Rapport projet de réseau

DELAR Emmanoe, RAKOTOARIJAONA Camille

25 avril 2017

Table des matières

1 Introduction

Lors de ce projet, réalisé en binômes, nous avions pour objectif de développer un jeu de bataille navale en réseau. Pour cela, nous avons utilisé le language de programmation objet Python. On est parti du code source fourni par notre enseignant. Ce code nous permettait de jouer contre la machine uniquement. Dès lors, nous avons dû l'améliorer afin de pouvoir jouer à 1 contre 1 sur le réseau.

2 Fonctionnement du jeu

2.1 Présentation du jeu

La bataille navale est un jeu de société dans lequel deux joueurs doivent placer des navires sur une grille tenue secrte et tenter de toucher les navires adverses. Le gagnant est celui qui parvient à torpiller complètement les navires de l'adversaire avant que tous les siens ne le soient.

2.2 Jouer contre la machine

Lorsqu'on lance le programme avec le nom du serveur en argument, si on choisi de ne pas jouer en réseau alors le jeu commence directement. On choisi la colonne et la ligne de la grille de l'ordinateur à viser. Quand c'est au tour de l'ordinateur, une fonction fait de même, choisie des coordonnées aléatoirement puis joue son coup. Et ainsi de suite jusqu'à la fin de la partie.

2.3 Jouer en réseau local

Si on choisi de jouer en réseau, comme demandé dans le sujet du projet, il faut dun côté, démarrer le serveur en lançant le programme ≪main.py≫ sans arguments, et du côté client, il faut lancer le programme avec ladresse ip du serveur en argument.
Lorsque ceux-ci sont lancés, le client aura la possibilité de choisir dabord si il veut se connecter au serveur, ou si il veut jouer seul face au robot comme décrit au dessus.
Si il décide de jouer en réseau contre quelqu'un, le programme attendra qu'une deuxime personne se connecte au serveur pour jouer aussi, et ainsi ils pourront jouer tous les deux.
Ensuite, chacun jouera à tour de rôle jusqu'a ce que la partie soit terminée, lorsque la partie est terminée chaque joueur verra le score, et le serveur se déconnectera et le jeu de chaque joueur aussi.
Il est à noter que les joueurs peuvent donc jouer tout seul peu importe le moment sans affecter le serveur lorsqu` ils jouent en local.

3 Structure du code

3.1 Serveur

Contrairement à la partie contre l'ordinateur, en mode réseau, les coordonnées choisies par le client sont envoyés au serveur qui les retransmet au deuxième joueur connecté. À l'aide du constructeur socket, nous avons alors créé un serveur de famille ipv6 en connexion TCP. Nous l'avons rattaché au port 7777 (choisi arbitrairement) puis nous avons lancé notre serveur en mode écoute. parler du numero de port.

La partie serveur est un peu complexe car il joue le rle de lien entre les joueurs. Au lancement de celui-ci, nous nous trouvons dans la fonction main principale, tout cela par la mthode select apprise en cours. Lorsque 2 joueurs sont enfin connects, le serveurs envoie aux deux joueurs un numro, 0 ou 1, en fonction de celui qui sest connect en premier laide de la mthode socket.send qui dfini le joueur qui jouera en premier et celui qui jouera en deuxime. La deuxime tape, est la partie la plus complexe et celle qui nous a pos le plus de problmes car cest celle qui consiste envoyer le jeu qui a t cre par le serveur. En effet nous avons fait le choix de de faire crer un jeu alatoire par le serveur dans ce cas-ci et de lenvoyer aux deux joueurs, ainsi ,implmenter lextension pour pouvoir avoir des spectateurs est plus simple car, nous avons modifi le code principal et on peut donc envoyer le jeu dautres joueurs aussi, ce qui peut permettre des spectateurs de voir le jeu se drouler. La fonction qui a t modifi est, la fonction randomConfiguration qui crer la table de bateaux alatoirement. Tout dabord dans la fonction tout se passe normalement comme dans la fonction prcdente, cependant, chaque fois que lon a stock une coordonne et une position de bateau, on les mets dans un tableau, ensuite, on envoie le tout aux deux joueurs travers une boucle parcourant le tableau. Cette partie nous a pos plusieurs problmes, nous avons par exemple eu un problmes au niveau de la taille des bateaux car les valeurs devaient tre envoyes en bytes et avaient des tailles variables, de 1 2 bytes car on avait des valeurs de 1 10 et 10 tait reprsent sur 2 bytes, on a donc dcid de retire 1 chaque valeur pour tre sur 0 9 et ainsi avoir une taille au maximum de 1 byte et chaque valeurs, reue de lautre ct par les joueurs serait incrmente de un, tout cela fait pour lenvoie des navires. Le deuxime problme tait aussi un peu complexe, mais aprs dbogage, on a pu voir que lon envoyait avant directement chaque valeur alatoire sans avoir au pralablement vrifi si on avait une configuration valide travers la fonction isValidConfiguration . On a alors compris quil fallait prendre les dernires valeurs de configuration valide et envoyer le tout chaque joueurs aprs. Et enfin comme la fonction basique, celle-ci retourne un bateau. On rpte alors ceci 2 fois et pour avoir la table de jeu. Ensuite, cest le dbut de la partie, la boucle qui fait jouer le serveur est la mme que celle qui fait jouer le robot ou chaque joueur, cependant, celui-ci se place dabord dans la partie du premier joueur de la boucle, et attendra que le joueur envoie son coup, il mettra le coup sur sa table de jeu et lenverra lautre joueur, tout cela avec la mthode recv et send. Et passera ensuite la partie du

joueurs 2 de la boucle et fera de mme. Et ainsi de suite jusqu ce que le jeu soit fini. Ensuite il sort de la boucle et ferme les socket et la fonction et le serveur.

3.2 Connexion client

Pour que le client se connecte au serveur, il nous faut le nom de l'hôte et le numéro de port.
Ensuite, on va créé une socket (client) compatible ipv6 qui va nous permettre d'ouvrir une connexion avec une machine locale et d'échanger des informations (à l'aide de la méthode "bind").
Pour récuperer les premieres informations, notamment le numéro du joueur, les navires, on intercepte les premiers bytes envoys par la socket serveur (à l'aide de la méthode "client.recv")
Avec de ces informations on génére une partie à partir d'une fonction Game
Une fois la partie lancée, si c'est au joeur actuel de jouer, il entre les coordonnées à viser. Ensuite, on va envoyer ces coordonnées (x,y) au serveur. Le serveur va retransmettre ces informations sous forme de byte á l'autre joueur (à l'aide de la méthode "client.send").
Par contre si c'est au tour de l'autre joueur, la socket client se met en mode réception et attend les coordonnées choisies par l'adversaire (à l'aide de la méthode "client.recv"). Ces données sont envoyées par le serveur.
Ce protocole est répété jusqu'à la fin de la partie.

3.3 Protocole de communication

Une fois la connexion établie, la communication serveur/client est très simple. Chaque information nécessaire à l'initialisation d'une partie est envoyé par le serveur bit à bit et ainsi récupéré du côté client. Il faut noter la nécessité à convertir les bytes recus en entier (int).

4 Extensions

5 Conclusion