



Cours administration réseaux NFS et NIS domaine LINUX

Département STI, 4ème année

Année 2020_2021

Module administration de réseaux V1

M. Szpieg

NFS et NIS généralités

- Network File System (NFS) et Network Information Service (NIS) permettent la création de bases de données et de ressources réparties sur un réseau.
 - ◆ NIS
 - Permet de partager des fichiers de configuration pour les hôtes du réseau.
 - Exemples :
 - ✦ le fichier `/etc/passwd` qui contient l'ensemble des comptes utilisateurs, le chemin de leur home directory ...
 - ✦ le fichier `hosts` qui contient des correspondances (numéro ip, nom de machine) permettant la résolution de noms fqdn .

NFS et NIS généralités suite

- ◆ NFS est un système de fichiers répartis. Pour les clients, les répertoires distants apparaissent comme des disques locaux.
Toutes les commandes locales agissant de manière transparente sur ces ressources partagées.

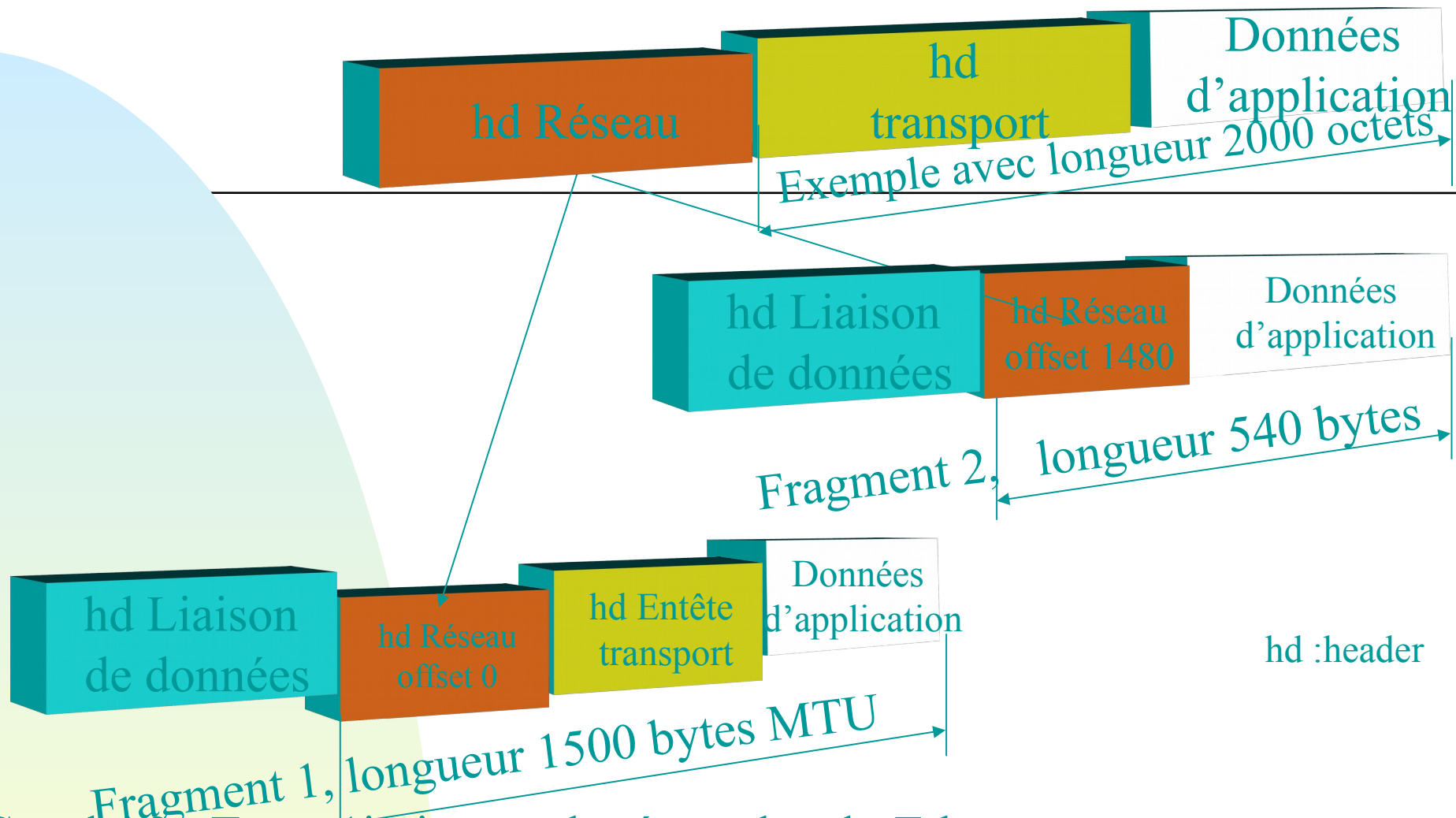
Exemple : `ls /promo2018/pdurand`

NFS et NIS généralités suite

- ◆ NIS a pour principal tâche de gérer des informations de configuration et de maintenir leur cohérence sur le réseau.
Les clients NIS font des requêtes vers les serveurs au lieu d'utiliser leurs propres copies locales.
- ◆ NFS s'appuie sur cette structure de base .

NIS généralités

- Base de données
 - ◆ Inventée par Sun Microsystems en 1985
 - ◆ Commune à un réseau
 - ◆ Dépendant d'un domaine NIS (**≠ domaine DNS**)
 - ◆ Organisée en « cartes » (*maps*) ou tables
 - ✦ Ensembles thématiques
 - ✦ 1 ligne ⇔ 1 entrée ⇔ 1 clé



Couche 2 : Transmission sur le réseau local : Ethernet

Couche 3 : Routage, segmentation : IP

Couche 4 : Ouverture fermeture de flux, port, séquençement : TCP

Couche 5 : Données d'application : Ex. http

12/09/07

NFS et NIS et leur intégration dans les couches réseau

(le fonctionnement des couches 1 à 4, la partie réseau est considérée comme acquise)

Nom de la couche	Protocole
Application	NFS et NIS
Présentation	XDR
Session	RPC
Transport	UDP ou TCP
Réseau	IP
Liaison	Ethernet
Physique	//

Les couches sessions et présentation (5 et 6)

La couche session régie la création et la durée de vie des connexions réseau. Le protocole utilisé sur cette couche est RPC « Remote Procedure Call ».

La couche présentation gère le format des données transmises sur cette connexion. Le protocole utilisé est XDR « eXternal Data Representation ». Cette couche est nécessaire car on ne manipule plus des données au niveau des octets mais des structures et des types plus élaborés float, chaînes de caractères, tableau, structure ...)

La couche présentation utilisée par NFS et NIS (XDR)

Elle permet des échanges de données structurées entre des machines de type hétérogène.

XDR permet de mettre sous une forme canonique les données structurées :

- ordre des octets, par exemple pour un int octet le plus significatif d'abord , le moins en dernier.
- Représentation des nombres à virgules flottantes
- Tableaux chaînes de caractères, structure, etc.

Le mécanisme RPC

- Basé sur le modèle client serveur :
 - ◆ RPC (Remote Procedure Call) :
Permet d'utiliser une procédure distante comme si elle se trouvait sur le calculateur client.
 - ◆ La plupart du temps RPC utilise le protocole transport UDP car la transaction consiste en l'envoi de paramètres sur le serveur distant et du retour du résultat de la procédure (Ex . mkdir toto). Puis la session se termine par libération de la ressource serveur et du client.
 - ◆ En cas de non réponse, le client peut réémettre la requête vers le serveur ou chercher un autre serveur.

