MODELE CLIENT-SERVEUR

On considère en général que ce modèle fait partie de la couche session. Il correspond à un fonctionnement en mode non connecté. La couche transport est bâtie sur un modèle symétrique. Dans la pratique les applications sont rarement symétriques, serveurs de fichiers par exemple. Dans ce modèle un client qui envoie une requête au serveur se comporte comme un programme qui appelle une procédure. Les commandes d'entrées-sorties sont alors remplacées par des "procédures stubs" incluses dans le langage de programmation comme des procédures locales.

Exemple : lire(id-fichier,ad_tampon,longueur)

détruire(id_fichier)

1 - IMPLEMENTATION

L'appel à distance comporte 10 étapes. L'espace d'adressage du client (resp. serveur) et de son stub étant le même les passages de paramètres par valeur se font tout à fait normalement en utilisant des présentations standard de données. Le seul problème est lié au passage par référence car l'espace d'adressage n'est pas le même sur le serveur et sur le client.

Une première approche consiste à effectuer une copie des paramètres : mécanisme de stockage/restitution. Cette technique peut échouer comme le montre le programme de la figure 2.

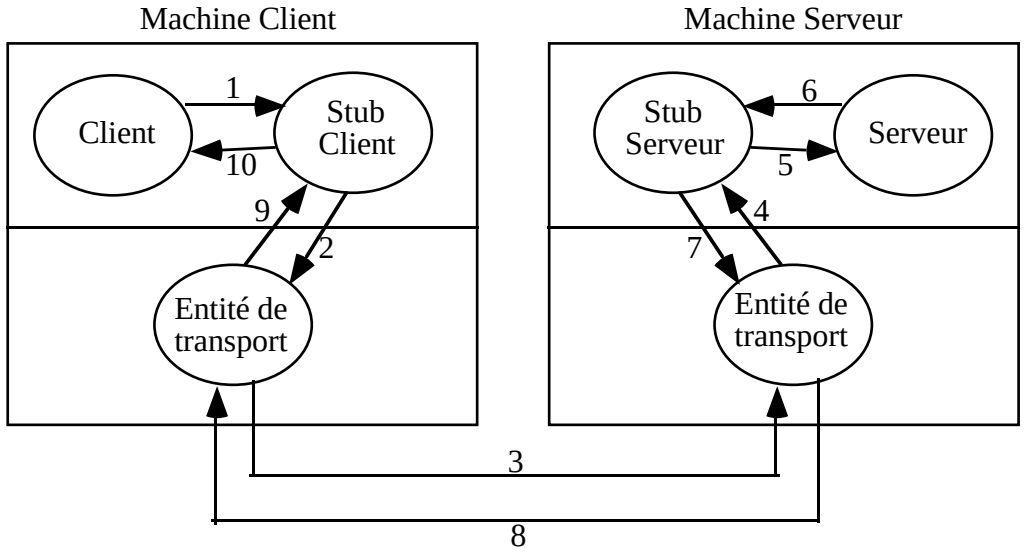

Figure 1. Appel de procédure à distance

```
void doubleincr(int * x,y)
        {*x++;*y++}
main()
{int a=0;
  doubleincr(&a,&a);
   printf(a,%n)}
```

Figure 2. Exemple de programme.

L'appel en local fournira la valeur 2, l'appel a distance si on ne prend pas soin des paramètres par référence fournira la valeur 1.

Les pointeurs posent les mêmes problèmes en plus accentués car les objets désignés peuvent être beaucoup plus complexes.

Ces problèmes sont en général peu ou pas résolus et il faut donc suivre très précisément le modèle implémenté.

2 - BINDING

Dans des réseaux traditionnels, les applications connaissent le port sur lequel se connecter. Les sessions sont établies à partir de port prédéfini.

Les applications construites au-dessus de TCP/IP ne connaissent que des ports standards. Cette liste se trouve dans le fichier ..... Bien évidement ce fichier ne peux contenir toutes les applications que l'on peut appeler.

L'attribution d'un numéro de port sur RPC s'effectue dynamiquement. Tout serveur à l'initialisation est enregistré dans une base de donnée. Pour cela il émet par une primitive enregistrer son nom, son port et un identificateur (entier 32 bits aléatoire).

Lorsqu'un client se connecte, la première connexion est celle à un serveur stub standard. Le client envoie une chaîne de caractère correspondant au service demandé. Le stub serveur renvoie, après consultation de la base de donnée, le port de connexion ainsi que l'identificateur. Puis le client émet sa requête auprès du serveur en utilisant la paire (port,entier).

L'identificateur permet à la machine serveur de trouver parmi les nombreux stubs serveur accédant au port celui à adresser. Il permet également en cas d'arrêt du serveur d'identifier les appels entrant qui correspondent à un ancien stub.

3 - SEMANTIQUE

La sémantique du RPC a une particularité par rapport aux appels locaux c'est la panne du serveur. La figure 3 est un diagramme temporel d'une requête sans arrêt du serveur.

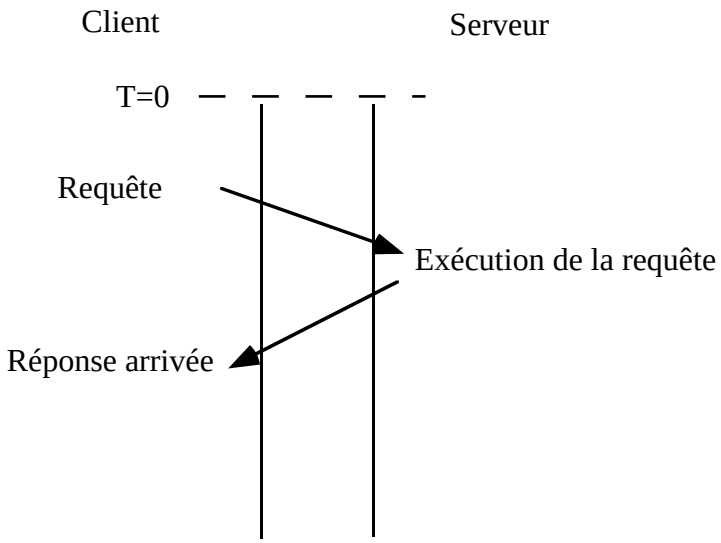

Figure 3. Un échange RPC sans panne.

Le serveur peut tomber en panne avant exécution de la requête ou après. Le premier cas de figure est résolu par la base de donnée. Dans le second cas, la panne peut avoir lieu juste après l'exécution mais avant transmission. Le client stub a trois attitudes:

1 - attendre indéfiniment,

2 - déclencher un temporisateur à l'émission de la requête et signaler l'anomalie,

3 - déclencher un temporisateur à l'émission de la requête et réémettre la requête.

Dans le dernier cas, la réémission entraînera une erreur puisqu'il y a changement du nombre entier dans le couple (port,entier) et le client stub peut recommencer l'opération de binding.

Ces trois situations peuvent être adoptées en fonction des situations traitées :

cas 1 - procédure utilisateur ne traitant pas les anomalies,

cas 2 - ajout d'un bloc en fin de fichier,

cas 3 - destruction de fichier, plus généralement idempotence de l'opération effectuée.