Systèmes de fichiers répartis

Gérard Florin
CNAM/CEDRIC

Plan du cours

- Introduction – Rappels systèmes de gestion de fichiers (‘File Systems’)
- Systèmes de fichiers répartis (‘Distributed File Systems’)
- Exemple de NFS ‘Network File System’
- Exemple de FTP ‘File Transfer Protocol’
- Conclusion

Introduction
Rappels systèmes de gestion de fichiers

Rappels : Notion de fichier

- **Fichier = Abstraction de mémoire permanente** : une séquence de données similaires non interprétées (des articles) souvent de taille un octet
- **Accès séquentiel :** on lit ou écrit des sous séquences d'items sur un pointeur courant.
- **Accès aléatoire :** accès direct dans le fichier sur une adresse calculée (par exemple en fonction d'une clé hachée).
- **Accès indexé :** utilisation d'un index accédé sur une clé pour obtenir une adresse dans le fichier.

Notion de système de fichiers

- **Système de fichiers :** logiciel dédié à
  - **La création et la destruction de fichiers**
  - **L'enregistrement d'articles dans un fichier**
  - **La restitution d'articles**
  - **Le nommage**
  - **Le partage**
  - **La protection**
- **Notion de répertoire :** un fichier d'un type spécial qui permet d'associer à un nom de fichier un identifiant de représentation interne.

Réalisation d'un système de fichiers : Organisation en couches en Unix

- **Module de gestion des répertoires** : relie les noms logiques de fichiers aux identifiants internes
- **Module de fichiers** : relie les identifiants internes aux fichiers proprement dits.
- **Module de contrôle d'accès aux fichiers** : vérifie les droits d'accès.
- **Module d'accès aux fichiers** : lecture et écriture des articles.
- **Module de gestion de l'espace physique** : alloue et désalloue les blocs physiques sur disque.
- **Module de gestion disque** : entrées sorties disque, gestion des tampons.

Désignation dans les fichiers centralisés Unix

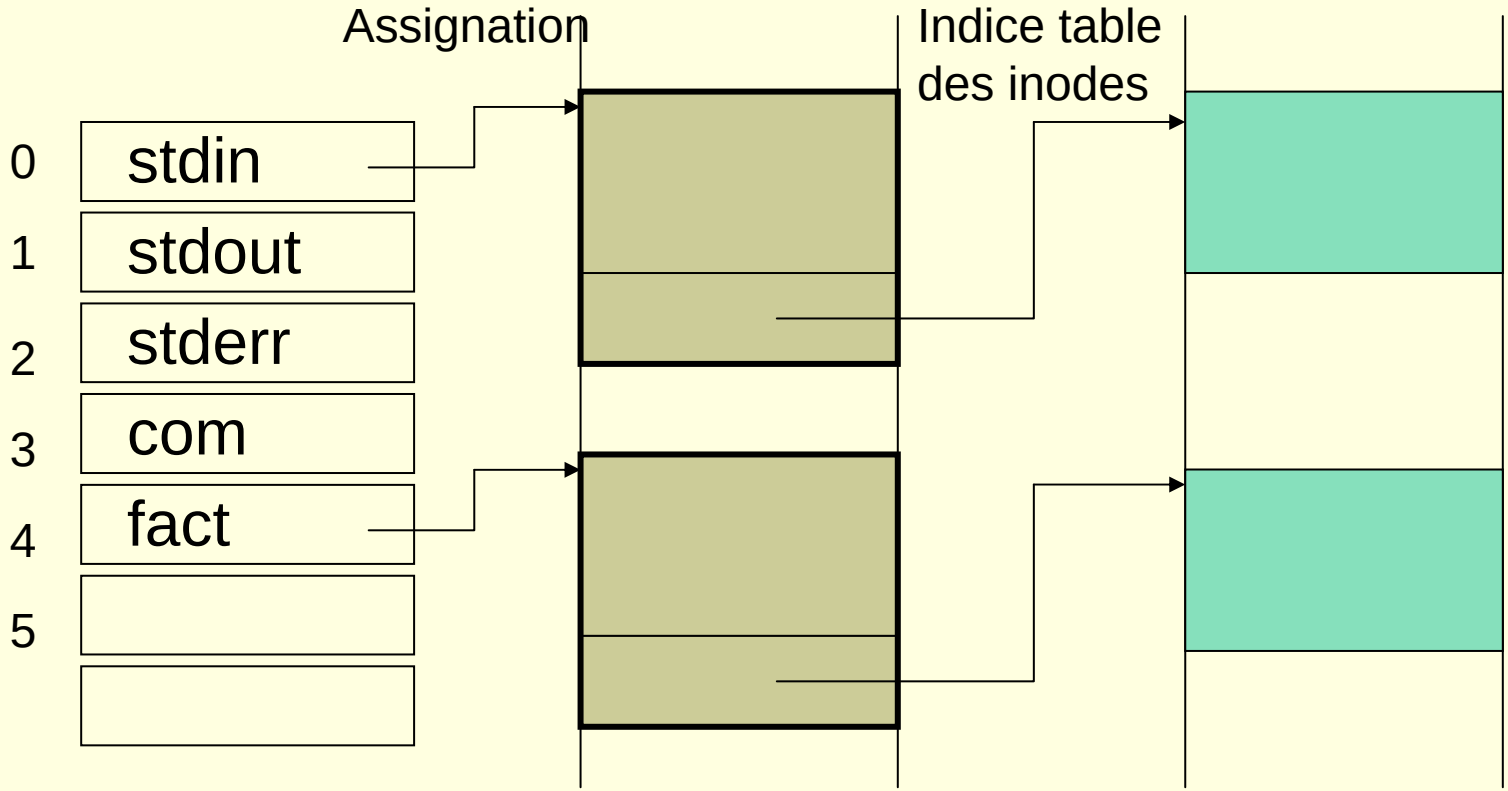


Table des noms logiques utilisés par un programme

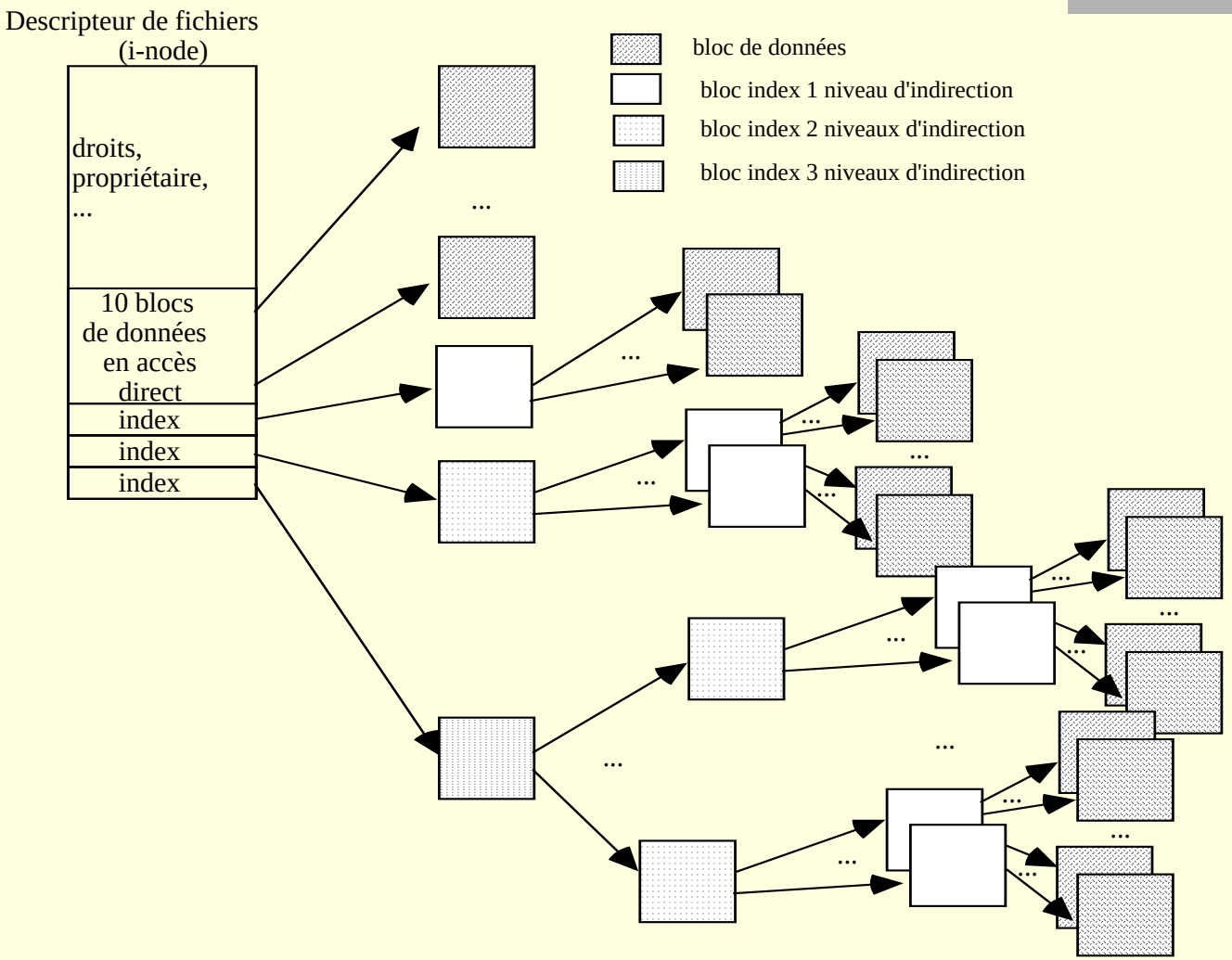
Table des noms de fichiers utilisés

Table des inodes (descriptifs physique des fichiers)

