M2 SRI

Systèmes de Fichiers Répartis

Plan

- Introduction
- FTP
- Newcastle Connection
- NFS
  - NFS v2, NFS v3, WebNFS, NFS v4
  - Spritely NFS
  - NQNFS
- Coda
- GFS

Définitions

Un système distribué:

Un ensemble d'ordinateurs ui appara!t comme un ensemble uni ue et cohérent " ses utilisateurs#

$%anen&aum et al#'

un système dans le uel les composants matériels ou lo(iciels communi uent et coordonnent leurs actions e)clusi*ement par échan(e de messa(es#

$+oulouris et al#'

Définitions

Un système distribué:

Un système dans le uel la panne d'un ordinateur dont *ous n'a*e, -amais entendu parler. emp/che *otre tra*ail de pro(resser0

$1eslie 1amport'

+aractéristi ues

+oncurrence

2bsence d'horlo(e (lobale

3annes indépendantes

3as de mémoire commune

+ommunication par 4messa(es4

%endances:

mobilité

ubi uité

3roblèmes

5étéro(énéité

+ode mobile

6u*erture

Sécurité

+oncurrence

+apacité de croissance $scalability'

%ransparence

%ransparences

2ccès $*access)*

7omma(e. synta)i ue. sémanti ue

8mplacement $*location*'

+oncurrence

Duplication

3annes $*Failure*'

Mi(ration 9 Mobilité

3erformance

:;1imites;<

6ptimisation. recherche de co=localité.

+onte)tes de nomma(es $cf 3lan >'

2utonomie

3arta(e temporaire de ressources $cf Sprite'

2dministration

2 pproches

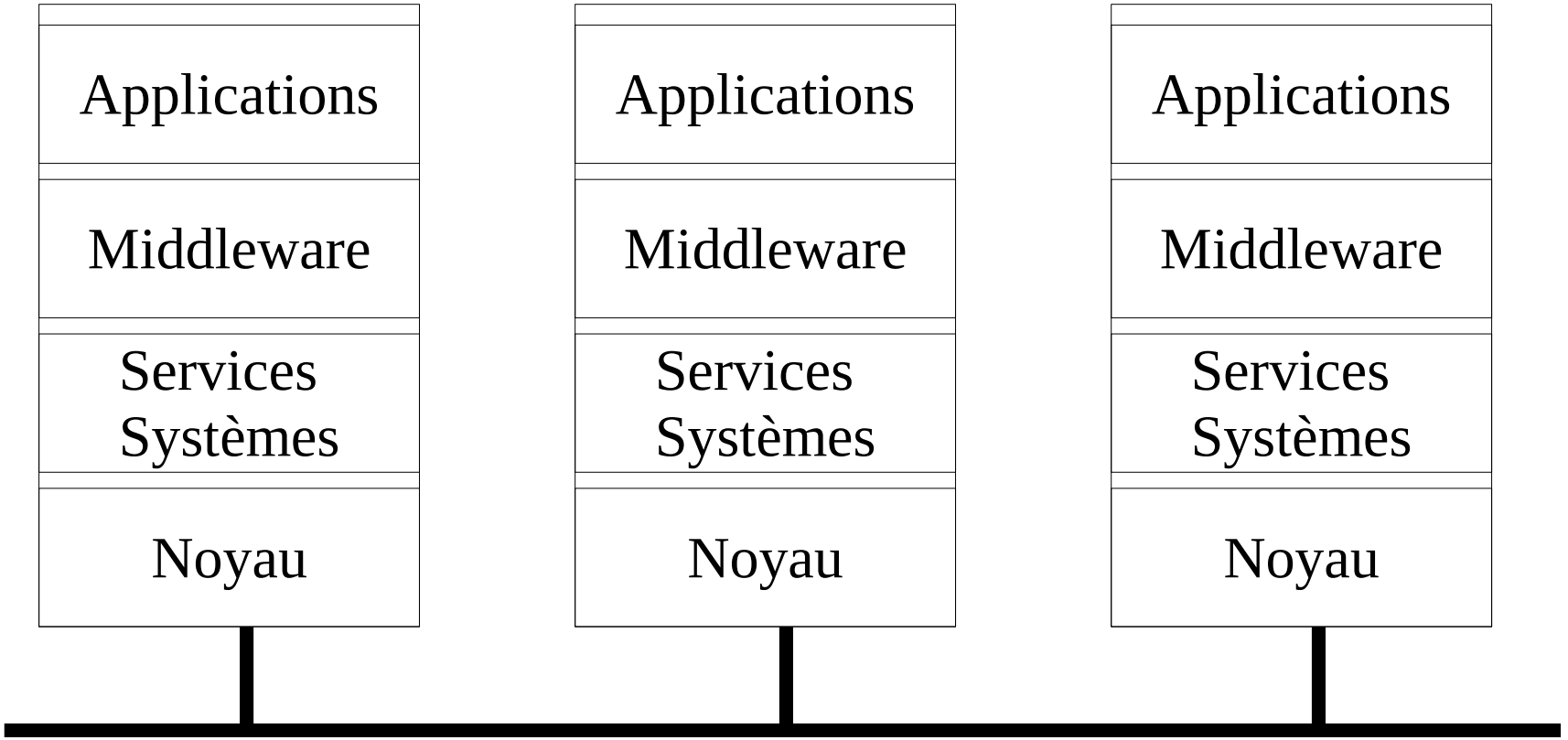
Systèmes fortement couplés:

Sin(le System Ima(e $%rue+luster. ?erri(hed'

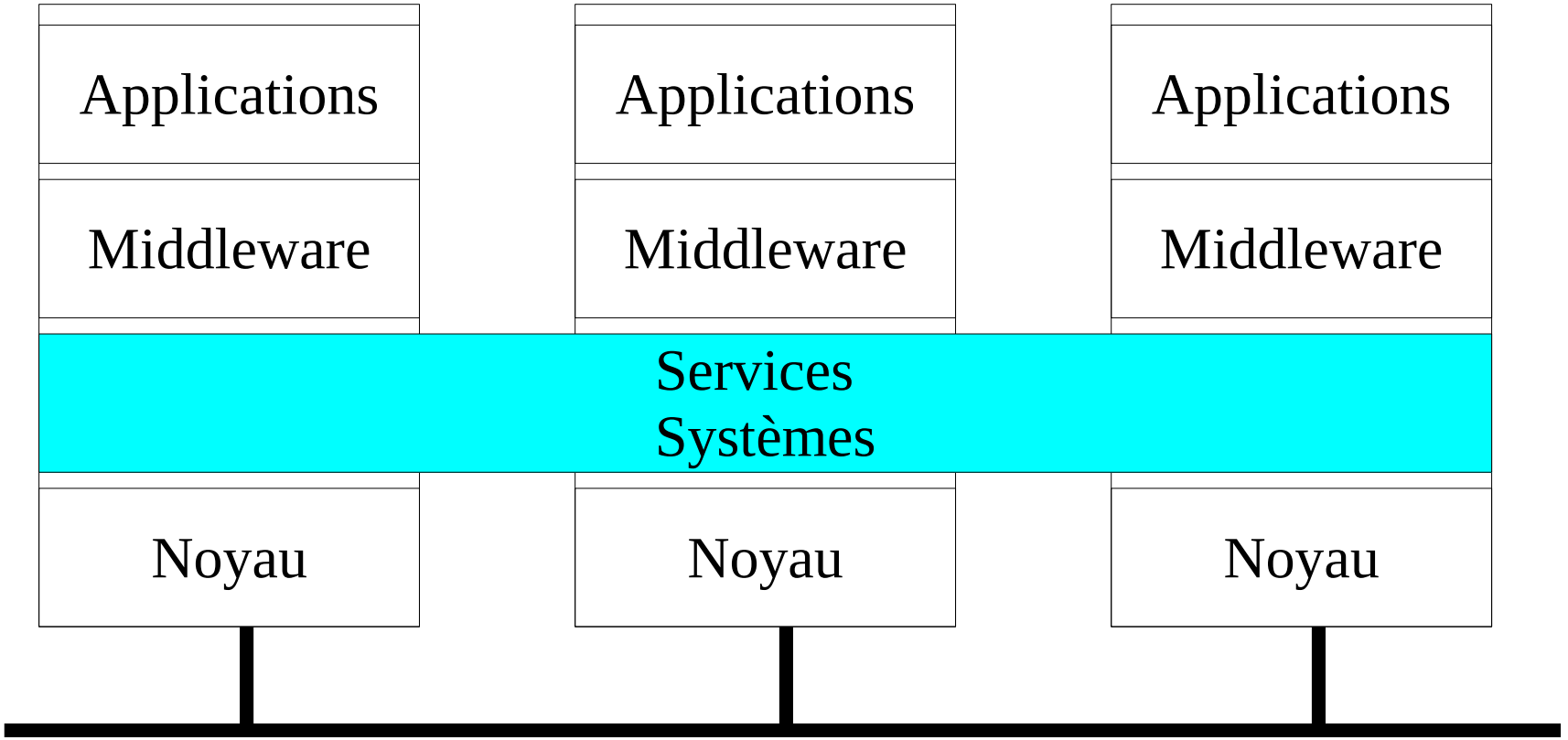
Systèmes faiblement couplés:

