSI.
(ALS "Application Layer Structure" ISO9545)

. Utilise en particulier les services communs prédéfinis au niveau application.

- Service d'établissement et de libération de connexions
(ACSE "Application Control Service Element" ISO 8649)

- Service d'invocation d'actions distantes (service de communications)
(ROSE "Remote Operation Service Element" ISO 9072)

Composants d'administration d'un réseau OSI



Vision fonctionnelle: ISO 7498-4, ISO 10164.

Vision organisationnelle: ISO 10040.

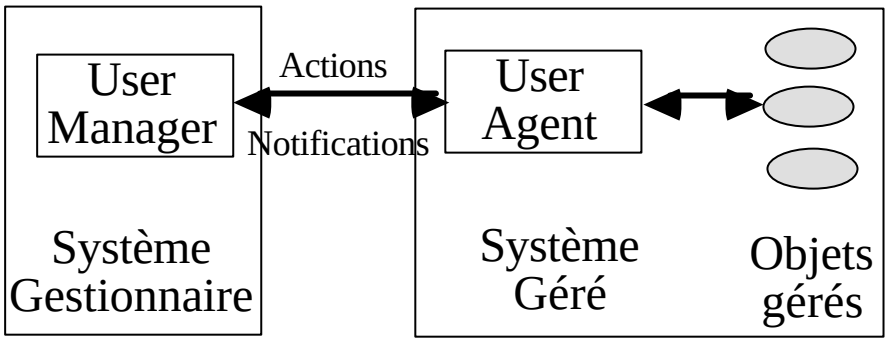
Vision informationnelle: ISO 10165.

Vision communicationnelle: ISO 7498, ISO 10040, ISO 9595, ISO 9596.

Gestionnaires et agents
"Managers" "Agents"

La normalisation est basée sur le dialogue "client serveur" entre:

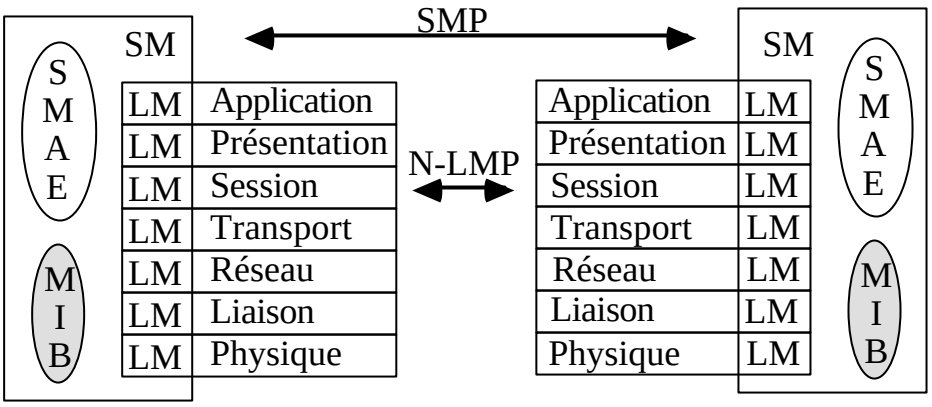
- un système d'administration ("user manager")
- un agent d'administration ("user agent")



Remarque

Un utilisateur peut-être simultanément agent (pour une opération) et gestionnaire (pour une autre opération).

2.2 L'organisation d'une architecture de gestion OSI



Ensemble SM ("System Management")

L'ensemble des définitions des mécanismes d'administration des objets (contrôle, coordination et supervision) ISO10040

Niveaux N-LM("N-Layer Management")

L'ensemble des activités de gestion entre systèmes au niveau de la couche N utilisant des protocoles spécifiques de chaque couche baptisés N-LMP.

SMAE - System Management Application Entity
LM - Layer Management
SMP - System Management protocol
LMP - Layer Management Protocol
MIB - Management Information Base.

2.3 Les informations de gestion

Norme SMI ("Structure of management Information" ISO 10165)

. L'ensemble des informations de gestion est représenté sous forme d'objets avec l'essentiel de la puissance du modèle objet:

- Existence d'attributs.
- Existence de méthodes de traitement.
- Possibilité de notification d'événements.
- ...

. Un objet d'administration est l'abstraction d'une ressource. Des mécanismes de chaque agent maintiennent le lien entre les objets de gestion et les ressources réelles.

. Un objet d'administration est une instance d'une classe correspondant le plus souvent à un type de ressource ou à un ensemble composé de ressources (PABX, réseau local).

. Certains objets (journaux, filtres) ne sont pas associés directement à la représentation d'une ressource matérielle ou logicielle.

Modèle des informations de gestion
"Management Information Model 10165-1

. Définit les principes de désignation des objets et de leurs attributs.

. Chaque objet possède deux types de caractéristiques:

- caractéristiques statiques ("ses attributs")

Exemple: pour un routeur le nombre de paquets par unité de temps sur une voie.

- caractéristiques dynamiques (les alarmes et notifications qu'il est capable d'émettre)

Exemple: pour une voie le franchissement d'un seuil de surcharge.

. Définit les principaux concepts objets utilisés dans les objets de gestion:

héritage, spécialisation, polymorphisme (pour les objets de gestion allomorphisme) et contenance).

. Les descriptions de types et la syntaxe de transfert sont basées sur ASN1 ("Abstract Syntax Notation") et ses règles d'encodage (BER "Basic Encoding Rules").

Relations entre classes et objets

. La relation de spécialisation (arbre)

Basée sur le mécanisme d'héritage qui permet de définir de nouvelles classes à partir des classes existantes.

. La relation de contenance (arbre)

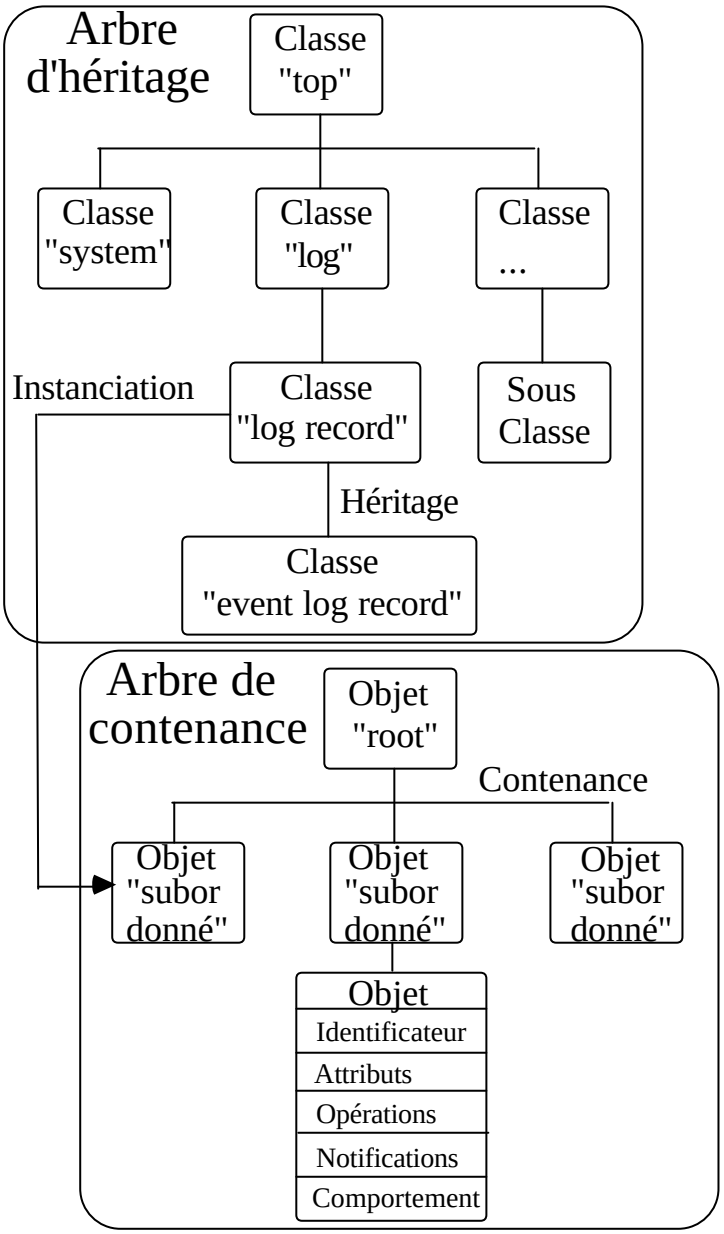
Elle caractérise la possibilité pour certains objets de contenir d'autres objets provenant d'une même classe ou de classes distinctes.

. Le nommage (arbre)

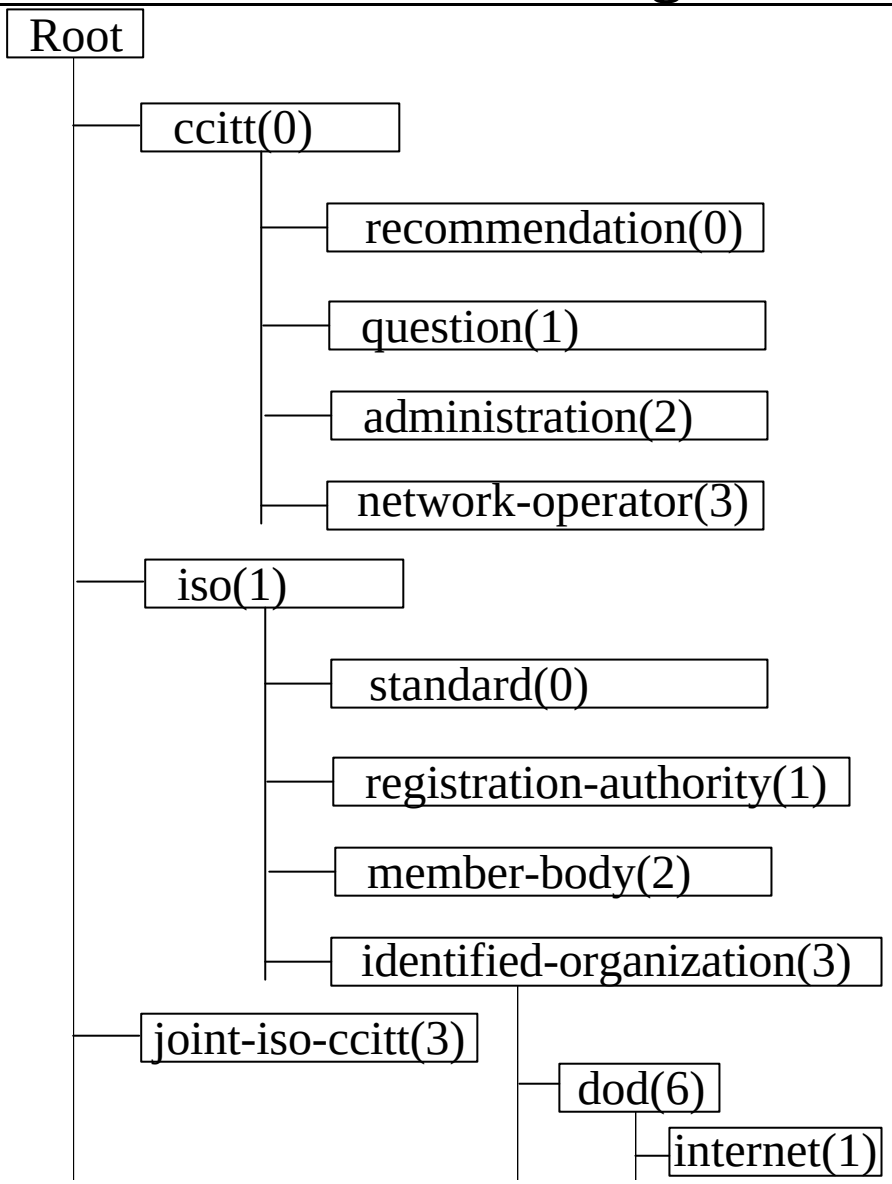
Un ensemble de noms enregistrés et dont l'usage est établi.

