Lab 3 : Programmation Socket et B8ZS

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, 10 mars, 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 de l'encodage et du décodage du flux B8ZS à l'aide du programme de socket client / serveur. Le premier but est de trouver deux algorithmes. L'algorithme #1 permet d'encoder des entrées binaires (ex : 100100) en B8ZS. L'algorithme #2 permet de décoder les entrées B8ZS et les retourner en format binaires. Le deuxième but important pour ce laboratoire est de créer un système client / serveur qui prend l'input du client pour savoir quel entrée binaire encoder. Il y a de la communication entre le client et le serveur pour bien accepter le message et finalement, le serveur de suite affiche le message encoder et décoder.

Explication de l'algorithme de notre solution :

Nous avons utilisé Python 3 pour notre solution de l'application d'encodage / décodage B8ZS.

Comment utiliser le code :

1. Assurer d'avoir télécharger tous les fichiers py, et les mettre dans le même répertoire.
2. Créer une instance de serveur dans le terminal de choix avec la commande suivante : *python3 server.py*
3. Créer une instance de client dans le terminal de choix avec la commande suivante : *python3 client.py*
4. Entrée le nombre binaire que vous désirez encoder dans le terminal du client.
5. Répétez l'étape 4 jusqu'à ce que tu aies encoder tous les nombres binaires désirez.

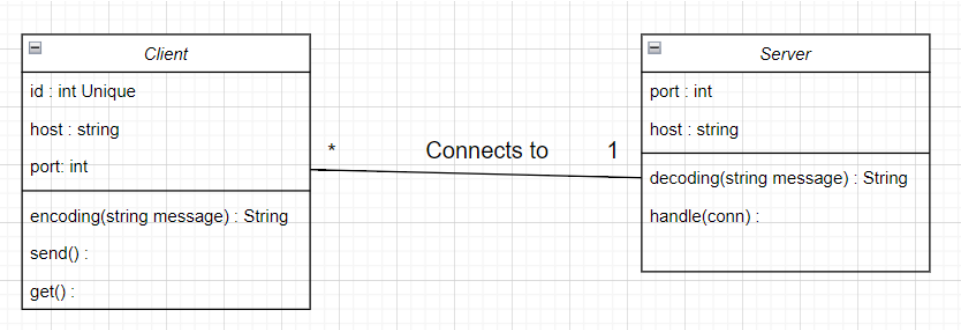
La partie la plus importante de cette application est la fonctionnalité d'encodage et de décodage. Pour encoder, nous vérifions s'il y a huit zéros de suite. Si oui, on remplace cette séquence par 000+-0-+ ou par 000-+0+- dépendant du change bit. Pour décoder, on prend l'inverse de ceci et on transforme les séquences 000+-0-+ et 000-+0+- en séquence 00000000.

De suite, il est important de comprendre le fonctionnement du serveur. Nous utilisons des sockets 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.

Dans le serveur, la fonction handle prend le texte entré par le client et fait un appel à la fonction decoding(message). Le processus est complété.

Finalement, 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, il suffit d'envoyer un code binaire pour encoder.