@eb

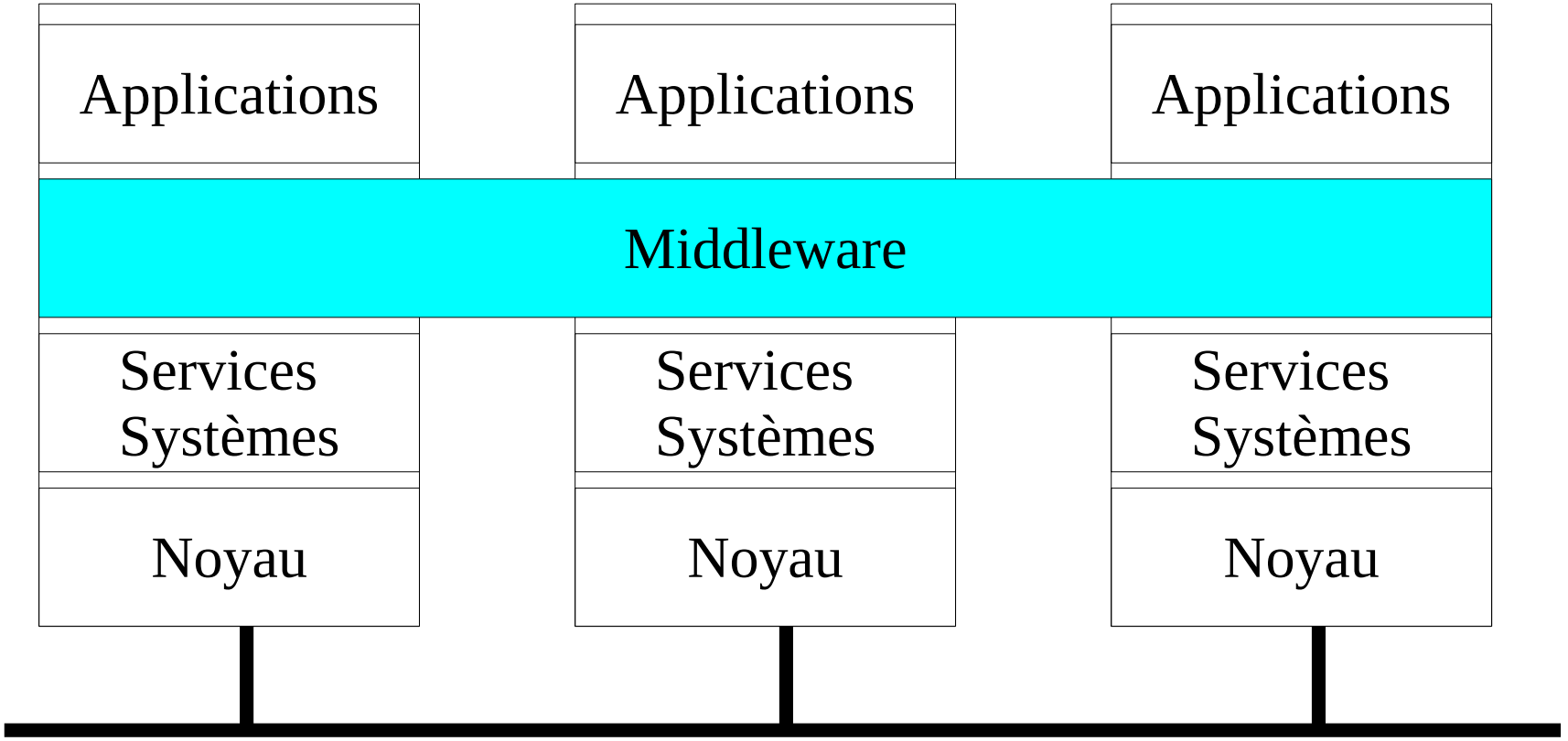
2rchitectures 1o(icielles



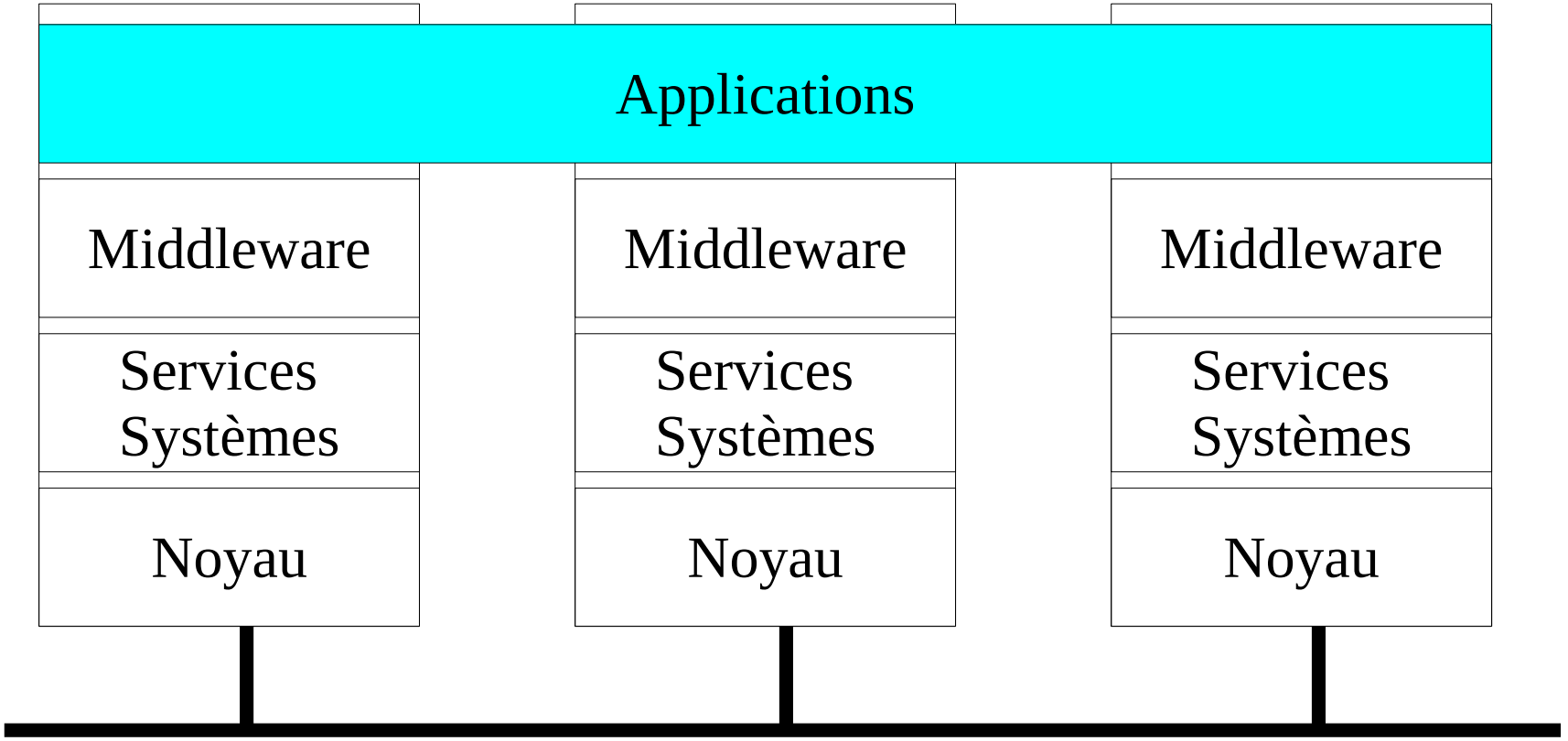
2rchitectures 1o(icielles



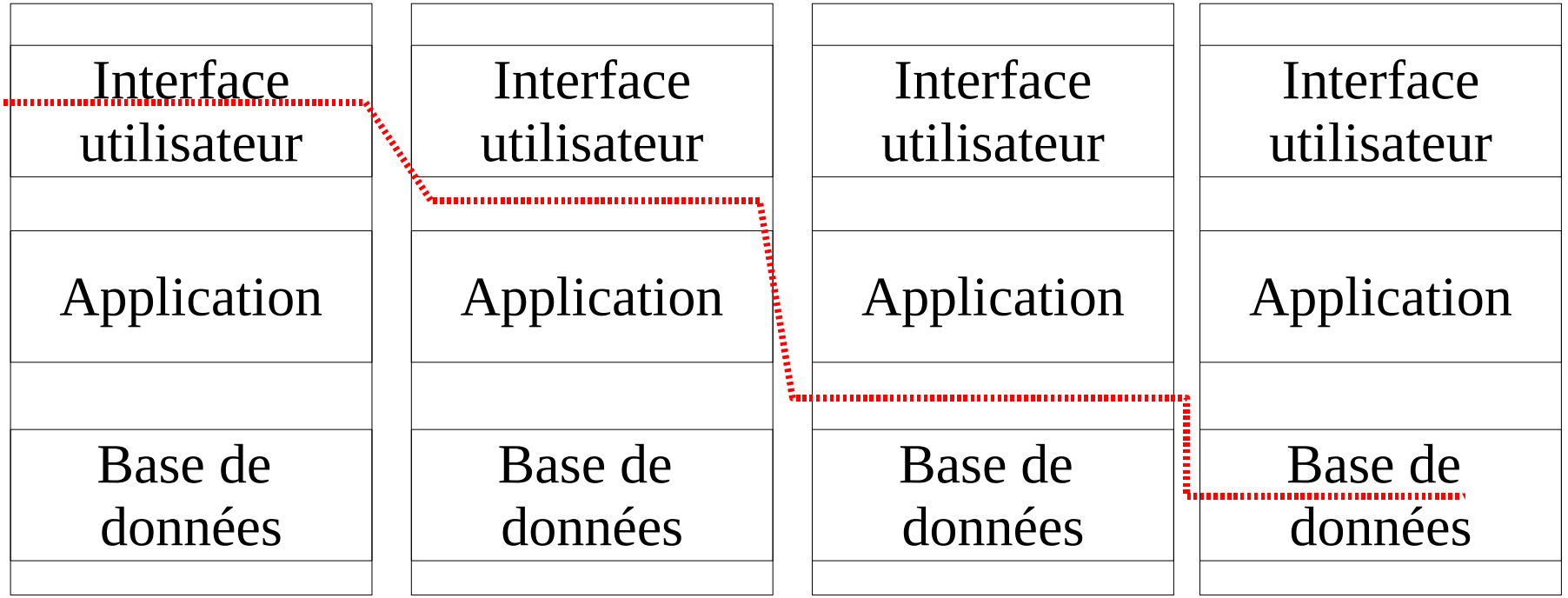
2rchitectures 1o(icielles



2rchitectures 1o(icielles



2 pproches + lients=Ser*eurs



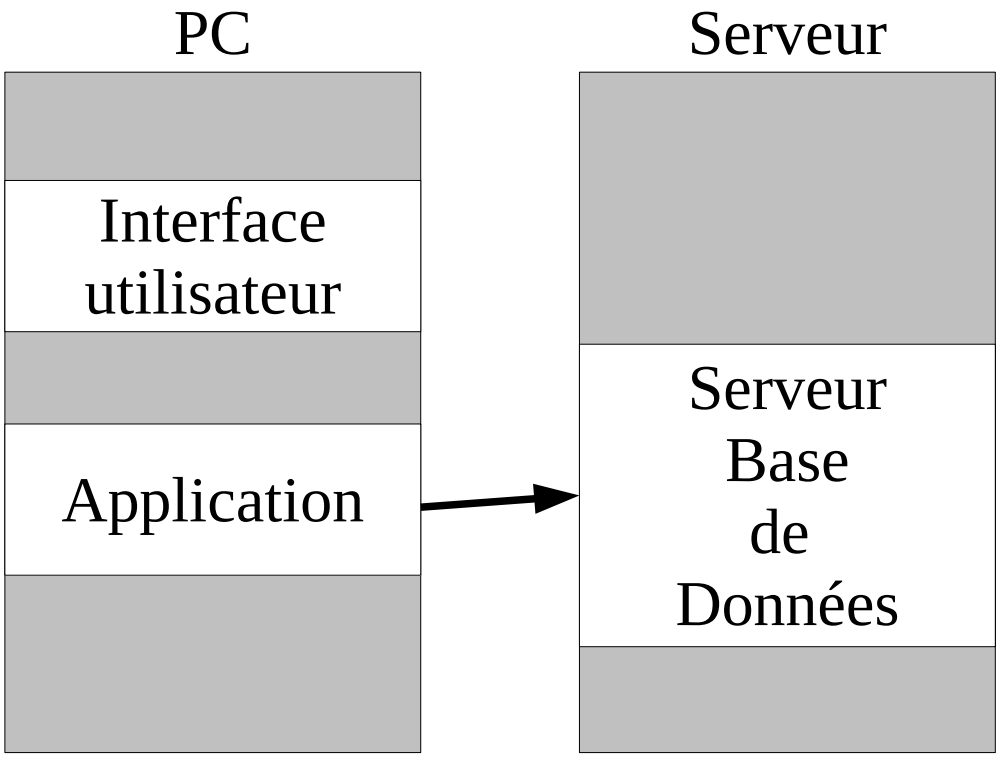
Déploiement: 2=%iers

Pour:

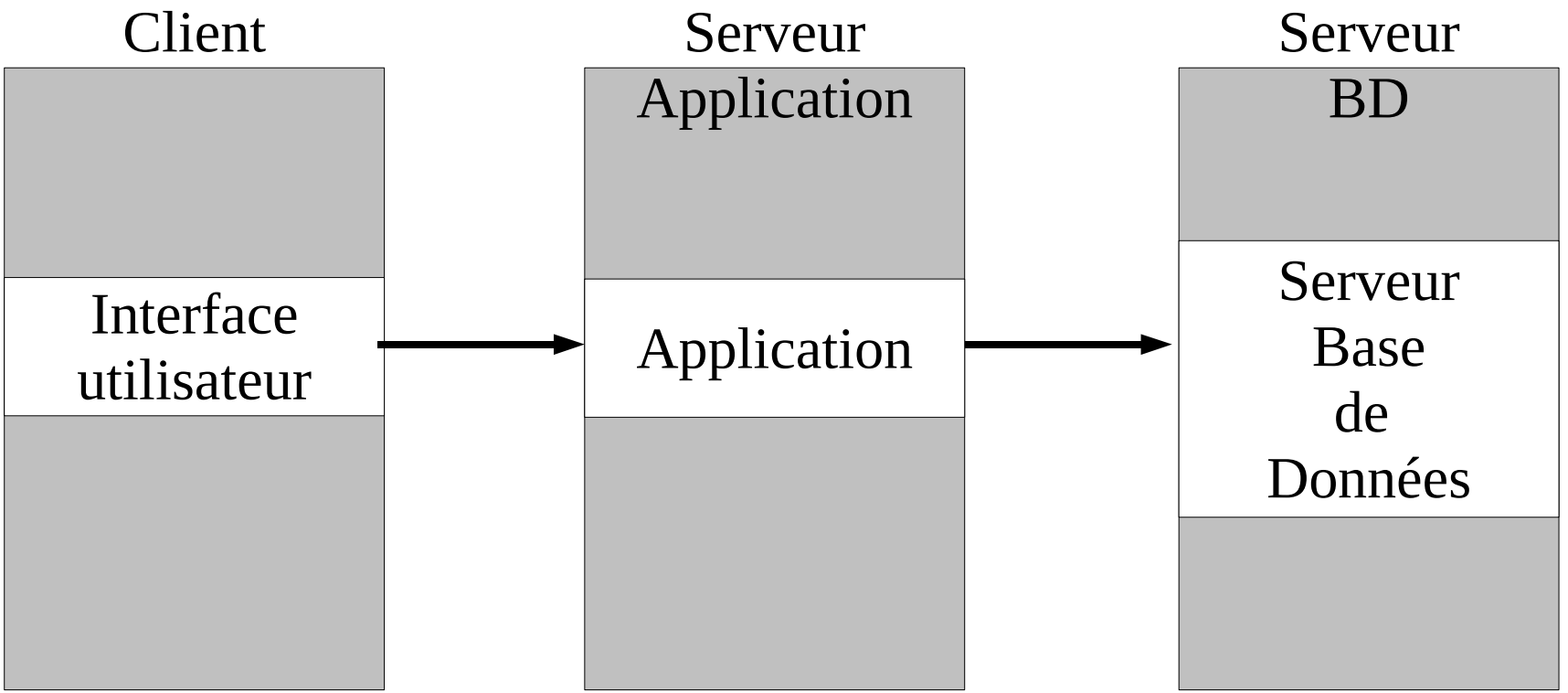
- Facilité de développement

Contre

- Compromission BD
- Administration
- Maintenance du code
- Sécurité
- Échelle
- Homogénéité
- Présentation fixe



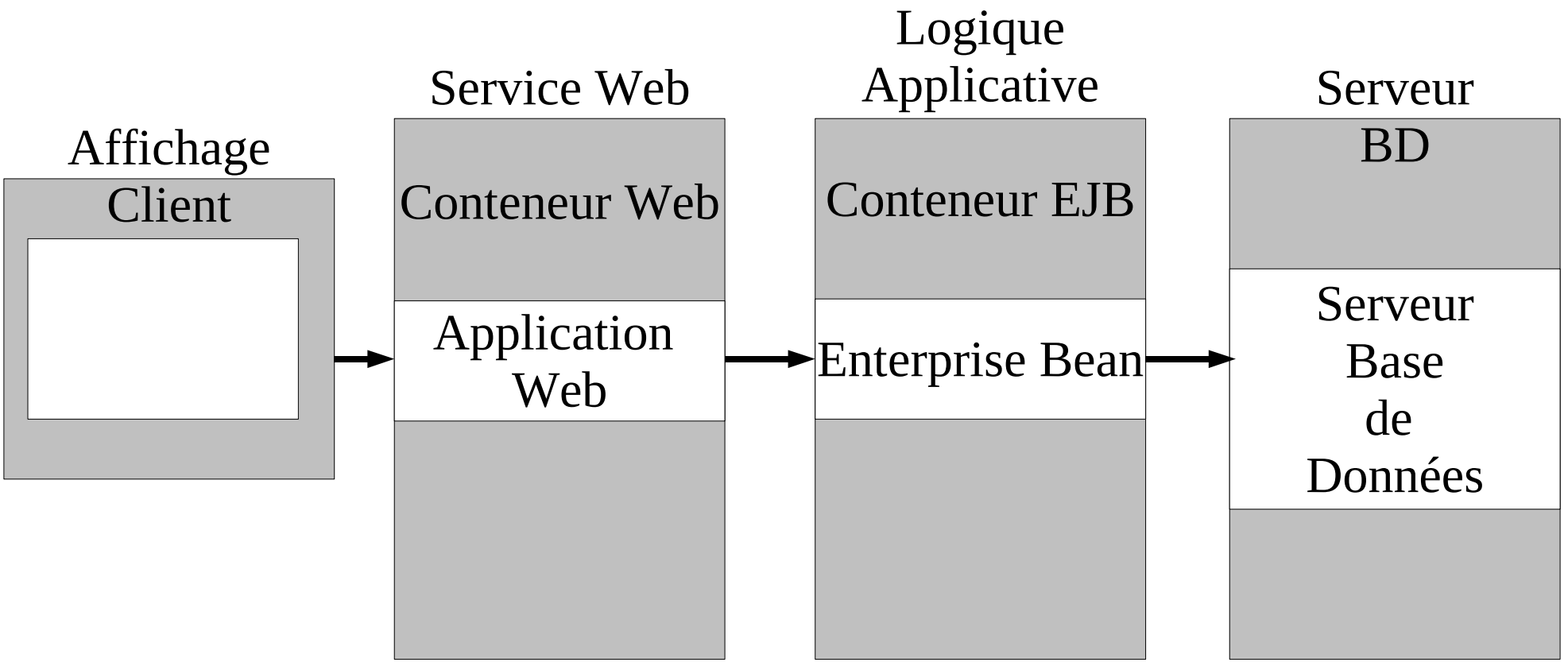
Déploiement: A=%iers



Déploiement: A=%iers

- Complexité
  - Application distribuée:
    - multithread, sécurité
- Portabilité
- ...