Les nouveaux noms doivent être créés dans ce contexte.

Arbres d'héritage et de contenance



Arbre de nommage



Niveaux les plus hauts de l'arbre de nommage

Guides pour la définition d'objets de gestion
"Guidelines for the definition of managed objects
10165-4

. Définition des règles et de la syntaxe pour la création de nouvelles classes d'objets.

. Décrit en particulier la notion de MIB ("Management Information Base"): un ensemble d'objets accessibles sur un agent.

Types d'objets

. Objets d'état

Décrivent les objets se rapportant aux différents niveaux d'une architecture de communication (attributs plutôt statiques).

. Objets de gestion

Ils permettent en particulier de contrôler l'émission des notifications d'événements et l'archivage de ces derniers dans des journaux.

. Objets d'inventaire

Décrivent statiquement les équipements (constructeur, version, ...).

2.4 Les fonctions de communication CMIS/CMIP

Le service CMIS ("Common Management Information Service")

Trois types d'usage

Pour des services confirmés (**C**)
ou non confirmés (**NC**)

Gestion des objets ("monitoring")

M-DELETE	Destruction d'un objet	C
M-CREATE	Création d'un objet	C
M-ACTION	Exécution sur un objet	C/NC
M-CANCEL-GET	Arret d'un get trop long	C/NC

Manipuler les attributs ("control")

M-GET	Recherche de valeur d'attribut	C
M-SET	Modification d'attribut	C
M-EVENT-REPORT	Notification d'événement	C/NC

Notifier la réalisation d'une action ("reporting").

Object create reporting
Notification de création C/NC
Object delete reporting
Notification de destruction C/NC
Attribute value change reporting
Notif changement valeur C/NC
Object Name change reporting
Notif changement nom C/NC

Spécifications particulières

1 **Vues** (Scoping): Sur la base de l'arbre de contenance on définit un ensemble d'objets sur lequel un filtre doit agir.

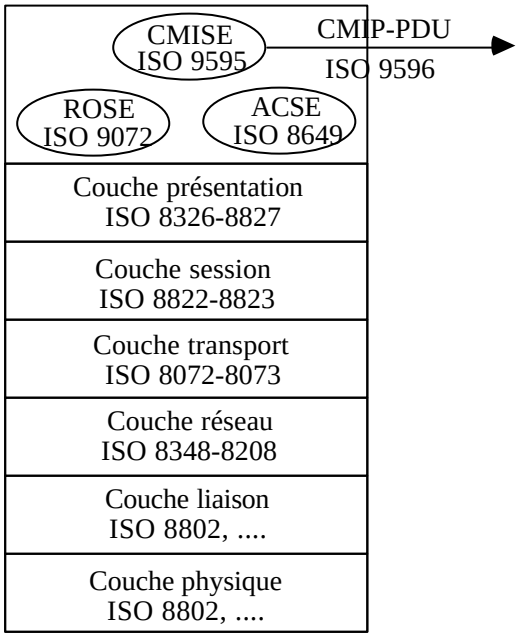
2 **Filtrages** (Filtering):Expression booléenne sur la présence ou l'absence de valeurs sur un objet sélectionné.

3 **Synchronization**: Spécifie si les accès doivent être atomiques ou sans règles précises de partage.

Les différents types de protocoles CMIP "Common Management Information Protocol"

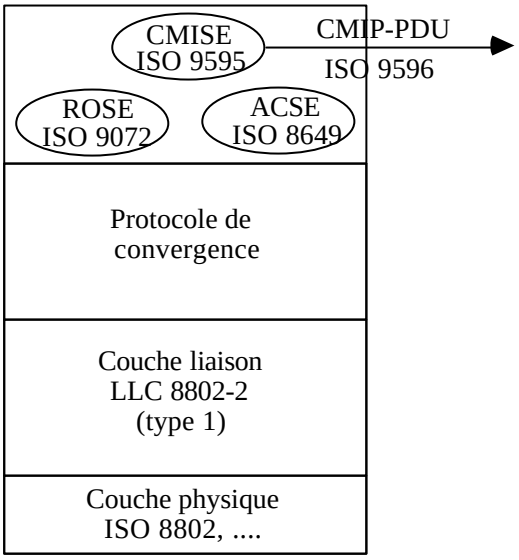
Mode standard: CMIP sur OSI

Inconvénient: nécessite de disposer d'une pile complète OSI et est jugé très encombrant en place mémoire (250 ko).



CMIP sur réseau local
CMOL "CMip Over Logical link control"

Une solution simplifiée pour se passer de la pile OSI et limiter l'encombrement (20 ko).

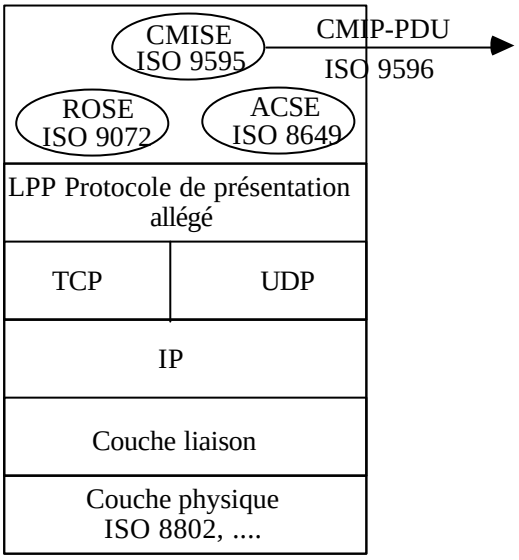


Protocole de convergence: pile OSI restreinte spécifique CMOL (ASN1, ...).
Niveau liaison dans les réseaux locaux IEEE 802 : LLC type 1 (sans connexion).

Solution IBM et 3-COM pour des réseaux non routables (HLM "Heterogeneous Lan Management" pour Lan manager).

CMIP sur TCP/IP
CMOT "CMip Over TCP/IP

Solution développée pour utiliser CMIP/CMIS en environnement TCP/IP et permettre l'utilisation conjointe de ce standard et de SNMP.



Protocole de présentation allégé: LPP Ligthweight Presentation Protocol: permet le transcodage ASN1.

Couches basses: celles de TCP/IP.

Conclusion gestion de réseau OSI

- **Avantages** : une réflexion en profondeur des fonctions d'administration à réaliser (analyse orientée objet, complétude, utilisable en milieu très hétérogène).

- **Inconvénients** : les solutions nécessitent des développements logiciels assez lourds, un encombrement mémoire plus important et les approches OSI sont en perte de vitesse.

Ces solutions sont un peu déployées dans certains produits et presque toujours en parallèle avec SNMP.

3

INTRODUCTION A LA GESTION

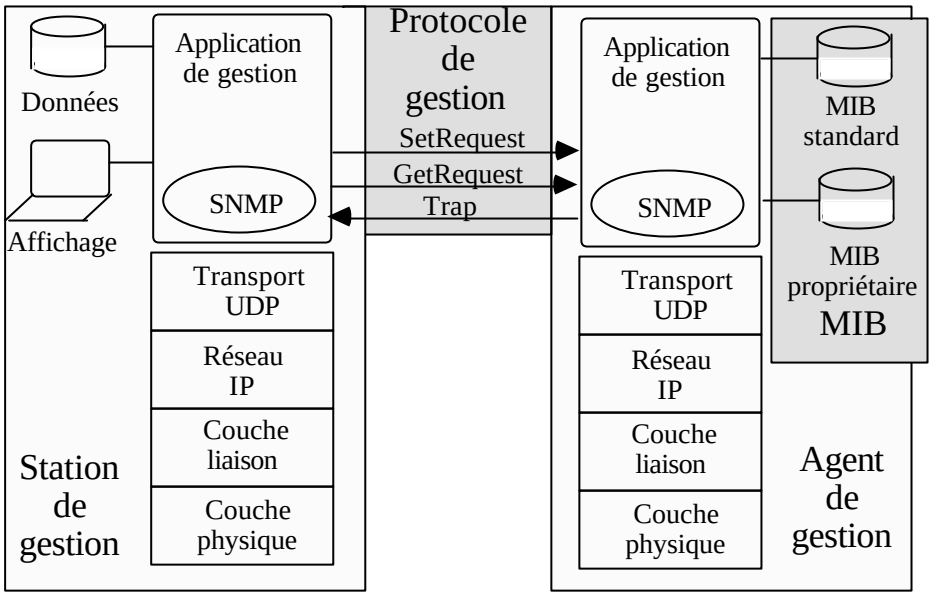
DE RÉSEAUX INTERNET

SNMP

"Simple Network Management Protocol"

3.1 Introduction à la gestion SNMP

Architecture de SNMP



Notions essentielles

- Station de gestion ("manager")

- Agent de gestion ("agent")

- Bases d'objets de gestion ("MIB")

- Protocole d'échange

Détail des concepts de base

Station de gestion ("Management station") ("Manager")

C'est un système informatique complet ou une application d'un système partagé qui sert d'interface à un opérateur.

Il comporte:

- Un logiciel d'administration capable de traduire un langage de commandes opérateur en accès distant, traitements ...
- Une base de données de variables concernant les ressources administrées.
- Un ensemble d'outils annexes utiles aux applications d'administration (analyse de données, diagnostics de pannes, ...)
- Une interface utilisateur (graphique) pour présenter les résultats des commandes.

Agent de gestion ("Management agent" "Agent")

C'est un logiciel **actif** qui tourne sur un équipement (calculateur, routeur, ...) sous forme d'un processus (démon UNIX) pour:

- le surveiller, enregistrer des valeurs
- répondre aux requêtes des stations d'administration
- générer sur événement important un message asynchrone ("trap").

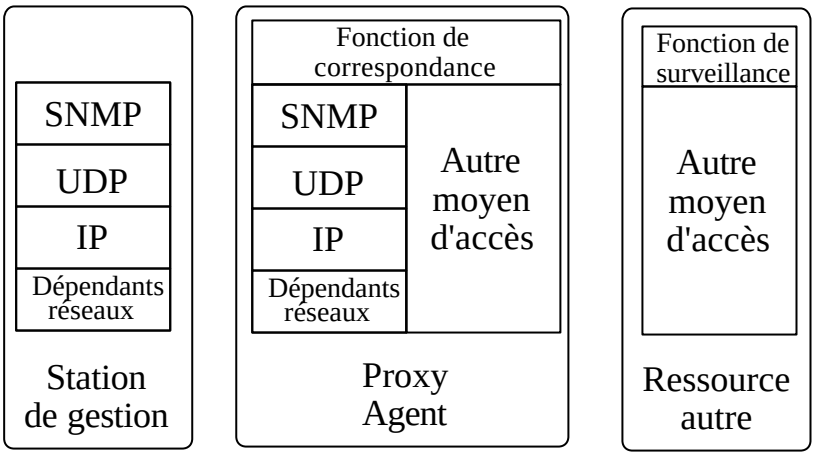
Base de donnée de gestion ("Management Information Base")

Une base de donnée avec ses règles de structuration et ses primitives d'accès pour stocker les variables manipulées.