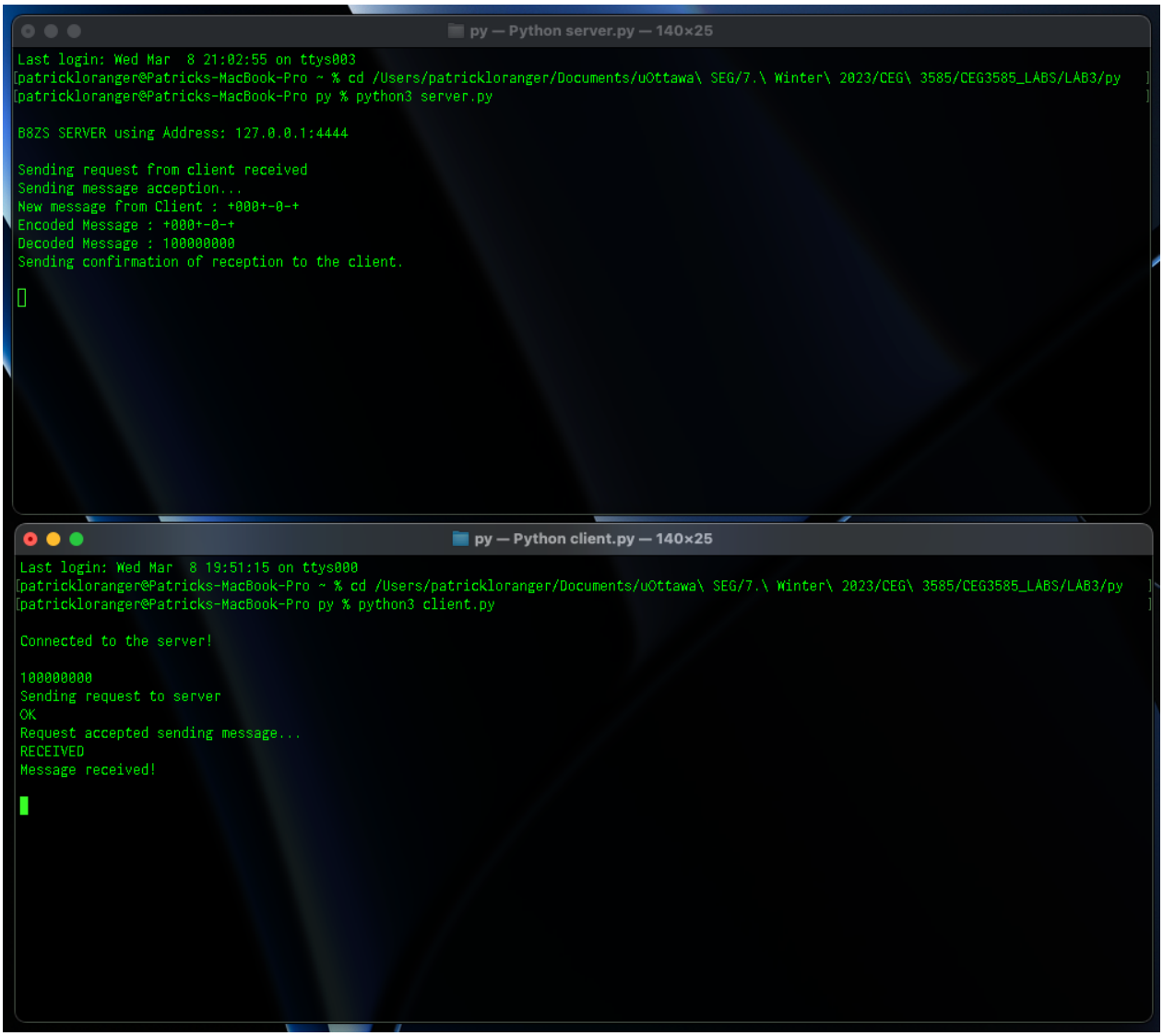
Document de conception et diagrammes UML :