Déploiement: Multi=%iers 9 B288



Plan

Introduction

FTP

N'est pas un système de fichiers répartis

Pas de transparence pour l'utilisateur

Conduit à des copies multiples

- Problème de maintien de la cohérence (sauf lecture seule)

Certains systèmes de fichiers « masquent » l'accès par FTP

Plan

Introduction

FTP

Newcastle Connection

NFS

- NFS v2, NFS v3, WebNFS, NFS v4
- Spritely NFS
- NQNFS

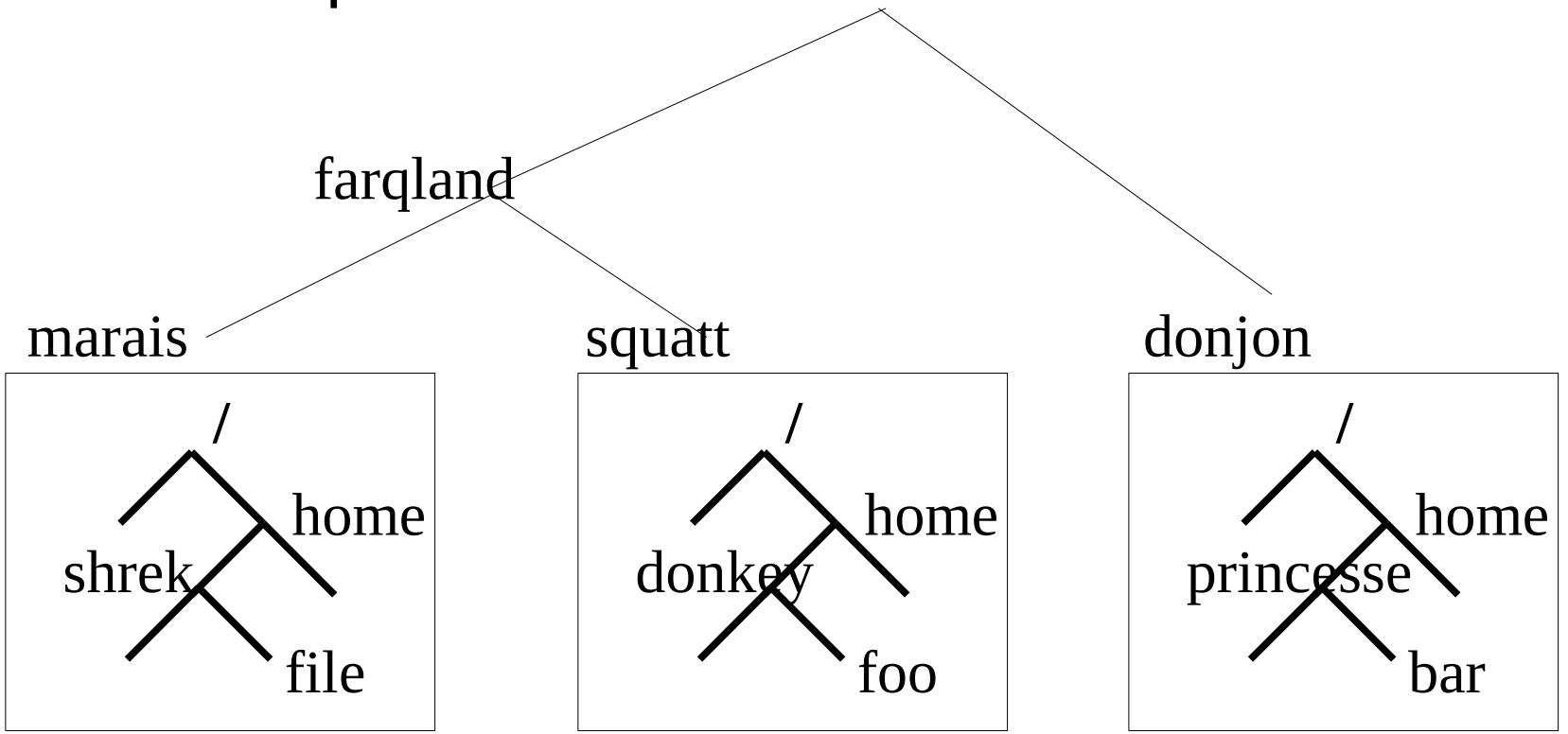
Coda

GFS

Newcastle Connection

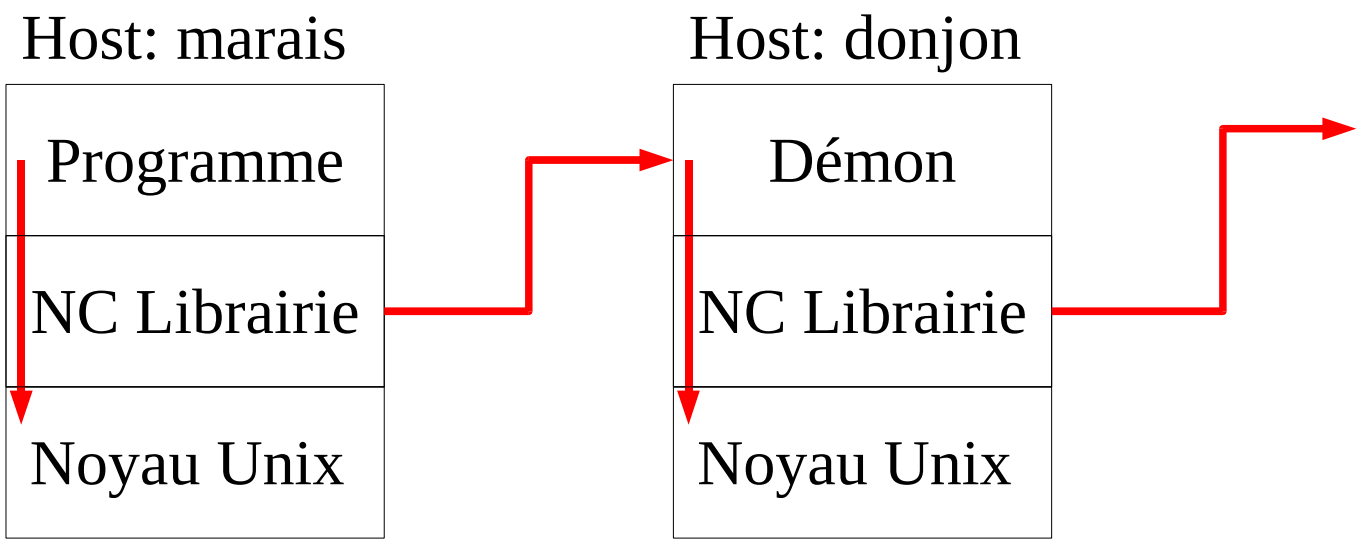
Transparence fournie par une librairie dédiée

Notion de "super-root"



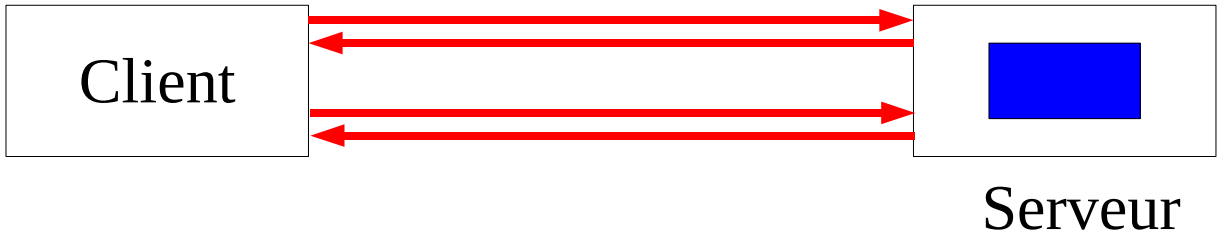
open ("/../../donjon/home/princesse/bar",...)

Newcastle Connection



Accès à un fichier distant

Envoi de toutes les requêtes et leurs paramètres au serveur

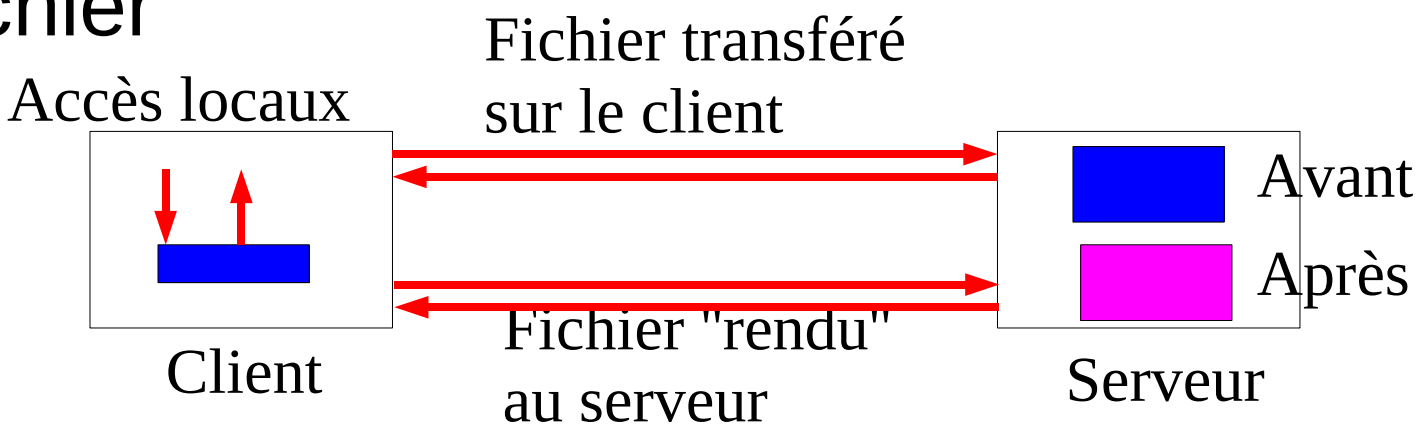


Le fichier reste sur le serveur

- Pas de problème de cohérence!
- Latence pour chaque opération
- Serveur = goulot d'étranglement

Accès à un fichier distant

Cacher localement sur le client tout ou partie du fichier



Latence réduite

Serveur moins chargé

Problèmes de cohérence

Plan

Introduction

FTP

Newcastle Connection

NFS

- NFS v2, NFS v3, WebNFS, NFS v4
- Spritely NFS
- NQNFS

Coda

GFS

NFS

- Protocole
  - S'appuie sur XDR et RPC (ONC)
  - Pas une interface
  - Ouvert, mis à disposition par Sun
- Largement répandu et utilisé
  - NFS v2 et NFS v3 déployés (sur UDP et TCP)
  - WebNFS
  - NFS v4
  - Variantes: Spritely NFS, NQNFS

Principes / Objectifs

Apparu quand les vitesses de transfert sur les réseaux locaux sont devenues acceptables

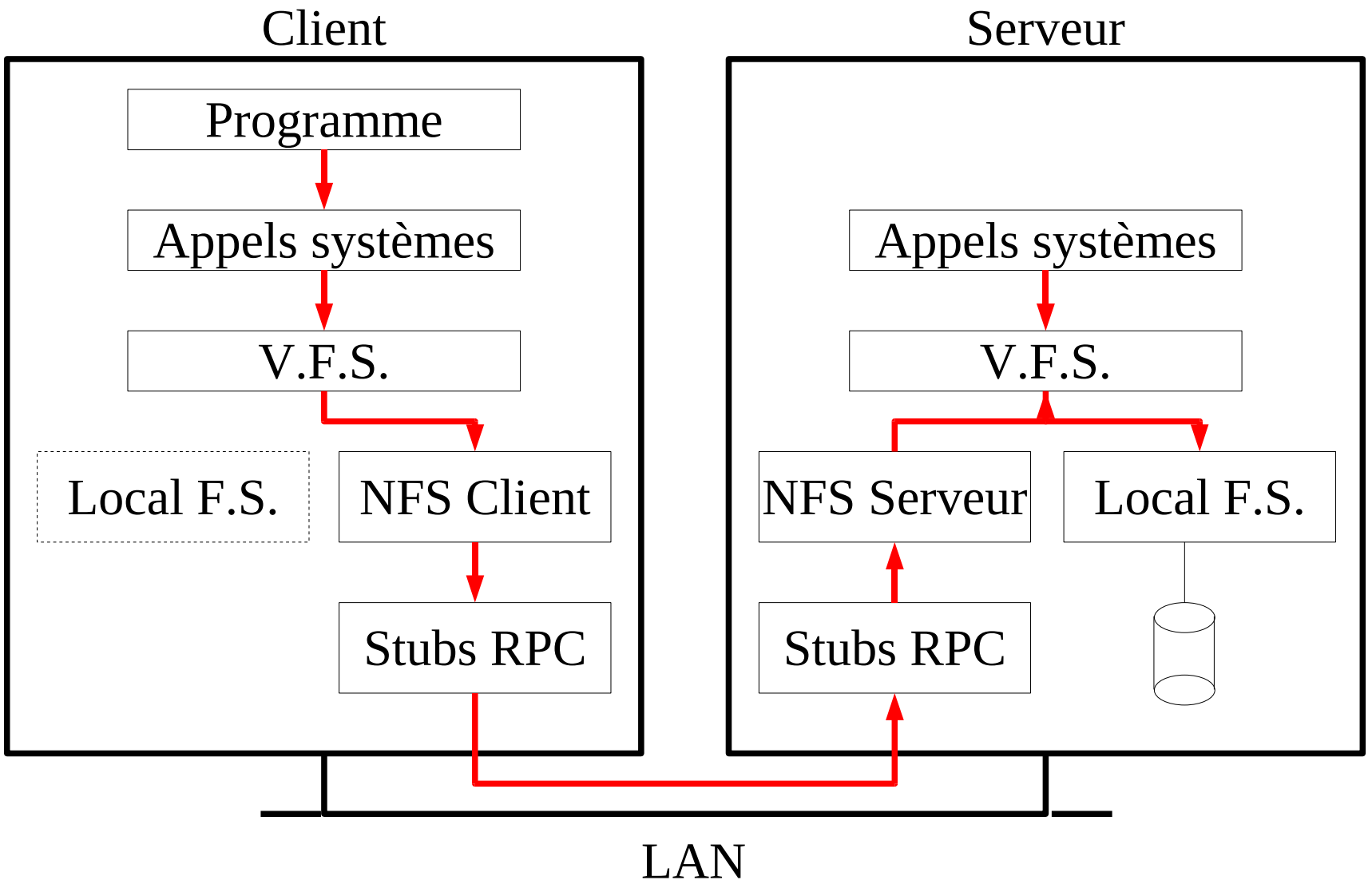
Minimiser l'état côté serveur

- Permettre un redémarrage facile du serveur

"Write sharing" rare

Flexibilité espace nommage du client

Architecture Typique



NFS: Modèle

Objets:

- Fichiers: séquence banalisée d'octets
- Catalogues: liste de noms simples de fichiers
- Liens symboliques interprétés par le client
- Fichiers spéciaux: périphériques locaux au client

File Handle:

- Indépendants du chemin d'accès au fichier
- Persistants

NFS: Modèle

- Chemins d'accès
  - Non nul, pas la chaîne nulle
    - Si nécessaire; doit être interprété par le client
  - '.': répertoire courant
  - '..' répertoire parent du répertoire courant
  - Évalués composant par composant par le serveur
    - Plus simple pour le serveur
    - Modèle Unix côté client
    - Lourdeur pondérée par le cache côté client