Prise de contact avec le serveur RPC

- Comment le client contacte le bon service ?
 - ◆ En général (pas pour RPC) les serveurs sont « standalones » ou gérés sur les systèmes UNIX par un super démon inetd (Internet démon) ou xinetd(version plus récente).
Celui ci attend les requêtes des clients sur les ports spécifiés dans /etc/services et pour les services définis dans (/etc/inetd.conf). Lorsqu'une requête arrive sur l'un de ces ports inet.d active le démon concerné ftp,telnet rlogin....
 - ◆ A chaque requête d'un nouveau client un nouveau démon est lancé (On parle de services multi-tâches) Ceci évite d'avoir un grand nombre de deamons (standalone) attendant sur chacun des ports et de mobiliser des ressources.

Prise de contact avec le serveur RPC

- Comment le client contacte-t-il le bon service ?
 - Les services RPC demandant une grande réactivité, utilise un autre démon « portmap »
 - Le serveur (Ex nfsd) s'enregistre au près du portmapper
 - Un client ayant besoin d'effectuer un appel de procédure doit obtenir l'adresse (Le port) du service.

Appel au démon portmap de la machine sur laquelle se trouve le serveur.

Démon portmap a un numéro de port connu et fixe (111).
 - Le portmapper renvoie ce numéro de port
 - Le client appelle le serveur

Principe de NIS

- Network Information Service permet de gérer des ressources réparties, pour cela des fichiers de configuration sont partagés.
- Quelques fichiers partagés

/etc/bootparams	Information pour les hôtes sans disques
/etc/ethers	Résolution N° MAC et nom (ARP)
/etc/group	Groupes d'utilisateurs EX : promo2017
/etc/aliases	Alias et liste d'utilisateurs pour le mail
/etc/netgroup	Définit des groupes d'utilisateurs et de machines
/etc/netmasks	Pour la gestion de sous réseaux
/etc/passwd	Les différents comptes utilisateurs, home directory
/etc/protocols	Donne la correspondance entre un n° de protocole et son nom
/etc/rpc	Identificateurs des programmes RPC
/etc/services	Table des ports réseaux affectés et des services

Etc.

Créer un serveur maître

- La commande qui permet de transformer un serveur unix en serveur NIS est `ypinit` .
 - ◆ Avant l'utilisation de cette commande vérifier que le serveur est sur un domaine (commande `domainname` sans paramètre). Si aucun domaine apparaît taper : `domainname nom_du_domaine`.
 - ◆ Puis `ypinit -m` où « -m » indique master. Ceci crée un ensemble de fichiers de données qui seront partagés et qui sont générés à partir des fichiers de la diapositive précédente.
 - ◆ Ces fichiers seront stockés sur le serveur dans `/var/yp/nom_du_domaine`.

Créer un serveur maître

◆ Exemple : de fichiers générés

Nom de table	Abrév.	Indexée par	Contenu	Intégration
bootparams		Nom de site	/etc/bootparams	Concaténation
ethers.byname	ethers	Nom de site	/etc/ethers	Remplacement
ethers.byaddr		Adresse MAC	/etc/ethers	Remplacement
group.byname	group	Nom de groupe	/etc/group	Concaténation
group.bygid		GID	/etc/group	Concaténation
hosts.byname	hosts	Nom de site	/etc/hosts	Remplacement
hosts.byaddr		Adresse IP	/etc/hosts	Remplacement
mail.aliases	aliases	Nom d'alias	/etc/aliases	Concaténation
mail.byaddr		Alias expansé	/etc/aliases	Concaténation
netgroup.byhost		Nom de site	/etc/netgroup	Remplacement
netgroup.byuser		Nom d'utilisateur	/etc/netgroup	Remplacement
netid.byname		Nom d'utilisateur	UID & GID	Dérivation
netmasks.byaddr		Adresse IP	/etc/netmasks	Remplacement
networks.byname		Nom de réseau	/etc/networks	Remplacement
networks.byaddr		Adresse IP	/etc/networks	Remplacement
passwd.byname	passwd	Nom d'utilisateur	/etc/passwd	Concaténation
passwd.byuid		UID	/etc/passwd	Concaténation
protocols.bynumber	protocols	Numéro de port	/etc/protocols	Remplacement
protocols.byname		Nom de protocole	/etc/protocols	Remplacement
rpc.bynumber		Identificateur RPC	/etc/rpc	Remplacement
services.byname	services	Nom de service	/etc/services	Remplacement
ypservers		Nom de site	Nom serveurs NIS	Remplacement

Créer un serveur maître

■ Les bases de données partagées

Fichiers remplacés et fichiers concaténés

- ◆ les fichiers remplacés comme `host.byname` vont remplacer les fichiers locaux lorsque NIS fonctionne, seuls ces fichiers seront consultés par le client NIS
- ◆ Fichiers concaténés, les données contenues dans ces fichiers seront ajoutés aux fichiers locaux

◆ Exemples des fichiers de l'école `/var/yp/insacvl`

`group.bygid group.byname hosts.byaddr hosts.byname mail.aliases`

`netid.byname passwd.byname passwd.byuid protocols.byname`

`protocols.bynumber rpc.byname rpc.bynumber services.byname`

`ypservers`

- ◆ Attention pour manipuler ces fichiers utilisez exclusivement `ypxfr` et `yppush` (Fichiers à trous!!!)

Créer un serveur esclave

- La commande est :
`ypinit -s nom_du_serveur_maître`
les tables sont alors dupliquées du serveur maître vers le serveur esclave.
- La fonction `yppush` (sur le serveur maître) permet de forcer cette mise à jour.
- La commande `ypxfr` (sur l'esclave) permet la mise à jour des tables du serveur esclave à partir des tables du serveur maître.
Ne pas oublier de régler le `crontab` pour que le serveur esclave mette à jour périodiquement ces tables (mode push) ou l'esclave consulte le maître (mode pull)

Définition d 'un domaine

- Les hôtes qui utilisent le même ensemble de tables générées par un serveur NIS, tables se trouvant généralement dans le répertoire, forment un domaine NIS :
`/var/yp/nom_du_domaine` forme un domaine.
- Attention le terme domaine est souvent utilisé en informatique pour désigner des entités différentes.

Le format DBM

- Ce format est composé d'enregistrements formés d'un couple clé/valeur.

Cette organisation est constituée de deux fichiers
« une table de hachage des index » (extension
« .dir ») et un fichier de données (extension « .pag »)

Une table contient les deux fichiers.

Pour permettre un accès rapide, le fichier de données peut contenir des blocs de données vides et le rend incompatible avec la copie de type classique (« commande cp,mv etc. ») donc n'utiliser la copie qu'au travers des commandes ypxfr et yppush prévues à cet effet.

Les tables NIS

- On appelle « table NIS » un ensemble de données homogène distribué par un serveur NIS pour compléter ou remplacer les données d'un client.
 - ◆ Exemple passwd.byuid
- Ces tables sont sur le serveur maître, et sont copiées sur le(s) serveur(s) esclave(s)
- Elles sont au format « dbm », DataBase Management. Format développé dans le cadre de UNIX BSD
- Les fichiers du serveur maître qui serviront à la génération de ces tables sont indiqués dans le fichier `/var/yp/Makefile`.
On peut enlever ou ajouter des fichiers à traiter dans ce fichier.

Les tables NIS

- Les fichiers de base du serveur maître sont au format ascii, les tables au format « dbm » donc on utilise « dbmmake » pour en faire la traduction.
- Les tables obtenues sont aussi parfois nommées les « DBM maps »
- Pour manipuler ces tables au format DBM, la bibliothèque standard ndbm est fournie aux programmeurs en langage « C »

Les tables NIS

- Si une table NIS remplace un fichier local, le fichier local est donc ignoré pendant le fonctionnement de NIS.
- Si une table NIS est concaténée à un fichier local, les données locales sont toujours consultées en premier.

