ECAM
Brussels Engineering School

jeudi 1 mars 2018
CBF, LUR, LRG

Projets de développement informatique : Labo 3

Let's chat together !

1 Contexte

Le but de ce labo consiste à réaliser une petite application réseau qui vous permettra de chatter avec vos camarades à travers le réseau. Lors de ce labo, vous devez mettre au point le protocole de communication entre les différents composants que vous allez mettre en place, tout en respectant quelques éléments imposés. Une fois le protocole défini, vous devez implémenter votre solution en binôme, chaque membre étant responsable de composants différents.

L'application de chat que vous allez développer ne sera utilisable qu'en mode console. N'hésitez pas à repartir des exemples vus au cours que vous pouvez retrouver sur le dépôt GitHub du cours [1].

2 Exercices

Pour vous aider à réaliser votre travail, et pour vous entrainer par rapport à la matière du cours concernant la programmation réseau et concurrente, voici quelques petits exercices à réaliser :

1. Créez un socket TCP pour vous connecter au serveur `www.vinci.be` sur le port 80. Une fois connecté, envoyez la chaine de caractères suivante sur le socket :
   ```
   GET / HTTP/1.0\n\n
   ```
   puis affichez la réponse renvoyée par le serveur. Réessayez en remplaçant `/` par `/shit` et comparez la réponse renvoyée par le serveur. Vérifiez, en particulier, le code de réponse HTTP (3 chiffres).

2. Repartez de l'exemple `Chat` vu au cours et ajoutez une commande `client` qui renvoie les informations du client connecté au chat, c'est-à-dire l'adresse IP et le port de la personne que l'on a rejoint. Testez votre programme.

3. Écrivez un programme Python qui récupère le nom de l'utilisateur connecté sur une machine, en passant par la commande `whoami` que vous appellerez en passant par `subprocess` ou `Popen`.

4. Écrivez un programme qui simule le lancement de 1000 dés en parallèle, à l'aide de threads. Chaque thread génère un nombre entier au hasard, compris entre 1 et 6, et le stocke dans une liste partagée à protéger à l'aide d'un lock. Le programme principale lance les 1000 threads et attend qu'ils aient tous fini avant de calculer la moyenne (exploitez les fonctions `sum` et `len`).

3 Travail

Votre application doit comporter une partie en mode client/serveur et une en mode peer-to-peer. Voici une idée d'implémentation :

— Le serveur permet de mémoriser la liste des clients disponibles pour chatter. Il retient pour chaque client son pseudo et son adresse IP.
— Le client va se présenter au serveur, ce qui fait qu'il sera disponible pour chatter. Il peut interroger le serveur pour obtenir la liste des clients disponibles.
— Ayant l'adresse IP d'une autre machine, une machine peut lancer un chat avec une autre en mode peer-to-peer, tout cela indépendamment du serveur.

1. Le dépôt GitHub du cours se trouve à l'adresse suivante : https://github.com/ECAM-Brussels/PythonAdvanced2BA. N'hésitez pas à le clone et à faire des `git pull` fréquemment pour télécharger les nouveaux commits sur votre machine.

3.1 Évaluation

Ce mini-projet comptera pour **4 points** dans le cadre de l'évaluation continue du labo. La grille d'évaluation est la suivante :

1. Le code source compile et s'exécute sans erreurs (1 point)
2. Le protocole de communication est bien défini (1 point)
3. Les connexions TCP et UDP sont bien utilisées (1 point)
4. Les erreurs possibles sont suffisamment bien gérées (1 point)

3.2 Délivrable

Pour ce faire, créez un dépôt GitHub sur le compte de l'un des deux membres du binôme et ajoutez-y les deux membres comme collaborateurs. Attention que lorsque vous travaillez à plusieurs sur un même dépôt GitHub, vous **devez** faire un `git pull` avant de faire le `git push origin master` afin de récupérer les nouveaux commits qui auraient été faits par l'autre membre. De plus, dans un premier temps, évitez à tout prix de travailler chacun de votre côté sur les mêmes fichiers [2].

En plus de votre code, ajoutez un fichier `README.md` qui décrit brièvement comment lancer et utiliser votre application ainsi qu'une description du protocole de communication que vous avez défini.

La deadline pour ce travail est le **mercredi 16 mars 2018 à 18h30**. Tout retard sera sanctionné par une note nulle.

2. On verra plus tard dans le cours comment gérer de telles situations et régler des conflits de commits.