Protocole d'échange de données ("Network Management Protocol")

Le protocole de communication entre une station d'administration et un agent.

Agent de procuration SNMP
"Proxy agent"



. Pour permettre d'administrer par SNMP des agents dans des réseaux non TCP-IP.

. Pour administrer des équipements trop simples (modems) pour supporter tout un agent SNMP (TCP/IP, SNMP, MIB).

. L'agent proxy SNMP reçoit les requêtes pour une ressource qu'il ne gère pas.

. Il les convertit dans une requête (simplifiée) à l'équipement dans un autre protocole.

. Il reçoit et transmet la réponse en SNMP.

Historique SNMP

. Depuis la création du réseau ARPA (1970) jusqu'aux débuts de l'INTERNET (1980) pratiquement pas de fonctions de gestion.

. Jusqu'en 1985-1988 utilisation des fonctions d'administration manuelles développées au moyen des échanges de messages entre routeurs ICMP autour de la suite TCP/IP.

. ping ("Packet Internet Groper") : site distant opérationnel ou non

. traceroute : édition d'un chemin

. En 1987 le protocole SGMP RFC 1028 ("Simple Gateway Monitoring Protocol") dédié uniquement à la gestion des routeurs

. En 1988 travaux sur SNMP généralisant SGMP à tous les types d'équipements.

SNMP version 1 RFC 1157 date de 1990.

MIB-II RFC 1213 date de 1991.

. En avril 1993 RFC 1444 à 1448 SNMP V2 (avec de nouvelles MIB).

Difficultés dues à l'existence de deux standards et de MIB successives

2.2 La structure des informations de gestion SMI "Structure of Management Information" RFC 1155

Principes généraux

. SMI définit la structure d'une MIB.

Deux versions successives principales MIB-I (114 objets) obsolète puis MIB-II (171 objets) en compatibilité ascendante avec la MIB-I.

. **SMI définit chaque objet stocké** dans une MIB identifie les types de donnés et la valeur de chaque objet.

Les objets d'administration SNMP n'ont pas toute la richesse du modèle objet. Ce sont de simples variables avec un type et quelques caractéristiques (lecture seule, ou lecture écriture)

. SMI définit les règles de description et d'encodage pour chaque objet

Utilisation de ASN1 "Abstract Syntax Notation 1" et BER "Basic Encoding Rules".

Structure d'une MIB standard

. Analogue d'une MIB OSI.

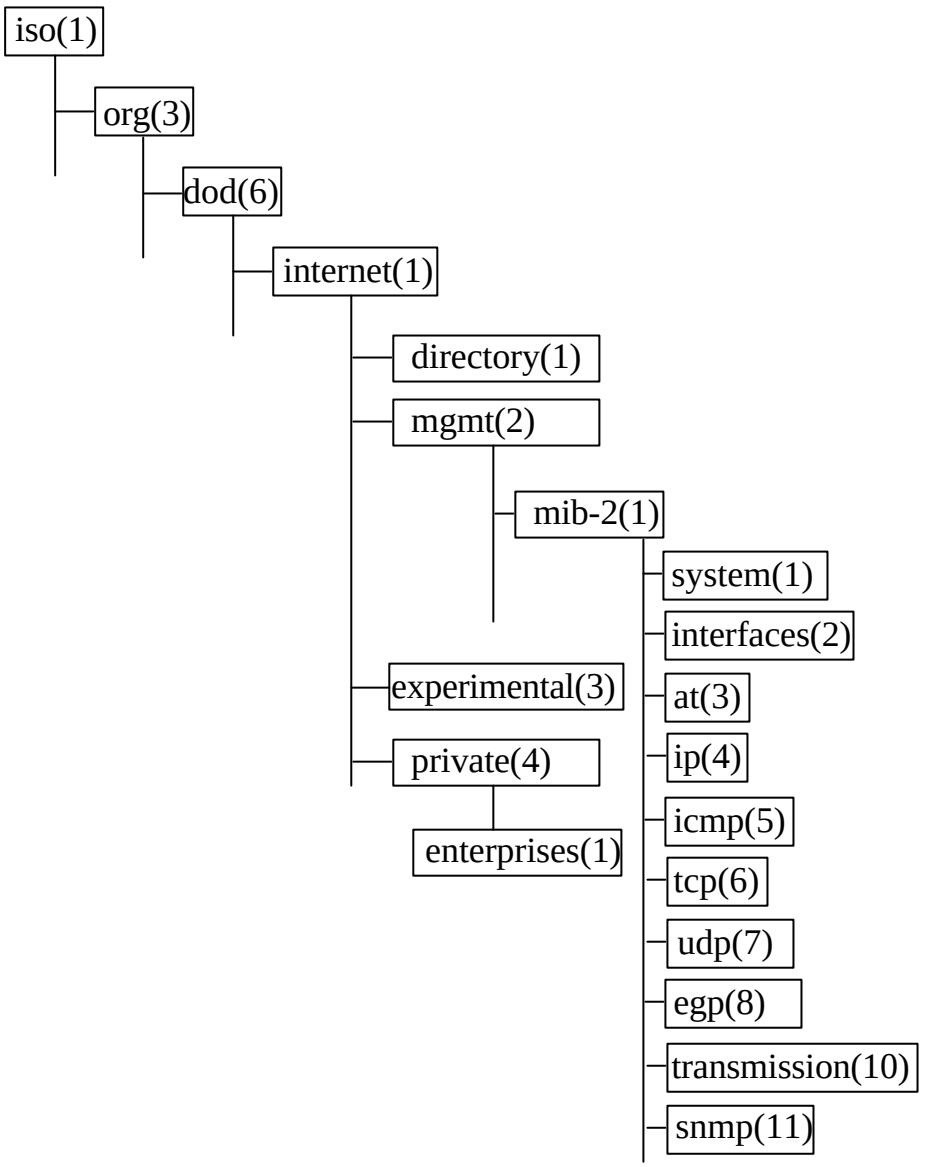
. Une MIB est une structure de données organisée comme un arbre dans laquelle sont rangés les objets administrés.

. Au sommet de l'arbre se trouvent les informations les plus générales.

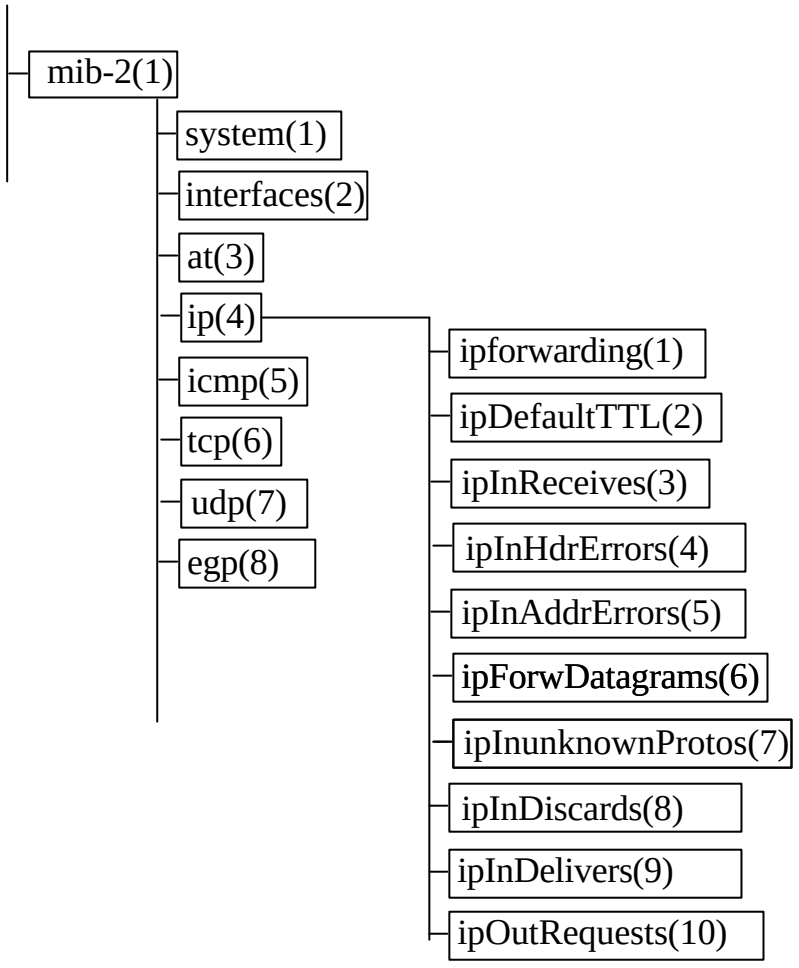
. Plus le niveau est bas plus l'information est détaillée.

. Les feuilles concernent les variables les plus spécifiques qu'un accès peut fournir.

MIB-II Début de l'arbre de nommage



Détail d'une branche: exemple IP



Commentaires arbre de nommage

L'arbre de nommage est commun OSI SNMP.

L'IAB ("Internet Activities Board") a un sous-arbre (de racine internet) sous dod ("Department Of Defence").

Les quatre noeuds sous internet ont la signification suivante

directory : sous-arbre pour les services d'annuaires

mgmt : sous-arbre des objets d'administration approuvés par l'IAB.

experimental : en expérimentation.

private : sous-arbre des objets définis unilatéralement.

La définition ASN1 d'un noeud quelconque est construite selon le mécanisme:

mib-2 OBJECT IDENTIFIER ::=
{ iso(1) org(3) dod(6) internet(1) mgmt(2) mib-2(1)}

Soit encore en valeur 1.3.6.1.2.1 qui préfixe toutes les définitions d'objets manipulés via la mib-

2.

Exemple de fonctionnement

L'objet ipDefaultTTL définit la valeur par défaut du TTL ("Time To Live") durée de vie de chaque paquets IP (en nombre de sauts).

La désignation de l'objet ipDefaultTTL est: iso.org.dod.internet.mgmt.
mib-2.ip.ipDefaultTTL
soit encore 1.3.6.1.2.1.4.2.

La valeur de la variable ipDefaultTTL est obtenue par le message (la commande)
GetRequest (1.3.6.1.2.1.4.2)

qui produit par exemple la réponse:
GetResponse ((ipDefaultTTL = 30))

Si l'on souhaite maintenant changer la variable ipDefaultTTL :
SetRequest ((ipDefaultTTL = 32))

qui produit la réponse
GetResponse ((ipDefaultTTL = 32))

Les autres MIB

Les MIB RMON (RFC 1271) "Remote MONitor"

Étudiées pour recueillir les informations provenant des sondes sur les réseaux locaux (analyseur de trafic, collisions, charge).

Neuf groupes de données

- Les statistiques
- Les historiques
- Les alarmes
- L'enregistrement des hôtes sur le réseau
- Les matrices de trafic (entre groupes)
- Les filtres
- Les paquets capturés
- Les événements générés par les agents

Les MIB privées

Chaque constructeur définit pour ses propres dispositifs et en fonction de leurs caractéristiques des MIB privées.

Ces MIB peuvent reprendre et étendre d'autres MIB en particulier les mib standards.

L'explosion du nombre des MIB privées et leur variété est une difficulté très importante pour la mise en oeuvre de SNMP.