Les tables NIS, la table netgroup

- Nis permet la création d'ensemble d'utilisateurs et de sites par l'intermédiaire de cette table.
- On peut, par exemple, définir un ensemble d'utilisateurs dérivés de la table passwd.byname pour leur donner un accès à une ressource spécifique. On forme ainsi un groupe réseau.
- Un groupe réseau est composé de un ou plusieurs triplets de la forme :
(nom de site, nom d'utilisateur, nom de domaine)
le nom de domaine contient le nom de domaine NIS pour lequel ce groupe est défini.
- Exemple de groupe :
admin_insacvl (-,lesage,insacvl),(-,hermance,insacvl),
...

Les tables NIS, la table netgroup

- Le nom de site peut ne pas avoir de sens suivant le contexte, par exemple, pour ajouter des utilisateurs à un fichier de mot de passe, seuls les noms d'utilisateurs ont un sens.
Dans ce cas un élément de triplet peut ne pas être spécifiés et être remplacé par le caractère « - »
- Attention à ne pas confondre ces groupes
« réseaux » définis par la table netgroup et le contenu des fichiers /etc/group qui donnent de manière explicite des droits aux membres d'un groupe localement.
Exemple /etc/group : bde:*:100:julien,laure,marie
- Un groupe réseau, est une abréviation, qui peut être utilisée à chaque fois qu'un nom d'utilisateur ou de site apparaît

Ajout table NIS aux fichiers locaux

- Le fichier `/var/yp/Makefile` contient le numéro d'UID à partir duquel les comptes sont exportés dans le domaine NIS

Cas particulier du fichier passwd

- /etc/passwd du client peut-être administré de manière plus fine :
 - ◆ Par entrée utilisateur :
+laure:*:
Cette ligne n'ajoutera que l'utilisatrice « laure »
 - ◆ Par mélange de champs :
+laure*:0:0:admin machine::
« admin machine » écrasera les données NIS
 - ◆ Insertion de données de la table « netgroup »
+@admin_insacvl
 - ◆ Détruire une entrée (cette info est mise en premier dans le fichier)
-@utilisateurs-dangereux ou -laure*:1234:21::

Priorité fichier local et table NIS

- Généralement le fichier est lu de manière séquentiel, aussi dès que l'information est pertinente la recherche s'arrête.
- Donc, ne laisser que le strict minimum dans les fichiers locaux.
- Le fichier passwd exemple 1
+laure:x:*:*:laure dupont:/home/laure:/bin/bash
 - ◆ Les UID et GID de la table NIS seront transmises mais les données homedirectory, nom complet, shell locaux écraseront les données de la table NIS

Priorité fichier local et table NIS

- Le fichier passwd exemple 2
+laure:hash_passwd:1234:34:dupont:/laure:/bin/bash
 - ◆ L'UID et le GID obtenus par « laure » ne correspondront pas aux valeurs données par le serveur NIS et présence du mot de passe hashé

Serveur NIS multi-domaines

- Un serveur NIS, est un serveur de données qui répond aux requêtes client pour toutes données se trouvant au format « dbm » dans le répertoire par défaut /var/yp.
- Dans ce répertoire un domaine apparaît sous la forme d'un sous répertoire :
Ex : /var/yp/insacvl
- Conclusion, si l'on veut rendre notre serveur multi-domaines il suffit de rajouter les sous-répertoires avec leurs bases de données en conséquences.

Services NIS et démons

- Démon sur le serveur

Un seul démon donne accès au service NIS son nom est ypserv

Un serveur NIS fournit un accès à un certain nombre de procédures classées en trois familles :

- ◆ procédures à l'usage des clients
type match de recherche à partir d'une clé
- ◆ procédures de négociation pour la maintenance des tables.
- ◆ procédures NIS internes
permet la mise à jour par transfert des tables entre serveur maître et serveur(s) esclave(s)

Services NIS et démons

- Démon sur le client
Un démon donne accès au domaine NIS pour un client c'est ypbind
- Lors du lancement d'un hôte, ypbind cherchera sur le réseau un serveur du domaine auquel il appartient. Un client peut appartenir à plusieurs domaines à la fois.
- ypwich affiche les serveurs auxquels l'hôte est lié.

Services NIS et démons

- Le démon `yppasswd` permet à un utilisateur quelconque du domaine de modifier son mot de passe sur le serveur maître. Il est donc démarré sur le serveur maître.
- Sur les postes clients NIS le fichier `/etc/passwd` contient la ligne suivante :
`+:0:0::`
Cette ligne indique que le client ajoutera en dynamique les comptes supplémentaires émanant d'un serveur NIS (pas utile dans toutes les distributions).
Donc elle ne doit pas apparaître dans le fichier des serveurs.

NFS généralités

- Network Files System (NFS)
permet pour le client d'avoir un accès transparent sur des disques distants.
 - NIS permet, entre autres, la centralisation des informations utilisateurs.
 - NFS permet la centralisation des accès disques
- NFS et NIS sont donc employés généralement en même temps.
 - NFS évite la duplication sur toutes les machines du répertoire /home avec tous les problèmes de synchronisation que cela implique.
 - Les ressources distantes ne peuvent être distinguées par l'utilisateur des ressources locales

NFS généralités

- Inventé par Sun Microsystems en 1989
- RFCs : 1094, 1813 (V3), 3010 (V4)
- Utilise les RPC
- Un client NFS
 - Envoie des commandes
 - Reçoit des réponses

NFS et RPC

- NFS utilise comme NIS le protocole RPC basé sur le principe client-serveur
- Un serveur NFS est un ordinateur qui partage un ou plusieurs systèmes de fichiers. (Ex : datas à l'INSA-CVL)
- Un client NFS peut utiliser un ou plusieurs systèmes de fichiers distants, ressources venant d'un ou plusieurs serveurs NIS ou pas. On peut monter un partage NFS sans pour autant disposer d'un serveur NIS.

Convention de nommage

- Si on avait développé des applications en local dans le répertoire `/usr/local/bin` et si on utilise NFS pour distribuer ces applications sur tout le réseau, il faut que le système distant apparaisse dans `/usr/local/bin`.
- Le disque qui contient le système de fichiers n'a pas d'importance.
- Attention des problèmes sur les liens symboliques se posent.

Installation de NFS coté serveur

- Les démons `nfsd` `rpc.mountd` `rpc.lockd` et `rpc.statd` `rpc.quota` doivent être actifs
 - NFS utilise les couches XDR et RPC
Pour rendre l'accès transparent le noyau utilise VFS puis un second niveau dans VSF utilise une notion de nœud virtuel `vnode` (virtual node) permettant d'émuler le comportement classique d'un système de fichiers UNIX sur des systèmes autres (EX `vfat` du DOS qui n'a pas la notion de propriétaire de fichiers, ni de liens entre autres).
 - `nfsd` permet l'accès à 16 procédures classées en 4 catégories : (opération sur: les répertoires, les fichiers, les liens symboliques, les systèmes de fichiers)

Exportation ressources NFS

- Le fichier `/etc/exports` du serveur contient les ressources mis à disposition sur le réseau
- Fichier `/etc/exports`
 - 1 ligne -- 1 répertoire partagé
 - Répertoire machine (droits) machine (droits) ...
- Exemple :
 - `/usr/doc client[1-3] (rw)`
 - `/home/durand/ client1(ro) client2(rw)`

Le répertoire `/home/durand` peut être monté à partir de la machine `client1` en lecture seule et par `client2` en lecture écriture.

Exportation ressources NFS

- Lorsque le service nfs démarre, la commande `/usr/sbin/exportfs` lit ce fichier, puis fournit à `rpc.mountd` et `rpc.nfsd` les systèmes de fichiers disponibles aux utilisateurs distants.
- Lorsqu'elle est lancée manuellement, la commande `/usr/sbin/exportfs` permet au super-utilisateur d'exporter ou de retirer sélectivement des répertoires sans redémarrer le service NFS.