Description physique des fichiers Unix: Informations dans les Inodes

- **Type du fichier** : ordinaire, répertoire
- **Protection** : droits d'utilisation (rwx).
- **Nombre de liens** : pour détruire finalement le fichier.
- **Propriétaire du fichier** : UID User identifier.
- **Groupe du fichier** : GID Group Identifier.
- **Taille du fichier** : en octets.
- **Adresses physiques directes de blocs** : 10 blocs.
- **Adresses physiques indirectes** : 3 blocs.
- **Date/heure** : dernier accès.
- **Date/heure** : dernière modification.
- **Date/heure** : dernière modification inode.

Organisation sur disque d'un fichier Unix

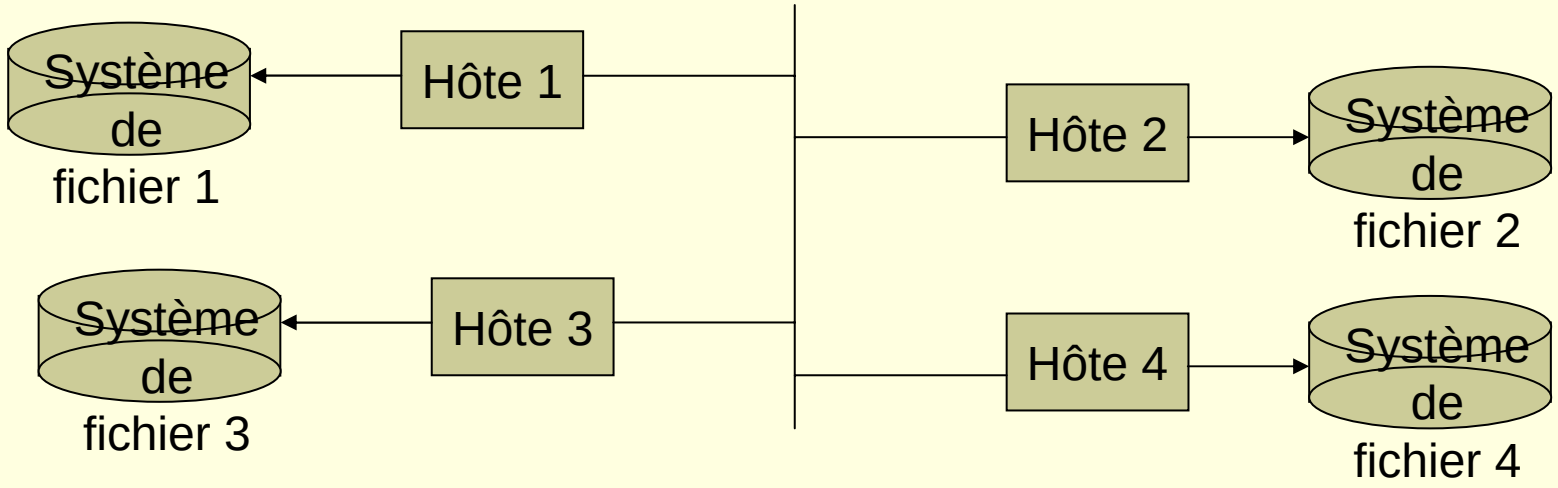


Systèmes de fichiers centralisés : La gestion des répertoires

- **Service de gestion des répertoires**
  - **Défini par un ensemble d'opérations sur des fichiers d'un type particulier répertoire**: créer, détruire.
  - **Manipulation de fichiers dans les répertoires**: créer, détruire un fichier, renommer des fichiers, déplacer des fichiers d'un répertoire à un autre.
- **Notion de système de fichier hiérarchisé : gestion hiérarchisée des répertoires**
  - Création de liens entre répertoires sous répertoires au moyen du système de **désignation.**
- **Création de liens 'symboliques':** dans certains systèmes de fichiers
  - Plusieurs noms logiques peuvent être associés au même descripteur physique (liens multiples sur le même inode en Unix) .

Systèmes de gestion de fichiers répartis (SGFR) (‘Distributed File Systems’ DFS)

Systèmes de fichiers répartis



- **Situation :** différents systèmes de fichiers attachés à différents hôtes d'un système réparti.
- **Objectif : Réaliser la transparence** des systèmes de fichiers à la répartition.
- **Autre terminologie** : **NAS Network Attached Storage (ex NFS).**

  Organisation différente: SAN 'Storage Area Network' un réseau haut débit (exemple Fibre Channel) permet des accès au niveau des blocs physiques à des disques attaché au réseau.

Systèmes de fichiers répartis : Principaux objectifs (1)

Résoudre les problèmes habituels posés par la répartition.

- **Problème de désignation (nommage, ‘naming’).**
  - Construire une désignation (un nommage) uniforme des fichiers cohérent avec les différents systèmes.
- **Problème de localisation.**
  - Retrouver dans le systèmes réparti quel hôte gère un fichier.
- **Problème d’accès distant.**
  - Accéder effectivement aux fichiers sur leur site de résidence.
- **Problème de concurrence.**
  - Gérer les accès concurrents en univers réparti.
- **Problème des pannes.**
  - Panne des serveurs de fichiers, panne du réseau ->partition du réseau.

Systèmes de fichiers répartis : Principaux objectifs (2)

- **Problème des performances.**
  - Assurer des performances acceptables (similaire à l'univers centralisé).
- **Problème de l'hétérogénéité**
  - Des processeurs, des systèmes d'exploitation.
- **Problème d'extensibilité ('Scalability').**
  - Fonctionnement dans de grandes configurations.
  - Passage à l'échelle
- **Problème de migration.**
  - Déplacement des fichiers d'un système à l'autre
- **Problème de réplication.**
  - Existence de copies multiples des fichiers

Utilisation des systèmes de fichiers répartis

Beaucoup de systèmes de fichiers répartis sont construits pour l'usage suivant

- **Les fichiers sont petits :** majorité de 10 à 100 Koctets.
- **Les lectures sont beaucoup plus nombreuses que les écritures.**
- **Les accès sont séquentiels :** l'accès aléatoire est rare.
- **La durée de vie d'un fichier est généralement courte.**
- **Le partage de fichiers en écriture est rare.**
- **Un processus n'utilise qu'un petit nombre de fichiers.**

Systèmes de fichiers répartis : Service et protocole

- **Service.**
  - La spécification des services que le système de fichiers répartis offre au client.
  - Comporte le service d'accès aux données: lecture, écriture aléatoire, écriture en fin de fichier ('appending')
  - Comporte le service de gestion des répertoires : création et destruction de fichiers dans des répertoires.
- **Serveurs de fichiers**
  - Un processus distant qui tourne sur une machine distante et implante un service de fichiers.
- **Protocole de fichier réparti**
  - Les échanges de requêtes qui transforment des primitives du service en opérations d'un serveur de fichier distant.