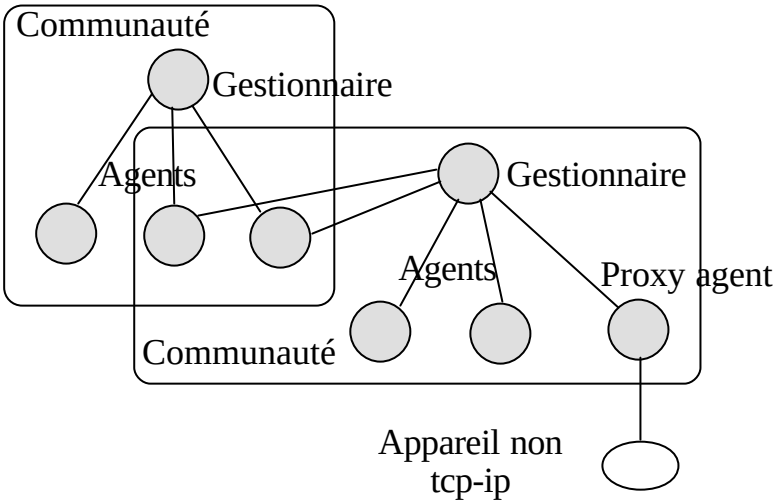
Quelques exemples de MIB privées

MIBNT	Pour Windows NT
LMMIB2	Pour lan manager et Windows NT
DEFLM	Pour lan manager
DEFNB	Pour Netbios
DEFORA	Pour Oracle
FMS	Pour les répéteurs 3-coms
CISCO	Pour les routeurs cisco

...

2.3 Le protocole de communication des informations de gestion

Principes généraux SNMP V1



Une relation gestionnaire-agent à un niveau

Les cinq primitives SNMP V1

GetRequest(0)	Requête pour lire une valeur d'instance
GetnextRequest(1)	Pour lire en séquence la valeur d'objet suivante dans l'arbre
GetResponse(2)	Réponse d'un agent
SetRequest(3)	Modifier la valeur d'un objet d'une MIB
Trap(4)	Notification d'événement exceptionnel

Modes de fonctionnement SNMP V1

- Interrogations cycliques ("Polling")

A intervalle régulier on interroge un agent
On construit une statistique, un graphique

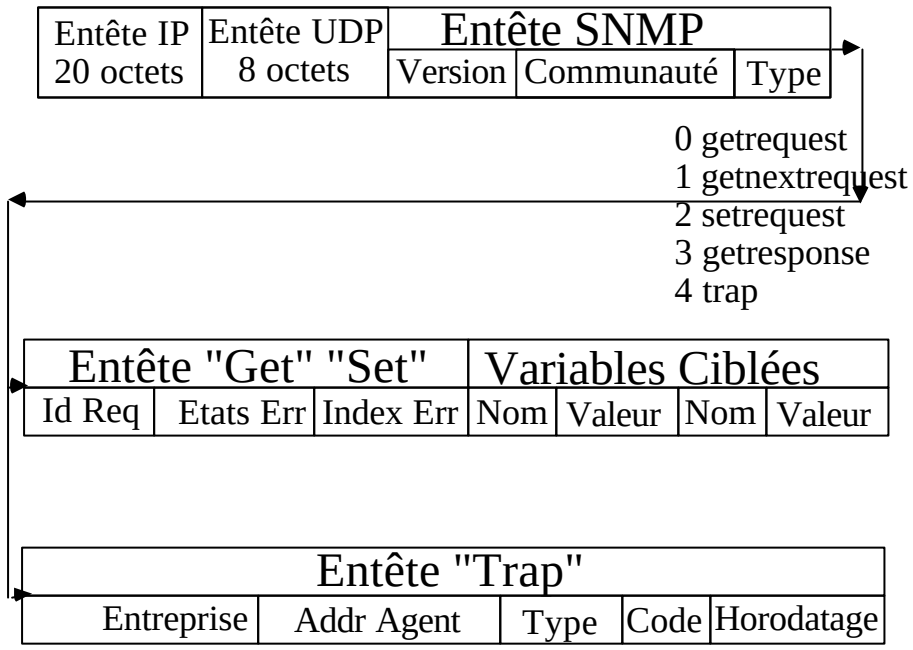
=> **Trafic réseau élevé.**

- Gestion événementielle ("trap")

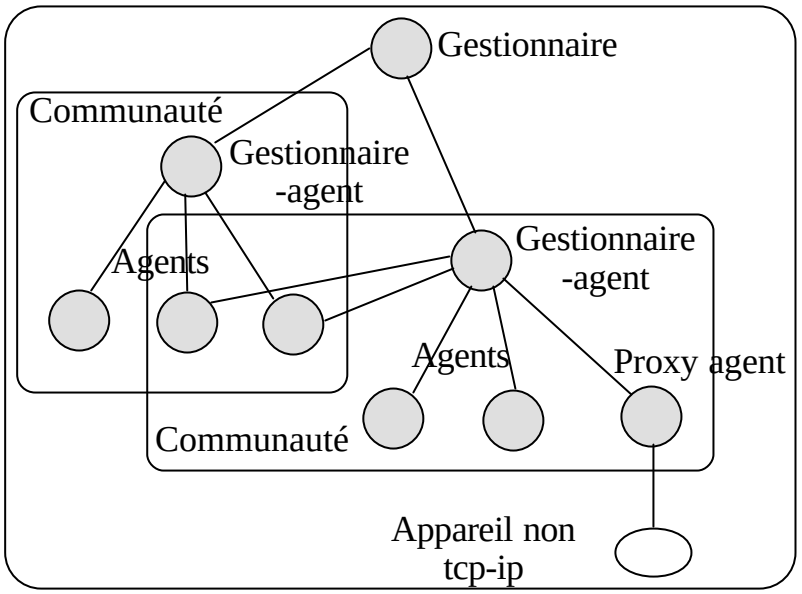
A l'initiative de l'agent

=> **Permettent de limiter le trafic.**

Format des messages SNMP



Principes généraux SNMP V2



- Communication entre gestionnaires selon un arbre à plusieurs niveaux.
- Support d'un ensemble de variables et de MIB plus importantes.
- Nouvelles primitives de communication.
- Amélioration des mécanismes de sécurité.

GetBulkRequest Lecture n valeurs d'un objet en une seule fois.

InformRequest Transmission de données d'un gestionnaire vers un agent.

SNMPv2-Trap Trappe format v2 (? v1).

Conclusion SNMP

- Les plates-formes SNMP éclipsent les anciens logiciels d'administration (grands systèmes).

- Elles sont simples à développer et permettent aux fournisseurs de matériels de se présenter sur le marché avec un agent de gestion (bon pour la plaquette commerciale).

- Nombreux problèmes liés aux choix simplificateurs de SNMP.

Faible ouverture aux autres architectures.

Nombreuses MIB privées.

Faiblesse des moyens de sécurité.

Surcharge des réseaux due aux scrutations périodiques.

Le diagnostic de panne reste très délicat hors des cas simples.

Absence d'une formalisation objet.

...

4

EXEMPLES DE LOGICIELS COMMERCIALISES

4.1 Les différentes classes de produits

Les hyperviseurs

- Destinés à offrir une vision globale pour l'ensemble des ressources de l'entreprise.
- Ils doivent nécessairement présenter une certaine ouverture à différentes classes de réseaux (SNA, DECNET, TCP-IP, ...) et différents standards de gestion en raison de l'hétérogénéité.

L'ouverture se fait souvent au moyen d'API normalisées pour des applications développées par des sociétés tierces.

Fonctions réalisées

. Découverte et cartographie de réseau

. Gestion des événements et des alarmes

. Émission et collecte de données spécifiques SNMP: affichage tableaux ou graphique, journalisation.

. Balayage de MIB: édition exhaustive.

Exemple de produits

Netview	IBM	ISM	Bull
Openview	HP	Polycenter	DEC

Les superviseurs

- Destinés à gérer un domaine particulier ils utilisent des standards d'administration spécifiques (typiquement SNMP sur un réseau uniquement TCP-IP) ou même des protocoles propriétaires.

- Ces logiciels peuvent être utilisés dans le cadre d'hyperviseurs à vocation plus générale.

Exemple de produits

Netview 6000	IBM	Spectrum	Cabletron
Sunnet Manager	SUN	NMS	Novell
Node manager	HP		

Les gestionnaires et agents propriétaires dédiés

- Destinés à gérer un ensemble limité d'équipements ou un seul appareil (modem, multiplexeur, commutateur, routeur, ...).
- Les grands fournisseurs d'équipements (CISCO) et d'hyperviseurs (HP Openview) passent des accords de compatibilité.

Conclusion

L'automatisation des tâches de gestion système et réseaux est un objectif en progrès mais **il reste énormément à faire**.

Les outils actuels sont très utiles mais ils sont essentiellement **aptes à scruter des variables** et à en **réaliser l'affichage** sous une forme plus ou moins conviviale.

Le processus de **compréhension** et de **décision** reste dans la plus grande partie des cas **humain**.

L'introduction des techniques de **l'intelligence artificielle distribuée ("multi-agent")** dans les logiciels d'administration est indispensable à terme pour assister les opérateurs mais reste encore du domaine de la recherche.

Bibliographie
Administration réseaux

Marshall. T. Rose "**The simple book: an introduction to management of TCP-IP based internets**" Prentice Hall 1991

William Stallings "**SNMP, SNMP V2 and CMIP The practical guide to network management standards**" Addison Wesley 1993

Gil Derudet "**L'aministration des systèmes distribués: une approche multi-agents**" , Mémoire d'ingénieur du CNAM octobre 1995

Yann Perrot "**Implantation d'un outil d' administration système d'information à la direction production transport de gaz de France**" Mémoire d'ingénieur du CNAM juin 1997.

L'ACCÈS AUX BASES DE DONNÉES DISTANTES.

G Florin

Plan du cours

Introduction

Les applications client-serveur en informatique d'entreprise: trois parties

- Une partie interface homme machine ("Présentation")
GUI: "Graphical User Interface"
OOUI: "Object Oriented User Interface"

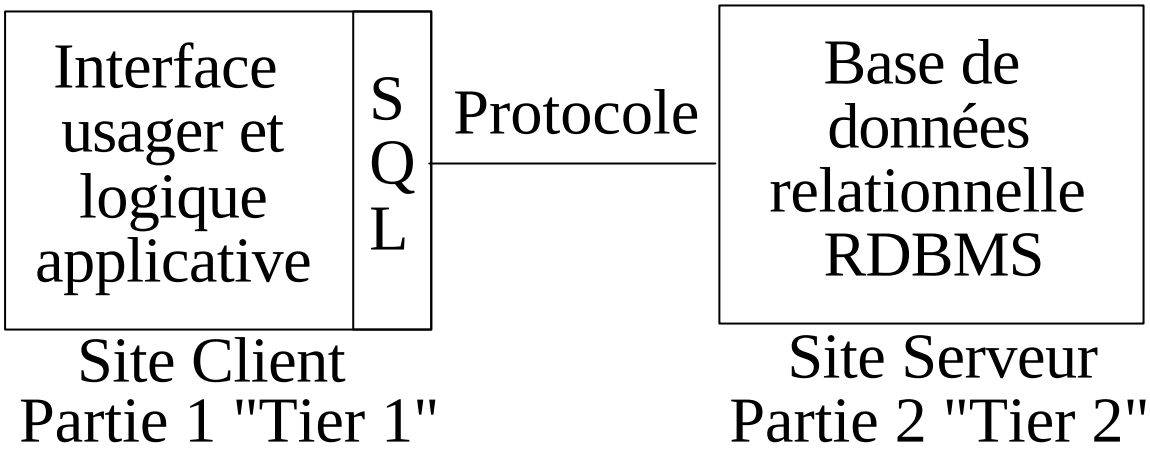
- Une partie application d'informatique de gestion. ("Business logic")

- Une partie base de données RDBMS ("Relational DataBase Management System")

Architectures d'applications client-serveur

Différents choix de placement des fonctions précédentes sur différents sites.