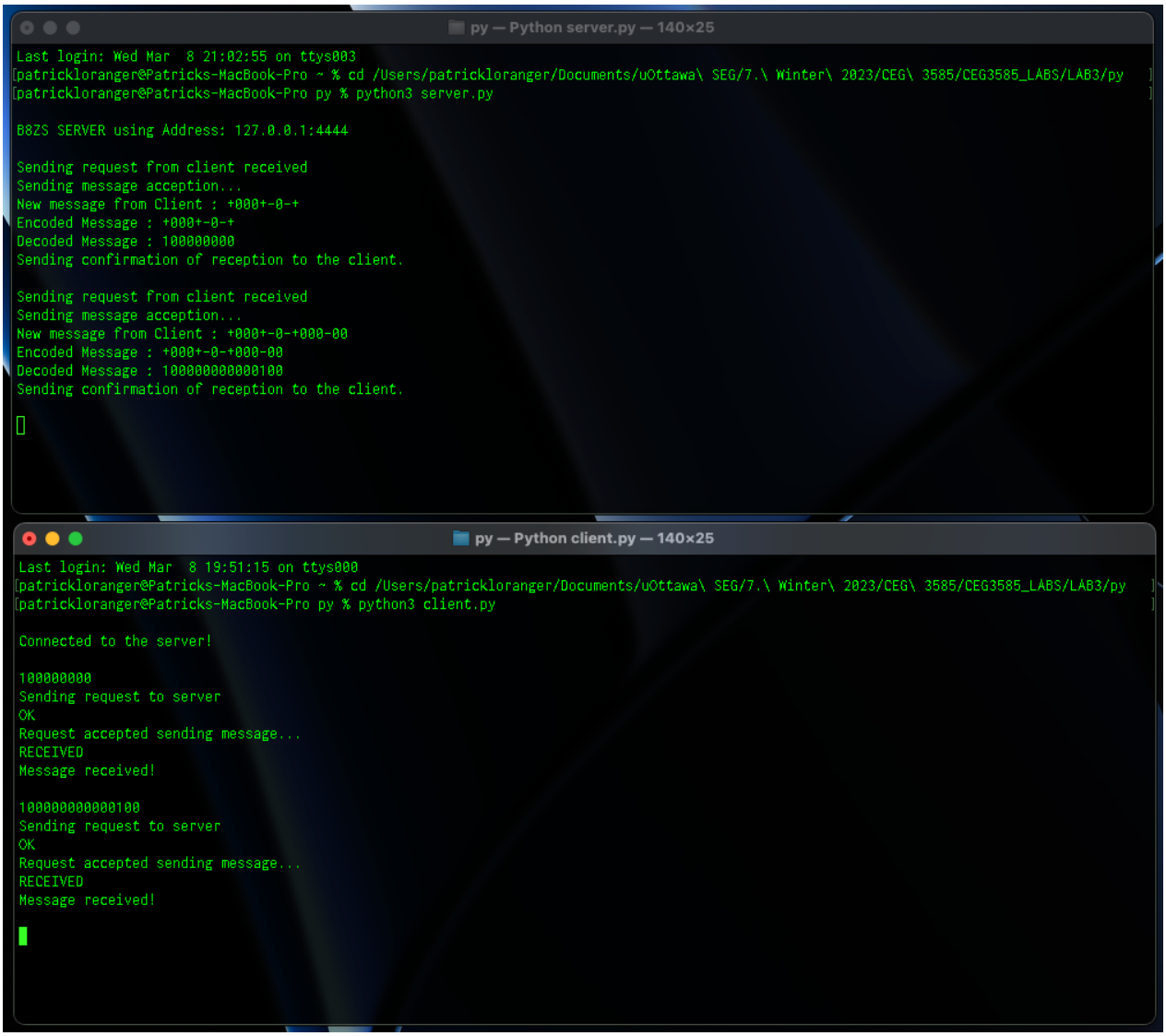
Captures d'écrans de la démonstration de l'application :

Les captures d'écrans sont de bonnes qualité, il suffit juste d'agrandir l'image si c'est difficile à voir.