Systèmes de fichiers répartis : quelques critères de classification

- **Communications :** utilisation des messages ou du RPC.
- **Serveurs :** avec ou sans état.
- **Serveurs :** mono ou multi processus
- **Désignation/nommage :** structure de l'espace de noms de fichiers.
- **Sémantique du partage :** "Unix", session, transaction.
- **Caches et duplication :** caches clients , caches serveurs, cohérence des caches
- **Tolérance aux fautes :** duplication, reprise
- **Sécurité :** modèles d'authentification, de protection …

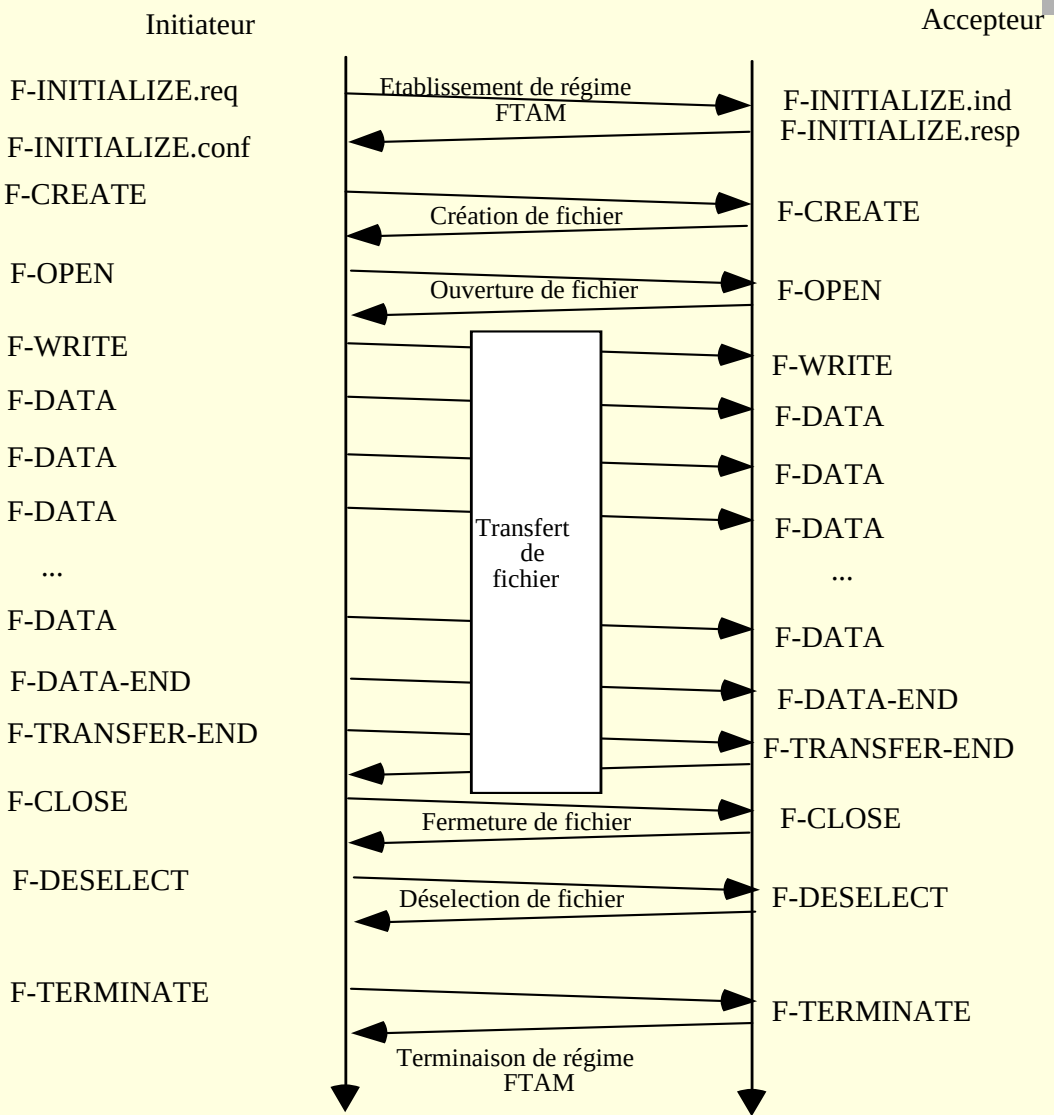
Systèmes de fichiers répartis : Le modèle de transfert de fichiers

- **Service de transfert de fichiers complets :**
  - Mode Upload / Download (pré chargement).
  - Service défini par une opération principale: transférer un fichier (lire à distance et écrire localement un fichier complet).
  - Les accès au fichier sont réalisés ensuite sur le site client.
  - Si nécessaire (après modifications) le fichier est retourné sur son site d'origine (le serveur).
- **Exemple type : FTP 'File Transfer protocol'**
- **Autres exemples** : Kermit, UUCP ('Unix to Unix Communication Program'), FTAM (OSI 'File Transfer Access and Management').

Systèmes de fichiers répartis : Transfert de fichiers : avantages /inconvénients

- **Avantages**
  - Simplicité très grande.
  - Accès au fichier non partagé (sauf sur le site client).
- **Inconvénients**
  - Si le volume est très important et si l'accès ne porte que sur une petite partie du fichier, transférer tout le fichier est inutile.
  - Le fichier occupe de la place sur différents sites.
  - Si les écritures sont concurrentes => partage des modifications très difficile (techniques d'invalidation ou réconciliation des versions)

Systèmes de fichiers répartis : Exemple de primitives pour un transfert FTAM



Systèmes de fichiers répartis : Le modèle d'accès distant

- **Service d'exécution de primitives d'accès fichiers à distance** sur le site serveur: Mode 'Remote Access' (page à la demande).
  - Le service est défini l'ensemble d'opérations sur fichiers (créer, détruire, ouvrir, fermer, lire , écrire, modifier des attributs etc …).
  - Les opérations sont réalisés sur le site serveur (le système de fichier ne s'exécute que sur le site serveur).
- **Exemple type NFS : 'Network File System'**
- **Autres exemples :** AFS ('Andrew File System', Coda, Microsoft DFS ('Distributed File System')

Systèmes de fichiers répartis : Modèle d'accès distant Avantages et inconvénients

- **Avantages**
  - On ne transfère pas tout le fichier: uniquement ce sur quoi on travaille.
  - On n'utilise pas d'espace disque sur le site client.
  - Les accès partagés entre clients sont possibles sur le serveur: une seule copie que l'on peut soumettre à un contrôle de concurrence.
- **Inconvénients**
  - Si de nombreuses opérations sont réalisées sur le même article: beaucoup d'échanges inutiles.
  - Système de fichiers plus complexe à développer que les outils de transfert complet de fichiers.

Systèmes de fichiers répartis : Evolution Historique

- **Années 1970 :** création des protocoles de transfert de fichiers FTP et premiers projets sur les accès distants
- **1974 Newcastle Connection** : espace unifié de fichier, idée de super racine
- **1979 Locus :** système unique de fichiers.
- **1984 NFS :** produit Sun => devenu standard universel.
- **1988 AFS / DFS :** initialement projet de recherche, puis produit.
- **1988 technologie RAID :** 'Redundant Array of Independent Disks')
- **Années 1990 disques en réseau** : SAN, NAS.
- **Années 2000 : FSU 'File Sharing Utilities'** ou **'Peer to Peer'** systèmes "pair à pair" (Napster, FreeNet, Gnutella, Groove etc …)

Désignation : Dépendance ou indépendance des noms à la localisation

- **Deux idées d'indépendance**
  - **Adressage :** le nom d'un fichier (un chemin d'accès) ne comporte aucune indication (directe) sur la localisation du fichier.
  - **Migration :** le nom d'un fichier n'est pas modifié lorsqu'on le déplace d'un support à un autre (on réalloue ses blocs).
- **Exemple de dépendance de la localisation**
  - **Structure des noms :** /serveur/rep1/rep2/rep3/Fich
  - **Le nom d'un fichier comporte celui du serveur** de résidence physique du fichier.
  - **Un fichier ne peut-être déplacé** sans changer de nom.
- **Exemple d'indépendance de la localisation**
  - **Structure des noms :** une arborescence unique globale à tous le système réparti /rep1/rep2/rep3/Fich
  - **Le nom d'un fichier ne donne pas d'informations sur l'emplacement physique du fichier** (l'emplacement est donné par consultation du catalogue).
  - **Un fichier peut-être déplacé** sans changer de nom.