Architecture client-serveur à deux parties (deux étages) "2-tiered Client-server architectures"



Problème à résoudre en réseau

Spécification du service SQL coté client et protocole entre les parties.

Améliorations

- Performances: procédures stockées

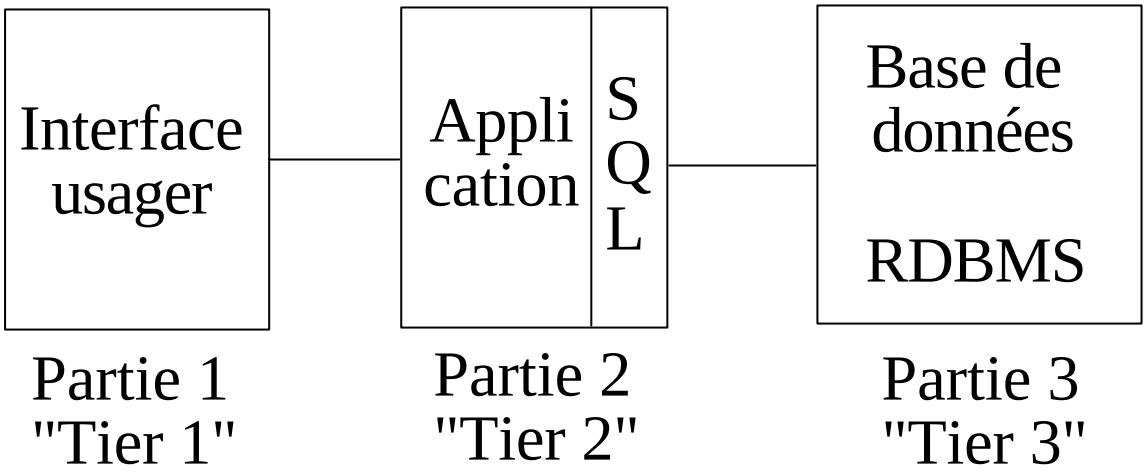
Une part de la logique applicative est stockée en dehors du client (plutôt dans la base de données) => Solution "2-1/2 tiered"

- Interopérabilité: passerelles

Vers d'autres bases de données

La base de données serveur sert de relais pour d'autres bases de données.

Architecture client-serveur à trois parties (trois étages) "3-tiered Client-server architectures"



Problèmes réseaux à résoudre

- Interface d'accès à l'application et protocole entre les parties 1 et 2:

Solution: invocation d'actions distantes: mode message, RPC (objets répartis)

- Service SQL et protocole entre l'application et la base de données

Même problème que précédemment entre les parties 2 et 3.

Les bases de données

. Existence de différentes générations de standards de bases de données (hiérarchique, relationnel, déductif, objet).

. Situation actuelle dominante: les bases de données relationnelles et le langage d'interrogation SQL:

SQL une API du service d'accès aux serveurs de bases de données relationnelles en théorie normalisée.

Le client-serveur de bases de données

. Besoin impératif dans les applications client-serveurs de base de données de réaliser l'accès à distance aux données des BD relationnelles par requêtes SQL.

- Développement de **solutions spécifiques** de chaque fournisseur de bases de données.

- Recherche d'une normalisation.

Difficultés du client-serveur de bases de données

. Existence d'un marché très large de fournisseurs DEC/RDB, Oracle, IBM/DB2, Informix, Sybase, Gupta/SQLbase ... avec:

- **nombreuses interfaces** qui sont des variantes des normes successives (SQL89, SQL92, SQL3) et surtout présentant des fonctions additionnelles diverses.

- des **spécifications** très différentes et très optimisées des **protocoles** de transmission à distance des requêtes.

. En fait dans un marché ou la concurrence est très vive, les fournisseurs:

- font apparaître leurs différences par des **extensions spécifiques.**
- proposent des **versions propriétaires** incompatibles pour tenir une clientèle captive
- la réalité des produits est **difficile à cerner** (sigles, variantes).

1 Rappel SQL "Structured Query Language"

Origines: bases de données relationnelles
(E.F Codd 1970).

Construction courant 1970 du langage de l'algèbre relationnelle.

Première version commerciale:
Oracle 1979.

Aujourd'hui: une centaine de produits commerciaux.

SQL: **un langage de requêtes utilisateur** pour accéder directement aux données (en interactif ou compilé en association à un langage évolué)

SQL: **un langage pour la définition et l'administration** des données.

SQL
devrait être un support commun naturel pour l'accès aux bases de données distantes en réseaux.

Versions successives des normes

SQL89 ou SQL ANSI

Une version de base à laquelle il est facile de se conformer.

Commandes: select, insert, update, delete, commit, rollback.

Sécurité: grant, revoke.

SQL92

Une version cinq fois plus longue que SQL 89.

Notion d'**agent SQL** produisant des directives SQL et de **serveur SQL**.

Notion de **connexions client-serveur.**

Notion de **contrôle transactionnel** (read-uncommited,read-commited,serializable)

Support du **SQL dynamique** (génération de commandes en cours d'exécution).

Support de **table provisoires**.

Support de nombreux **types** de données.

Diagnostics d'erreurs (variable sqlstate commande get diagnostics)

Support des **contraintes de domaines**.

Problèmes spécifiques en réparti

Procédures stockées

Des **extensions procédurales** à une approche qui était à l'origine purement déclarative.

En fait en client-serveur la notion de procédure stockée devient un **mécanisme de RPC**.

On peut aller beaucoup plus vite qu'en transférant les commandes SQL une à une.

Baptisé aussi "**transactionnel léger**" en raison des possibilités transactionnelles.

Le plus souvent les procédures stockées ne sont pas soumises aux techniques de validation à deux phase.

Déclencheurs ("Triggers")

Des actions réalisées sur des événements **relatifs aux données** (modification de valeur, franchissement de seuil ,...) en mode synchrone ou asynchrone.

Ces actions sont souvent **définies sous forme de procédures stockées**.

Problèmes des procédures stockées et des déclencheurs en univers réparti

Inexistence de standards entre différents fournisseurs pour ce qui devient un RPC en réparti.

SQL3

Orientation objet associée à SQL

SQL/Framework :

Cadre de la normalisation SQL3

SQL/Foundation:

Notions de rôles, déclencheurs, requêtes récursives, collections, et objets (types abstraits de données et héritage).

SQL/CLI:

Une interface d'accès au standard SQL3

SQL/Persistant Storage Module:

Persistance, procédures stockées, exceptions

SQL/Bindings:

Liaison avec les langages évolués existants

SQL/Transactions:

Liaison avec le transactionnel (XA)

SQL/Temporal:

Gestion temporelle de l'évolution

2 L'accès à des bases de données distantes d'un seul fournisseur

Une solution simple pour qui peut l'accepter

Tout est le plus souvent très spécifique d'un fournisseur: difficulté à changer mais cohérence et efficacité de la solution.

Les services rendus

. Fournir une couche réseau qui offre une interface d'accès aux services standards d'un logiciel de BD (API SQL).

. Offrir l'accès à distance aux fonctions d'administration de la base de données.

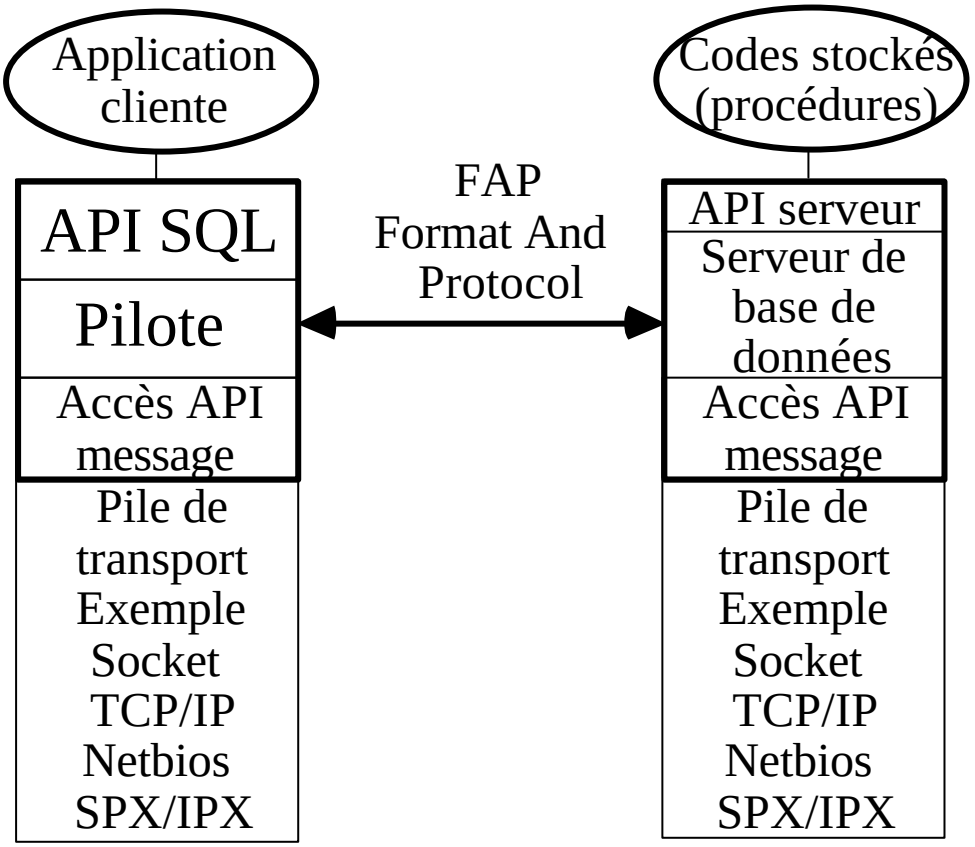
. Offrir des outils d'interrogation et de développement (graphiques) d'applications sur le poste client.

. Accéder à des bases de données d'autres fournisseurs via des passerelles.

L'organisation de la couche application accès distant BD propriétaire

Très grande spécificité des solutions.

Organisation des logiciels à mettre en oeuvre



L'API SQL

L'interface d'appel fournisseur définit l'ensemble des requêtes acceptées coté client:

. au moins le standard SQL 89

. des extensions spécifiques

Nombreuses CLI ("CAll Level Interface") spécifiques de chaque éditeur très optimisées (très efficaces):

Oracle OCI ("Oracle Call level Interface")

Sybase ("Sybase open Client")

Le pilote SQL

Réalise le protocole de présentation de requêtes SQL propre au fournisseur

- Définit les formats de messages
- Définit les enchaînements légaux.

Ce protocole est baptisé FAP

"Format And Protocol".

Spécifique de chaque serveur il est directement traité par le serveur au moyen d'un simple "écouteur".

L'interface d'accès au transport.

Compte tenu de la diversité des piles de protocoles réseaux les fournisseurs doivent supporter plusieurs API de communication.

3 L'accès à des bases de données distantes en univers ouvert

. Problème des utilisateurs qui ne peuvent rester mono-fournisseur

"fédération des bases de données" au moyen d'interfaces standards et de protocoles d'échanges de requêtes standards.

