Rendu Projet Réseau: Chat type IRC

Elliot Renel, Théo Dupont

30 avril 2019

Table des matières

1 Introduction

Nous allons présenter ici les différentes fonctionnalités des versions 0 et 1 de notre serveur chat de type IRC. Nous irons ensuite montrer le Protocole réseau mis en oeuvre pour ces applications serveur/client.

2 Fonctionnalités V0

2.1 Connexion au serveur

Tout d'abord, il faut que le serveur soit en cours fonctionnement, en lançant l'application "server.py". Après cela, le client se connecte au serveur via l'application "client.py" qui le connecte automatiquement sur le port 1459.

2.2 Arrivée sur le serveur

Lorsque le client est connecté, il doit alors choisir un pseudonyme (ou nickname en anglais). Le client ne peut pas choisir un nickname déjà utilisé dans le serveur. Lorsqu'il a choisi, le serveur l'acceuille et lui donne la liste de tous les serveur actifs, ou un message si il y en a aucun. Le client a alors accès aux commandes ne demandant pas d'être dans un canal : /JOIN, /BYE, /LIST et /HELP

/HELP permet d'afficher toutes les commandes disponibles sur le serveur, et /LIST donne comme a chaque connexion, la liste de tous les serveurs actifs.

Pour chaque commande, si l'entrée est mauvaise, par exemple une erreur de nickname, ou si un argument n'est pas donné, le serveur enverra un message d'erreur en indicant la commande /HELP.

2.3 Rejoindre un canal

Le client peut alors rejoindre un canal, avec la commande /JOIN <channel> Si <channel> est déjà existant, on rejoint le canal, sinon, il est crée et est ajouté à la liste des canaux. De plus, si l'on crée un canal, on obtient les droits d'administration dont nous détaillerons les fonctionnalités plus tard.

2.4 Quitter un canal

Lorsque l'on a rejoint un canal, on peut à tout moment le quitter avec la commande /LEAVE. Si l'on était l'administrateur du canal, l'administrateur devient le client suivant par ordre d'arrivée. Si il n'y a plus personne après la sortie du client, le canal est supprimé.

2.5 Discuter dans un canal

Une des premières actions que l'on peut réaliser dans un canal, c'est d'envoyer un message à toutes les personnes présentes dans le canal. Pour cela, il

suffit d'écrire le message simplement sans commande devant. Cela enverra automatiquement le messages aux clients concernés. Bien sur le client reçoit aussi les messages qui lui sont destinés.

2.6 Commandes basiques dans un canal

Le client a également accès à plusieurs commandes depuis un canal, par exemple /LEAVE, dont nous avons parlé précédemment. La commande /WHO affiche au client la liste des utilisateurs présents dans le canal, et affiche l'administrateur du canal de la manière : "@<nick>@" pour le reconnaître des autres utilisateurs. /MSG <nick> <message> permet au client d'envoyer un message à une personne en particulier dans le canal. Seule la personne concernée recevra ce message.

2.7 Commandes d'administrateur dans un canal

L'administrateur possède plusieurs commandes additionnelles par rapport aux autres utilisateurs. La commande /KICK <nick> permet d'exclure la personne désignée du canal. La personne se retrouve alors hors du canal, avec la possibilité de le rejoindre à nouveau, ou d'en rejoindre un autre. La commande /REN <channel> permet à l'administrateur de renommer le canal avec un nouveau nom. Toutes les personnes sont concervées dans ce canal.

2.8 Quiiter le serveur

Lorsque l'on n'est plus dans un canal, on peut utiliser la commande /BYE pour quitter le serveur. Le serveur réalise alors la déconnexion entre le client et lui même. Remarque, l'utilisateur ne peut pas utiliser cette commande tant qu'elle est dans un canal.

3 Fonctionnalités V1

Dans cette V1, toutes les fonctionnalités de la V0 sont concervées, seules des modifications de gestion ou des améliorations des fonctionnalités sont réalisées.

3.1 Changer de nom

L'utilisateur peut alors changer de nom a tout moment, avec la commande /NICK <name>. Le client sera reconnu alors avec ce nouveau nom. La règle d'avoir une seule personne avec le meme pseudo s'applique toujours.

3.2 Message privé amélioré

L'utilisateur peut désormais parler en privé à plusieurs personnes dans le canal à la fois avec la commande /MSG <nick1 ;nick2 ;nick3...> <message> Le serveur enverra donc à nick1, nick2, nick3 .... le message indiqué.

