Rapport - Projet de réseau

Adrien Chinour
Camille Meyrignac

27 avril 2017

Table des matières

1 La couche transport

Pour passer le jeu en réseau, il a fallut d'abord définir la couche transport, car c'est elle qui fait le lien entre les couches physiques et la couche session utilisé par notre jeu. On a donc d'abord réaliser :

- la fonction permettant de créer un serveur lorsque l'utilisateur entre la commande ./main.py
- la fonction permettant de connecter un client a un serveur déjà existant avec la commande ./main.py IP PORT avec IP correspondant à l'ip du serveur et PORT le port sur lequel le serveur écoute.

Les fonctions sont assez simple, la première créer un socket (avec l'option

socket.SOCK_STREAM pour utiliser la version TCP) pour le serveur qui écoute sur le port 7777 en boucle. Et la deuxième créer un socket pour le client et le connecte au serveur. Code des deux fonctions :

```
def createServer():
    s = socket.socket(socket.AF_INET6,socket.SOCK_STREAM,0)
    s.bind(('',7777))
    s.listen(1)
    return s

def createClient(IP,port):
    s = socket.socket(socket.AF_INET6,socket.SOCK_STREAM,0)
    s.connect((IP,int(float(port))))
    return s
```

2 Notre protocole personnel

Maintenant que notre couche transport est prête il faut mettre en place notre protocole permettant au serveur et aux clients de communiquer.
Comme pour la partie precedente nous avons créer deux fonctions une executer par le serveur pour lire les messages des cliens et l'autre executer par les clients pour lire les messages du serveur.

Notre protocole est basé sur un prefix qui contient le type de message envoyé. Cela nous permet de transmettre le minimum d'information et que le client est le choix de son affichage. Il peut par exemple modifier sa couche application en créant une interface graphique sans que cela affecte les communications serveur/client. Par exemple pour indiquer au client que le serveur attent un identifiant pour le connecter au serveur il lui envoie 'US'.
Pour le plus important - c'est-à-dire l'envoi de la partie en cours aux joueurs et aux observateurs - notre protocole envoi le prefix 'YT' (your turn) ou 'WT' (wrong turn) suivi des 200 caractères composant la grille du client. De cette manière on évite a chaque joueur de connaitre la partie de l'adversaire et d'avoir à effectuer des calculs inutiles puisqu'il a juste a convertir les 200 caractères de la grille comme il le désire.

Dans l'autre sens, si le client a reçu l'information commençant par 'YT' alors les coordonnées qu'il tape sont envoyées au serveur à l'aide du prefix 'AS' (add shot) pour que le serveur ajoute le tir à la partie en cours.

3 Les extensions réalisées

Pour rendre le programme plus intéressant et plus complet en plus du simple échange coordonnées / partie. Nous avons réaliser plusieurs extensions surtout pour améliorant les échanges serveurs / clients.

3.1 Système de compte

3.2 Possibilité de rejouer

Pour éviter d'avoir à redémarrer le serveur après la fin d'une partie, on a ajouter a notre protocole un prefix 'END' suivi du numéro du joueur gagnant que le serveur envoi au client pour lui signaler que la partie est terminé. Le client a alors le choix entre jouer la prochaine partie ou devenir observateur. Il répond à l'aide du prefix 'PLAY' suivi de sa réponse.
Dès que deux clients ont décidé de rejouer alors une nouvelle partie commence.

```
#partie serveur
elif(m.startswith('PLAY')):
    if m.lstrip('PLAY') == '0' and nbp < 2:
        joueur[nbp] = (socket,users[sockuser[socket]])
        nbp += 1
        socket.send(("WC"+str(nbp)+sockuser[socket]).encode())
        print(str(nbp))
        if nbp == 2:
            print("restart game")
            sendToAll(l, server)
    else:
        socket.send(("WC0" + sockuser[socket]).encode())

#partie client
elif(m.startswith('END')):
    print("Partie terminé : Le joueur " + m[3:] + " à gagné!")
```

```
    message = input('Envie de jouer ? (o/n):\n')
    socket.send(('PLAY'+format(message.capitalize())).encode())
```

3.3 Les observateurs

En plus des clients "joueur", il est intéressant d'ajouter des clients "observateur" qui peuvent voir la partie en cours.
Pour faire ça, on a ajouté un cas particulier à notre fonction qui envoie la partie aux clients pour qu'elle envoie les bateaux des deux joueurs. Le serveur envoi donc les 200 caractères de la partie avec le prefix 'WT' car ce n'est pas à lui de jouer.

```
# pour les observateurs
else:
    data = "WT"
    for i in range(2):
        data = data + getConfiguration( game.boats[i],
                                        game.shots[(i+1)%2],
                                        showBoats=True)
    socket.send(data.encode())
```

3.4 Connexions sécurisées