Une passerelle SQL ouverte devrait:

- Accepter des requêtes SQL dans une interface normalisée.

- Les transformer en un format d'échange (FAP encodage et protocole) commun.

- Ce format devrait être supporté par les différents serveurs de bases de données.

- La passerelle devrait être capable de localiser les serveurs et d'offrir des services d'accès aux catalogues.

Les difficultés de l'interopérabilité

- Différences entre API SQL.

- Différences entre les pilotes.

- Différences entre les protocoles FAP.

- Multiplicité des approches en matière d'administration des bases de données.

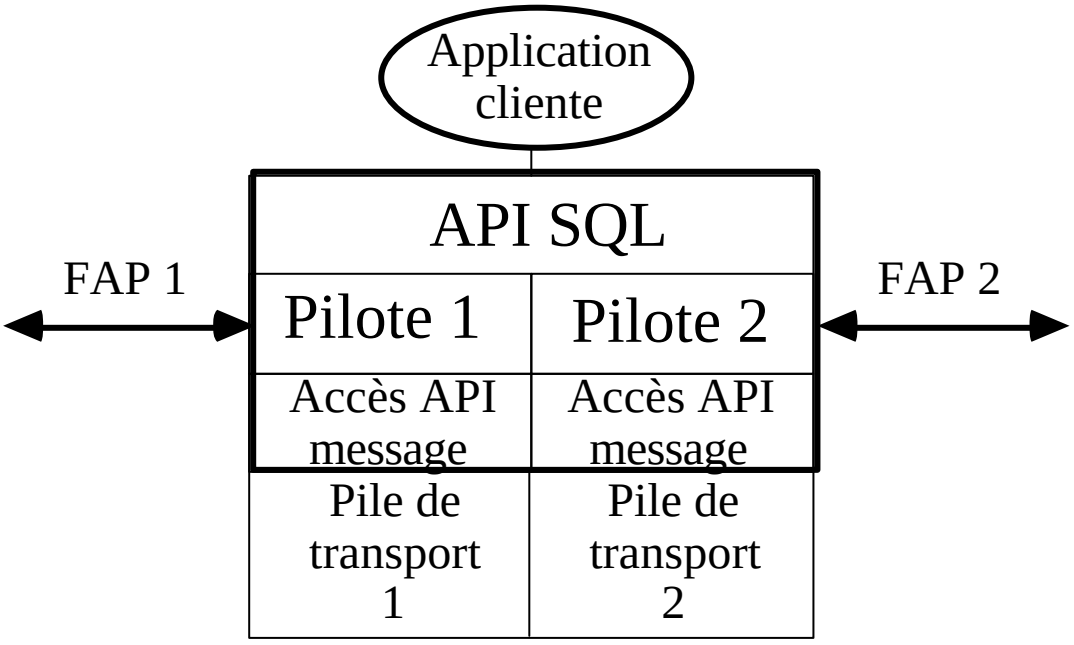
3.1 Normalisation du service (interface SQL commune)

Choix d'un dialecte SQL unique.

Développement de pilotes pour différents progiciels de bases de données.

Support de l'ensemble des pilotes et des piles de communication sur les stations clientes.

L'administration des données et les serveurs de bases de données sont spécifiques comme précédemment.



De nombreuses candidatures au statut d'API SQL commune

Les "normes"

ESQL "Embedded SQL" de l'OSI

CLI "Call Level Interface" du SAG "Sql Access Group".

CLI de l'X/Open .

Les standards propriétaires

ODBC "Open Data Base Connectivity" de Microsoft.

EDA/SQL "Enterprise Data Access" de IBI "Information Builders Incorporated".

ESQL "Embedded SQL"

Une API intégrée langage

- Standard OSI pour l'incorporation d'instructions SQL dans les langages évolués (apparu dans la norme SQL92)

ADA, Pascal, C, Cobol, Fortran, PL1.

- Les instructions sont incorporées au moyen de directives de séparation (genre exec sql) et doivent être traitées par un pré-compilateur.

- Le pré compilateur génère du code pour un serveur de base de données précis de sorte qu'il faut recompiler pour tout changement de serveur.

Exemple : IBM

L'API CLI du SAG "Call Level Interface" du SAG "Sql Access Group"

. Le SAG (consortium de plusieurs dizaines d'éditeurs de bases de données) emmenés par Digital et Tandem.

. Une API "système" pour l'accès indépendant du fournisseur aux serveurs offrant la conformité au standard SQL 89.

. Définie par un ensemble de 23 primitives.

- 3 appels de connexion au serveur
- 5 appels de préparation de requêtes
- 2 appels d'exécution de requêtes
- 7 appels de récupération de résultats
- 3 appels de terminaison de commande
- 3 appels de terminaison de session.

. De très nombreux utilisateurs moyennant une variété importante d'extensions.

SQL/CLI de l'X/Open et OSI

X/Open gère maintenant la CLI SAG.

X/Open a défini une version étendue de la CLI SAG normalisée OSI 9075-3.

31 primitives.

AllocHandle	BindCol
BindParam	Cancel
CloseCursor	Connect
CopyDesc	DescribeCol
Disconnect	EndTran
ExecDirect	Execute
Fetch	FreeHandle
GetCol	GetCursorName
GetConnectAttr	GetDescField
GetDescRec	GetDiagField
GetDiagRec	GetEnvAttr
GetStmtAttr	NumResultCols
Prepare	ReleaseEnv
SetCursorName	SetDescField
SetDescRec	SetEnvAttr
SetStmtAttr	

L'API ODBC de microsoft "Open DataBase Connectivity"

. Une interface définie par extensions à partir la CLI du SAG (depuis 1992).

. Ensuite de très nombreux rattachements à ce standard de facto. Existence sur les principaux OS-PC de pilotes pour un grand nombre de fournisseurs de bases de données.

. Mais standard difficile à suivre en raison de plusieurs évolutions successives avec des variations importantes (V1, V2, V3 ..).

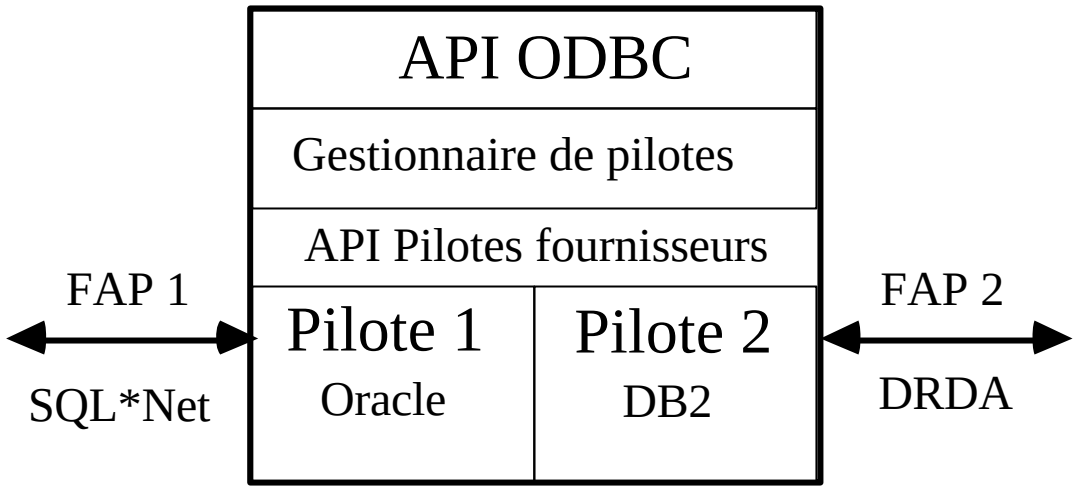
Exemple ODBC 2.0

. Noyau: les 23 primitives du SAG.

. Niveau 1: 19 appels pour les accès aux catalogues des bases de données, les gros objets et des fonctions spécifiques des pilotes.

. Niveau 2: 19 appels supplémentaires pour des accès utilitaires.

Organisation générale de ODBC



- Couche de réception des requêtes ODBC.

- Couche de gestion des pilotes (à quel pilote passer une requête?)

- Selon un format standard pour les pilotes ODBC.

- Chaque pilote de bases de données encapsule les requêtes dans le format propre à chaque fournisseur de bases de données.

Difficultés de ODBC

- Inexistence d'un standard stable.

- La plupart des éditeurs de bases de données fournissent des ODBC qui supportent leurs propres extensions (sauf les fournisseurs qui s'intéressent uniquement à ODBC: Visigenic, Intersolv, ...).

- Incertitude sur l'approche finalement retenue par microsoft relativement à l'accès distant aux bases de données.

Probablement intégrée à OLE-DCOM? orientée objet?

- Un standard fonctionnant en milieu ouvert est toujours plus lent qu'un logiciel spécialisé accédant une seule base:

=> Reproche multiples de mauvaises performances ... un peu améliorées avec le temps.

3.2 Normalisation du protocole de communication (FAP)

Approche classique de la normalisation: définir l'interface du service (SQL CLI) et le protocole de communication (FAP).

Différentes tentatives:

Les normes

- RDA "Remote Database Access"
de l'ISO et du SAG

Les standards propriétaires

-DRDA "Distributed Relational Database"
de IBM

- EDA/SQL "Enterprise Data Access"
de IBI "Information Builder Incorporated"

RDA ("Remote Database Access") de l'ISO/SAG ISO/IEC 9579-1 et 9579-2

Historique

- Travaux anciens sur l'interopérabilité dans les bases de données a l'OSI.
- La première version RDA SQL date de 1993.
- Processus de normalisation encore en cours.

Objectifs

- RDA définit essentiellement un protocole de transport des requêtes SQL entre un client RDA et un serveur RDA (pour des requêtes de SQL89 et de SQL 92).

- RDA cherche à définir également une API de service compatible SQL 92.

- Destiné à fonctionner sur la pile OSI, il est également portable comme tout protocole d'application OSI sur internet TCP/IP.

Quelques caractéristiques RDA

- Pas de validation à deux phases.

- RDA utilise le format de présentation ASN1 ("Abstract Syntax Notation one") BER ("Basic Encoding Rule")

=> Les données sont encodées et décodées systématiquement (utile en univers ouvert mais coûteux pour des systèmes identiques).

=> Chaque valeur est typée de sorte que pour une grosse table on a un encombrement important.

Des améliorations sont à l'étude pour la optimisation des règles d'encodage pour des tables avec de nombreuses lignes

=> Utilisation d'un format "Packed Encoding Rules" au lieu de BER.

RDA est à ce jour très peu implanté par les fournisseurs.

DRDA "Distributed Relational Database" de IBM

. A l'origine le format utilisé par DB2 et qui prouve son fonctionnement interopérable sous de nombreuses API (ESQL, CLI SAG, CLI Xopen, ODBC).

. Poussé par IBM pour devenir un standard de facto pour l'interopérabilité des bases de données relationnelles (licence et codes cédés à bas prix).

. Repris comme moyen d'accès possible par une dizaine au moins de serveurs de bases de données (Oracle, Sybase, ...).

