

Compte rendu final - Projet Systèmes et réseaux

Dusoleil Yoan

Décembre 2022

Contents

1	Introduction	2
2	Choix techniques	2
2.1	Les Sockets	2
2.2	Architecture Master/Slave	2
3	Déroulement d'une partie	3
3.1	Lancement du serveur	3
3.2	Lancement d'un joueur	3
3.3	Tour de jeu	4

1 Introduction

Ce projet a été réalisé dans le cadre du module Systèmes et réseaux. Le but du sujet était de développer le jeu du nom de "6 qui prend". Pour ce faire, l'application a été développée en C afin de bénéficier des sockets permettant donc de jouer en ligne. L'architecture du projet est la suivante :

- Un processus "Gestionnaire de jeu" (ici le serveur) : Son but est de gérer les différentes connexions des sockets clients et de rythmer le jeu à l'aide des différentes fonctions permettant d'ordonner aux clients d'effectuer des actions d'affichage/demande de saisie.
- Un processus "Joueur" (ici le client) : Ce processus reçoit des ordres venant du processus Gestionnaire de jeu. Il peut recevoir 3 types d'ordres : "show" qui indique au client d'afficher au joueur le prochain message que le client va recevoir de la part du serveur, "prompt" qui lui indique de demander au joueur de saisir quelque chose et de répondre au serveur avec le contenu saisi par l'utilisateur.
- Un processus "Robot" : Ce processus est exécuté depuis le serveur si jamais on souhaite ajouter un robot à la partie. Il consiste à répondre au serveur avec des valeurs aléatoires.

2 Choix techniques

2.1 Les Sockets

Les sockets, bien que bloquants dans leur opération, permettent une communication entre des processus en réseau. Le fait qu'ils soient bloquants ici n'est d'ailleurs pas un problème puisque le jeu du "6 qui prend" est un jeu en tour par tour.

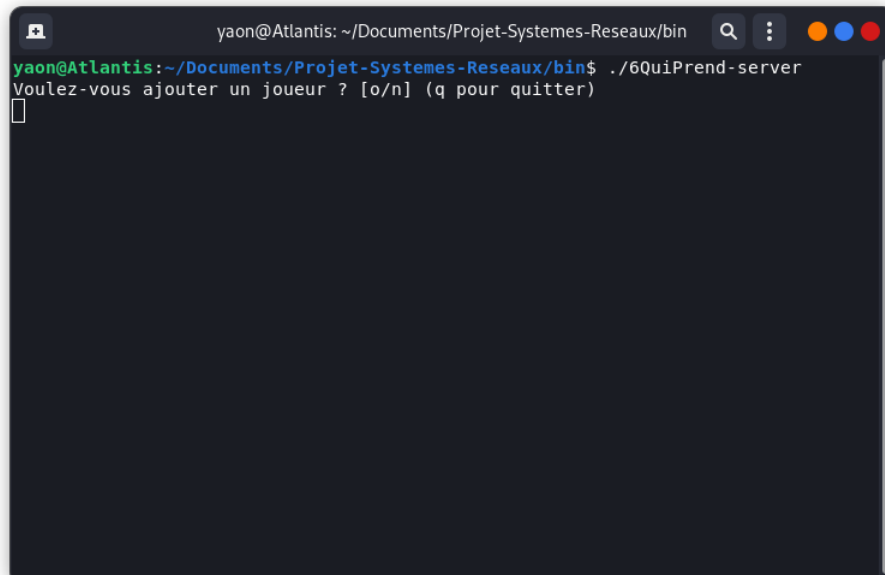
2.2 Architecture Master/Slave

Le serveur ici fait office de "maître" et le client "d'esclave". Une des limitations de cette architecture étant, dans une situation où le serveur cesse de communiquer, le client restera bloqué en attente d'une directive. Cette architecture a été choisie à cause du manque de temps pendant la réalisation de ce projet. En effet, la mise au point de la communication interprocessus semblait plus importante que la mise au point du jeu en lui-même, c'est pourquoi, le moyen de développer le jeu le plus vite possible après la démonstration est de développer l'entièreté du jeu côté serveur et d'asservir complètement le client aux directives du serveur.

3 Déroulement d'une partie

3.1 Lancement du serveur

Le serveur peut-être démarré avec le fichier "server_exec" à la racine du projet. Ce script shell compile tous les fichiers source du serveur et exécute le serveur. Par la suite, l'utilisateur est accueilli par la fenêtre suivante :

A terminal window titled 'yaon@Atlantis: ~/Documents/Projet-Systemes-Reseaux/bin'. The prompt is 'yaon@Atlantis:~/Documents/Projet-Systemes-Reseaux/bin\$'. The user has entered './6QuiPrend-server'. The output is 'Voulez-vous ajouter un joueur ? [o/n] (q pour quitter)' followed by a cursor on a new line.