Désignation : Trois catégories principales de systèmes de désignation

- **Nom de machine + chemin d'accès local :** /machine/chemin_local ou machine:chemin_local
  - Les premiers systèmes de fichiers répartis ('Newcastle Connection')
- **Montage :** d'un système de fichier distant sur la hiérarchie locale (extension du montage en local).
  - SUN NFS
- **Construction d'un système de nommage complètement nouveau** : dans lequel tous les fichiers reçoivent un nom universel (qui a la même forme pour tous les serveurs et tous les clients).
  - Systèmes de fichiers : Chorus, Mach, Amoeba, Apollo

Partage de l'accès aux fichiers : Problème du contrôle de concurrence

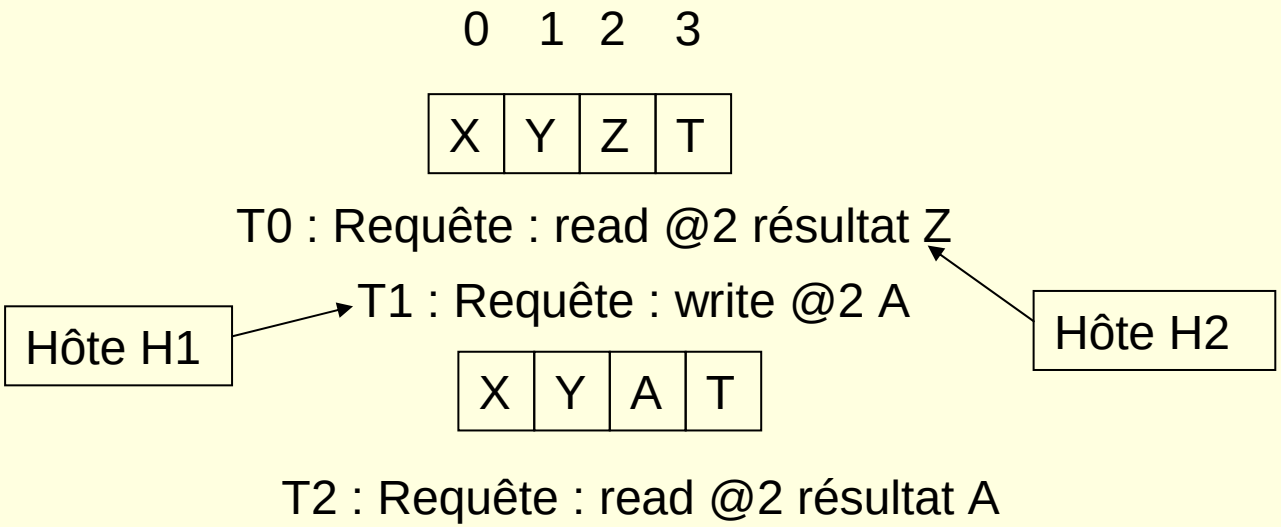
- **Terminologie multiple :** Contrôle de concurrence, de cohérence, de consistance ('Consistency Semantics')
- **Objectif :** définir les entrelacements légaux d'opérations lire et écrire (l'instant ou modifications apportées à un fichier par un usager sont observables par les autres usagers).
- **Indispensable :** pour prédire le comportement d'un programme qui utilise un fichier lorsque ce fichier est partagé.
- **Délimitation des accès :**
  - aucune délimitation,
  - entre une opération d'ouverture et opération de fermeture,
  - des instants quelconques de début et fin de transaction séquence

Partage de l'accès aux fichiers : Les quatre versions

- **On ne présente en général que quatre versions**
- **1) Sémantique UNIX :** chaque opération est instantanément visible pour tous les autres usagers.
- **2) Sémantique de session :** aucune modification n'est visible jusqu'à ce que le fichier soit fermé.
- **3) Fichiers non mutables :** aucune modification n'est autorisée.
- **4) Approche transactionnelle :** toutes les modifications sont visibles par tous ou par personne.

Partage de l'accès aux fichiers : La sémantique Unix : définition

- **Définition :** une écriture est visible immédiatement par tous les autres usagers qui ont ouvert le fichier (s'ils effectuent une lecture ces usagers obtiendront la nouvelle valeur écrite).
- **Autre terminologie :** cohérence/consistance atomique, les requêtes sont traitées en séquence dans l'ordre des dates de leur émission.

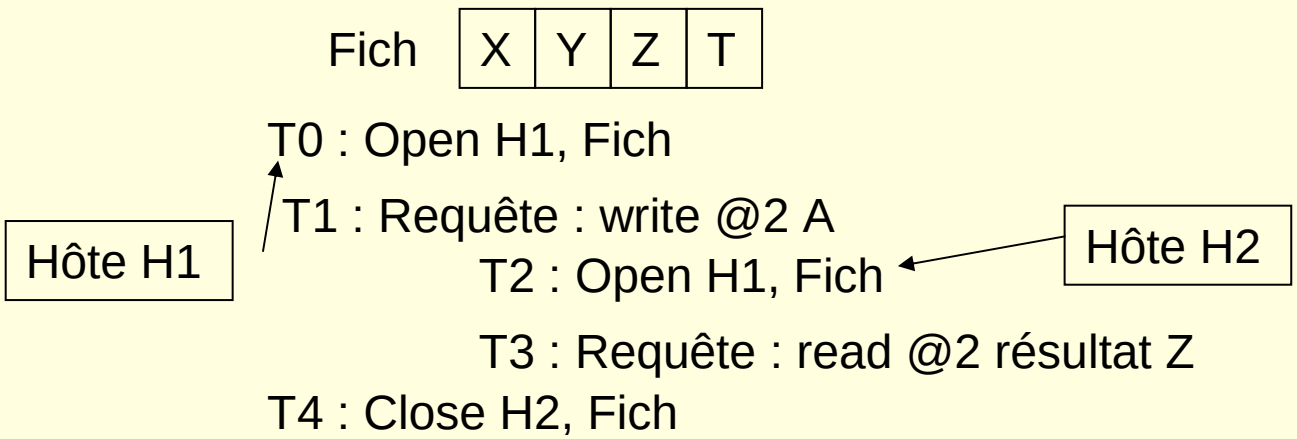


Partage de l'accès aux fichiers : La sémantique Unix : réalisation

- **Sémantique facile** à réaliser en univers centralisé (si les clients utilisent un seul cache).
- **Sémantique plus difficile** en univers réparti
  - **Besoin de nombreux messages :** par exemple si chaque usager utilise un cache lorsqu'une modification est réalisée les copies sont invalidées puis modifiées toutes.
  - **Assez simple en réparti:** si l'on accède à un seul serveur de fichiers distants, que les clients n'ont pas de cache. => Le serveur peut traiter les requêtes en séquence dans l'ordre temporel.

Partage de l'accès aux fichiers : La sémantique de session

- **Définition :**
  - Chaque intervalle entre une ouverture (open) et une fermeture (close) forme une session. Les modifications d'un usager sont visibles par lui.
  - Tous les autres usagers ayant déjà ouvert le fichier ne peuvent voir ces modifications.
  - Lorsqu'un fichier est fermé les modifications deviennent visibles aux autres usagers qui commencent une session après la fermeture.



Partage de l'accès aux fichiers : Réalisation de la sémantique session

- **Solution de base :**
  - Chaque utilisateur travaille sur sa propre copie du fichier (au moins pour ce qui concerne les enregistrements modifiés).
  - Lorsque le fichier est fermé les modifications sont reportées sur la copie du serveur.
- **Problème :** que deviennent les écritures générées en parallèle par plusieurs usagers

Partage de l'accès aux fichiers : Fichiers non mutables 'Immutables' (1)

- **Trois opérations seulement sont prévues :**
  - 1) **Création d'un fichier :**
    - Par écriture complète du fichier à partir de la mémoire centrale ou d'un autre fichier.
    - Des que le fichier est créé (qu'il est déclaré partageable il ne peut plus être modifié par personne).
  - 2) **Destruction du fichier.**
  - 3) **Lecture du fichier.**
  - Une modification de fichier ne peut être obtenue que par recréation d'un autre fichier.

Partage de l'accès aux fichiers : Fichiers non mutables 'Immutables' (2)

- **Solution rare :**
  - Qui correspond néanmoins à un certain profil d'utilisation (de fichiers de petite taille en éditeur de texte).
  - Qui facilite le partage mais le problème des modifications en parallèle subsiste.
- **Problème :** des fichiers de grande taille souvent modifiés => utiliser en parallèle d'autres systèmes de fichiers.
- **Exemple :** système réparti Amoeba.

Partage de l'accès aux fichiers : L'approche transactionnelle

- **Délimitation de transactions :** comportant ouverture, fermeture et accès à des fichiers.
- **Garantie des propriétés transactionnelles 'ACID' :** sur les lectures et écritures aux fichiers.
- **Transactions sérialisées:** tout se passe comme si les transactions (suites d'opérations) avaient été réalisées en séquence.
- **Réalisation :** moniteur transactionnel.
- **Concurrence entre les accès limitée** : si de nombreuses écritures sont effectuées.
- **Tolérance aux pannes :** validation à deux phases.

Serveurs de fichiers : avec ou sans état (‘stateless’ vs ‘Sateteful’)

- Le serveur de fichier conserve ou non des informations relatives aux opérations des différents clients en activité.

