Lab 1 : Application de Bavardage

CEG 3585 [A] – Introduction à la communication de données et au réseautage

Hiver 2023
École de science informatique et de génie électrique
Université d'Ottawa

Professeur : Mohamed Ali Ibrahim, ING., PhD.

Vendredi, 3 février, 2023
CEG 3585 [A] Lundi Groupe 7:

Patrick Loranger (300112374)
Pierre Akladios (300114467)

But et théorie du problème :

Ce laboratoire présente une relève de conception et d'implémentation d'une application de clients et serveur de bavardage utilisant le protocole TCP IP. On utilise des sockets pour bien communiquer entre clients. En bref, le but est d'implémenter quelques différentes fonctionnalité au code. La première fonctionnalité est celle de broadcast. Pour ceci, un client peut envoyer par le serveur un message à tous les autres clients en même temps. La deuxième fonctionnalité est celle du message privée. Un client peut déterminer à qui il/elle veut envoyer son message, et le message serait seulement reçu par le destinataire en question.

Explication de l'algorithme de notre solution :

Nous avons utilisé Python 3 pour notre solution de l'application de bavardage.

Comment utiliser le code :

1. Créer une instance de serveur dans le terminal de choix avec la commande suivante : *python3 server.py*
2. Créer une instance de client dans le terminal de choix avec la commande suivante : *python3 client.py*
3. Répéter l'étape 2 jusqu'à cinq fois pour avoir plusieurs clients connecté au serveur.
4. Lire les commandes dans les terminal de client et envoyer des messages!

En commençant avec l'explication du serveur. Nous utilisons des sockets et des threads pour bien suivre le protocole TCP IP. Premièrement, on fait la connexion au host 127.0.0.1 et au port 4444. On fait les étapes préliminaires pour bien établie le socket dans le serveur.

La fonction send(conn) est utilisé comme handler pour tous les messages envoyées via le serveur.

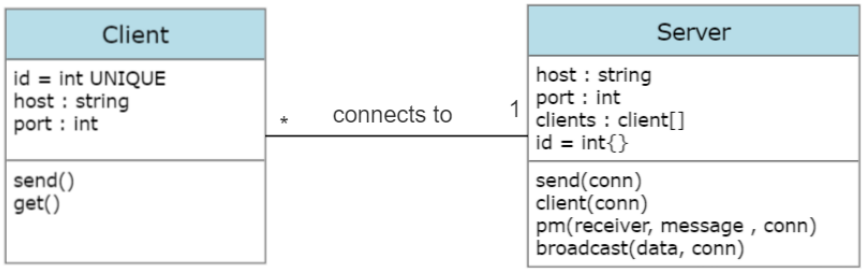
La première fonctionnalité de notre code est la fonction pm. Si le message entrer par le client commence avec la commande « pm » (private message), suivi d'un id de client, le handler (send) va considérer la commande et va faire un appel de la fonction pm, qui envoi le message seulement au client avec le id spécifier dans la commande.

La deuxième fonctionnalité de notre code est la fonction client. La commande pour cette fonction est « conn ». Si un client entre « conn » dans son terminal, le code fait un requête et affiche les identifiants de tous les clients actifs connectés au serveur. Ceci est pour savoir à qui envoyer un message privé.

La dernière fonctionnalité de notre code est la fonction broadcast. Si le client envoi seulement un message avec aucune commande, ce message fait un broadcast à tous les clients actifs dans le serveur. Tout le monde reçoit le message dans leurs terminal.

De suite, nous avons le code pour le ficher client. On initialise les host, port, socket de la même manière que dans le code du serveur. En se connectant étant un client, une liste de commande ce fait imprimer sur l'écran, « conn », « pm » et broadcast. De suite, il y a deux fonctions, un pour envoyer les messages, et un autre pour les recevoir. On commence les threads à la fin.

Document de conception et diagrammes UML :



Captures d'écrans de la démonstration de l'application :

NB : tous les captures d'écrans sont de haute qualité. Veuillez agrandir l'écran pour mieux voir.