Montage NFS client

- Montage manuel

Le format de « mount » sur le client est :

```
mount -o <options> <serveur>:</remote/export>  
      </local/directory>
```

- Montage systématique

Dans le fichier /etc/fstab ajouter la ligne :

```
<server>:</remote/export> </local/directory> nfs  
<options> 0 0
```

- Montage opportuniste :

On utilise l'utilitaire nommé automount, qui montera et démontera les systèmes de fichiers NFS automatiquement, économisant ainsi des ressources. Le fichier auto.master contient des lignes se référant à chacun de ces points de montage.

Montage NFS client

- Montage opportuniste suite :
Le fichier de configuration principal
 - « /etc/auto.master » liste des points de montage
<mount-point> <map-type>
EX : /home /etc/auto.home
 - Puis le fichier /etc/auto.home contiendra
* servernfs:/home/&
- Si le démon autofs est lancé alors le répertoire :
/home/durand sera monté si l'utilisateur Durand se connecte.

Installation de NFS côté serveur

- ◆ Toutes ces opérations sont réalisées par le noyau, en passant par le démon `nfsd` pour rendre le système plus réactif on lance plusieurs démons `nfsd`.
- ◆ Le protocole NFS est « sans état », c'est à dire que le client conserve les informations nécessaires pour accéder au serveur mais celui-ci ne garde pas un historique des requêtes de chaque client.
- ◆ Le démon `rpc.lockd` permet à un processus d'avoir accès à un fichier ou a une partie d'un fichier. NFS étant sans état, il convient au démon `rpc.statd` de s'occuper des réservations de ressources. Pour mettre un verrou sur un fichier, le démon `rpc.lockd` client demande à `rpc.lockd` du serveur de noter et de suivre le verrouillage `/var/lib/nfs/sm`.

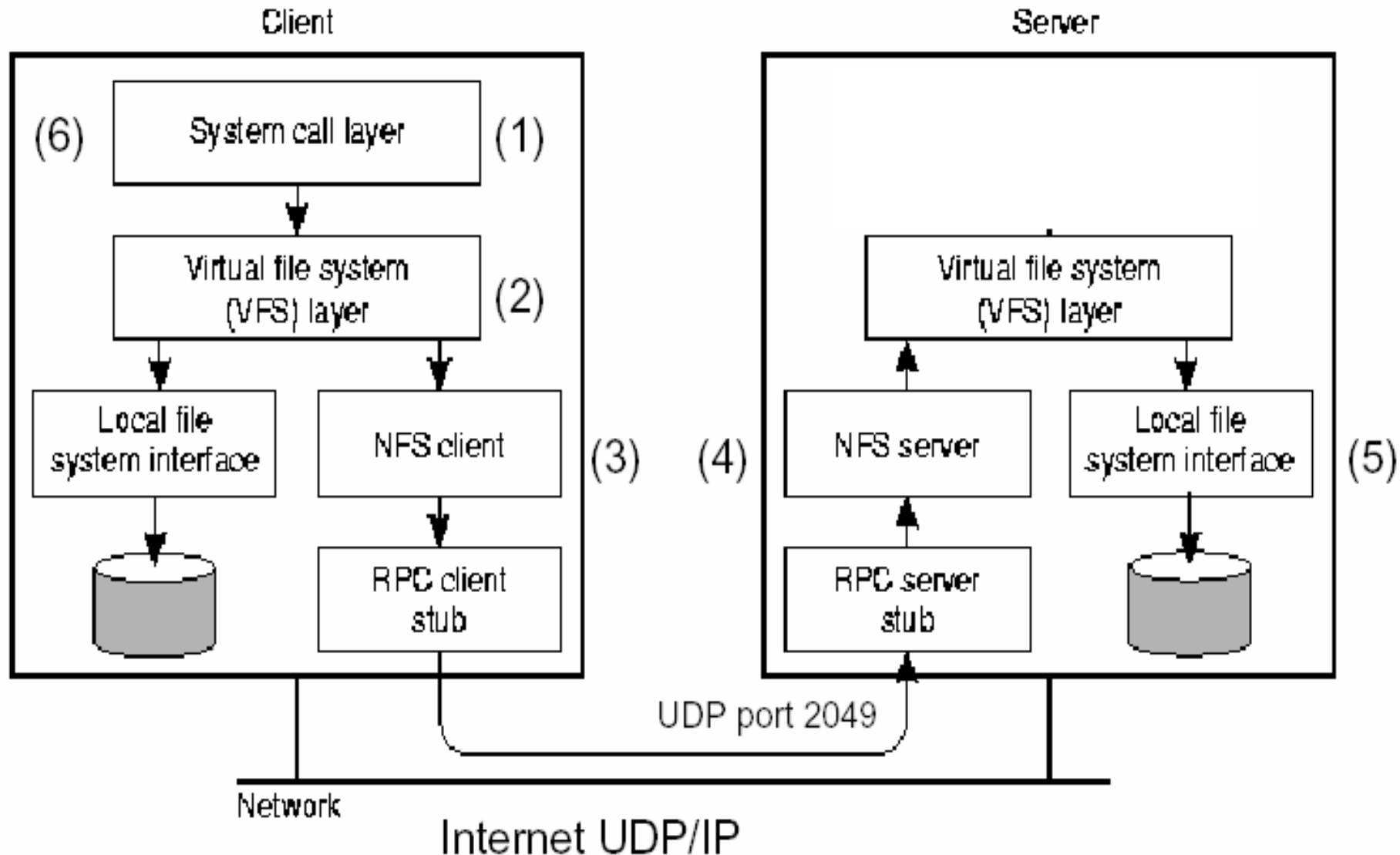
Installation de NFS côté serveur

- ◆ Le démon `rpc.mountd` permet les opérations de montage et de démontage. Ces opérations peuvent s'effectuer de trois manières :
 - ✦ montage au démarrage par `/etc/fstab`
 - ✦ montage en cours de session par l'utilisation de `mount`
 - ✦ par l'auto-monteur qui monte automatique le système distant selon les besoins de l'utilisateur. Cette technique permet une gestion centralisée sur le serveur NIS des ressources partagées.
- ◆ Le démon `rpc.quotad`
 - ✦ gestion des quota sur le disque distant

Installation de NFS côté client

- ◆ Le démon biod peut être installé sur le client (pas obligatoire) il permet d'implémenter, à l'image de ce que fait le noyau, un mécanisme de cache pour la lecture et l'écriture de fichiers. Il permet d'optimiser la connexion.

Rappel sur NFS



Rappel sur NFS

- NFS v2 - 1985

- Utilise UDP
- Performances médiocres en écriture

- NFS v3 - 1994

- Utilise UDP ou TCP
- Performance accrue en écriture

NFS v4

- Sécurité via l'utilisation de kerberos pour l'authentification et cryptage ;
- La fiabilité avec l'utilisation de TCP par défaut ;
- Réplication, failover et récupération des sessions en cas de panne du serveur.

RPC: REMOTE PROCEDURE CALL

- ◆ RPC permet l'exécution des procédures sur une machine distante.
- ◆ En d'autres termes, ils permettent de créer des programmes répartis en fournissant un mécanisme d'appel de procédures distantes.
- ◆ Système de communication Client/Serveur

BUTS

- Facilité:
 - ◆ faciliter la transformation des applications locales en applications réparties;
- Distribution:
 - ◆ utiliser une application sur plusieurs machines;
- Transparence:
 - ◆ appeler une procédure distante comme si elle était locale;
- Indépendance du transport:
 - ◆ TCP, UDP;

RPC

- Ils se programment (peuvent se programmer) avec l'aide de RPCGEN qui génère le code d'interface réseau à partir d'une description des procédures à repartir.
- RPC est un outil efficace, flexible et facile pour distribuer une application
- Les RPC utilisent le modèle Client/Serveur pour construire des applications réparties.

