Aissaoui Siham & Mahras Asmae

Rapport de projet : Résolution de noms

Introduction

Ce programme a pour but de répondre à une requête d'un client aux serveurs pour la résolution d'un nom de domaine.
Il traite la requête du client par le biais d'une hiérarchie composée de 3 serveurs qui ont pour but de communiquer l'adresse ip et le numéro de port correspondant au nom de domaine demandé par le client.

Implémentation

Représentation des données

Toutes les données sont stockées dans des structures de données élaborées pour communiquer les informations nécessaires au bon fonctionnement coté client ainsi que coté serveur. On retrouve une structure pour les informations concernant le serveur (nom, adresse ip, numéro de port) notamment pour la lecture du fichier contenant les informations des serveurs racine. Viens ensuite une deuxième structure plus détaillée pour les informations des serveurs contenant un pointeur vers la structure serveur citée plus haut et 3 autres données représentant les 3 parties du nom de domaine nécessaires à la résolution du nom `<nom_de_machine>.<sous_domaine>.<domaine>`.
On aura aussi une structure pour stocker la réponse du serveur coté client contenant (id, horodatage, nom de domaine, code de retour, liste des serveurs correspondants).

Fonctionnalités

En ce qui concerne les choix d'implémentation, nous avons choisis de mettre a profit les données transmises entre les partis.
Tout d'abord, l'identifiant de la requête est non seulement un moyen de discerner les requêtes mais aussi l'étape à laquelle le client se trouve dans la résolution de son nom avec :

res = identifiant mod 3 {
- si res = 2 => on est sur le serveur domaine
- si res = 1 => on est sur le serveur sous domaine
- si res = 0 => on est sur le serveur nom

Ensuite, le code de retour a pour but de communiquer le nombre de serveurs correspondant à la requête client avec comme valeur 0 si on ne trouve pas de serveur correspondant et n si on trouve n serveurs correspondants. Ce qui nous permet au final de n'allouer que l'espace nécessaire en fonction du code de retour.
L'automatisation de la hiérarchie des serveurs a été traité dans le programme client à travers 3 boucles imbriqués interdépendantes modélisant un parcours en profondeur de la hiérarchie pour trouver l'adresse du nom demandé par le client.

Outils
Le makefile sert à compiler les programmes c et générer les exécutables. Il permet également de supprimer les exécutables et de générer une archive .tar.
Le script quant à lui lance 3 serveurs simultanément pour recevoir les requêtes client et les traiter.

Gestion des pannes
Le programme s'assure que le client reçoive une réponse à sa requête même en cas de panne de serveur. Pour cela, on a utilisé un timeout pour permettre au client de contacter un nouveau serveur si le premier ne répond pas au bout de quelques ms. Si le client ne trouve pas d'adresse correspondant à sa requête, il remonte la hiérarchie pour parcourir à nouveau une autre adresse.

Manuel utilisateur
Compilation
Compiler le programme :
```
→ make
```

Supprimer les exécutables :
```
→ make clean
```

Générer une archive tar :
```
→ make dist
```

Lancement du programme

Compiler le programme :
```
→ make
```

Exécuter le script :
```
→ ./script.sh
```
pour lancer les serveurs.

Le programme server prend en argument un numéro de port et un fichier contenant la liste des serveurs.
Pour modifier les numéros de port des serveurs, il faut les modifier sur le fichier script.sh.

Lancer le programme client :
Ouvrir un autre terminal puis lancer la commande :
```
→ ./cli <fichier_serveurs_racines>
```
pour entrer le nom à résoudre sur l'entrée standard
ou
```
→ ./cli <fichier_serveurs_racines> <fichier_requêtes>
```
pour entrer les noms à résoudre à partir d'un fichier.