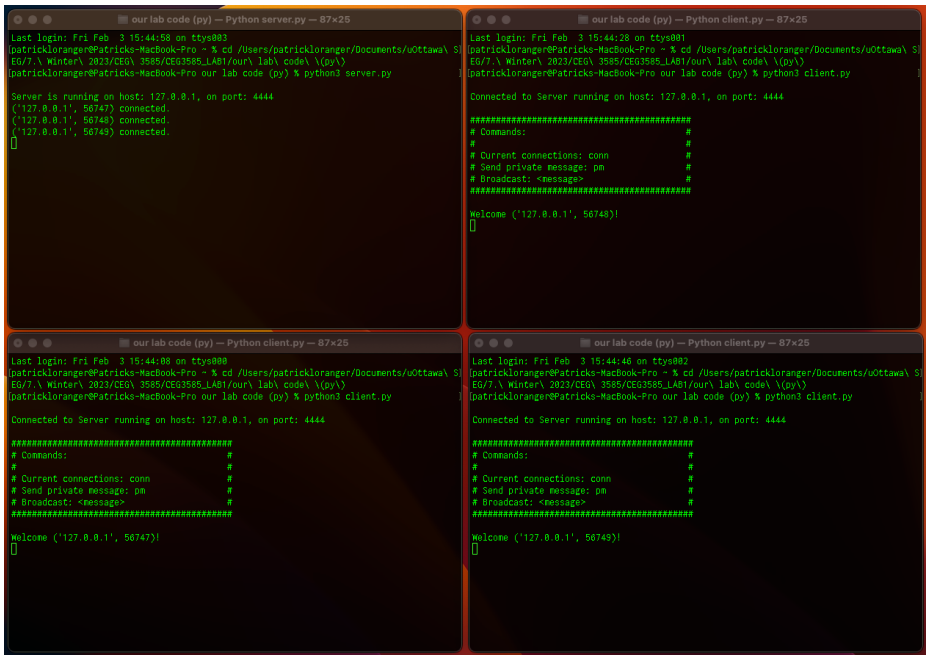


Figure 1 : Initialisation du serveur et 3 clients différents.

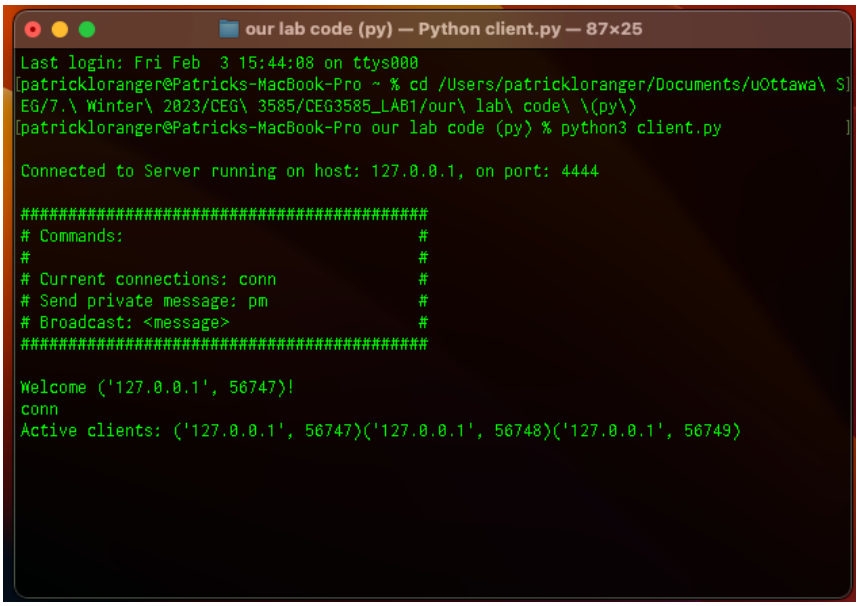


Figure 2 : Le client 56747 entre la commande « conn » pour voir la liste de clients actifs dans le serveur. Le terminal montre : « Active clients : […] »