Avantages des serveurs sans état

- Pas d’espace mémoire occupé sur le serveur dans des tables.
- Pas de limites sur le nombre de fichiers ouverts.
- Pas de problèmes liés aux pannes des clients.

Avantages des serveurs avec état

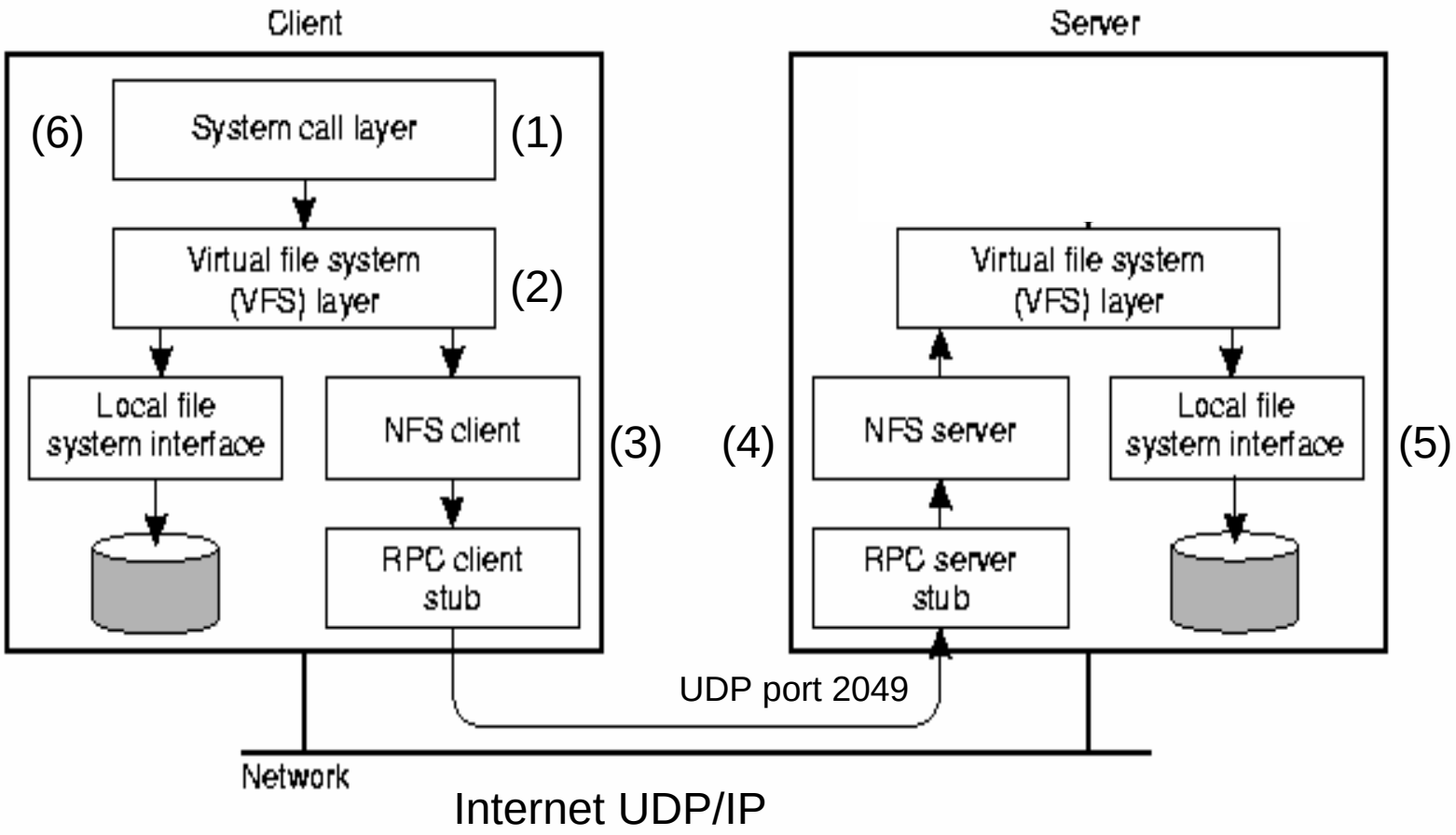
- Messages de requêtes courts (informations à transporter).
- Meilleures performances en temps d’exécution.
- Lecture/écriture en fin de fichier facile.
- Gestion du verrouillage possible.

Le système de fichiers répartis SUN NFS (‘Network File System’)

Système de fichiers répartis NFS
Généralités : Choix de conception

- **Développé par SUN IETF :** RFC 1094 (année 1988)
- **Objectif :** permettre à toute application utilisant des fichiers distants d'accéder à ces fichiers de la même manière que s'ils étaient locaux.
- **Différents choix de conception :**
  - **Réalisation de la désignation** par montage d'arborescences sur l'arborescence locale.
  - **Serveurs de fichiers sans état**.
  - **Pas de modification d'un programme client** c'est le système d'exploitation local qui est modifié.
  - **Pas d'implantation du serveur de fichiers par processus** => modification système coté serveur (performances).
  - **Réalisation des échanges de NFS en RPC** : utilisation de l'architecture ONC SUN avec RPC et présentation XDR.

Généralités NFS : Architecture de NFS



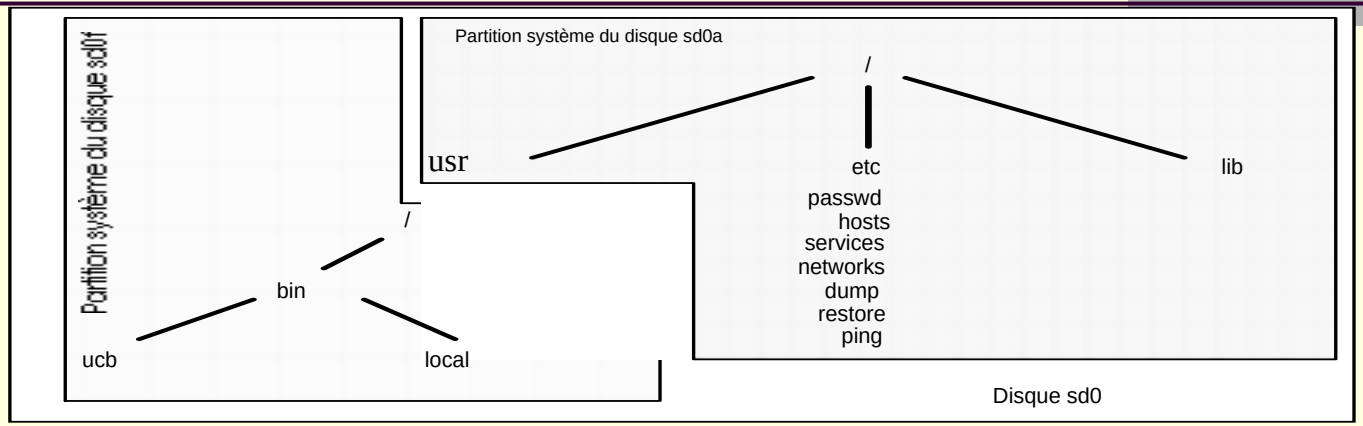
Généralités NFS : Opérations réalisées pour un accès

- **1) Réception** d'un appel système d'accès fichier.
- **2) Sélection** entre traitement local du fichier ou traitement distant : utilisation de la couche VFS 'Virtual file System' qui appelle soit 'local file access' soit 'NFS client box'.
- **3) Préparation** d'un appel RPC par le client NFS pour faire réaliser (utilisation UDP pour l'efficacité en performances).
- **4) Réception** par le serveur généralement sur le port UDP 2049 mais on peut aussi utiliser un port quelconque alloué pour la circonstance (port éphémère).
- **5) Traitement de la requête** par le système de fichier local via la couche VFS serveur.
- **6) Transmission de la réponse** : résultat de la requête qui est délivrée au client.