NFS: Modèle

- Serveur sans état (en fait à état minimum)
  - NFS v2 et v3
  - "Permis", pas imposé par le protocole
  - Pas de requête open/close
- Opérations non-idempotentes
  - Dues au RPC "at least once"
  - Détection par cache
- État minimum:
  - Facilité de basculement sur un autre serveur en cas de défaillance

NFS: Modèle

- NFS v2 v3
  - Ne couvre pas le montage
  - Mount protocole
  - Ne couvre pas le verrouillage de fichiers
    - Fait par "Lock Manager"
- NFS v4
  - intègre montage et verrouillage

NFS v2

File handles: fixes 32 octets

Ex: fsid / inode / numéro de génération

Opérations:

Null, getattr, setattr, *root,* lookup, readlink,

read, *writecache*, write, **create, remove, rename**

link, symlink, mkdir, rmdir, readdir, statfs

NFS v3

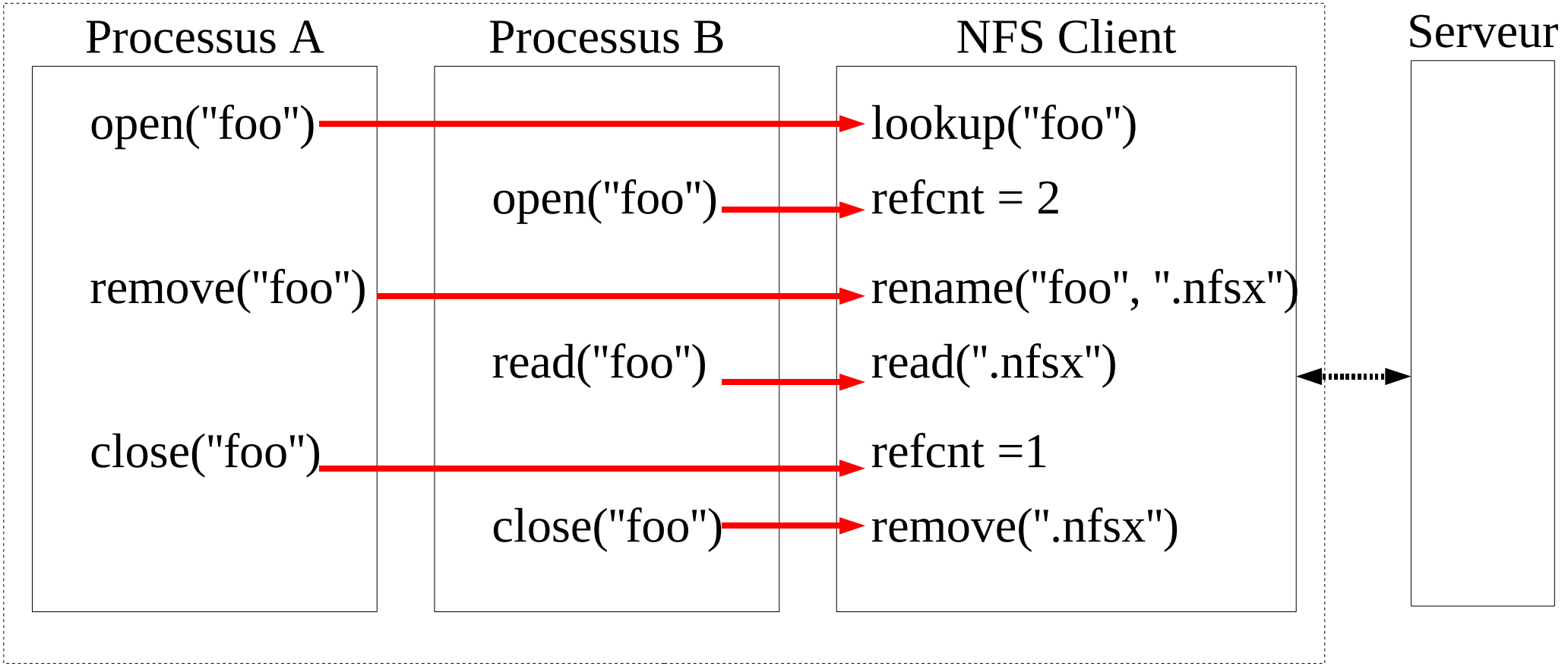
File handles: var

NFS v3

Nouvelles procédures:

- Access
- mknod
- readdirplus
- commit

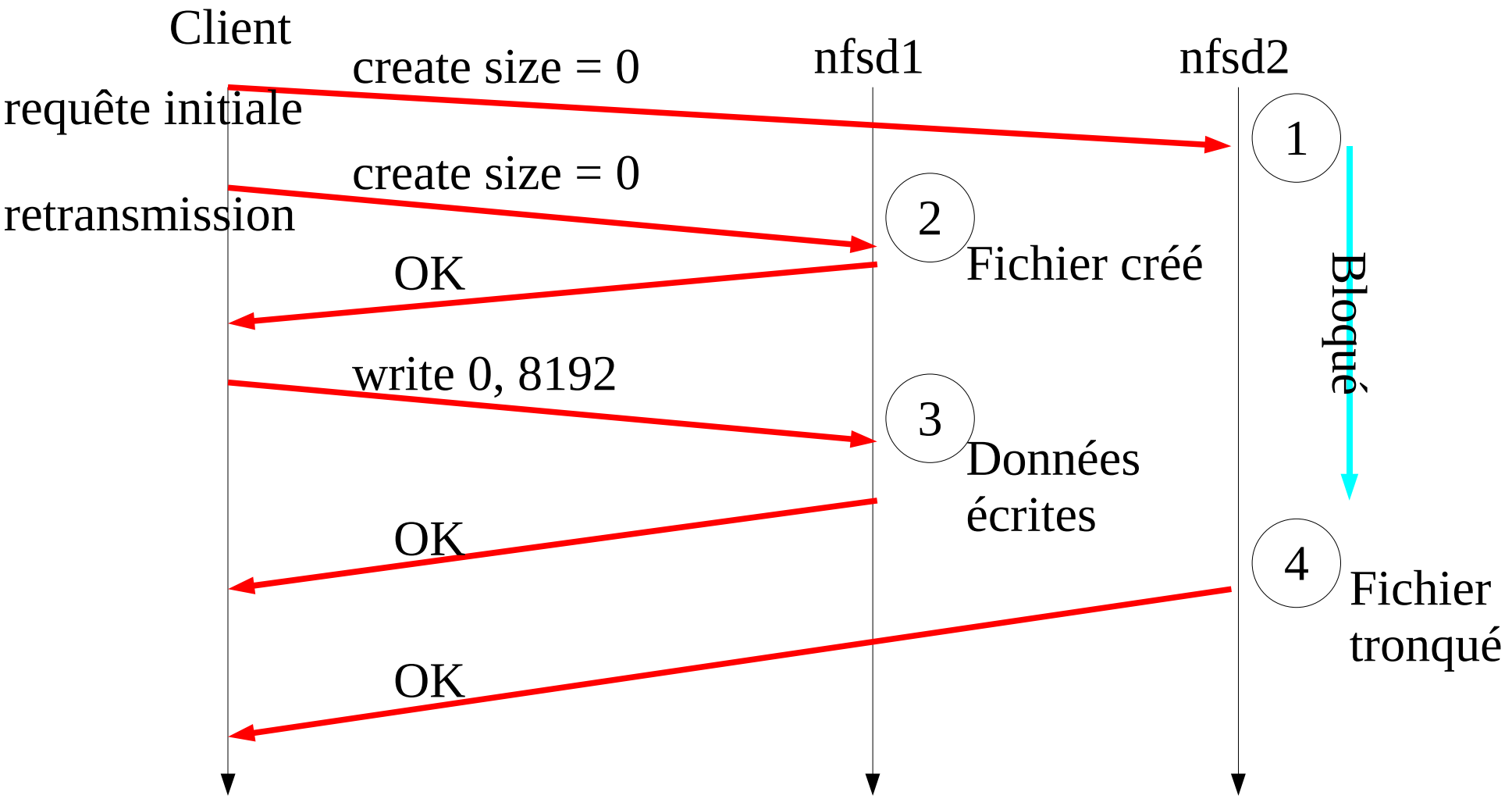
Implémentation: dernier close



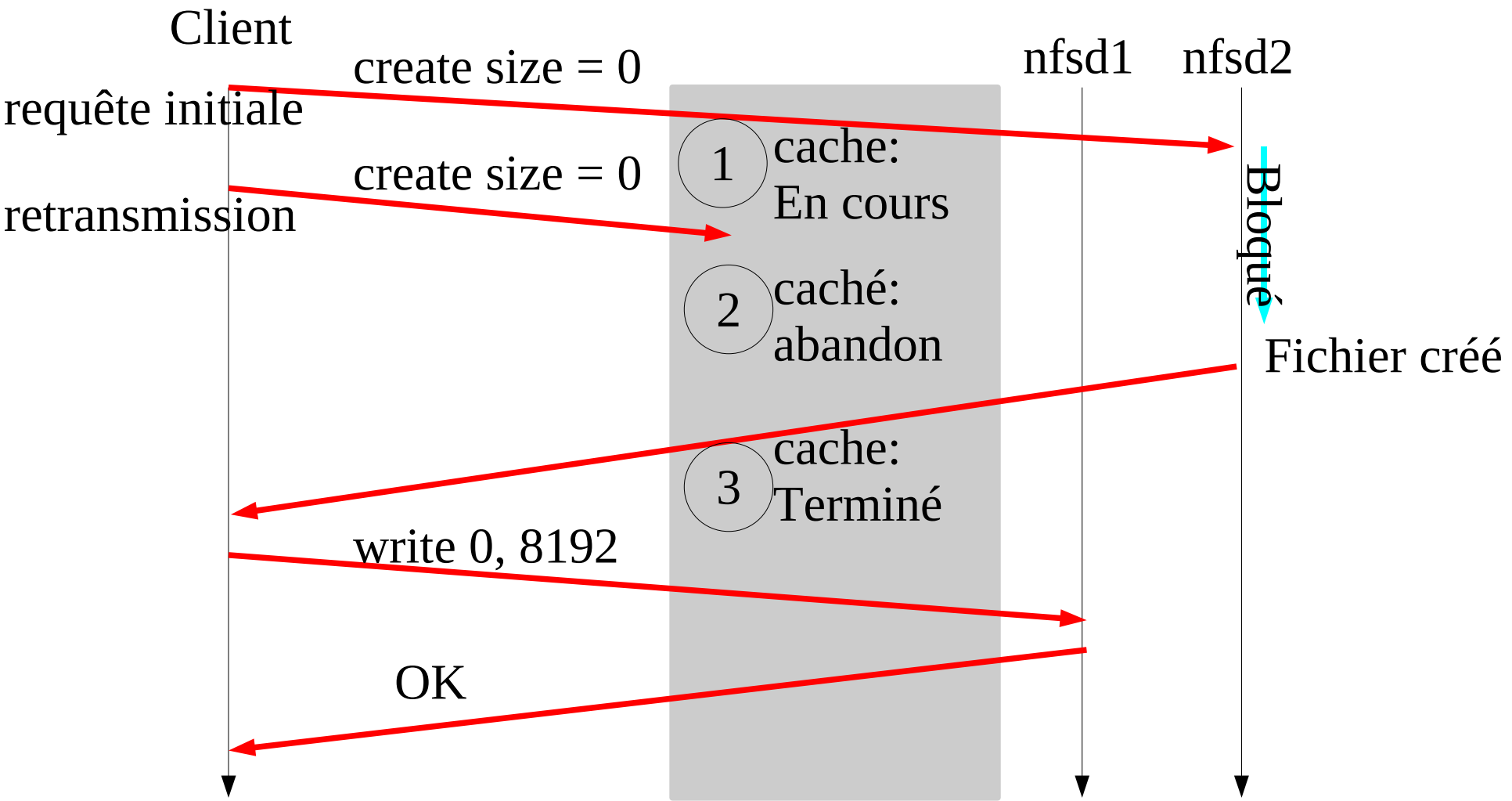
Implémentation: Concurrence

- Serveur:
  - Plusieurs démons ou threads (nfsd)
- Client:
  - Requête émise par le processus (thread)
  - Entrées / sorties asynchrones
    - Write back, read ahead
    - gérées par démons (threads) dédiés (biod)

Implémentation: Cache de requêtes dupliquées



Implémentation: Cache de requêtes dupliquées



Implémentation: Cache de

Implémentation: Serveur et Stockage stable

Le serveur doit normalement répondre à une requête d'écriture après avoir sauvegardé les données sur mémoire stable

- disque
- NVRAM

NFS v3: attribut d'écriture + commit

- UNSTABLE
- DATA_SYNC
- FILE_SYNC

Retransmission adaptative (UDP)

- Time-out calculé dynamiquement
- Retransmission avec "backoff" exponentiel:
  - 2, 4, 8, 16, 32, 64, 128, 256
  - souvent limité à borne max (8)
- Timeout calculé en fonction de la requête
  - lookup, getattr,...
  - setattr, read,...
  - write, mkdir,...

