Réseau - Projet 2023-2024

D.Moreaux

17 septembre 2023

1 Vue d'ensemble

Le projet consiste en la réalisation d'un système de chat en ligne utilisant le protocole TCP.

Ce système sera constitué d'un serveur programmé en langage C et d'un client réalisé en Python + Tkinter. Ils utiliseront l'interface *socket* pour communiquer ensemble.

Un document détaillant le protocole sera réalisé avant de commencer la programmation du client et du serveur.

1.1 Le système de Chat

Le système n'aura aucune persistance : si le serveur est arrêté, la liste des canaux de discussion, les messages et les utilisateurs seront oubliés et lors du relancement du serveur, ces listes seront de nouveau initialement vides.

Le serveur permet aux utilisateurs de se connecter sans leur demander de nom d'utilisateur ou de mot de passe. Une fois connecté, l'utilisateur aura un displayname temporaire jusqu'à ce qu'il soit changé par le programme client. Il est possible de changer le displayname autant de fois qu'on le désire.

Les utilisateurs peuvent demander la liste des canaux et rejoindre un ou plusieurs canaux. Si on tente de rejoindre un canal vide, ce dernier sera créé et l'utilisateur qui l'aura rejoint sera marqué opérateur. Un utilisateur peut être opérateur dans plusieurs canaux.

Les opérateurs d'un canal peuvent changer le titre du canal, donner ou retirer le droit opérateur à d'autres utilisateurs et éjecter un utilisateur du canal (il pourra le rejoindre de nouveau).

Un utilisateur pourra envoyer un message dans un canal ou à un autre utilisateur. Il pourra également voir le titre d'un canal et la liste des membres du dit canal.

Il lui sera également possible de quitter un canal qu'il a rejoint précédemment ou de quitter le serveur (se déconnecter).

1.2 Le serveur

Le serveur sera réalisé en C sous Linux. Il disposera de deux plusieurs listes : la liste des utilisateurs, la liste des canaux et, pour chaque canal, la liste des utilisateurs connectés et leur droit. Ces listes seront des listes liées (simples ou doubles, au choix) afin de ne pas imposer de limite (autre que la capacité mémoire).

Si plusieurs fichiers source sont utilisés, un fichier Makefile permettra la compilation du programme.

Aucune librairie spécifique (hormis la librairie du langage C) ne devra être utilisée sur le serveur.

Les trames réseau seront définies sous la forme de structures et on prévoira une union de ces structures pour désigner un paquet réseau.

1.3 Le client

Le client sera réalisé en Python à l'aide de la bibliothèque Tkinter. Il proposera une interface graphique permettant de communiquer avec le serveur.

Cette interface devra prévoir des boutons et autres pour les différentes fonctionnalités (pas de commandes texte).

Pour échanger des informations avec le serveur, on utilisera les fonctions pack() et unpack() pour structurer les données selon la situation. La communication réseau sera faite exclusivement avec l'interface socket (similaire à l'interface du langage C).

2 Références

Les différentes références suivantes pourront vous aider dans la réalisation du programme :

- Man-pages Linux sur l'interface socket. Des points de départ intéressants sont socket(2), socket(7), tcp(7), ip(7)
- Circlenet.odp : présentation décrivant la couche réseau de CircleMUD, un programme de MUD.
- https ://www.circlemud.org/pub/CircleMUD/3.x/ : sources de CircleMUD 3.1. Principalement le fichier comm.c
- documentation Python en ligne

3 Environnement de travail

Vu l'aspect cross-platform de Python, le client pourra être lancé à partir d'une machine Linux, Windows ou MacOSX.

Le serveur sera hébergé dans une machine virtuelle soit bridgée, soit avec du port forward. La mise à jour des sources sur la machine serveur se fera à l'aide de SSH/SCP (PyCharm/CLion/... ou WinSCP au choix).

L'environnement de développement pour le code en C est au choix de l'étudiant. L'environnement pour la partie Python sera PyCharm.

4 Milestones

1. Définition du protocole sur un document papier.
2. Analyse : Mockup du client.
3. Analyse : structures de données sur le serveur.
4. client : interface graphique sans connexion réseau.
5. client/serveur : connexion, nickname, déconnexion.
6. client/serveur : rejoindre un canal, liste des utilisateurs du canal, liste des canaux, quitter un canal.
7. client/serveur : changement du titre d'un canal, mise à jour du titre chez les clients, octroit/retrait du droit opérateur, ejection d'un utilisateur.
8. client/serveur : envoi d'un message dans un canal, envoi d'un message privé.

Les points 4 et suivants peuvent être modifiés. Les trois premiers points doivent être faits en premier et validés avant le congé de Toussaint [1].

1. idéalement même bien plus tôt