Généralités NFS : La famille des protocoles NFS

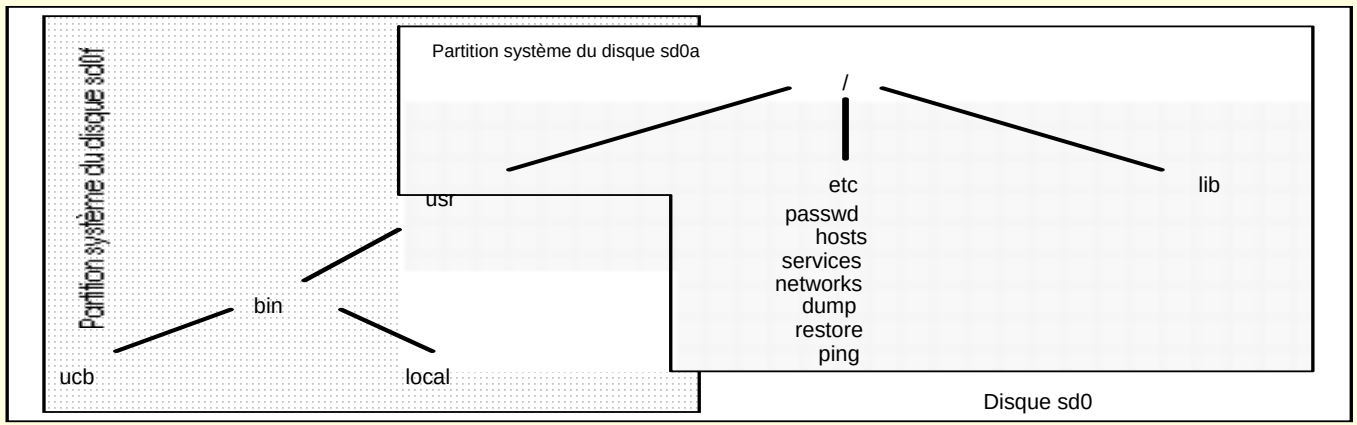
- **NFS :** Protocole proprement dit de réalisation des accès distants.
- **Mount** : Protocole pour le montage d'arborescences distantes sur l'arborescence locale (permettant la désignation en univers réparti).
- **Lock Manager** : Protocole de gestion de verrous pour le contrôle de concurrence (avec état).
- **Network Monitor** : Protocole de surveillance du réseau.
- **RPC, XDR, PortMapper** : protocoles définis dans le cadre du RPC SUN pour réaliser les communications, faire des conversions de représentation, déterminer les numéros de ports utilisés dans les RPC

NFS : le protocole de montage
Rappel le montage en local



Avant montage de la partition sd0f :

% cd /usr ; ls -l
%



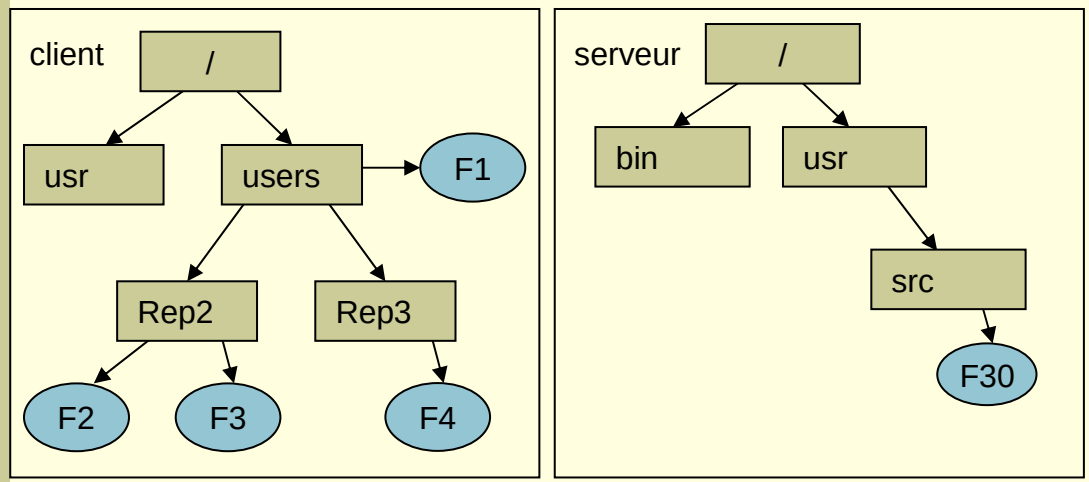
Après montage de la partition sd0f :

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

NFS : le protocole de montage
Montage de systèmes de fichiers distants

- **Extension :** de la notion de montage existant entre systèmes de fichiers locaux à un montage de fichiers distants.
- **Fichier /etc/exports** : Définition par le serveur d'une liste des répertoires qui peuvent être montés à distance sous la forme nom de répertoire, liste d'accès des utilisateurs et machines autorisées à effectuer le montage
  - **Exemple :** ligne du fichier /usr -access=client
  - **Remarque :** Le nom d'un répertoire correspond à une sous-arborescence (pas à un système de fichier).
- **Command mount :** Définition par le client du point ou il désire accrocher l'arborescence distante.
  - mount –t nfs nom_serveur:/chemin_serveur /chemin_client

Systèmes de fichiers répartis NFS : Exemple de montage distant



Fichier exports /usr -access=client

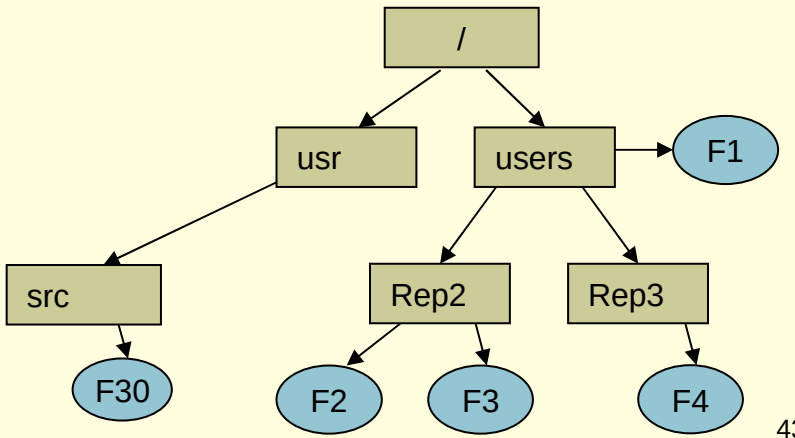
$ mount –t nfs serveur:/usr /usr

Référence de fichiers

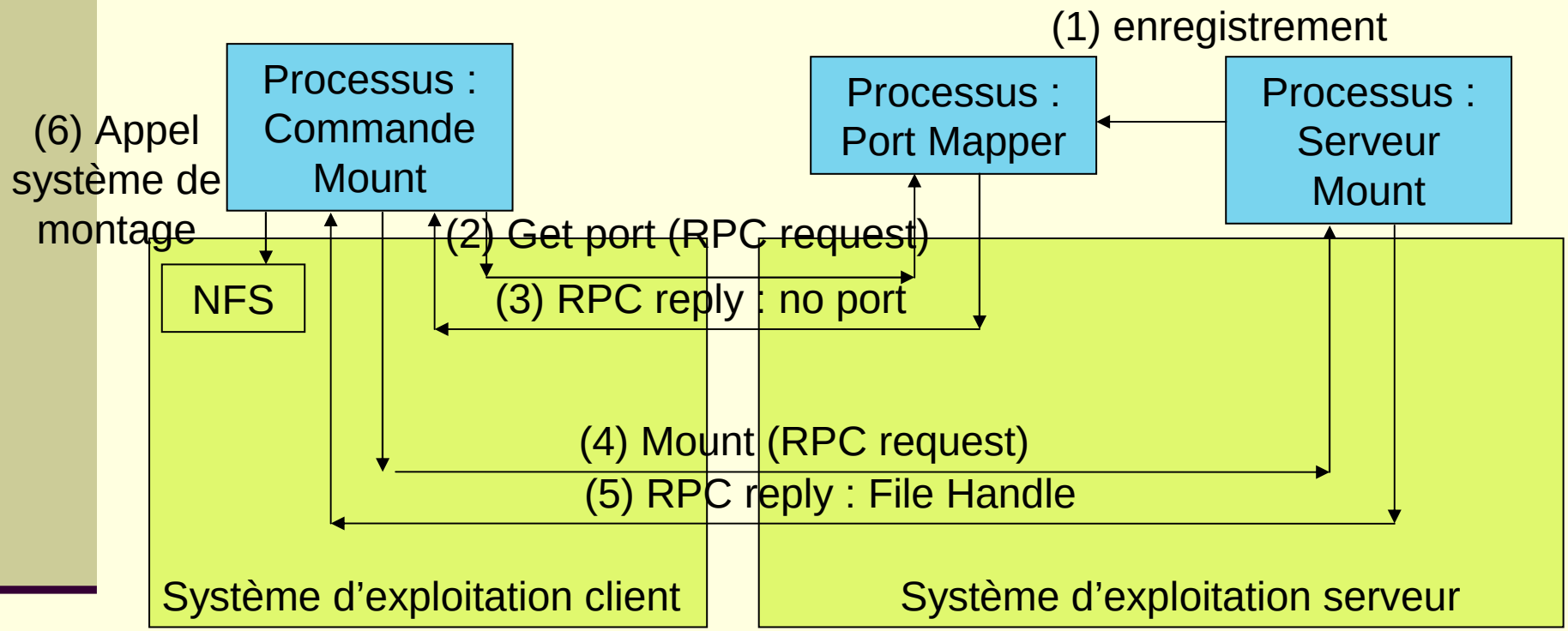
Distant : /usr/src/F30

Local : /users/rep3/F4

Vision client après montage



NFS : Fonctionnement du montage



- Utilisation de trois processus usager (hors système).
  - Le processus client de montage
  - Le processus serveur de montage ('Mount daemon')
  - Le gestionnaire de ports pour RPC ('PortMapper')