Quatre points

- Accès à une seule base de données distante "Remote Unit of Work"
- Accès à plusieurs bases de données distantes avec possibilité de validation à deux phases.
- Possibilité d'utiliser des procédures stockées "Stored Procedures"
- Sécurité DCE (identification des usagers et authentification des requêtes selon l'approche

DCE Kerberos)

EDA/SQL de IBI

"Enterprise Data Access"
de "Information Builder Incorporated"

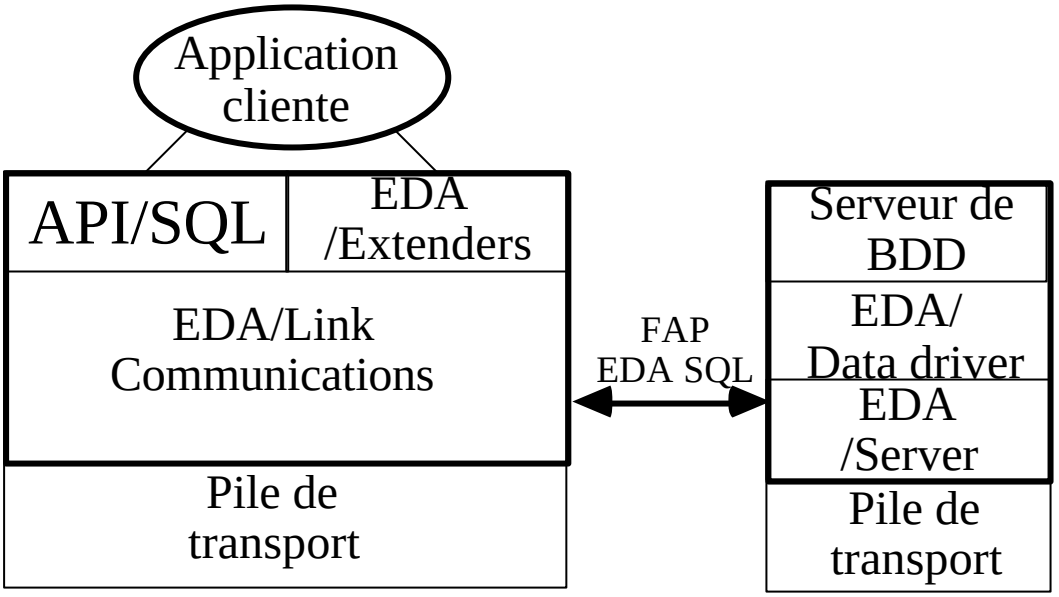
- Un produit constructeur client-serveur développé depuis très longtemps dans le but de faire interopérer des bases de données relationnelles (IBI ne fabrique pas de système de bases de données).

- Le protocole de présentation FAP de EDA-SQL possède des pilotes pour environ 50 serveurs de bases de données différents.

- Les requêtes sont transmises telles que si elles sont simples ou précompilées en requêtes simples (peu de travail coté client).

- Le pilote est plutôt placé du coté du serveur de bases de données.

Architecture EDA SQL



API/SQL est l'interface de EDA/SQL (CLI à vocation commune fondée sur SQL 89).

EDA/Extenders supporte de multiples formes d'extension pour d'autres API base de données

EDA/Link est l'interface de communication pour une douzaine d'API de communication.

EDA/Server est un pilote qui convertit les requêtes au format de la base de donnée cible.

EDA/data driver convertit les données proprement dites (couche présentation).

Conclusion accès bases de données distantes

- La transmission à distance des requêtes d'accès à des bases de données pourrait être un problème résolu de la couche application:

. Choix d'un service (dérivé de SQL)
. Choix d'un protocole

- Le problème est extrêmement complexifié et rendu très opaque par le jeu des éditeurs de bases de données ou d'interfaces clients-serveurs.

. Les commandes de bases (déclaratives) sont enrichies de multiples fonctionnalités.
. En particulier les procédures stockées et les déclencheurs mêlent des RPC synchrones ou asynchrones dans un protocole de mode message.
. Les propositions propriétaires non standardisées sont trop nombreuses.

Nombreux problèmes à résoudre pour une homogénéisation des approches et peu de volonté d'y arriver

Perspectives

- Renouvellement important du problème d'accès aux données avec l'arrivée des bases de données orientées objets.

Langages **SQL3 et OQL** ("Object Query Language" de l'ODMG) successeurs de SQL.

Une API peut-elle émerger et s'imposer?

- Renouvellement important du problème de transmission des requêtes en relation avec l'arrivée de standards systèmes d'objets répartis et langages associés:

CORBA, DCOM/ACTIVEX, JAVA

Peut-il émerger un protocole (sinon une approche de communication style RPC objet) qui s'impose?

Peu probable à court ou moyen terme

Bibliographie
Accès bases de données distantes

R. Orfali, D. Harkey, J Edwards "**Guide de survie client serveur**" Wiley deuxième édition

Norme OSI 9075-3 "**SQL Call Level Interface (SQL/CLI)**"

INTRODUCTION

AUX COLLECTICIELS

"GROUPWARE AND WORKFLOW"

G Florin

Plan du chapitre collecticiels

1 Introduction

2 Aspects données: gestion électronique de documents

2.1 Numérisation des archives papier.

2.2 La gestion électronique de documents composites multimédias

3 Aspect contrôle: planification

4 Conclusion

1 Introduction

"S'il existait un concours de la définition la plus floue dans la catégorie client-serveur le groupware gagnerait haut la main"
Client/Serveur: Guide de survie Wiley

Une définition orientée "groupware"

L'ensemble des outils informatiques permettant à des groupes de personnes de travailler collectivement de la façon la plus simple possible en maximisant l'efficacité.

Des outils pour le travail coopératif des personnes sans insister sur l'organisation.

Une définition orientée "workflow"

L'automatisation des procédures ou des applications informatiques des tâches manuelles sont effectuées par des participants qui appliquent un ensemble de règles pour réaliser une action globale dans l'entreprise.

Travail coopératif utilisant divers moyens dans

le cadre d'une planification prédéfinie.

La terminologie en anglais et en français

Groupware: Collecticiel.
Travail de groupe,
Informatique de groupe.

Workflow : Planification (des tâches).
Ordonnancement, "Fluxgiciel"

Workflow management:
Gestion de la planification

CSCW
"Computer Supported
Cooperative Work"

TCAO
"Travail Collectif Assisté
par Ordinateur"

Un domaine d'application de l'informatique répartie

Problème du collecticiel posé dans un cadre réparti (client-serveur)

- Chaque personne utilise un ou plusieurs postes de travail.

- Le groupe met en place des ressources informatiques collectives pour la réalisation automatisée de tâches.

Développement de nombreux progiciels intégrés répondant à ces besoins.

Exemples:

"Notes" Lotus/IBM,
"Collabra Share" Netscape,
"Exchange" Microsoft,
"Groupwise" Novell.

Un domaine d'intérêt pour les théories et sciences des organisations

Un champ d'application pour une approche organisationnelle:

1) Accélérer des procédures administratives souvent encore à support manuel sur papier (en faisant pénétrer les possibilités réseaux).

=> Améliorer la productivité par l'a pénétration des applications réparties

2) A l'occasion bousculer des procédures administratives établies et repenser globalement l'organisation du travail ...

BPR "Business Process Re-engineering"

Une définition souvent donnée du travail de groupe ("groupware")

Réunion de cinq technologies de base

. **La GED gestion électronique de documents.**

Composites
Multimédia.

. **La planification des tâches**

. **La planification des activités des personnes et la gestion des agendas.**

. **La gestion des conférences et l'accès partagé aux documents.**

. **Le courrier électronique.**

Une classification données/contrôle pour l'informatique de groupe

- Partage/communication des données

Rassemble plutôt les outils qui se préoccupent des données (textuelles, son, images, vidéo...)

Stockage et partage de l'accès (bases de données)

Communication (messageries).

- Spécification du contrôle

Planification des tâches ("workflow")

Rassemble les outils qui se préoccupent du contrôle (de l'ordonnancement des tâches pour réaliser une action complexe)

- planification des circulations d'objets (de documents).

- planification des actions des personnes

Organisation de réunions

=> tenue des agendas.

Planification des projets, négociation et distribution des tâches, rappel d'échéances.

Une classification: la matrice moment/lieu (Johansen)

Classifier différents types de relation de communication entre personnes selon deux critères.

Temps

Caractère synchrone ou asynchrone de la relation

Synchrone	Même moment.
Asynchrone	Moments différents.

Espace

Localisation des personnes

- Même lieu
- Lieux différents

Z Point de vue adopté: la localisation physique des personnes.

Remarque: La localisation électronique de l'application ou des documents avec lesquels la personne dialogue n'est pas considérée car très difficile à définir en approche répartie.

Quelques exemples d'applications selon la classification moment/lieu

Même moment / Lieux différents

- Téléphone/ conférences téléphonique
- Rédaction coopérative
- Partage de fenêtres (tableau blanc)
- Visio conférence
- Accès partagé aux données.

Moment différent/ Lieux différents

- Messagerie textuelles
- Messageries vocales
- Accès non partagé aux données.

Même moment / Même lieu

- Réunion de personnes => planification des réunions/gestion des agendas.

Moments différents / Même lieu

- Partage de ressources matérielles entre des personnes (imprimantes, bureau, ...)

=> réservation des ressources

Historique

Recherches sur l'interaction homme machine depuis les années 1960 (Stanford Research Institute).

Techniques d'ordonnancement des tâches.

Recherches sur les applications réseaux et informatiques réparties (exemple protocole RJE "Remote Job Execution" exécution de tâches à distance).

Apparition du terme "Groupware" 1981

Logiciel "Lotus notes" 1989

Congrès Groupware 1992 San José

2

Gestion électronique

des documents

2.1 Numérisation des archives de documents papiers

- **Numérisation** d'un document papier.

- **Vérification de la qualité** obtenue.

- **Association et saisie d'attributs** numériques permettant la recherche et l'accès ultérieur au document (type, identification, date, ...)

- **Compression et sauvegarde** dans une base de données.
  => Utilisation de la notion SQL de blob "Binary Large OBjects" (controversée)
  => Meilleure solution en traitant directement les objets multimédia.

- **Exploitation:** Accès aux documents, visualisation, impression, transmission réseau.

2.2 La gestion des documents/objets composites multimédias

Un document est généralement construit à partir de multiples documents

Par recopie et agrégation, création d'une relation de contenance entre objets.

document imbriqué ou **composite**,

Par **référence** à d'autres documents (lien, référence d'objet).

Les documents assemblés ou reliés peuvent appartenir à de multiples types de données, avec différents codages issus de multiples outils:

textes, dessins, images, sons, vidéo.

L'outil de gestion doit offrir

- des possibilités **d'archivage et de restitution** sur des critères variés dépendant du contenu.
- des possibilités de **modification** par les outils associés aux divers types de documents.

OpenDoc

Produit par le consortium CIL d'éditeurs de logiciels ("Component Integration Labs") en particulier IBM (avec SOM) et Apple (avec Bento), Adobe, Lotus, Novell, OMG ...

Implantation de documents composites avec des fonctionnalités diverses

- Partager l'affichage dans une fenêtre.

- Stocker plusieurs documents dans un même fichier conteneur.

- Échanger des documents avec d'autres applications par le biais de couper/coller, glisser déposer, presse-papier.

- Coordonner les actions locales sur les objets au moyen de scripts (architecture OSA "Open scripting Architecture")

- Réaliser les actions à distance par Corba.

Ole
"Object Linking and Embedding"

- **Le modèle d'objet composite** de Microsoft

- Offre un environnement complet de **composants** en local au moyen d'interfaces définissant un contrat entre composants.

- En 1996 microsoft définit environ 100 interfaces pour réaliser un ensemble de fonctions avec les mêmes objectifs que Opendoc en local.