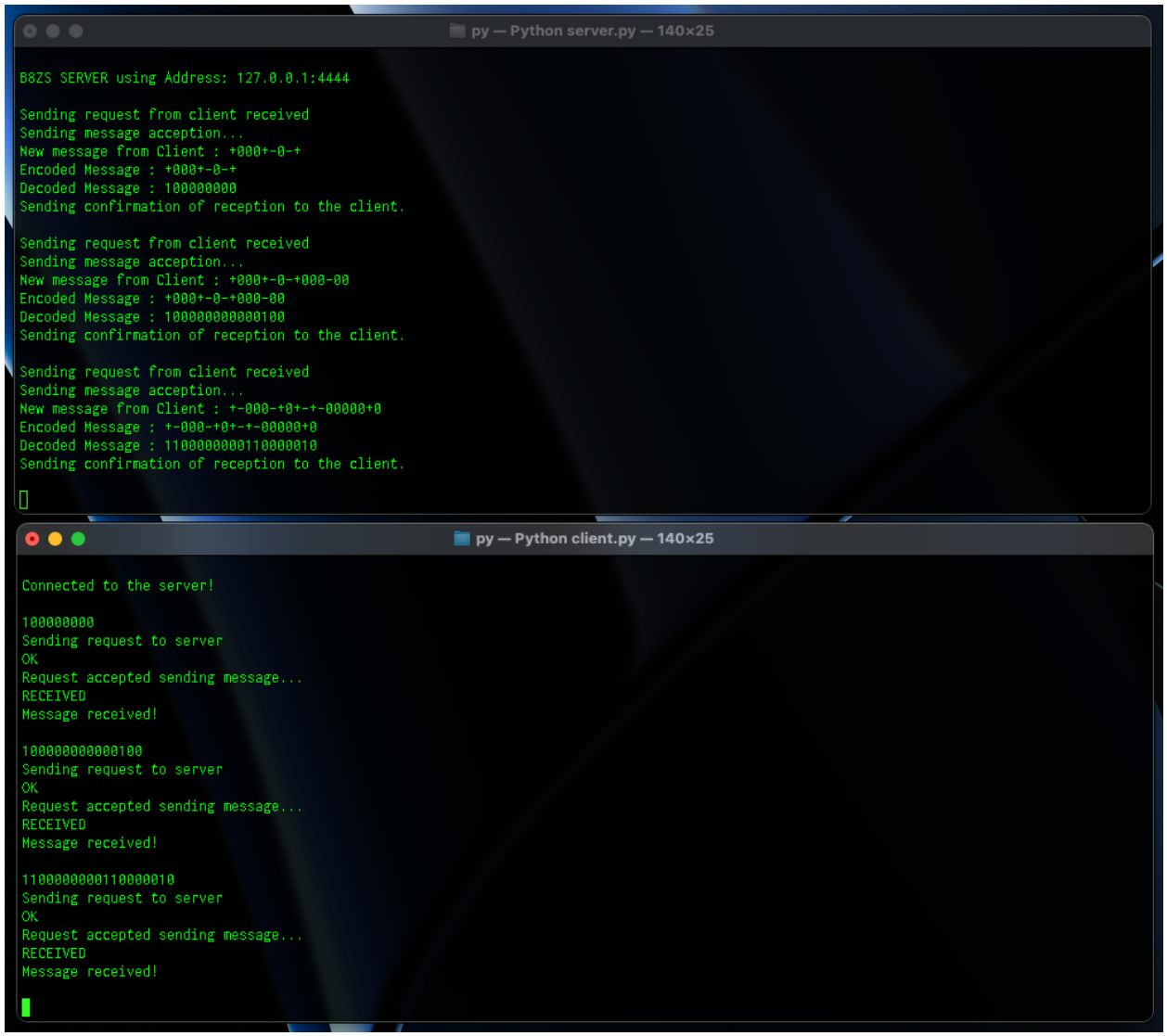
Le serveur est mis en marche, et le client se connecte au serveur.



