Sarah Zouari – Anaïs Rivière

Classe : B2A

Groupe : 15



## Contexte :

Dans le cadre du projet de développement logiciel de deuxièmes années de bachelor de l'école Paris Ynov Campus, nous avons choisi le sujet de matchmaking.

Nous présentons donc un jeu de puissance 4 multi-joueur en ligne.

## Le serveur :

Nom du fichier : server.py

Le serveur est divisé en deux parties, une partie qui en fonction de la connexion du client lui attribut son id (voir fichier jeu.py)

```
while True:
    conn, addr = s.accept()
    print("Connected to:", addr)



    idCount += 1
    p = 0
    jeuId = (idCount -1)//2
    if idCount % 2 == 1:
        jeux[jeuId] = Jeu(jeuId)
        print("création d'un nouveau jeu")
    else:
        jeux[jeuId].ready = True
        p = 1




    start_new_thread(threaded_client, (conn, p, jeuId))
```

Et une partie qui récupère/renvoie les informations

```python
def threaded_client(conn, p, jeuId):

    global idCount
    conn.send(str.encode(str(p)))

    reply = ""

    while True:
        try:
            data = conn.recv(4096*5).decode()

            if jeuId in jeux:
                jeu = jeux[jeuId]

                if not data:
                    break
                else:
                    if data == "reset":
                        jeu.reset()
                    elif data != "get":
                        jeu.choix(p, data)

                    reply = jeu
                    conn.sendall(pickle.dumps(reply))

            else:
                break
        except Exception as e:
            print("Failed try")
            print(e)
            break

    print("Lost connection")

    try:
        del jeux[jeuId]
        print("closing game", jeuId)
    except:
        pass
    idCount -= 1

    conn.close()
```

Quand on lance le serveur, on peut constater qu'il attend la connexion

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

PS C:\Users\Anais\Desktop\puissance4-projet-dev> & C:/Users/Anais/AppData/Local/Programs/Python/Python38/p
ython.exe c:/Users/Anais/Desktop/puissance4-projet-dev/server.py
waiting for a connection
```

## Le client :

La classe network permet de se connecter :

```
class Network:
    def __init__(self):
        self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server = "192.168.209.177"
        self.port = 5555
        self.addr = (self.server, self.port)
        self.p = self.connect()
```

Une fois la connexion établie avec le premier client, celui-ci attend la connexion d'un deuxième client :

```
En attente d'un deuxième joueur
En attente d'un deuxième joueur
En attente d'un deuxième joueur
En attente d'un deuxième joueur
En attente d'un deuxième joueur
```

Puis une fois le deuxième client connecté, celui-ci va attendre que le premier joueur joue :

```
ok
attendre que le joueur 1 joue...
ok
```

Le premier joueur peut jouer :

```
ok
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]]
Joueur 1 choisi la colonne (0-6)
```

Une fois que le premier joueur a joué, nous pouvons constater que le plateau du deuxième joueur a bien été mis à jour. Les informations transitent donc correctement entre les machines.

La partie se déroule ensuite comme une partie de puissance 4 classique.

Lorsque qu'un des joueurs gagne, les deux joueurs sont avertis de leur victoire ou de leur défaite :

```
 [0. 1. 2. 2. 2. 0. 0.]]
Joueur 1 choisi la colonne (0-6)4
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 2. 1. 0. 0.]
 [0. 1. 2. 2. 2. 0. 0.]]
test
Tu as gagné! Bravo!
ok
```

```
OK
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 1. 2. 0. 0.]
 [0. 0. 1. 2. 1. 0. 0.]
 [0. 1. 2. 2. 2. 0. 0.]]
Le joueur 1 a gagné!
```