- En distribué DCOM "Distributed Component Object Model" veut rivaliser avec CORBA

Approche supportée par la puissance de microsoft mais plus ancienne aussi bien en approche objet qu'en distribution que OpenDoc Corba.

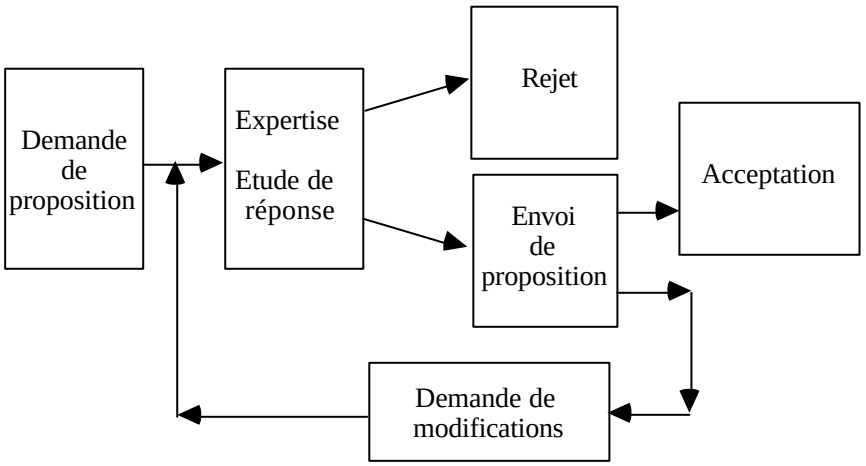
3

Planification "Workflow"

Introduction

- Modélisation des actions successives (manuelles ou automatiques) pour réaliser une opération complexe (ordonnancement des tâches).
- Associée à une circulation de documents papiers ou électroniques dans l'entreprise.
- Assez souvent linéaire mais dans sa généralité:
    - existence de fonctionnements en boucle
    - existence d'actions parallélisées
    - existence de fusions de résultats
    - existence de traitements des exceptions

Exemple du début d'une négociation de contrat



Modèles de planification

. Fonctions à réaliser

Modélisation graphique ou langage
Outils graphiques pour concevoir
la logique de cheminement

Invocation d'opérations
par l'utilisation des outils de communication
Activation de processus par message.
Activation d'objets en RPC.
Envoi de courrier électronique.
Émission de bons de travail.

Suivi d'un travail.
Avancement courant.
Identification des blocages.

Le consortium WFMC "The WorkFlow Management Coalition"

- WFMC est un consortium des fournisseurs majeurs de logiciels de planification pour promouvoir :

Un langage commun

Définition d'une terminologie (en anglais et en français)

Une API client WAPI "Workflow Application Programming Interface/Interchange"

Pour invoquer des opérations sur des moteurs d'exécution

L'interopérabilité entre des moteurs d'exécution. "Workflow Interoperability"

Pour que des moteurs d'exécution puissent communiquer et exécuter de manière coordonnée des processus.

Le modèle de référence de WFMC: principales notions (1)

- Un **processus** (quelquefois appelé procédure, flux de tâche) est un ensemble coordonné d'actions reliées **en série** ou **en parallèle** dans le but d'atteindre un objectif prédéfini.

Exemples:

Procédure d'atterrissage d'avion.
Processus de remboursement de note de frais.
Procédure de déclaration de sinistre.

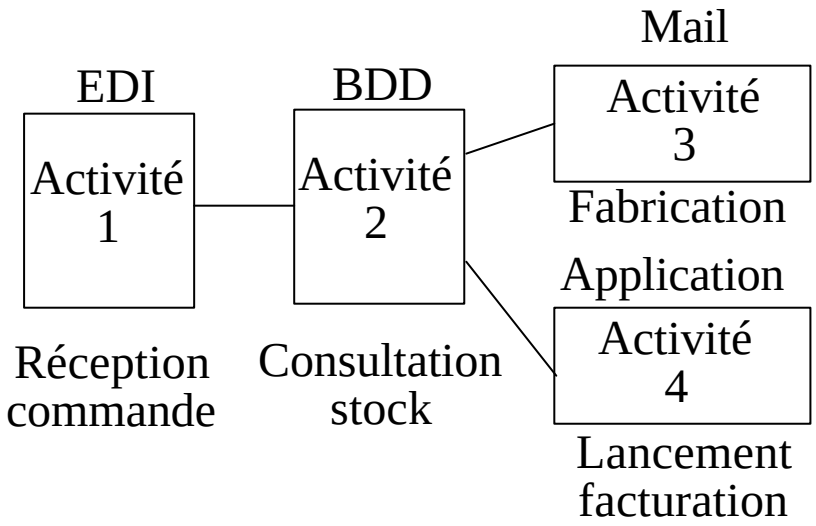
- Un gestionnaire de flux de tâche (moteur d'exécution, "process engine") définit, gère et exécute des processus en **exécutant des tâches dont l'ordre d'exécution est prédéfini** dans une représentation informatique de la logique du processus.

- Une **activité** est l'une des étapes élémentaire d'un processus. Elle peut-être manuelle (emballage, ouverture de courrier, saisie) ou exécutée automatiquement par un outil (lecture optique de document, accès base de données).

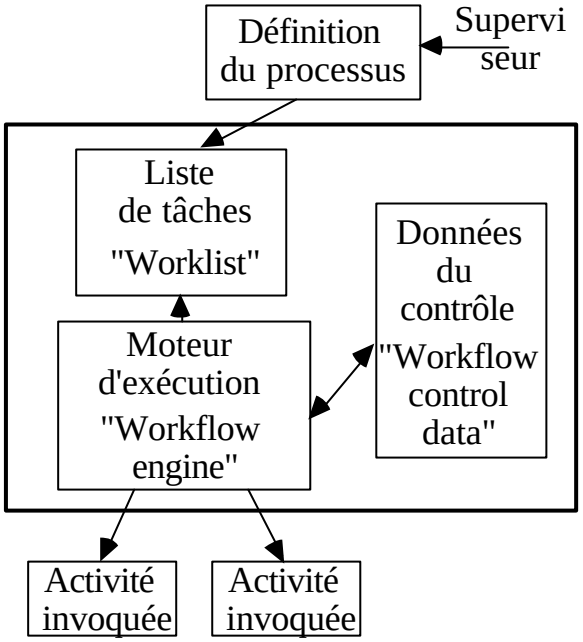
Le modèle de référence de WFMC: principales notions (2)

- Une **instance** de processus ou d'activité (un cas d'activité) est une exécution particulière.

- **Un bon de travail** ("work item") est la représentation informatique d'un travail à effectuer dans une instance d'activité.



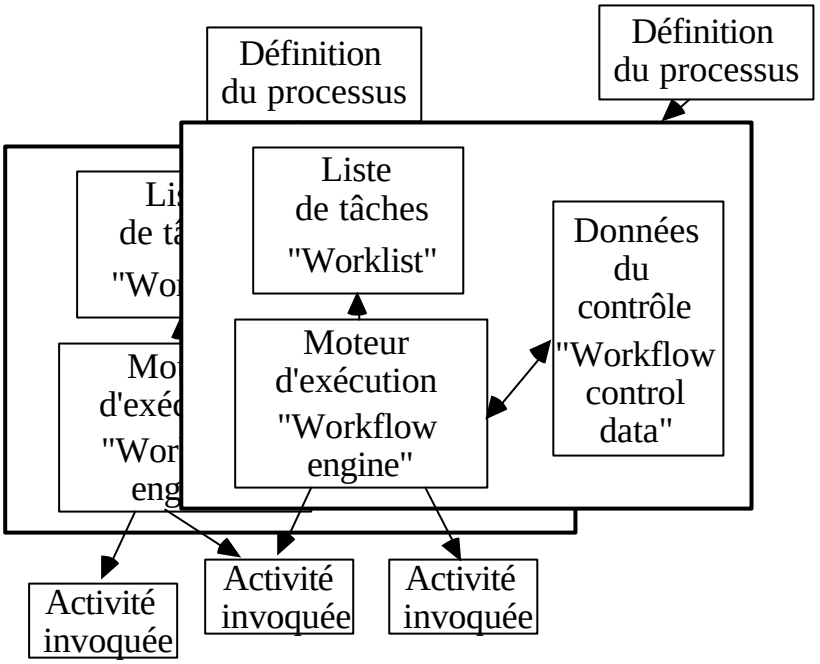
Planification centralisée



. Solution classique d'ordonnancement ou on automatise la planification des tâches à partir d'un seul système informatique:

- un serveur central: un site coordinateur qui détermine ce qui doit être lancé à partir de la connaissance des plans de travail et de l'état global.

Planification distribuée



. On automatise la planification des tâches en univers réparti:

- Chaque site détermine sa propre activité à partir de la connaissance de son plan de travail et de la connaissance totale ou partielle de l'état global.

- Les différents acteurs de la planification coopèrent à la réalisation d'un but.

Notion de système multi-agent.

L'API client du modèle WFMC

Fonctions de connexion (2 primitives)

Connexion, deconnexion a un moteur

Contrôle et statut d'un processus (23 primitives)

Définition d'un processus
Création d'une instance de processus
Lancement d'un processus
Lecture et mise à jour du statut courant

Contrôle et statut d'une activité (13 primitives)

Définition d'une activité
Lecture et mise à jour des attributs

Administration d'une liste de tâches (18 primitives)

Ouverture/fermeture d'une liste
Lecture et mise à jour des attributs.

Modélisation des plans

Itinéraire ("route")

La suite des activités réalisées lors de l'exécution d'un plan dans un cas précis.

Sensibilisation ("transition condition")

La règle définissant dans un état donné les activités exécutées.

Aiguillage ("OR-split)

Un itinéraire s'ouvre sur plusieurs itinéraires possibles, on en exécute l'une ou l'autre.

Jonction ("Or Join")

Jonctions d'itinéraires alternatifs.

Branchement multiple ("AND-split)

Séparation d'un itinéraire en plusieurs itinéraires différents activés en parallèle.

Rendez-vous ("AND join")

Plusieurs activités parallèles convergent vers un itinéraire unique

Itération

Répétition jusqu'à satisfaction de condition.

Un exemple: Lotus Notes

Réunion des fonctions suivantes sur de nombreuses piles de communication

- Un service de **bases de données** avec **réplication** de documents **multi-médias**.

- Un service de **courrier électronique**.

- Un environnement **client à interface graphique**.

- Des services **d'administration, de nommage et de sécurité** répartis.

- Des outils de développement d'applications notes (orientées objet): des **scripts** utilisables pour **automatiser des tâches** répétitives.

- L'implication de nombreux **fournisseurs** dans des produits **annexes** s'interfacant avec notes.

Conclusion

- Passage des différents acteurs du collecticiel à l'internet (navigateurs, présentateurs de documents, courriers, désignation et sécurité)

- Tendance à l'intégration des domaines connexes (bases de données, transactionnel, téléphonie, ...).

Le collecticiel a en fait pour projet de traiter de tout ce qui touche aux applications réparties d'entreprise.

Il pourrait peut-être s'il parvient à maturité fédérer dans des progiciels intégrés des ensembles de plus en plus large d'outils du client-serveur.

Bibliographie

David. Hollingsworth "**The workflow reference model**" The workflow management coalition specification 1994
http://www.aiai.ed.ac.uk:80/project/wfmc/

R. Orfali, D. Harkey, J Edwards "**Guide de survie client serveur**" Wiley deuxième édition