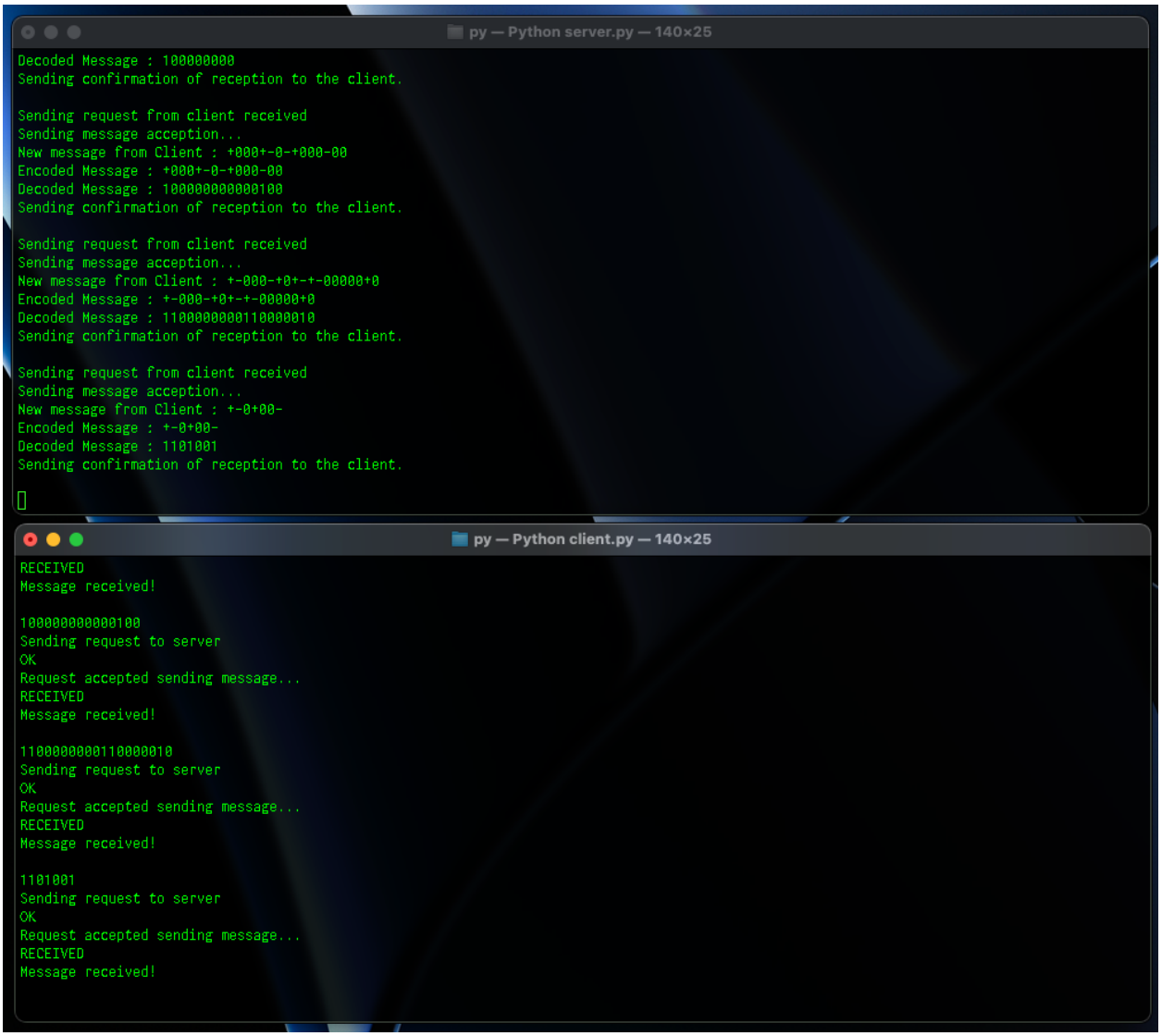
Le client encode et envoie le message : 100000000.
Le serveur accepte et décode le message.



Le client encode et envoie le message : 100000000000100.
Le serveur accepte et décode le message.



Le client encode et envoie le message : 1100000000110000010.
Le serveur accepte et décode le message.



Le client encode et envoie le message : 1101001.
Le serveur accepte et décode le message.

Discussion :

En participant à ce laboratoire on a eu l'opportunité d'apprendre à propos des techniques d'encodages et de décodage B8ZS. Cette technique est utilisée pour faire l'encodage et le décodage de messages binaires pour envoyer des signaux d'une machine à l'autre. Le but de ce format de transmission est de prévenir le fait d'avoir trop de 0 consécutifs envoyé dans un message. On devait trouver une manière de vérifier qu'il y avait vraiment 8 0 consécutifs. Il fallait aussi inclure les 2 types de substitution possible dans le code. Soit 000+-0-+ et 000-+0+-. On a aussi implémenté une manière de vérifier quel des 2 substitutions utilisée en vérifiant si la dernière impulsion était positive ou négative. Dans ce laboratoire, il fallait aussi, implémenter un système client-serveur qui permettrait de transmettre le message encodé du client vers le serveur et le serveur devait décoder ce message. Les détails du fonctionnement de ce système sont discutés dans une section précédente du laboratoire. Ce système était inspiré par le code implémenter dans le cadre du laboratoire 1. 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.

Conclusion :

En conclusion on a réussi à accomplir les tâches nécessaires pour implémenter le système client-serveur qui permet d'encoder et de décoder des messages binaires selon la technique d'encodage B8ZS. Le langage de programmation choisi était python à cause de la disponibilité des librairies et pour pouvoir réutiliser du code du laboratoire 1. Les but du laboratoire qui était d'apprendre à propos de la technique d'embrouillage B8ZS en développent un système client-serveur avec des fonctions d'encodage et de décodage de messages a été accomplie.