NFS : Fonctionnement du montage (2)

- Démarrage du serveur de port (PortMapper) quand le site serveur démarre.
- Démarrage du processus serveur de montage.
- (1) Le processus serveur de montage s'enregistre auprès du serveur de port.
- Un client effectue une commande mount montage d'une arborescence serveur sur son arborescence locale.
- (2) Le client demande au serveur de ports le numéro de port du processus serveur de montage.
- (3) Le serveur de port fournit le numéro.
- (4) Le processus client (commande mount) émet un RPC pour obtenir une poignée file handle du système de fichier distant.
- (5) Réponse de la poignée.
- (6) Le processus client (commande mount) émet une commande au NFS local pour associer la poignée et le point de montage local. A partir de ce moment le NFS local utilisera cette poignée pour accèder à distance au serveur de fichier réel.

Système de fichiers répartis NFS : Notion de poignée fichier ‘File Handle’

- **Quand un client ouvre un fichier :** il reçoit une structure de données concernant le fichier (‘File Handle’).
  - Contient essentiellement l’identifiant du fichier pour réaliser l’accès sur le bon fichier
  - Structure ‘opaque’ : le client ne manipule pas la structure.
- **Le client doit fournir au serveur cette poignée** : pour chaque nouvelle requête effectuée .
- **Exemple Serveur de fichiers UNIX:** informations ‘handle’
  - **Numéro du système de fichier** : ‘filesystem id’.
  - **Numéro de l’i-node** : identifiant interne du fichier dans le fs.
  - **Numéro de génération** : ce numéro change à chaque fois qu’un numéro d’inode est réutilisé pour un nouveau fichier pour éviter les ambiguités entre fichiers anciens et nouveaux.
  - **Eventuellement du bourrage :** pour compléter la taille selon les versions (versions à taille fixe 32 , 64 octets, version à taille variable définie par la longueur).

Fichiers répartis NFS : Interface virtuelle VFS ‘Virtual File System’

- **Interface séparant l’utilisation d’un fichier de son implantation.**
  - **Nouvelle interface d’E/S fichier avec le noyau Unix** qui reçoit toutes les requêtes considérées par VFS comme génériques.
  - **Réalisation effective des opérations** par les différents systèmes de fichiers (systèmes de fichiers locaux ou distants).
  - **VFS implante la désignation** en réparti par montage d’arborescence.

Fichiers répartis NFS : Concepts de l'interface virtuelle VFS

- **Concepts VFS représentés par des structures de données :**
- **Notion de "Virtual File System" (système de fichier virtuel)**
  - Tous les systèmes de fichiers connus sont représentés par des VFS qui sont identifiés par un ident numérique interne.
  - Un VFS est un descripteur qui contient un identifieur du système de fichier réel et un pointeur sur le V-node sur lequel il est monté.
- **Notion de "Virtual node V-node" (fichier virtuel).**
  - Tous les fichiers connus sont représentés par des V-nodes qui sont identifiés par un identifiant numérique interne.
  - Un V-node est un descripteur qui contient un pointeur sur le système de fichier auquel il appartient et éventuellement sur le système qui est monté sur ce point.

Interface virtuelle VFS ‘Virtual File System’ : Structure de données VFS

- **Descripteur d’un système de fichiers virtuel en C (structure de donnée décrivant un VFS)**

```
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 */
};
```

Interface virtuelle VFS ‘Virtual File System’ : indicateurs et opérations VFS

- **Différents indicateurs associés à un VFS /*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) */
```

- **Différentes opérations sur un VFS**

```
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 */
};
```

Interface virtuelle VFS ‘Virtual File System’ : Structure de données V-Node

- **Descripteur fichier virtuel en C (structure décrivant un 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 */
};
```

Interface virtuelle VFS ‘Virtual File System’: Opérations prévues sur fichier

- **Liste des opérations possibles sur fichier virtuel**

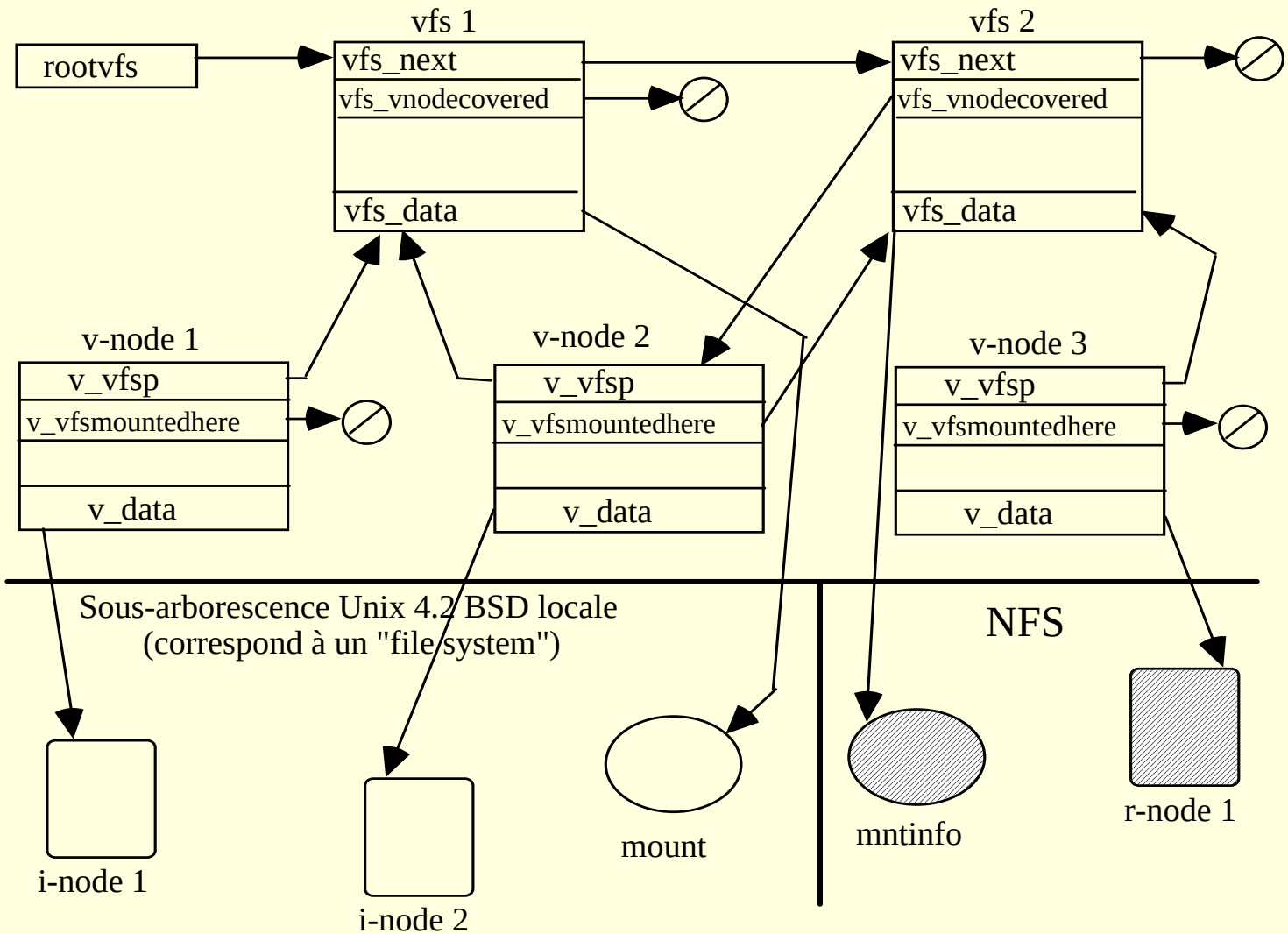
```
/* vnode operations */
struct vnodeops {
    int (*vn_open)();       int (*vn_close)();      int (*vn_rdwr)();
    int (*vn_ioctl)();      int (*vn_select)();     int (*vn_getattr)();
    int (*vn_setattr)();    int (*vn_access)();     int (*vn_lookup)();
    int (*vn_create)();     int (*vn_remove)();     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)();
};
```