Write behind, write gathering

Client: write behind

Si le processus écrit séquentiellement, le client NFS entreprend de transférer les blocs au serveur

Serveur:

Sur réception d'un bloc, légère attente dans l'espoir de regrouper des blocs consécutifs en une seule opération

Implémentation: cache client

Cacher information retournée par serveur:

- accès consécutifs plus rapides
- désengorger le réseau
- désengorger le serveur

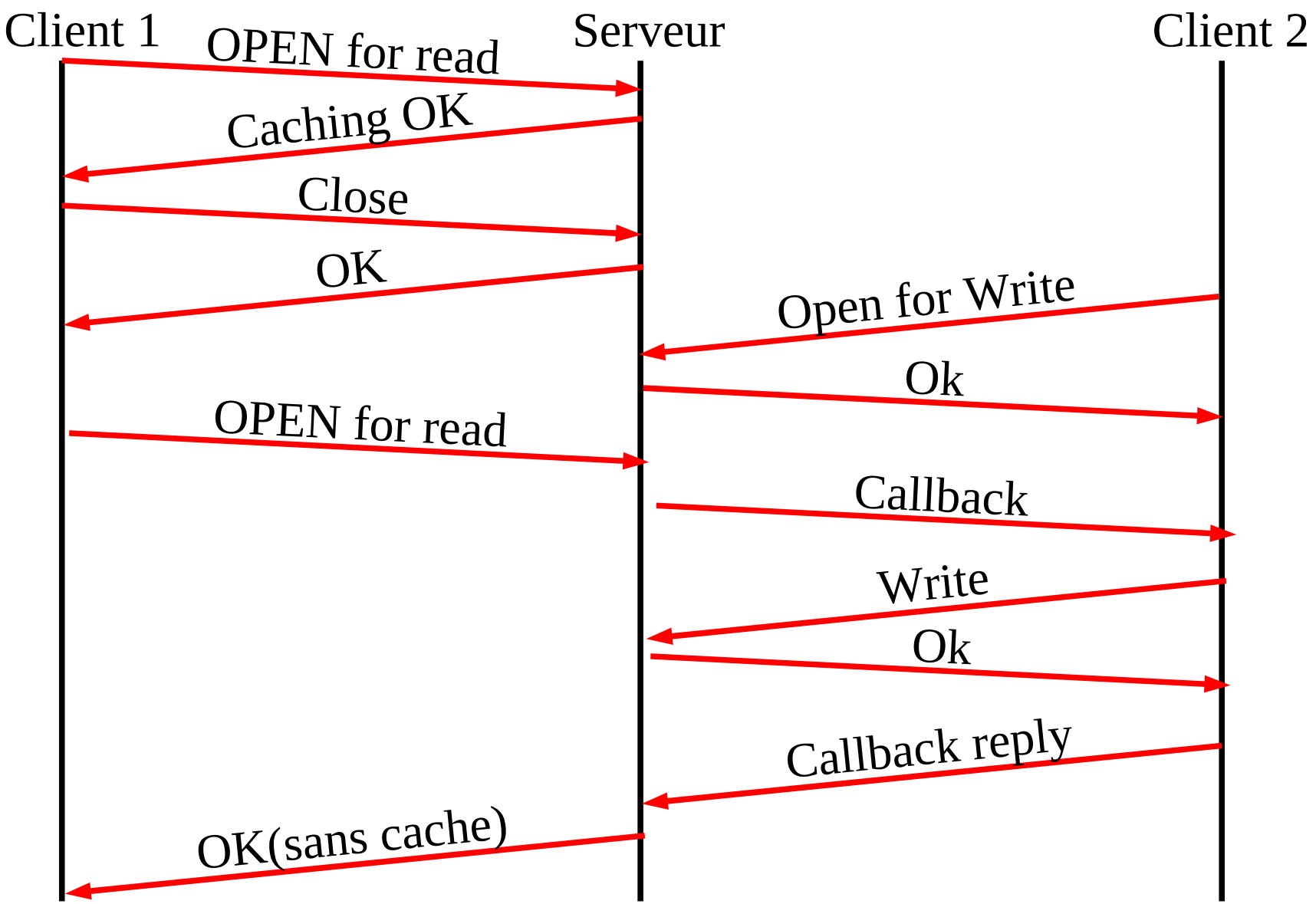
Gestion cohérence:

- Temps de dernière modification
- Temps de présence dans cache client
- Solaris:
  - Fichier: de 3 sec à 30 sec
  - Répertoire: de 30 sec à 60 sec.

Spritely NFS

- Basé sur NFS v2/ NFS v3
- Fournir une sémantique "copie unique"
- Obtenir de meilleures performances via de meilleurs mécanismes de cache.
- Addition au protocole
  - OPEN (mode) / CLOSE
  - "Callback" du serveur vers le client pour révocation de droits.

Spritely NFS



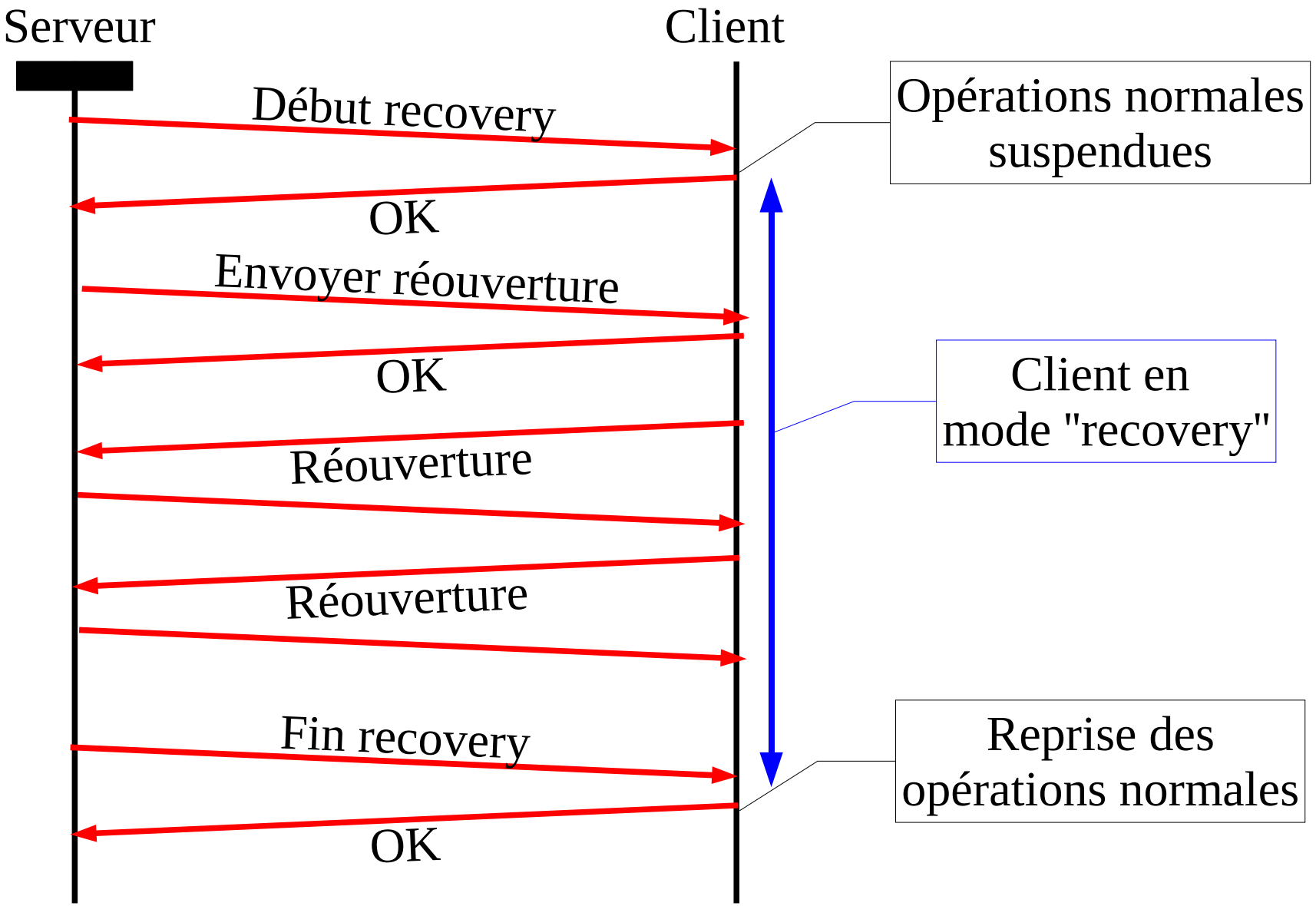
Spritely NFS

Pas de nécessité de flusher les blocs modifies par un client lors d'un close

Les séquences suivantes peuvent se derouler entièrement sur le client:

- create, write, close
- open, read, close, unlink
- Cas typique entre 2 passes d'un compilateur...

Spritely NFS: Recovery



NQNFS

Not Quite NFS

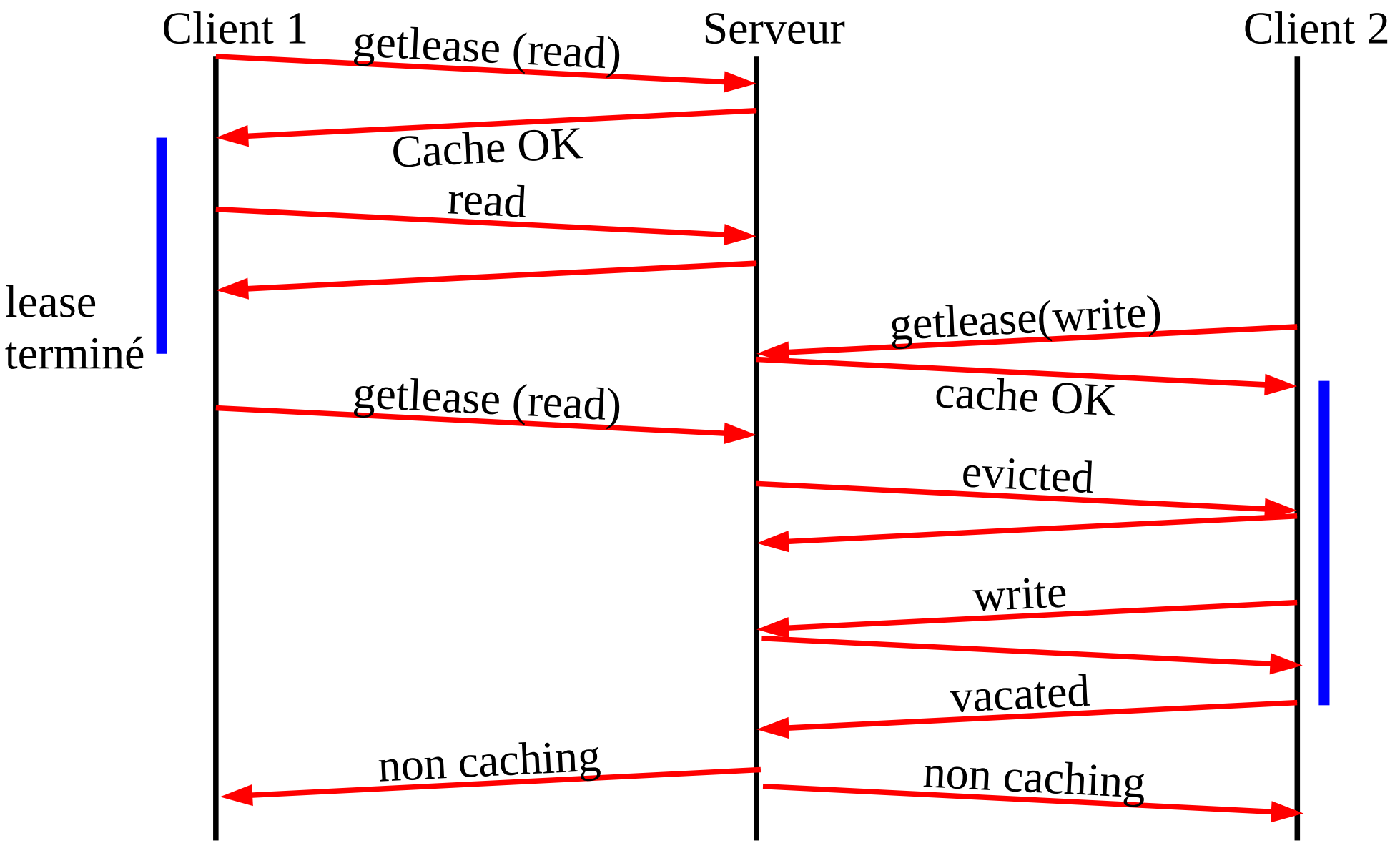
Buts identiques à ceux de Spritely NFS

- Sémantique "copie unique"
- Meilleures performances

Addition au protocole:

- GETLEASE

NQNFS


Client 1
Serveur
Client 2
getlease (read)
Cache OK
read
lease
terminé
getlease(write)
cache OK
getlease (read)
evicted
write
vacated
non caching
non caching

NQNFS: recovery

Le serveur attend simplement que les "loyers" fournis précédemment soient expirés.

- Durée du "loyer" courte (~1mn)
- Après ce délai: plus d'état

Le serveur sert cependant les requêtes d'écriture.

WebNFS

URL: nfs://server:port/home/shrek/file

Implémentation évitant la lourdeur

- du protocole de montage
- de la résolution du chemin composant par composant

Public file handle

- répertoire unique défini par le serveur
- "tout à zéro"

lookup multicomposants

NFS v4

File handle 128 octets

mknod remplacé par create

create (regular file) remplacé par open!

Nouveaux appels

- open
- close

lookup: multi-composants

Attributs fichiers retournés avec chaque réponse

Attributs "diversifiés"

NFS v4

Sécurité enrichie et renforcée

Possibilité d'opérations composées dans une requête unique (ex: lookup, open, read)

Montage et Verrouillage intégrés

Délégation

NFS v4: share reservation

Current file denial state

Request access	NONE	READ	WRITE	BOTH
READ	Succeed	Fail	Succeed	Succeed
WRITE	Succeed	Succeed	Fail	Succeed
BOTH	Succeed	Succeed	Succeed	Fail

(a)

Requested file denial state

Current access state	NONE	READ	WRITE	BOTH
READ	Succeed	Fail	Succeed	Succeed
WRITE	Succeed	Succeed	Fail	Succeed
BOTH	Succeed	Succeed	Succeed	Fail

(b)

Résultat d'un open:
(a) quand un processus demande un accès partagé
(b) quand un processus demande un déni

NFS v4: Délégation

Plan

- FTP
- Newcastle Connection
- NFS
  - NFS v2, NFS v3, WebNFS, NFS v4
  - Spritely NFS
  - NQNFS
- Coda
- GFS