QUAND DOIT-ON UTILISER RPC?

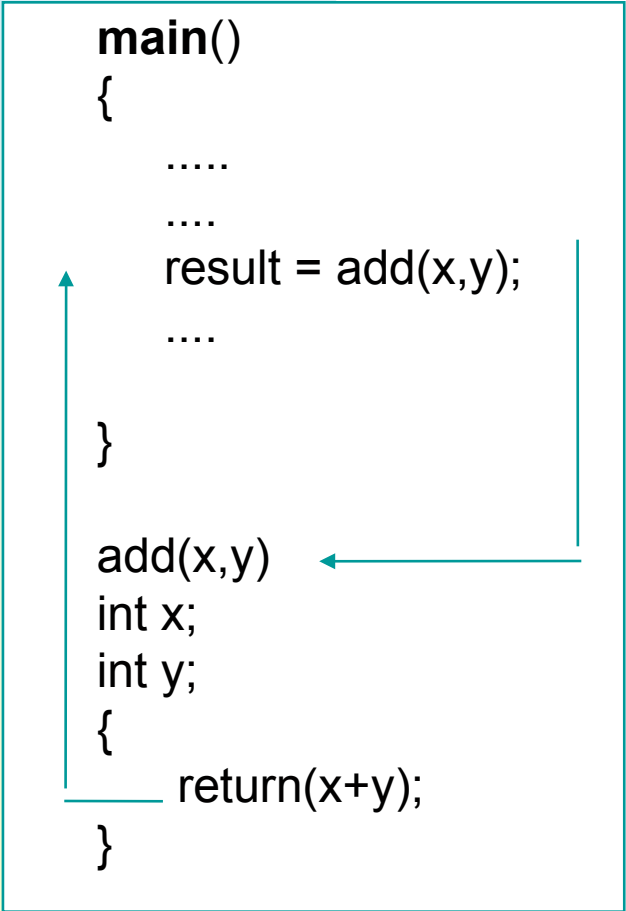
- Convertir une application développée localement en une application distribuée.
- Comme beaucoup de programmes sont divisés en procédures, cette tâche est plus facile.
- Les procédures sont exécutées à distance

FONCTIONNEMENT

Appel de procedure locale

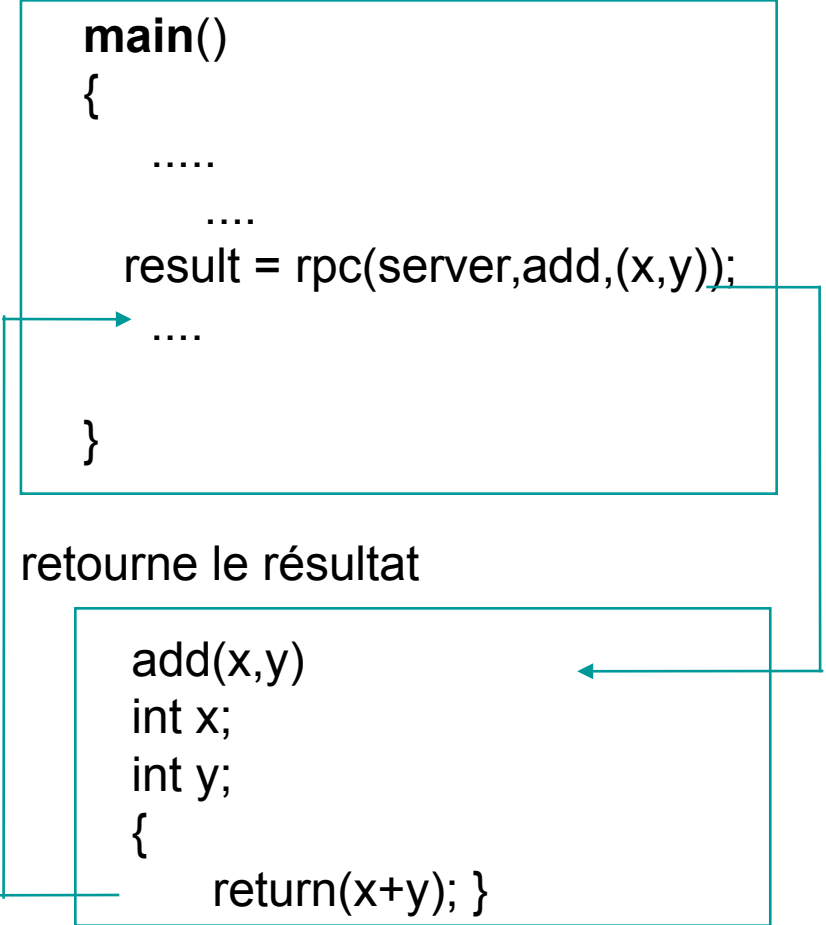
```
main()
{
    ....
    ....
    result = add(x,y);
    ....
}

add(x,y)
int x;
int y;
{
    return(x+y);
}
```



Appel de procedure à distance

```
main()
{
    ....
    ....
    result = rpc(server,add,(x,y));
    ....
}
```



retourne le résultat

```
add(x,y)
int x;
int y;
{
    return(x+y); }
```

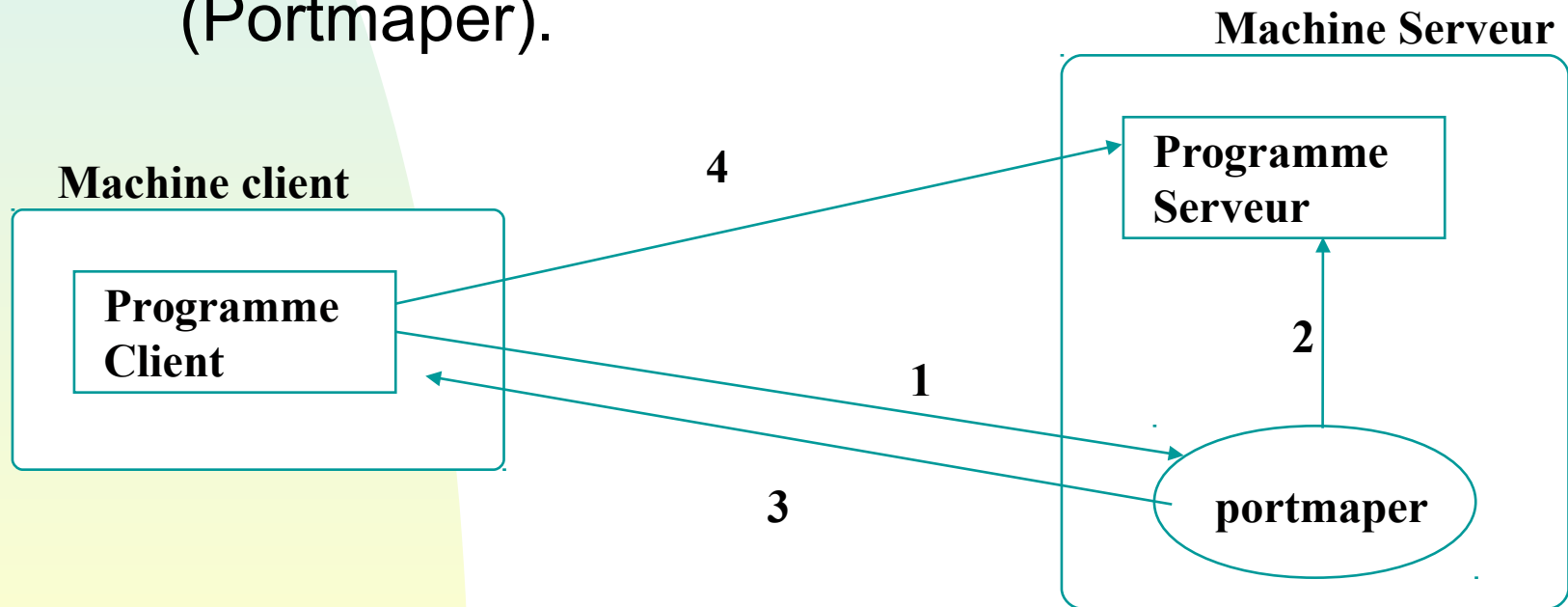
Client

part
l'exécution
sur le serveur

Serveur

ARCHITECTURE RPC

- Trois composantes dans l'architecture RPC:
 - ◆ L'application client
 - ◆ Les procédures du serveur
 - ◆ Le programme qui résout les ports (Portmap).