Interface virtuelle VFS ‘Virtual File System’: Attributs d’un fichier virtuel

- **Liste des attributs possibles sd’un fichier virtuel**

```
/ * 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 */
};
```

Interface virtuelle VFS: Relations entre VFS-VNODE et fichiers réeels

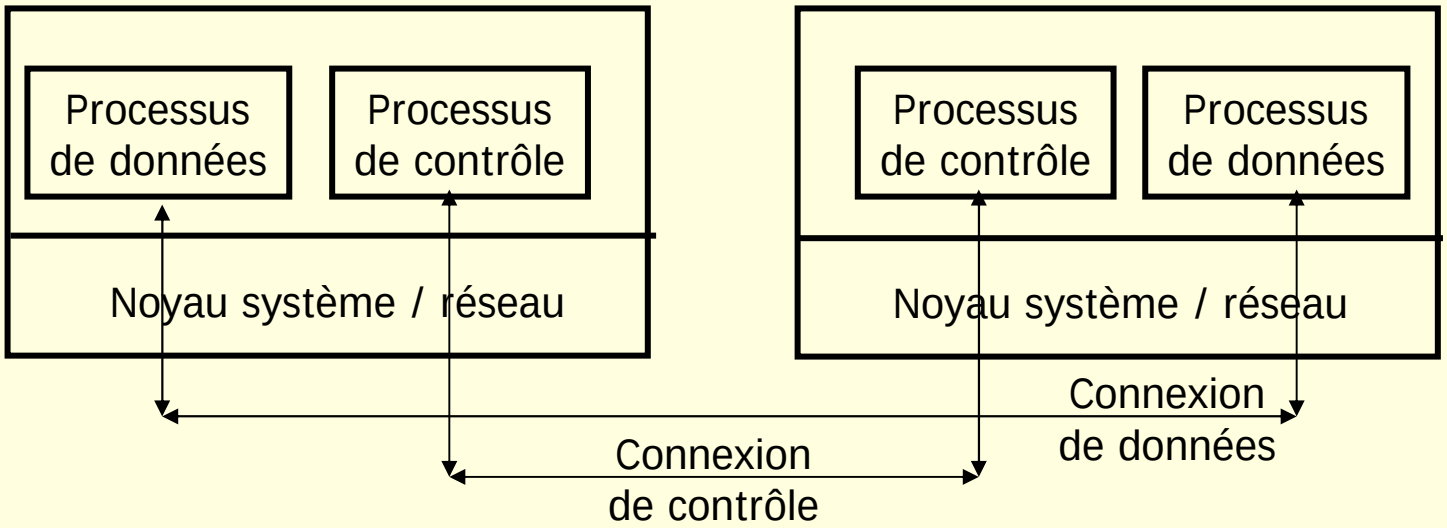


Le transfert de fichiers
FTP
(‘File Transfer Protocol’)

FTP : Généralités

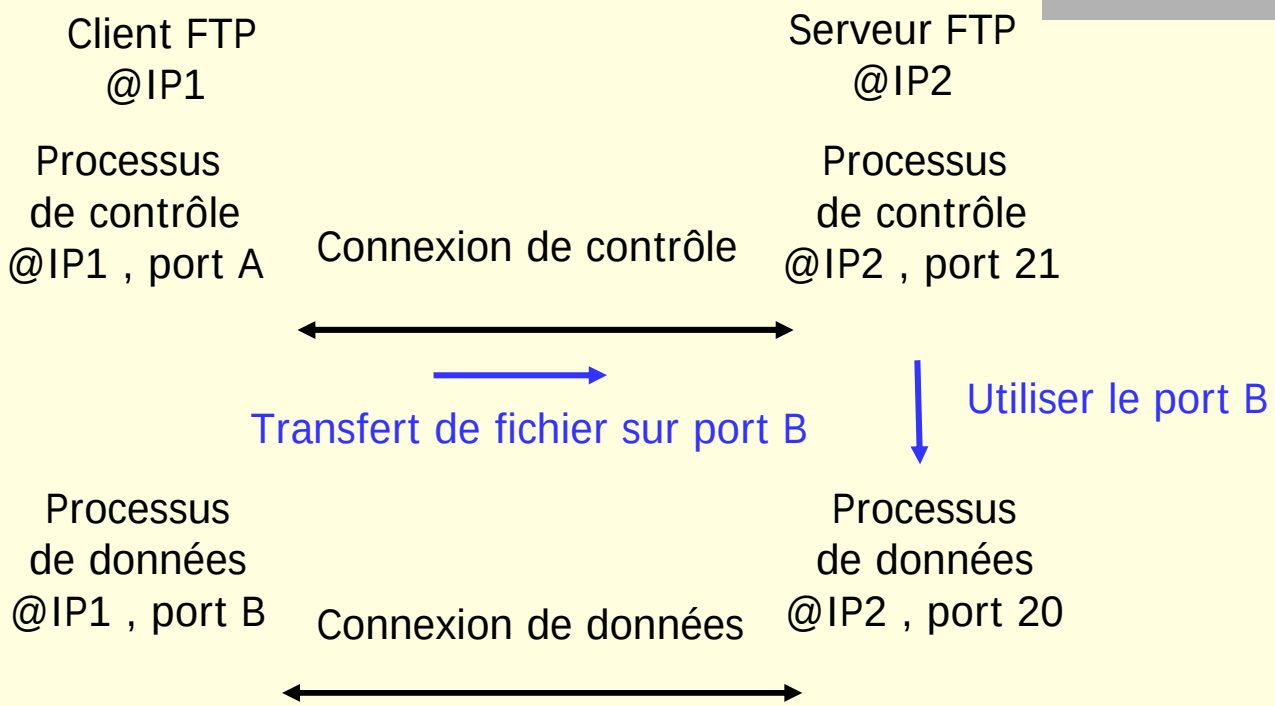
- **Utilise le transport TCP**
- **Pour transférer des fichiers** entre deux hôtes (éventuellement de nature différente)
- **Interface utilisateur:** ensemble de commandes en mode ligne ou graphique (nombreuses interfaces conviviales par exemple ‘leechftp’)
- **Quelques commandes essentielles**
  - Authentification sur le site distant.
  - Transfert d’un fichier.
  - Liste des fichiers dans un répertoire.
  - Mode caractère ASCII NVT ou binaire.

FTP : Fonctionnement



- **Processus de contrôle et de données:** Deux processus coté client, deux coté serveur.
  - Contrôle : Gère le dialogue utilisateur et lance le processus de données.
  - Données : Effectue concrètement chaque transfert de fichier.
- **Connexion TCP de contrôle** : entre processus de contrôle
  - Achemine les directives de contrôle (transfert, liste de répertoire …)
  - Quand la connexion de contrôle est terminée les transferts données sont abortés.
- **Connexion TCP de données** : entre processus de données
  - Une connexion TCP est créée pour chaque transfert de fichier.

FTP : Utilisation des ports TCP



- Le processus de contrôle serveur est en attente sur le port 21.
- Un processus serveur de données est créé pour chaque transfert de fichier et attend sur le port 20.
- Le client utilise un port A quelconque pour dialoguer avec le serveur.
- Le client choisit un autre port libre B qui est transmis au processus de données serveur par la connexion de contrôle.

TFTP : Trivial FTP

- **FTP trop riche dans certains cas :**
  - **Fonctions diverses pas toujours utiles** : authentification, liste de répertoire.
  - **Implantations :** support de plusieurs connexions TCP en parallèle
- **TFTP une version très simplifiée** : créée pour le transfert du binaire au démarrage d'une machine sans disque.
  - Programme de petite taille.
  - Peu de fonctions disponibles.
  - Simplification par l'utilisation de UDP (blocs de 512 octets).
  - Contrôle d'erreur nécessaire avec UDP (par acquittement positif).
  - Principaux types de primitives d'accès fichiers
    - lecture (fichier)
    - écriture (fichier)
    - transfert d'un bloc avec numéro
    - acquittement positif (numéro)
    - code d erreur, message d'erreur

Bibliographie

- E. Gressier Cours systèmes de fichiers répartis. Réseaux et communications B CNAM Paris
- R. Stevens : TCP/IP Illustrated Volume 1 The protocols Addison Wesley 1993
- Coulouris , Dollimore & Kindberg “Distributed Systems”, Troisième Edition, Addison Wesley, 2000 (Ch. 6 & 18)
- A. Tanenbaum “Distributed Operating Systems”, Prentice Hall, 1995
- Arkady Zaslavsky Distributed and mobile file systems http://www.csse.monash.edu.au/courseware/cse5501/mdcs-l-04