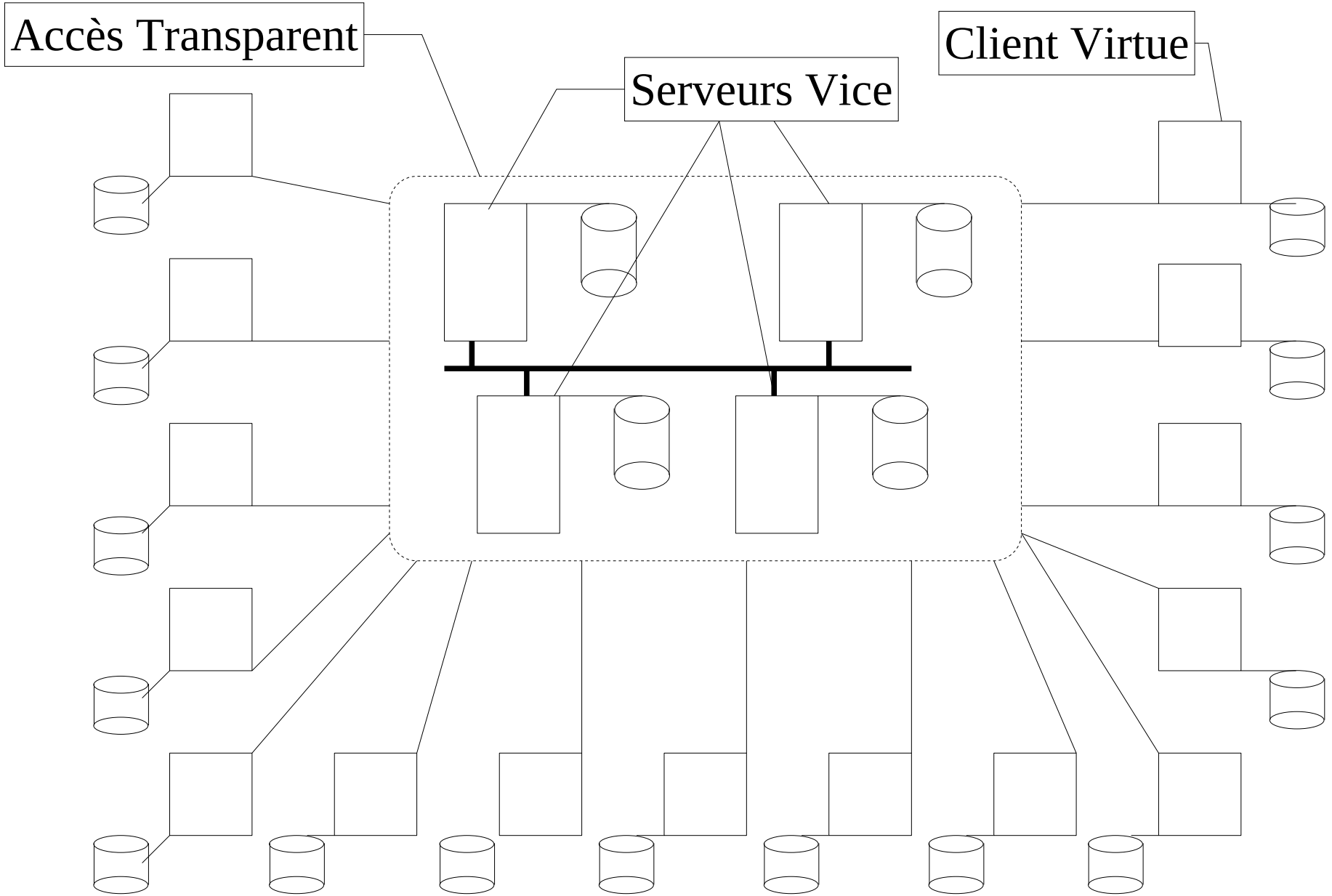
Coda

- Successeur de Andrew File System (AFS)
- Support pour clients mobiles
  - Opérations en mode déconnecté
    - Ré-intégration des clients déconnectés
- Résistance aux pannes
  - Serveurs multiples lecture / écriture
  - Résolution des conflits entre serveurs
  - Gestion des partitions réseau entre serveurs

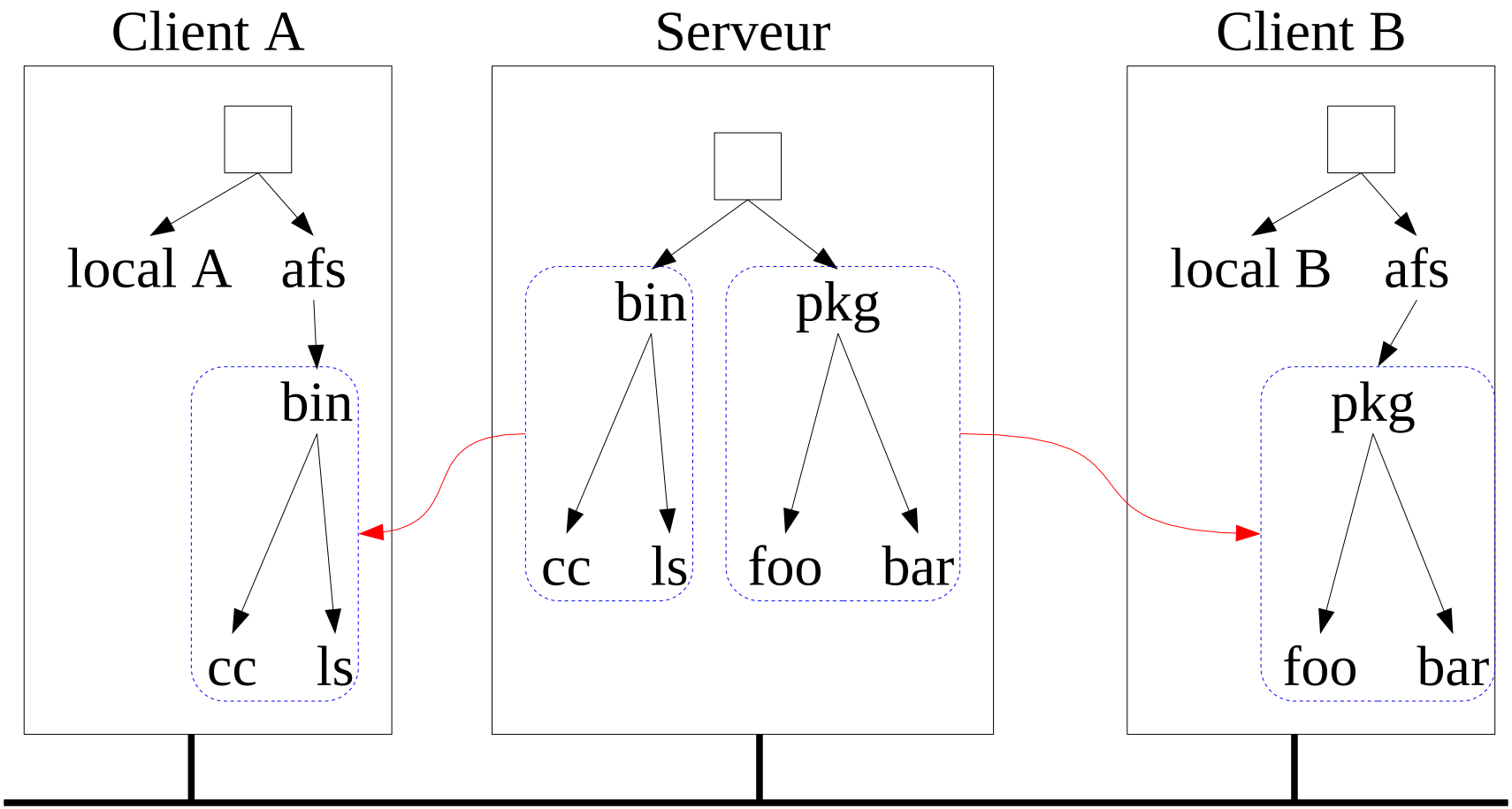
Coda

- Performances
  - Cache persistent côté client des données, répertoires et attributs
  - "write back"
- Sécurité
  - Kerberos
  - Access Control Lists (ACL's)

Coda

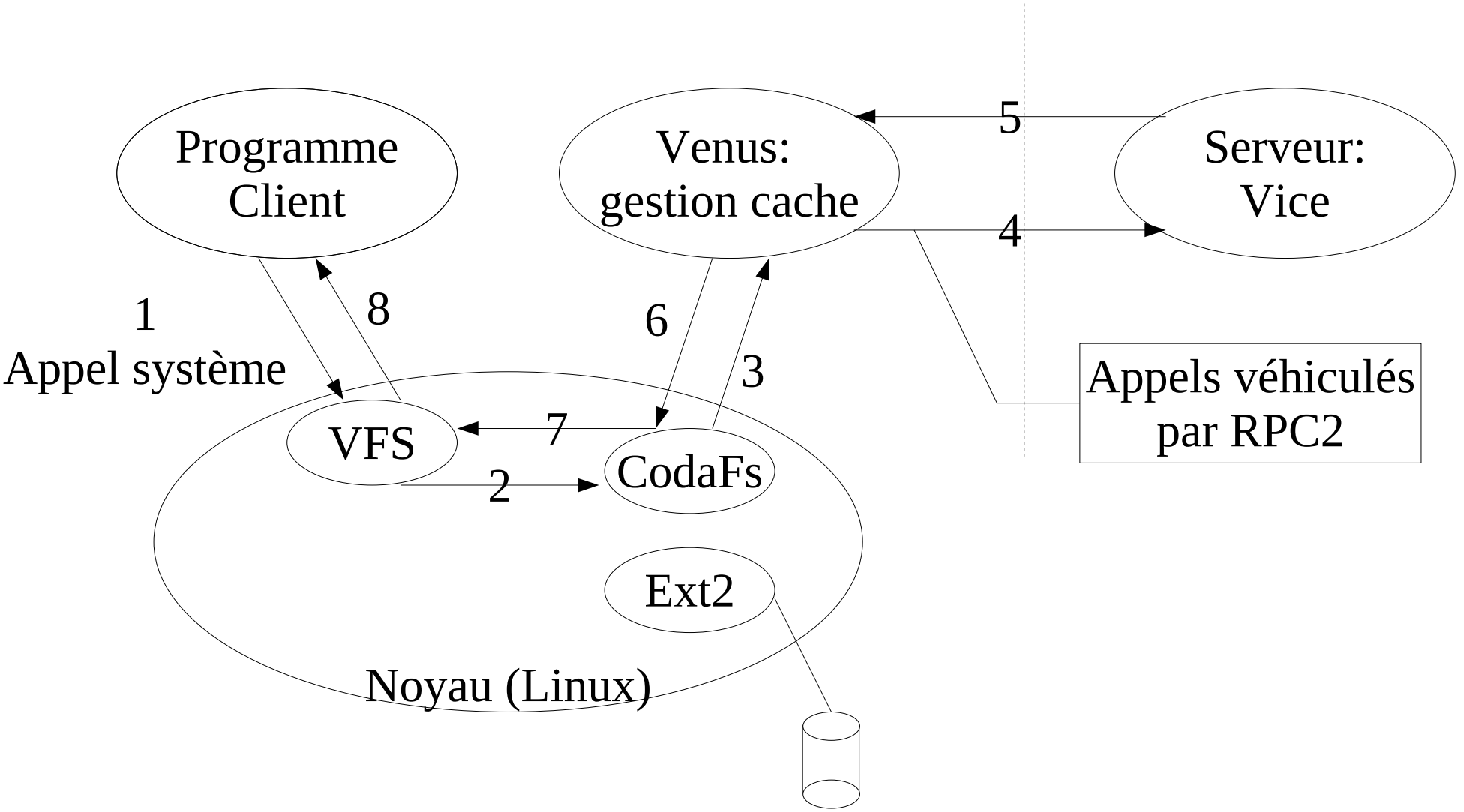


Coda: Nommage



Espace de nommage commun imposé par le(s) serveur(s)

Coda Venus et Vice



Coda

Au premier accès, Venus amène une copie du fichier sur le disque local,

Les lectures / écritures suivantes se font ensuite localement (via ext2) sans "passer" par Venus.

Si le fichier a été modifié, il est renvoyé au serveur lors de la fermeture.

Coda: Déconnexion

Les modifications sont conservées dans un journal (sur disque: Client Modification Log)

- Pas d'erreur notifiée à l'utilisateur
- Modifications ré-intégrées automatiquement lors de la re-connection
- Optimisées (ex: modifications suivies de suppression)

Les lectures de fichiers absents du cache ne peuvent pas être servies.

Coda: Déconnexion

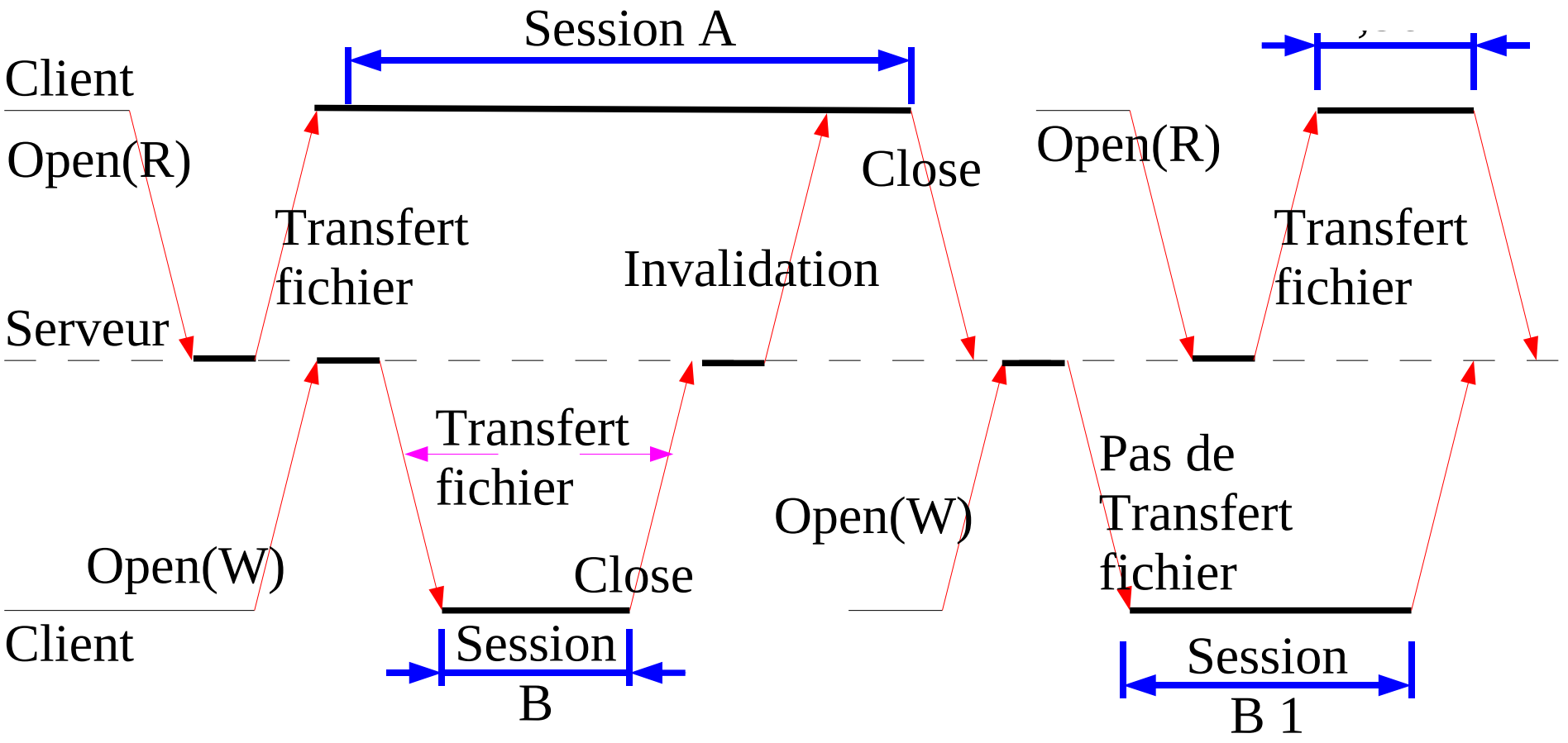
"File Hoarding"

- Cache les fichiers les plus utilisés (binaires,...), automatiquement par espionnage,
- Maintenus à jour par Venus

Conflits lors des ré-intégrations

- Réconciliation automatique ou manuelle

Coda: Partage de fichiers



Utilisation des copies locales de fichiers dans Coda

Coda: Identificateurs

Fichier: identifie par un "File Handle"