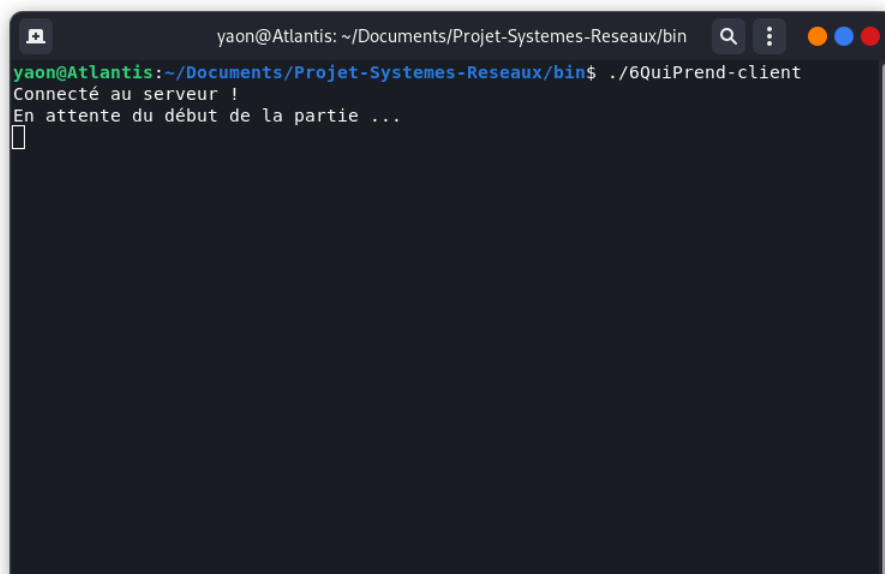
```
yaon@Atlantis: ~/Documents/Projet-Systemes-Reseaux/bin
yaon@Atlantis:~/Documents/Projet-Systemes-Reseaux/bin$ ./6QuiPrend-server
Voulez-vous ajouter un joueur ? [o/n] (q pour quitter)
█
```

L'utilisateur possède alors 3 possibilités de saisie :

- "o" : Le serveur demande alors à l'utilisateur si il souhaite ajouter un joueur humain ou un joueur robot, se met en attente d'une connexion de la part d'un joueur. Une fois la connexion établie, le serveur demandera à nouveau à l'utilisateur ce qu'il souhaite faire.
- "n" : Lance la partie si au moins 2 joueurs sont connectés.
- "q" : Quitte le programme.

3.2 Lancement d'un joueur

Le client peut-être démarré avec le fichier "client_exec" à la racine du projet. Comme pour le serveur, ce script shell compile l'entièreté des fichiers source du client et exécute ce dernier. L'utilisateur est ensuite accueilli par la fenêtre suivante :

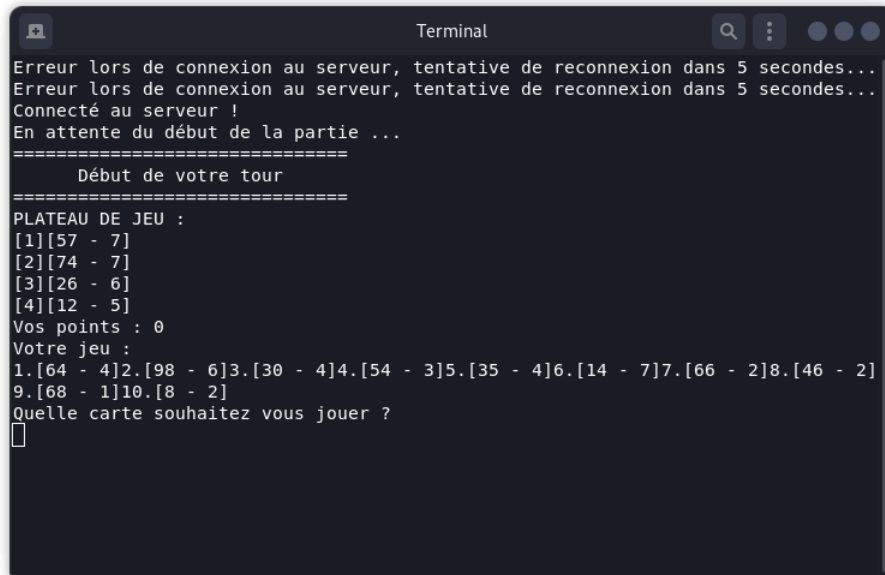
A terminal window titled 'yaon@Atlantis: ~/Documents/Projet-Systemes-Reseaux/bin'. The prompt is 'yaon@Atlantis:~/Documents/Projet-Systemes-Reseaux/bin\$'. The user has entered './6QuiPrend-client'. The output is 'Connecté au serveur !' followed by 'En attente du début de la partie ...' and a cursor on a new line.

```
yaon@Atlantis: ~/Documents/Projet-Systemes-Reseaux/bin
yaon@Atlantis:~/Documents/Projet-Systemes-Reseaux/bin$ ./6QuiPrend-client
Connecté au serveur !
En attente du début de la partie ...
█
```

Si le client n'arrive pas à se connecter au serveur, une nouvelle tentative sera effectuée à intervalle de 5 secondes entre chaque tentative.

3.3 Tour de jeu

Une fois la partie lancée, chaque joueur verra la fenêtre suivante au début de son tour :



```
Terminal
Erreur lors de connexion au serveur, tentative de reconnexion dans 5 secondes...
Erreur lors de connexion au serveur, tentative de reconnexion dans 5 secondes...
Connecté au serveur !
En attente du début de la partie ...
=====
Début de votre tour
=====
PLATEAU DE JEU :
[1][57 - 7]
[2][74 - 7]
[3][26 - 6]
[4][12 - 5]
Vos points : 0
Votre jeu :
1.[64 - 4]2.[98 - 6]3.[30 - 4]4.[54 - 3]5.[35 - 4]6.[14 - 7]7.[66 - 2]8.[46 - 2]
9.[68 - 1]10.[8 - 2]
Quelle carte souhaitez vous jouer ?
█
```

Le joueur peut donc choisir la carte qu'il souhaite jouer puis choisir la rangée dans laquelle il souhaite la jouer. Le traitement permettant de savoir quelles cartes sont jouables et quelles rangées sont permises est fait côté serveur. Le serveur redemande juste au client de saisir une nouvelle valeur si l'utilisateur entre une valeur incorrecte.