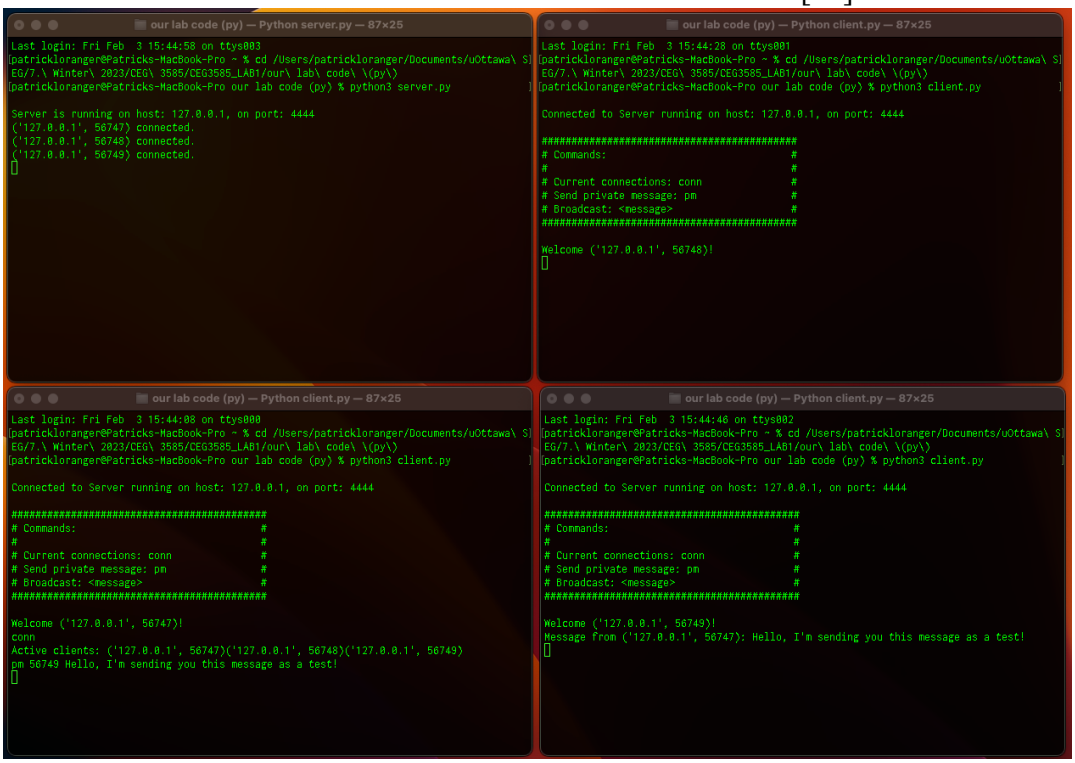


Figure 3 : Le client 56747 entre la commande « pm » et « 56749 » pour envoyer un message privé au client 56749. Le client 56749 le reçoit, et le troisième client ne le reçoit pas.

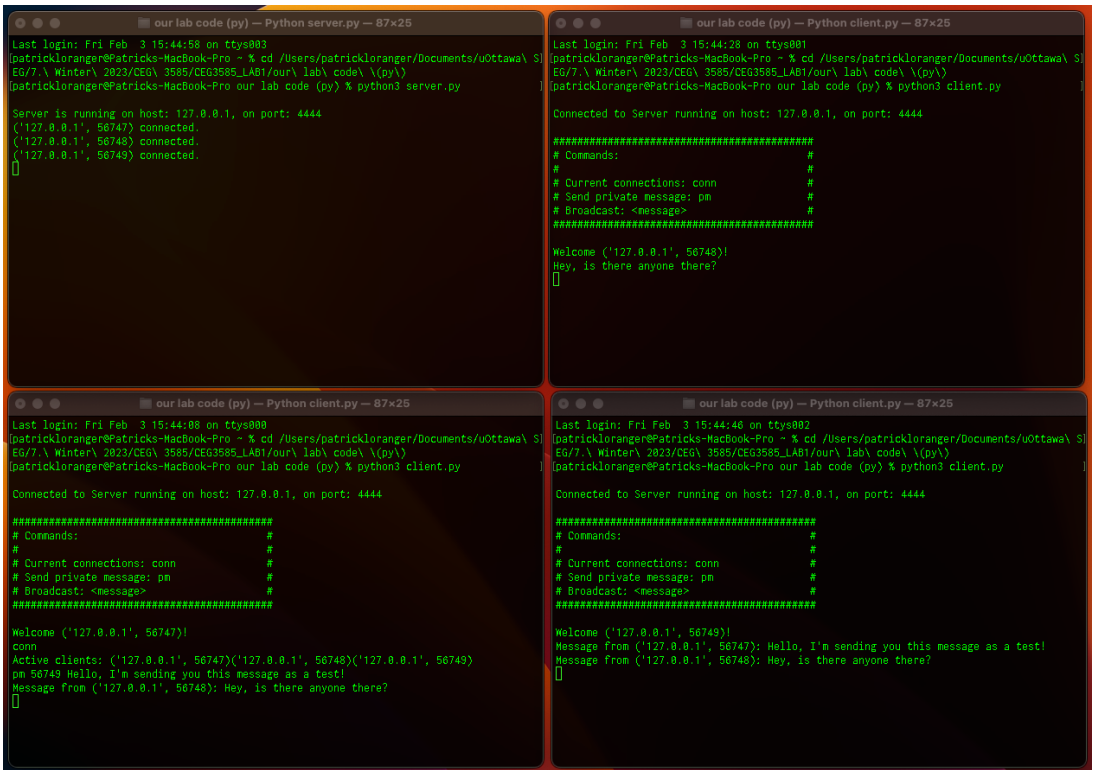


Figure 4 : Le client 56748 envoie un message broadcast à tous les clients dans le serveur. Le message est reçu par tout le monde.