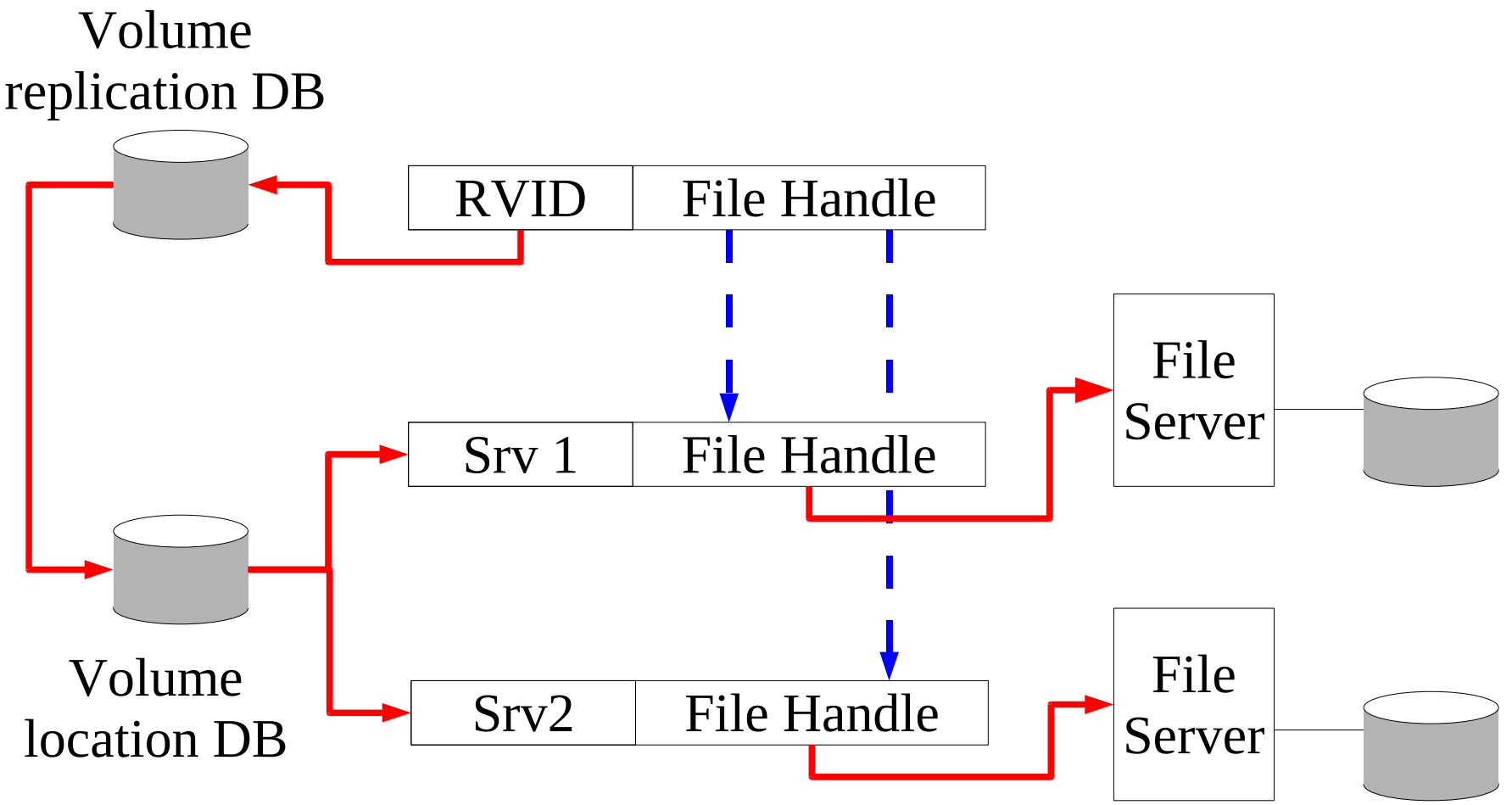
Volume physique identifié par "**VID**"

- Volume Identifier
- Géré par "Volume location DB"

Volume logique identifié par "**RVID**"

- Replicated Volume Identifier
- Géré par "Volume replication DB"

Coda: Identificateurs



Coda: Serveurs Répliqués

Volume Storage Group (VSG):

- Ensemble des serveurs ayant une copie d'un volume donné

Accessible Volume Storage Group (AVSG):

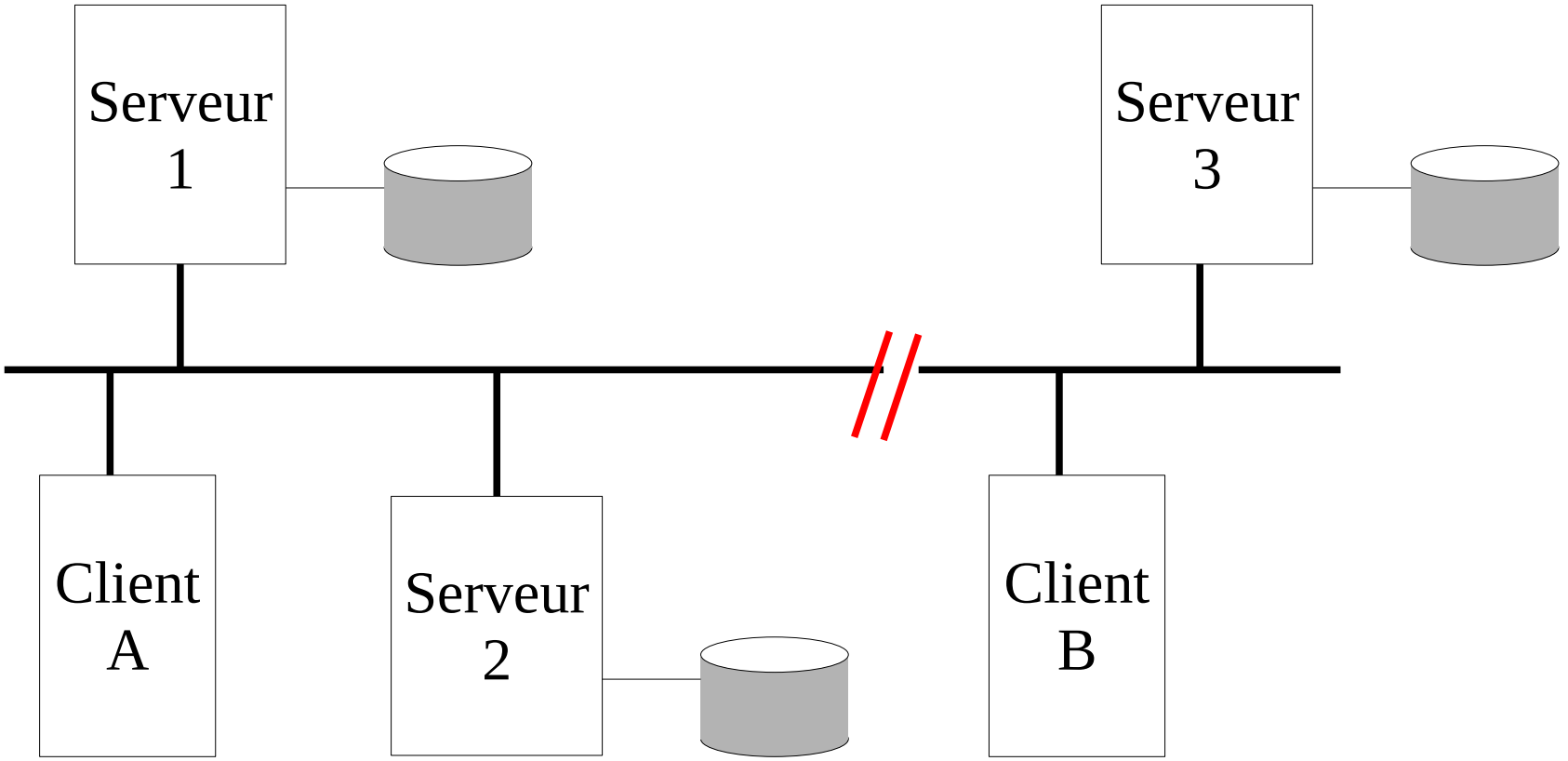
- Sous-ensemble accessible à un instant donné

Read One, Write All

Quand un serveur est en panne

- AVSG # VSG

Coda



Partition réseau: AVSG du client A disjoint de AVSG du client B

Coda: Détection des conflits

Chaque serveur gère un 'Coda version vector' pour chaque fichier (analogue aux vecteurs temporels) noté: CVVi(**f**)

A $t = 0$

$CVV_1(\mathbf{f}) = CVV_2(\mathbf{f}) = CVV_3(\mathbf{f}) = [1, 1, 1]$

Après écriture de A

$CVV_1(\mathbf{f}) = CVV_2(\mathbf{f}) = [2, 2, 1]$; $CVV_3(\mathbf{f}) = [1, 1, 1]$

Après écriture de B

$CVV_1(\mathbf{f}) = CVV_2(\mathbf{f}) = [2, 2, 1]$; $CVV_3(\mathbf{f}) = [1, 1, 2]$

Intermezzo

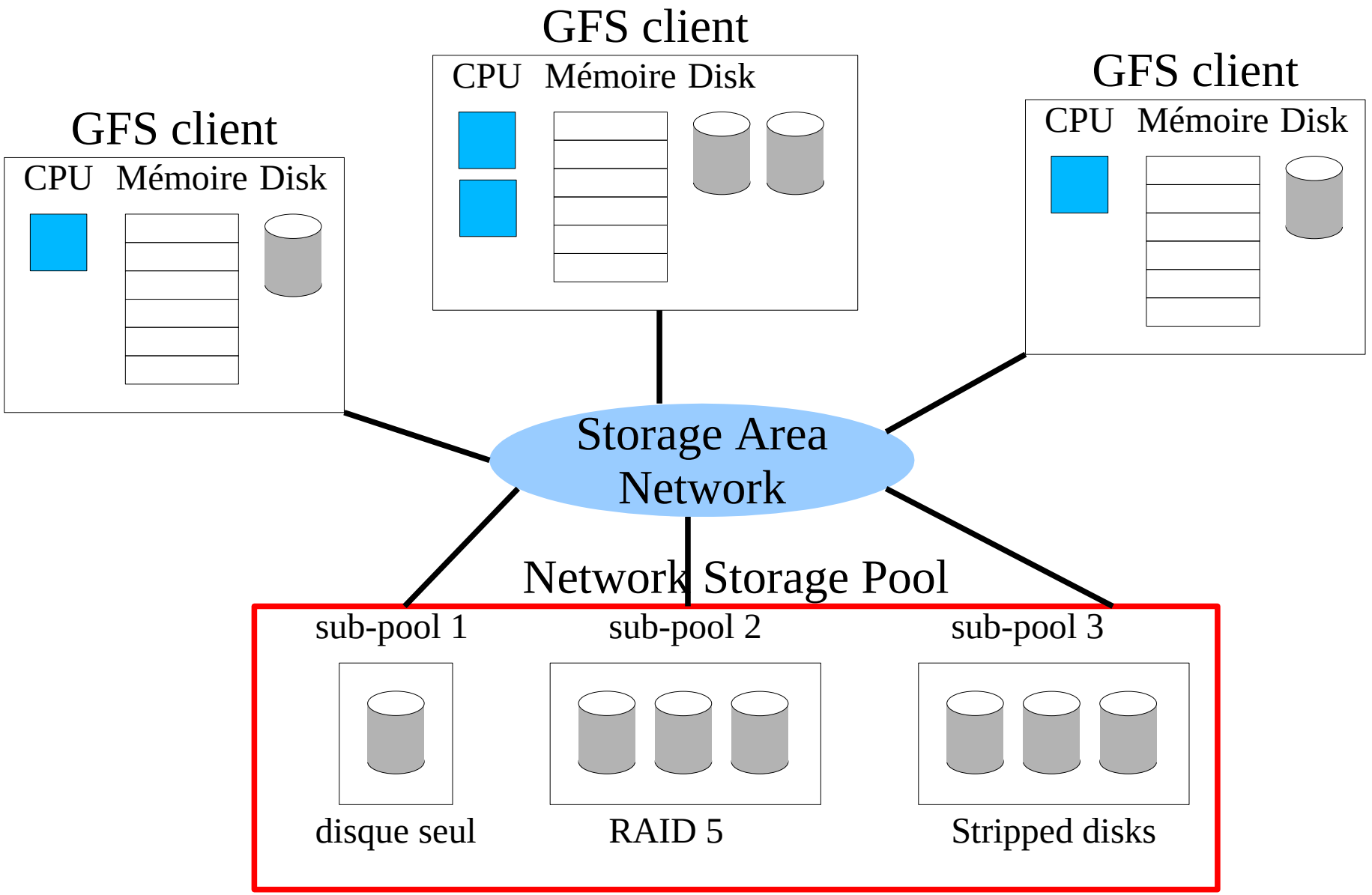
Voir aussi Intermezzo

- Mêmes capacités que Coda
- "Sur-couche" sur systèmes de fichiers existants (ext3fs)
- Protocole basé sur HTTP 1.1

Plan

- FTP
- Newcastle Connection
- NFS
  - NFS v2, NFS v3, WebNFS, NFS v4
  - Spritely NFS
  - NQNFS
- Coda
- GFS

GFS


GFS client
CPU Mémoire Disk
GFS client
CPU Mémoire Disk
GFS client
CPU Mémoire Disk
Storage Area Network
Network Storage Pool
sub-pool 1
sub-pool 2
sub-pool 3
disque seul
RAID 5
Stripped disks

GFS

Pas de "Serveur" centralisé

Disponibilité:

- Si un client tombe en panne, un autre peut continuer son travail: ils accèdent aux mêmes fichiers

