DM Res1 — Serveur de discussion

Gaétan Richard

1 Présentation

Dans ce devoir, il s'agit d'utiliser `netcat` (ou `nc`) pour programmer un mini client-serveur. Il nécessite le développement d'un programme appelé par netcat côté serveur.

Il s'agit de programmer un serveur de mise en relation. Le serveur sert à mettre en relation deux personnes ayant les mêmes centres d'interêts. Pour cela chaque personne indiquera au serveur un sujet de discussion avec leur ip et un port sur lequel il est prêt à écouter. Si celui-ci n'existe pas encore, le serveur lui indiquera de se mettre à l'écoute. Sinon, le serveur lui donnera l'adresse et le port d'un autre client à l'écoute sur ce sujet pour le contacter directement.

Un serveur principal sera toujours à l'écoute, sur un port "bien connu" des clients (dans notre cas le port `5555`). Un échange se déroulera toujours de la façon suivante : Un client établira la connexion et lancera une demande. Le serveur répondra puis fermera la connexion.

2 Protocole

On veillera a respecter **scrupuleusement** le protocole afin d'avoir des applications compatibles.

Le serveur doit régir à deux commandes :

- `TALK` *sujet ip port* : si sujet ne fait pas partie de la liste des sujets, le serveur répond `LISTEN` et place ce sujet dans la liste des sujets. Sinon, le serveur renvoie `CONN` *ip port* où *ip* et *port* sont celle du client ayant demandé l'ouverture du sujet. Le sujet est alors fermé.
- `LIST` : donne la liste des sujets de discussion ouverts. La réponse est de la forme `+OK` n où n est le nombre de sujets suivi de la liste des sujets chacun sur une ligne.

Voici un exemple de transactions. Les lignes précédées de `S>` indiquent les lignes écrites par le serveur. Les lignes précédées de `C>` indiquent celles du client. Ces préfixes ne font bien sur pas parti du protocole.

```
C> LIST
S> +OK 0

C> TALK pigeon 10.130.196.10 2467
```

```
S> LISTEN

C> LIST
S> +OK 1
S> pigeon

C> TALK pigeon 10.130.196.15 1487
S> CONN 10.130.196.10 2467

C> LIST
S> +OK 0
```

Question 1 *Réaliser un fichier* ***serveur.sh*** *permettant de lancer le serveur. Pour cela, on pourra utiliser l'option* `-c` *de* ***nc.traditional****.*

Si des problèmes de délai se posent, il existe l'option `-q 1` de **nc.traditional** qui permet d'attendre une seconde avant de fermer la socket.

3 Client

Afin que ce logiciel puisse être utilisé par des novices. On désire cacher la partie protocole et réaliser un client shell **client.sh**.

Ce client se lancera :

- soit avec un seul argument *ip* : dans ce cas, il contactera le serveur à cette ip et affichera la liste des sujets ouverts ;
- soit avec deux arguments *ip sujet* : il contactera alors le serveur à cette ip, et créera (en écoute ou en connexion selon la réponse) une connexion directe avec un client pour discuter.

Voici quelques exemples d'utilisation de clients. On suppose que le serveur est à l'adresse `127.0.0.1`.

```
$ ./client.sh 127.0.0.1 pomme
En attente de pair
...

$ ./client.sh 127.0.0.1
Sujets ouverts:
pomme
téléphone

$ ./client.sh 127.0.0.1 pomme
Connexion en cours
....
```

4 Consignes

Ce devoir est à rendre pour le **21 février 2012** avant **23h59**. Il doit être envoyé sous la forme d'une archive **tar.gz** ou **tar.bz2** à l'adresse `gaetan.richard@unicaen.fr`.

L'archive devra contenir l'ensemble des fichiers de scripts commentés, un fichier texte `INSTALL` donnant les instructions (et des exemples) pour lancer le programme. L'ensemble des scripts **doit** pouvoir fonctionner sur les machines de la salle TP. Vous joindrez également un fichier texte `README` indiquant les personnes ayant fait le travail ainsi que le travail effectué.

Le travail est à faire seul ou par deux. La notation tiendra compte du travail réalisé, de la façon dont celui-ci a été réalisé (commentaires, ...) ainsi que du respect des consignes.