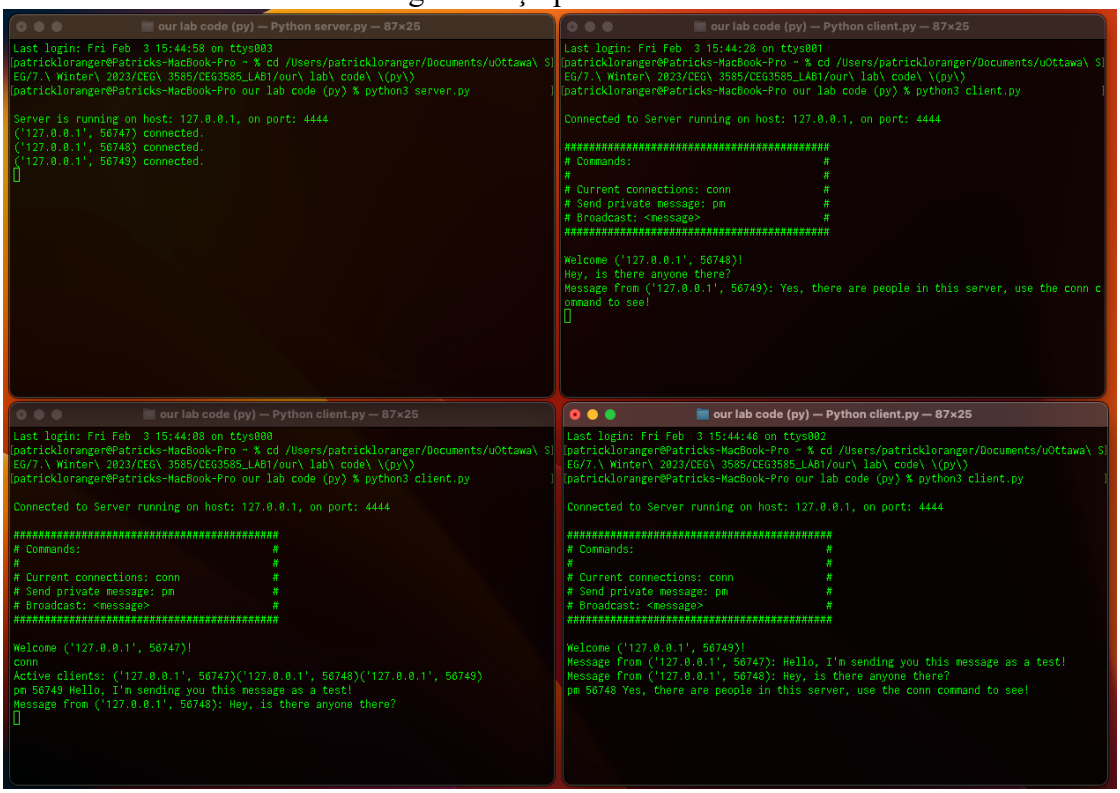


Figure 5 : Le client 56749 réponds au message broadcast avec un message privé. Il explique au client comment faire des messages privés. Seulement le destinataire reçoit le message.

Discussion :

En participant à ce laboratoire on a eu l'opportunité d'apprendre à propos du fonctionnement du protocole de communication TCP/IP surtout à propos de la couche de transport. Ce protocole est tres commun dans le domaine du réseautage et à beaucoup d'application. Par exemple, la possibilité de communiquer entre un client et un serveur. Cette application du protocole est exactement ce qu'il fallait accomplir dans le laboratoire. L'application de bavardage devais pouvoir accomplir deux rôles principaux. La possibilité d'envoyer un message d'un client vers un autre et l'option d'envoyer un message du serveur vers tous les clients. On développer plusieurs fonctions dans le fichier client.py et le fichier serveur.py qui nous ont appris à propos des détails de fonctionnement de ce protocole. L'explication de ces fonctions et comment les utiliser est expliquer dans les sections précédentes de ce document. L'implémentation du code a été faite en python à cause que ce langage offre plusieurs librairies qui sont utiles à la résolution de ce problème. Par exemple la librairie socket qui permet d'initialiser des sockets qui vont représenter le serveur et les clients pour pouvoir leur donner un moyen de communiquer. Aussi, la librairie de threading qui a permis d'accomplir la fonctionnalité d'envoyer un message d'un client vers un autre.

Conclusion :

En conclusion on a réussi à accomplir les tâches nécessaires pour implémenter l'application de bavardage. Le langage de programmation choisi était python à cause de la disponibilité des librairies. Le but du laboratoire qui était d'apprendre à propos du protocole TCP/IP en développent une application de bavardage a été accomplie.