Partage de charge

Gestion "unifiée" des disques

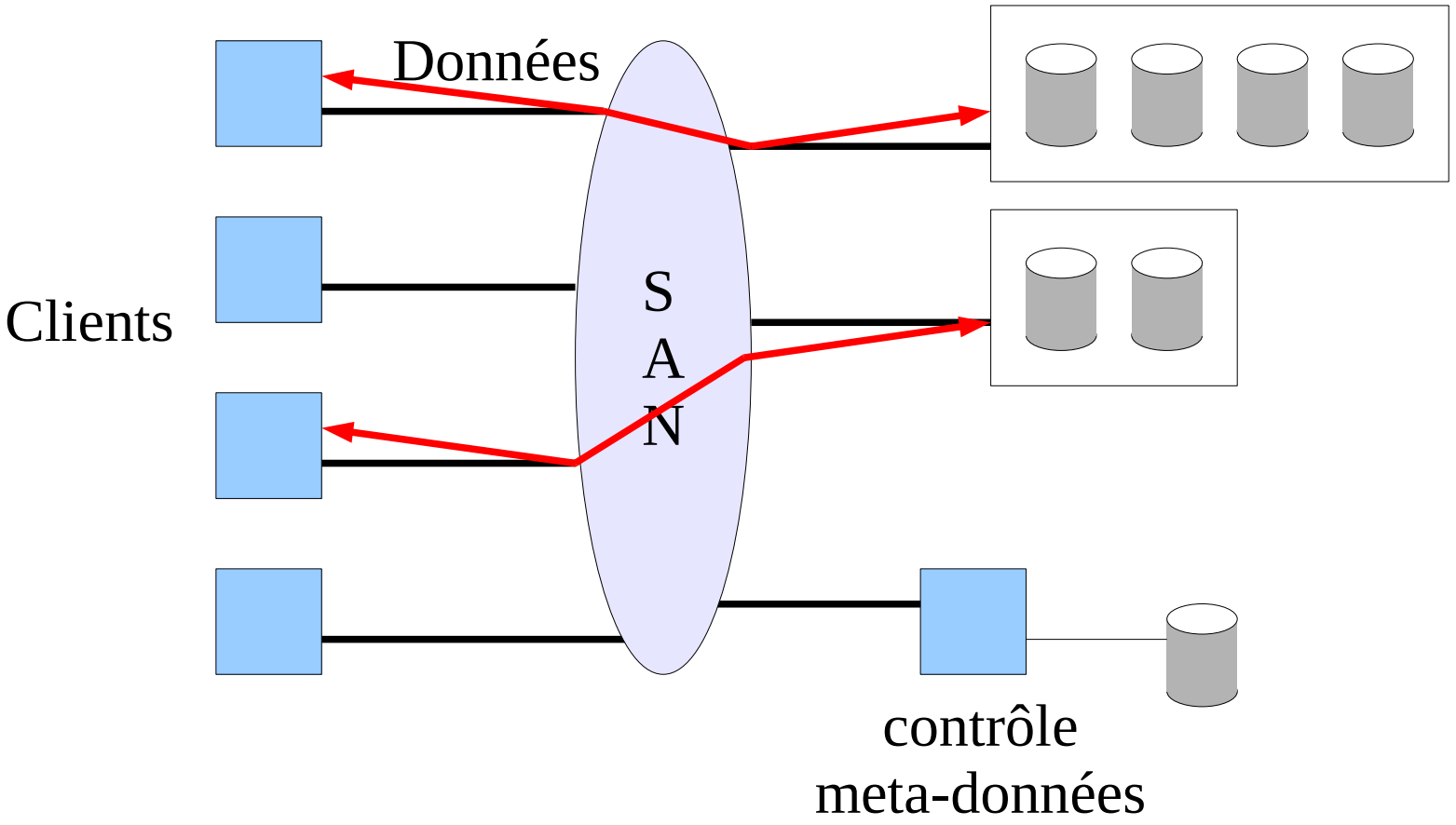
Croissance facilitée

Shared Disk File Systems

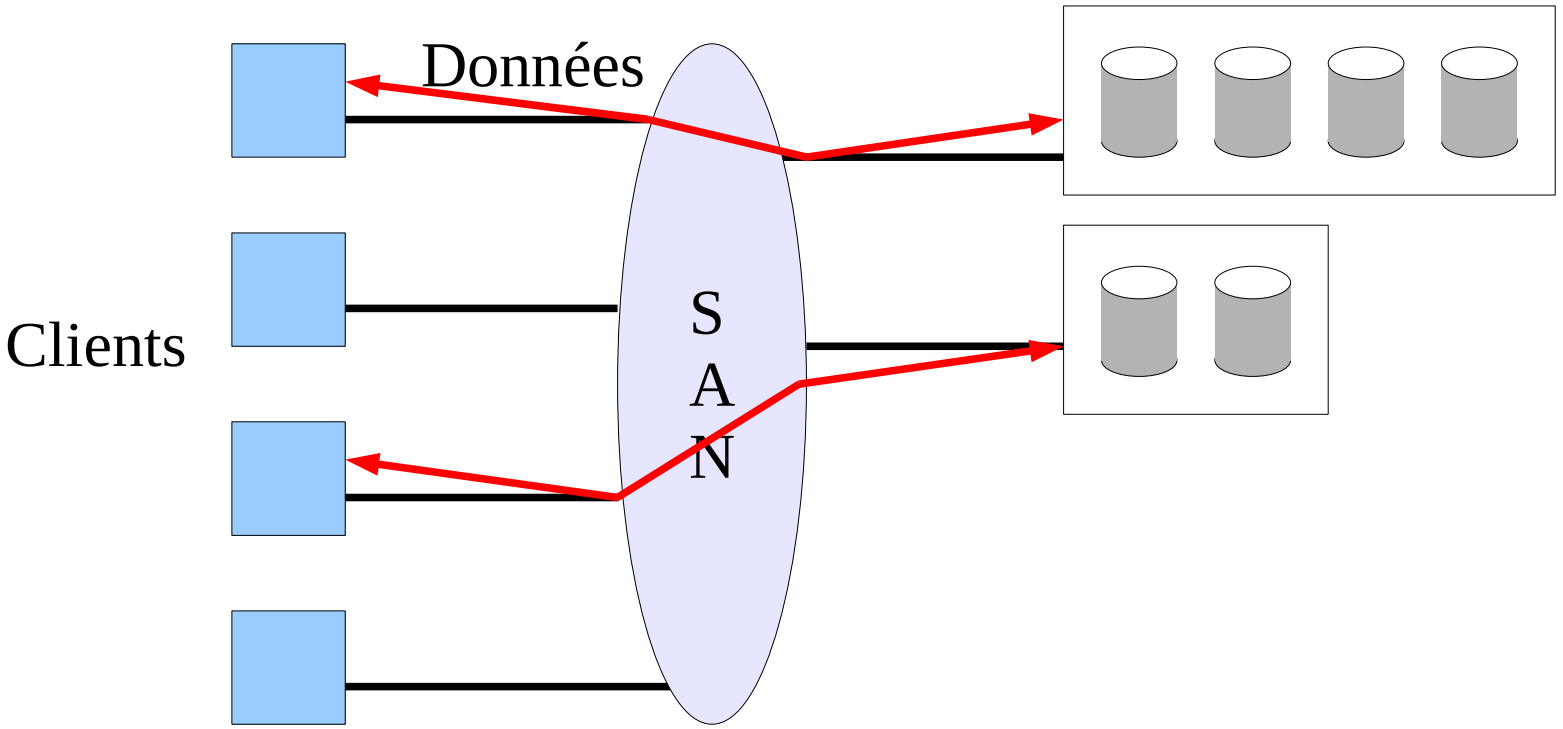
Deux approches possibles:

- Asymétrique
  - xFS
- Symétrique
  - GFS

SDFS: Asymétrique



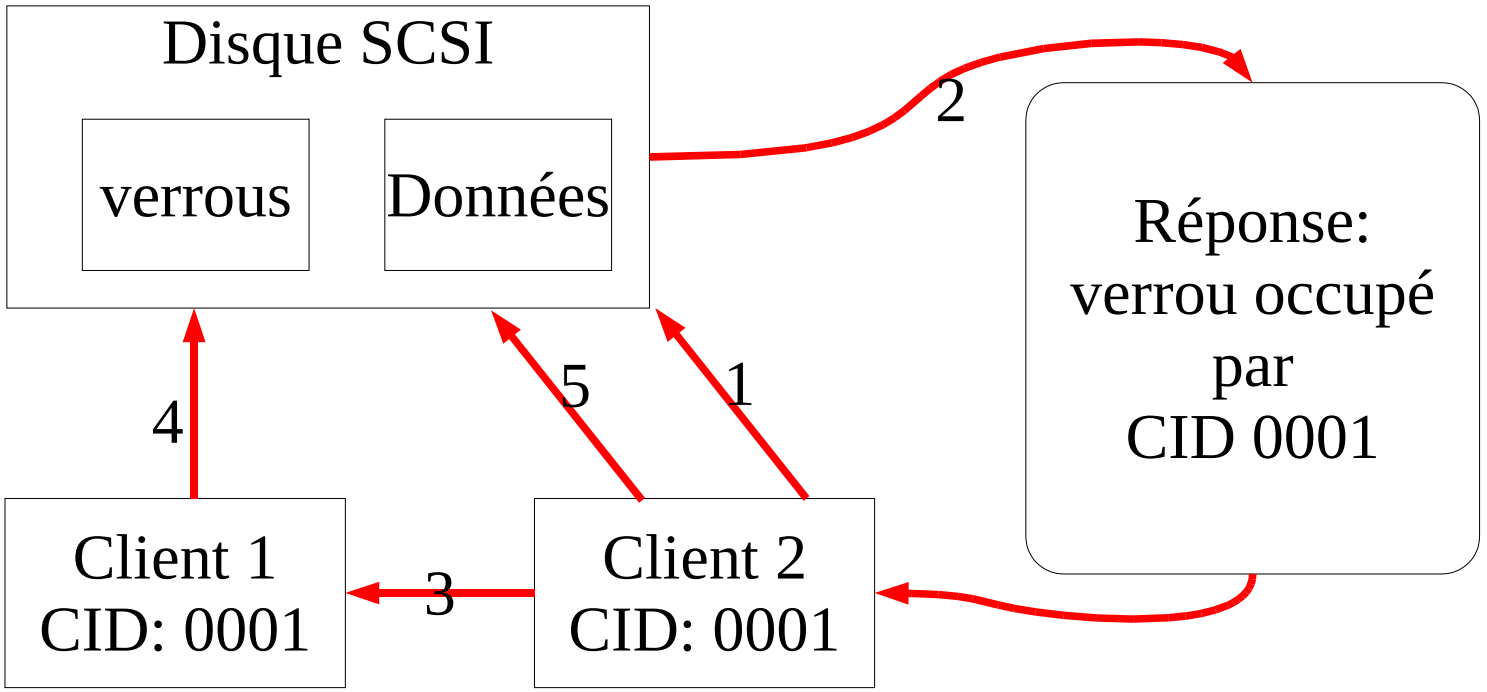
SDFS : Symétrique



GFS

- Utilise une approche "symétrique"
- Système de fichiers monte simultanément sur chacun des nœuds
- Panne d'un nœud quasi-transparente
  - Nécessité de rejouer le journal du client défectueux
- Contrainte:
  - Accès parallèle nécessite des mécanismes de verrouillage entre nœuds

GFS: Verrous



1: Demande de verrou
2: Verrou déjà pris, demander a CID 0001
3: Relâcher le verrou SVP,
4: Verrou relâché, données mises à jour
5: Prise de verrou

GFS

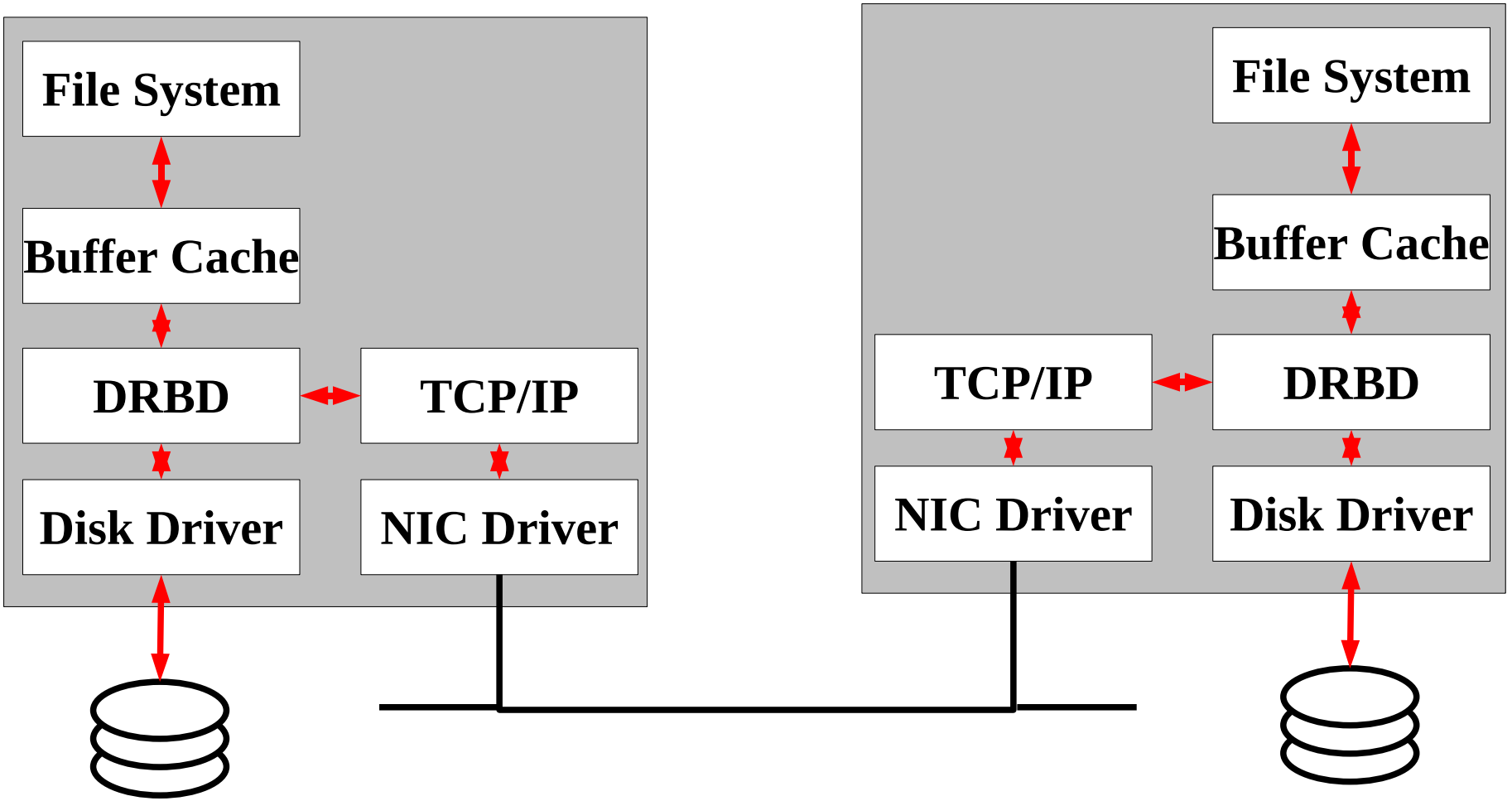
- Support pour verrouillage:
  - Disques avec protocole DMEP
    - Device Memory Export Protocol
  - Espace de verrouillage accessible via IP (ex: machine Unix)
- Verrous associés aux blocs
  - 1 inode = 1 bloc
- 1 journal par nœud client

Disque Miroir par logiciel

DRBD: Dual Remote Block Device

- Associe (RAID 1) un disque local et un disque distant
- Vu comme un disque local
- Un site "Primaire" un site secondaire
- Évolutions pour supporter état "shared" pour GFS

DRBD


File System
Buffer Cache
DRBD
TCP/IP
Disk Driver
NIC Driver
File System
Buffer Cache
TCP/IP
DRBD
NIC Driver
Disk Driver

Autres

WebDAV

- Partage et contrôle de version sur le Web

Frangipani

- Très similaire à GFS

Lustre