IDENTIFICATION DES PROCÉDURES ET DES PROGRAMMES

- Numéro de programme
- Numéro de version
- Numéro de procédure:
 - ◆ Le numéro de procédure se donne par le programmeur et ses valeurs dépendent du nombre de procédures à déclarer.
 - ◆ On attribue ainsi la valeur 1 à la première procédure, 2 à la seconde, 3 à la troisième procédure, etc...

LES NUMÉROS DE PROGRAMMES

- Le numéro de programme est attribué suivant le tableau suivant:

Numéro hexadeximal	Domaine d'attribution
00000000-1FFFFFFF	sun
00000000-3FFFFFFF	Administrateur local
40000000-5FFFFFFF	développeur
60000000-FFFFFFFF	réservé

RPCBIND

- Conserve la liste de correspondances entre les numéros de programmes et de versions et les numéros de port.
- Le serveur s'enregistre auprès du démon portmap c'est-à-dire qu'il enregistre un numéro de programme et de version.
- Le portmapper est un programme deamon sur le serveur. Il enregistre les services RPC et retourne leur identificateur de port TCP,UDP lorsqu'on lui donne un identificateur de programme RPC. Il est lui même un service RPC qui écoute sur le port 111.
- Le portmapper peut aussi répondre à une requête de broadcast de la part d'un client.

RPCGEN

- **rpcgen** aide à générer des applications avec RPC.
- C'est un pré-compilateur
- Permet de donner un coup de main aux programmeurs face à XDR
- En entrée, il accepte un langage RPCL, il génère en sortie les fichiers en C.

RPCGEN (2)

- Génère les modules d'interfaces des programmes.
- Similaire au niveau syntaxe et structure au langage C
- Par défaut, il génère:
 - ◆ Fichiers en-tête (header) du côté serveur et du côté client
 - ◆ Ensemble de routines XDR traduisant chaque type de données définies dans le fichier en-tête
 - ◆ Source du serveur
 - ◆ Source du client

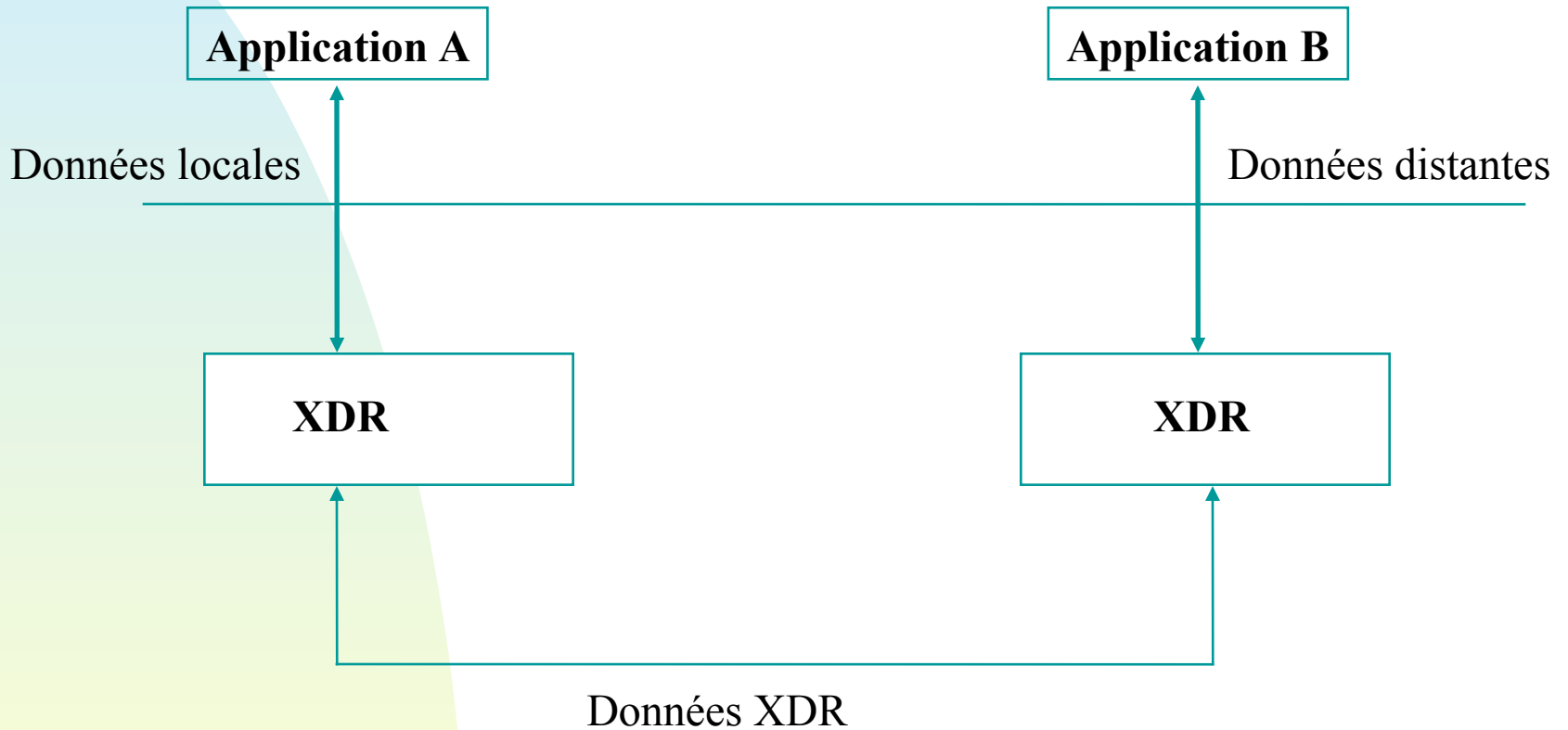
LA REPRÉSENTATION INTERNE DES DONNÉES

- Comme les RPC font appel à des procédures distantes, le problème de représentation interne des données se pose.
- On utilise XDR : eXchange Data Representation
- RPC utilise un format et un protocole standard de représentation, de description et de codage des données.
- Pour simplifier la tâche du programmeur, l'utilitaire **rpcgen** génère des fichiers dont les données sont sous le format XDR.