3.3 Connexion et gestion de plusieurs canaux

Dans cette version, l'utilisateur peut désormais se connecter à plusieurs canaux en meme temps. L'utilisation de /JOIN n'est donc plus restreintre qu'a si on est déjà dans ce canal ou non.

Cependant, l'affichage des messages n'aurait pas beaucoup de sens si l'on affichait tous les messages de tous les canaux ou le client est présent. On utilise donc un canal "courant" ou l'utilisateur serait actif et ou se passeraient les commandes liées à un canal en particulier, comme /MSG , /LEAVE ou encore /REN. Ainsi, "/KICK Bob" fonctionne uniquement si le canal courant contient Bob dans les clients, et ne l'exclura que de ce canal, pas des autres.

On ajoute alors la commande /CURRENT qui envoie au client le canal courant de celui-ci, et la commande /CURRENT <channel> qui change le canal courant pour le nouveau <channel> Attention, cette dernière ne fonctionne que si l'on a rejoint le canal <channel> avant avec /JOIN <channel>

Remarque, il faut maintenant quitter tous les cannaux pour pouvoir utiliser la commande /BYE.

3.4 Commandes administrateur ajoutées

Tout d'abord, sachez le fait que puisque l'on peut désormais rejoindre plusieurs canaux en même temps, on peut également être administrateur de plusieurs canaux.

L'administrateur peut ne plus être seul à s'occuper d'un canal. En effet, entre en jeu la commande /GRANT <nick> qui donne au client visé les droits d'administration du canal. Et vice versa, un administrateur peut révoquer les droits d'administration à un utilisateur avec la commande /REVOKE <name>

3.5 Envoyer et recevoir un fichier

Un autre ajout de la V1 permet à un client d'envoyer des fichiers à un autre. Pour cela, disont que Alice veut envoyer un fichier à Bob, Alice va ainsi utiliser la commande /SEND <nick> <path> avec Bob à la place de <nick> et le chemin du fichier à ouvrir dans <path> (avec le nom du fichier lui même).

De son côté, Bob reçoit une notification lui indiquant qu'Alice lui a envoyé un fichier. Il peut ainsi le récupérer avec la commande /RECV <path> où path contient le chemin d'accès au fichier (avec le nom du fichier crée).

Attention : Alice doit d'abord envoyer avant que Bob puis recevoir. Il recevra un message d'erreur si aucun fichier ne lui ai déstiné au moment où il tape la commande. La commande /RECVF peut aussi être utilisé pour attendre qu'un fichier soit reçu par Bob et ainsi évité le message précédent, mais n'est pas recommandé car bloque Bob jusqu'à se qu'il recoive le fichier.

Diclaimer : Un fichier temporaire est crée par le serveur puis est supprimé dès que Bob demande à recevoir le fichier.

3.6 Commande HISTORY

La commande /HISTORY permet de consulter les messages précèdement écrits dans le channel courant. Cela affiche spécialement tous les messages du channel qui sont passés par le serveur, même si le client n'est pas présent au début de la conversation. Cet historique est alors sauvegardé par le serveur et consultable par n'importe quel utilisateur dans le canal.

4 Protocole réseau mis en oeuvre

Nous utilisons ici le protocole TCP car il est plus simple et il possède plus de fonctionnalités adéquates que UDP, notamment pour l'envoi de files.

Côté client on crée une socket et l'on tente de se connecter au serveur via le port 1459.

Côté serveur, la socket attend la connexion d'un client. Lorsqu'un client essaie de se connecter, le serveur l'accepte et l'ajoute à la liste de sockets, puis l'invite a envoyer des données necessaires pour son identification.

A partir de là, côté client, le script client.py va interpréter chaque message écrit par l'utilisateur en récupérant la commande tapé et assigne un chiffre hexadécimal au premier caractère des données envoyées qui identifiera la commande demandée par le client (l'un est aussi prévu pour les messages classiques). client.py envoie ainsi ce byte suivi des autres données tapées par l'utilisateur. Le serveur, lors de la récupération du message, va ainsi donnée le chiffre hexadécimale et le reste du message à la fonction switchman qui décidera quoi faire. Ainsi le serveur redistribue chaque tâche a une fonction en particulier, et le serveur fait quitter le client lorsque celui-ci se déconnecte.

Nous tenons à remercier nos professeurs pour l'aide qu'ils nous ont apporté, et d'avoir réussi à lire ce rapport malgré les fautes d'orthographes commise.