XDR

- Standard pour la description et l'encodage des données
- Permet de transférer les données de différentes architectures : VAX, IBM-PC, Cray, etc...:
- Couche: Présentation du modèle ISO
- Très similaire à ASN.1 !!!!! (Abstract Syntax Notation, X.409)

XDR (2)



XDR REPRÉSENTATION

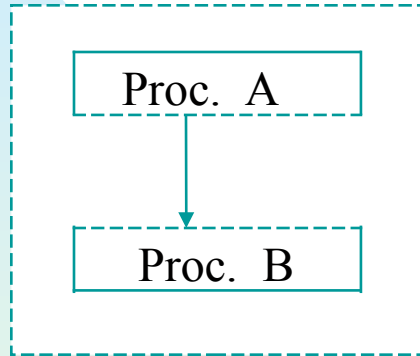
- Tout est représenté sous forme de multiples de 4 octets (32 bits)
- Les octets sont numérotés de 0 à $n-1$

ÉTAPES DE MISE EN OEUVRE

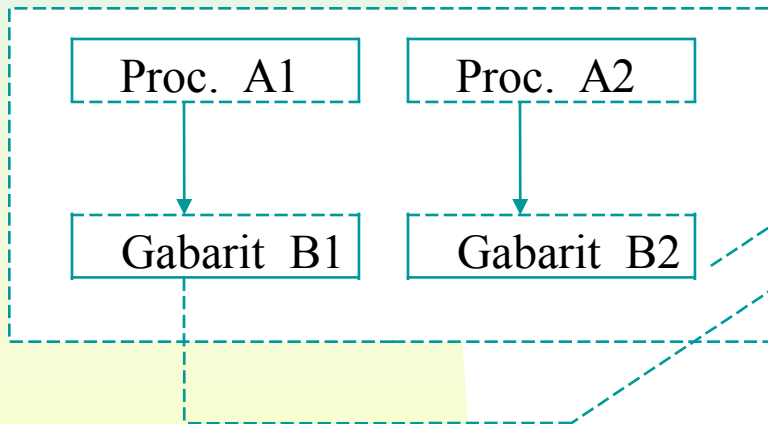
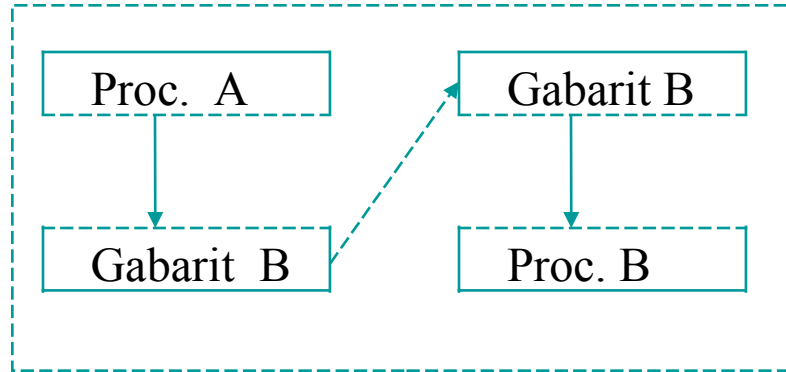
- Pour écrire une application à l'aide des RPC il faut:
 - ◆ Écrire l'interface de haut niveau dans le langage RPCL fichier.x
Il sera compilé par **rpcgen**.
 - ◆ Les fichiers générés sont: fichier**clnt.c**, fichiers**svc.c** et fichier**xdr.c**,
fichier.h
 - ◆ Écrire le programme client;
Compilation et édition des liens du programme client en y incluant les
fichiers de suffixes fichier**clnt.c** et fichier**xdr.c**
 - ◆ Écrire le programme serveur:
Compilation et édition des liens du programme serveur en y incluant
les fichiers de suffixes fichiers**svc.c** et fichier**xdr.c**

COMPOSANTES DE RPC

Architecture en local:



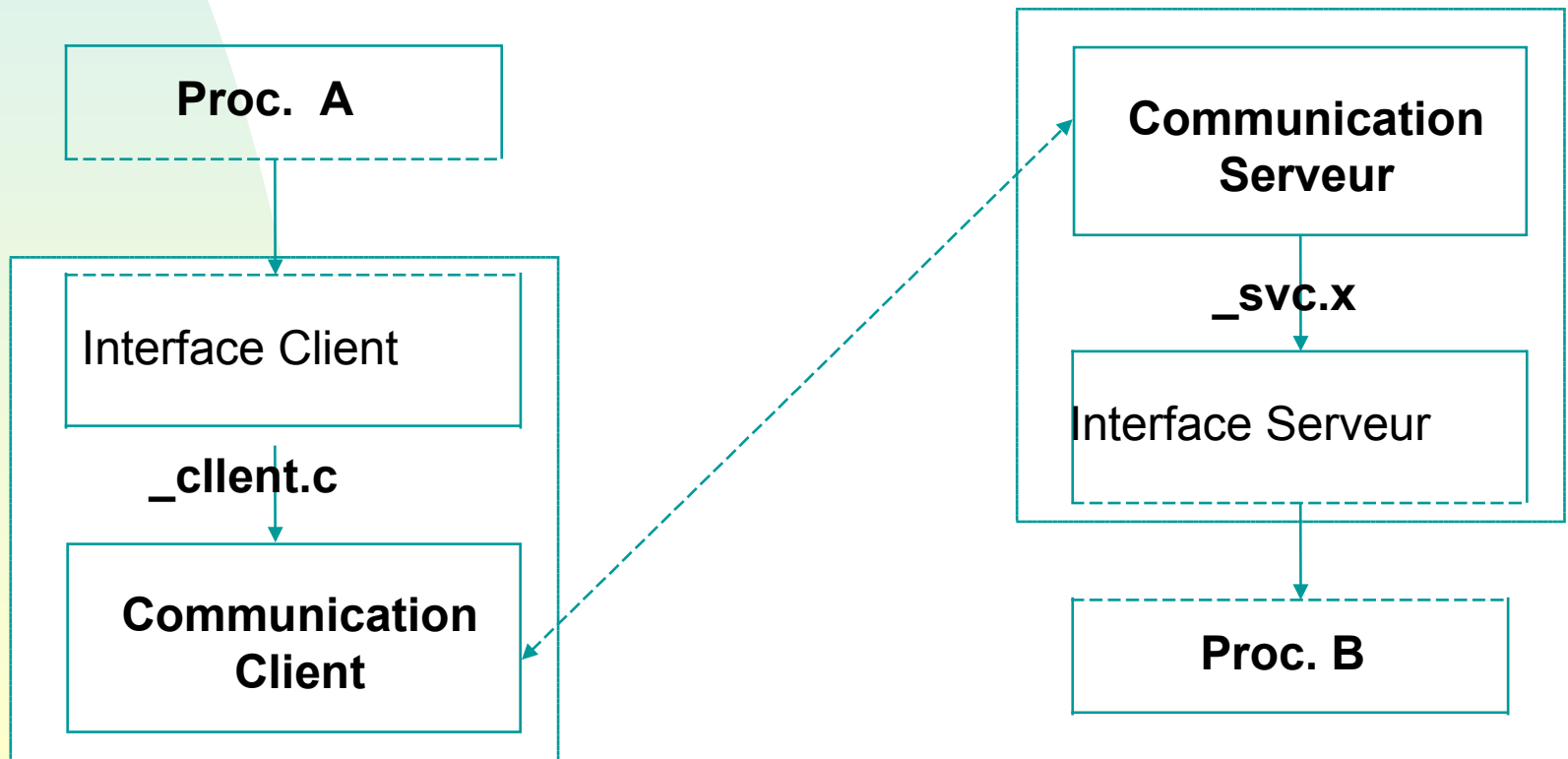
Architecture en RPC:



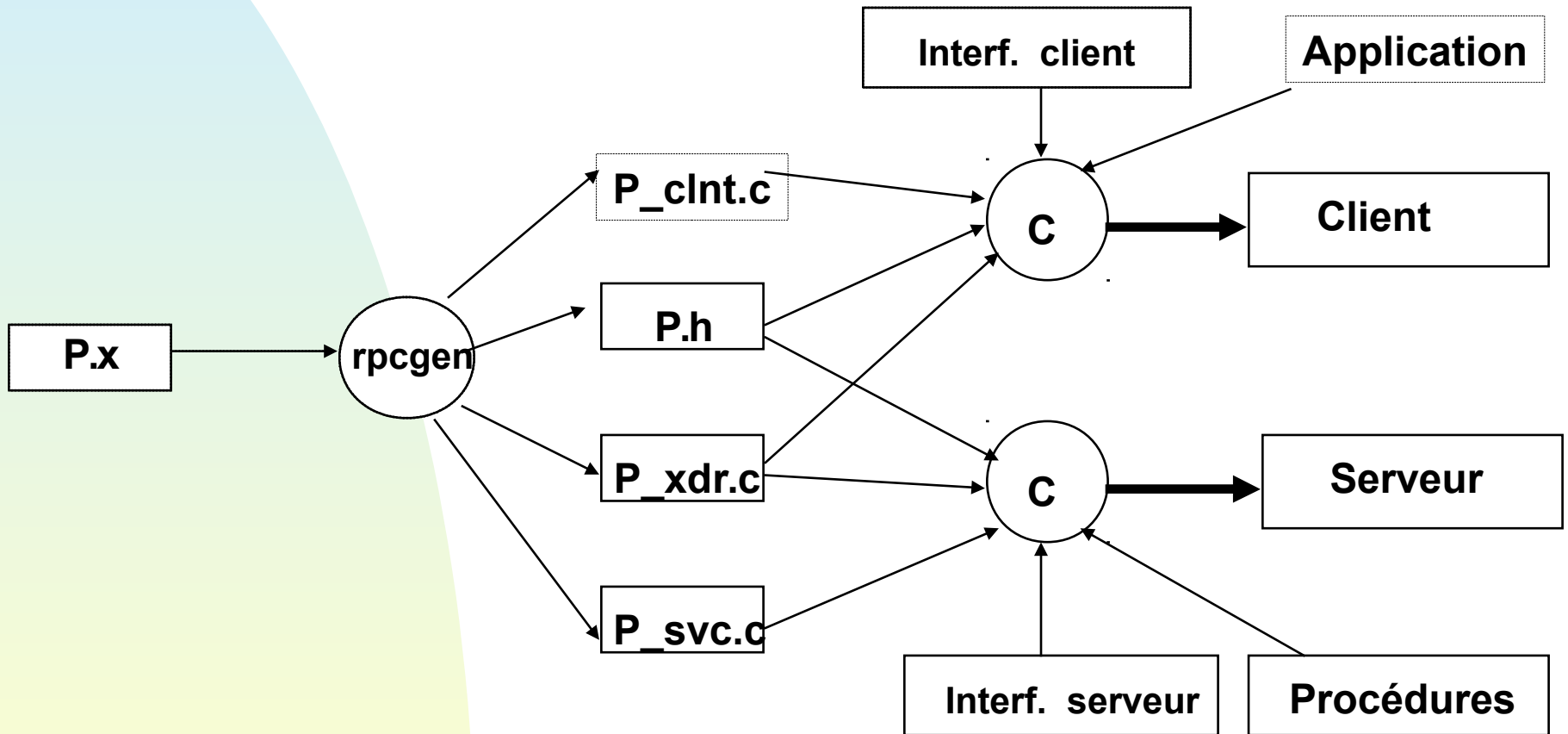
UTILISATION DE RPCGEN

Rpcgen génère pour chaque gabarit deux parties:

- Une partie commune à toutes les applications RPC pour la communication Client-serveur
- Une interface avec le programme d'application



PROGRAMMES GÉNÉRÉS PAR RPCGEN



LES NIVEAUX DE PROGRAMMATION RPC

- Haut niveau:
 - ◆ A ce niveau, le réseau, le système d'exploitation et la machine sont transparents. Le programmeur code simplement en C sans se soucier d'autre chose
- Niveau intermédiaire:
 - ◆ Ce niveau est le niveau propre de RPC. Le programmeur n'a pas besoin de tenir compte du niveau socket, Unix, et des bas niveaux des mécanismes d'implantation.
 - ◆ La majorité des applications utilise ce niveau.
- Niveau expert:
 - ◆ Ce niveau donne suffisamment de détails pour que le programmeur puisse avoir le contrôle.
- Bas niveau

SERVICES HAUT NIVEAU DE RPC

- RPC offre des service de haut niveau appelé Service Library Routines (voir **/usr/include/rpcsvc**)

rnusers()	retourne le nombre d'usagers sur la machine distante
rusers()	retourne l'information sur les usagers de la machine distante
havedisk()	déterrmine si la machine distante a un disque
rstat()	accède aux données de performances de la machine distante
rwall()	écrit sur la machine à distante
yppasswd()	met à jour le fichier passwd du service NIS (Network Information Service)

EXEMPLE DE RPC DE HAUT NIVEAU

```
#include <stdio.h>
```

```
Int main(int argc,char **argv)
```

```
{   int num;
```

```
    if(argc != 2) {
```

```
        fprintf(stderr," utiliser: rnusers machine hote\n");
```

```
        exit(1);    }
```

```
    if((num = rnusers (argv[1])) < 0 ) {
```

```
        fprintf(stderr," erreur rnusers \n");
```

```
        exit(1);} 
```

```
    printf("%d usagers sur %s\n",num,argv[1]);
```

```
    exit(0)}
```

LA COMMANDE RPCINFO

La commande rpcinfo permet de manipuler les services RPC sur un serveur.

rpcinfo -p

program	no_version	protocole	no_port	program	no_version	protocole	no_port
100000	2	tcp	111 portmapper	100005	2	tcp	1025 mountd
100000	2	udp	111 portmapper	100005	3	udp	1025 mountd
100024	1	udp	1024 status	100005	3	tcp	1025 mountd
100024	1	tcp	1024 status	100003	2	udp	2049 nfs
100007	2	udp	787 ypbind	100003	3	udp	2049 nfs
100007	2	tcp	789 ypbind	100021	1	udp	1026 nlockmgr
100011	1	udp	906 rquotad	100021	3	udp	1026 nlockmgr
100011	2	udp	906 rquotad	100021	4	udp	1026 nlockmgr
100005	1	udp	1025 mountd	100009	1	udp	946 yppasswdd
100005	1	tcp	1025 mountd	100004	2	udp	882 ypserv
100005	2	udp	1025 mountd	100004	1	udp	882 ypserv

Pour détruire le service 100005 Version 1

rpcinfo -d 100005 1

TYPES DE SERVEURS RPC

Deux types de serveurs :

Avec état: Le serveur conserve des informations sur les transactions des clients

Sans état: Aucune information sur les opérations du client n'est mémorisée. C'est le client qui s'en occupe.

Deux types d'opérations RPC:

Idempotentes: Les opérations peuvent se répéter.
Les effets ne s'accumulent pas

Non idempotentes: Les opérations qui se répètent avec un effet cumulatif

ERREURS DANS LES REQUÊTES

Une erreur peut se produire dans les cas suivants:

1. Requête perdue:

On peut la refaire

2. La réponse est perdue:

Une répétition de données aura lieu car la position courante du fichier a été changée depuis la dernière fois.

Solution: numéroter les requêtes

3. Le serveur est tombé:

La solution dans 2 n'est pas valable car le numéro de requête dans le serveur est perdu également.

Solution: Utiliser des requêtes idempotentes.

REQUÊTES IDEMPOTENTES

La procédure `put_block` suivante est idempotente.

```
int put_block(int file, int offset, int count, char *data)
{ int return_code = 0;
  if (lseek(file, offset, SEEK_SET) == -1) // On se positionne à la
    return_code = -1;                      // position offset=nbytes
  else
    return_code = write(file, data, count);
  return (return_code);}
```

```
static offset=0;
int write_file (int fd, char *buf, int nbytes)
{ int nbytes;
  if (nbytes = put_block(fd, nbytes, buf) != -1 )
    offset +=nbytes;
  return (nbytes);}
```