2020 - 2021

Tennis en Réseaux

Projet-2 INFO0503

TONNELLE Nathan & MEDDOURI Sofiane
GROUPE S5O7A

Table des matières

Introduction

Le but de ce projet était de développer 2 applications permettant de mettre en relation plusieurs utilisateurs pour jouer au tennis (une application utilisant exclusivement le protocole UDP et une autre exclusivement le protocole TCP).

Sur le terrain de jeu, on peut distinguer 5 positions différentes numérotées de 1 à 5 et correspondant, respectivement, à gauche, milieu-gauche, milieu, milieu-droite et droite. Au début de chaque partie, tous les joueurs sont placés au milieu du terrain (la position n° 3). Le joueur qui sert en premier est choisi aléatoirement. À tour de rôle, les joueurs choisissent de tirer sur une des 5 positions possibles. Le joueur d'en face, son adversaire, devra rattraper la balle, sans pour autant savoir où elle est tirée, il a le choix entre rester à sa place actuelle, ou de se déplacer sur une position en plus ou en moins, par rapport à sa position actuelle.

La balle sera rattrapée ou non en suivant ces règles :

- Si le joueur est sur la même position que la balle, alors elle est rattrapée.
- Si le joueur se situe sur une case adjacente à celle de la balle, alors il y a 50 % de chance qu'elle soit rattrapée.
- Dans le cas contraire, la balle n'est pas rattrapée.

Dans le cas où le joueur manque la balle, son adversaire marque un point, un nouveau jeu commence et c'est le joueur qui a gagné le point qui sert. La partie s'achève dès qu'un joueur arrive à avoir 5 points, minimum, avec au moins 2 points d'écart.

Exemple de déroulement d'une partie

La partie suivante se déroule entre les joueurs nommés : Sofiane et Nathan.

- Sofiane et Nathan sont à la position 3 (position du milieu)
- Sofiane sert.
- Sofiane choisit de tirer en 4 (position milieu-droite)
- Nathan se déplace en 4
- Il réussit à rattraper la balle
- Nathan relance en 5 (position droite)
- Sofiane ne se déplace pas.
- Il manque la balle (Nathan gagne le point, le score est de 1 – 0 pour Nathan)
- Sofiane et Nathan se remettent à la positon 3
- Nathan sert.
- Il choisit de tirer en 2 (position milieu-gauche)
- Sofiane se déplace en 4.
- Il manque encore la balle (Nathan gagne le point. Le score est de 2 – 0 pour Nathan).
- Sofiane en a marre et décide d'arrêter la partie.
- Nathan gagne par forfait

Dans ce rapport, nous allons tout d'abord décrire pour chaque application et pour chaque opération, les diagrammes d'échange et le format des messages JSON échangés. Enfin, dans une deuxième partie, nous vous expliquerons comment installer et utiliser nos applications (compilation, configuration et exécution), et cela sans utiliser d'IDE.

Diagrammes d'échanges

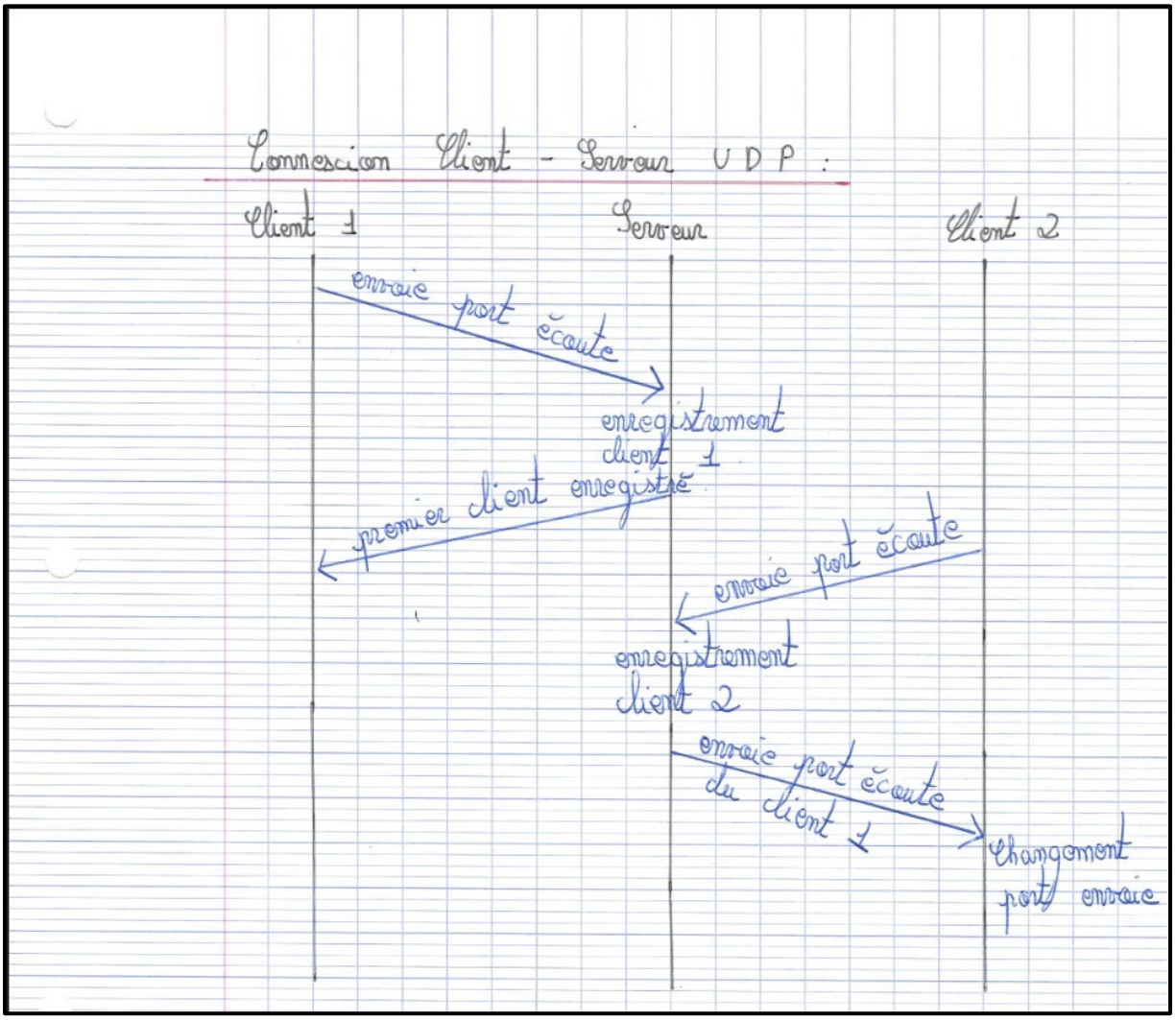
La connexion client – serveur en UDP

Nous allons d'abord vous présenter comment fonctionne la connexion client – serveur en UDP avec l'aide de diagramme d'échanges.

Avant d'entrer dans le vif du sujet, il faut tout d'abord savoir que c'est le serveur qui s'occupe de mettre en relation les différents clients.

Le client s'enregistre auprès du serveur. Si c'est le premier client, le serveur enregistre simplement son adresse IP ainsi que son port. Si c'est le deuxième client, il lui envoie en retour l'adresse du client précédent.

Voici la photographie contenant le diagramme d'échanges client – serveur :



Dans ce diagramme nous avons pris l'exemple de 2 clients qui souhaitent s'enregistrer auprès du serveur.

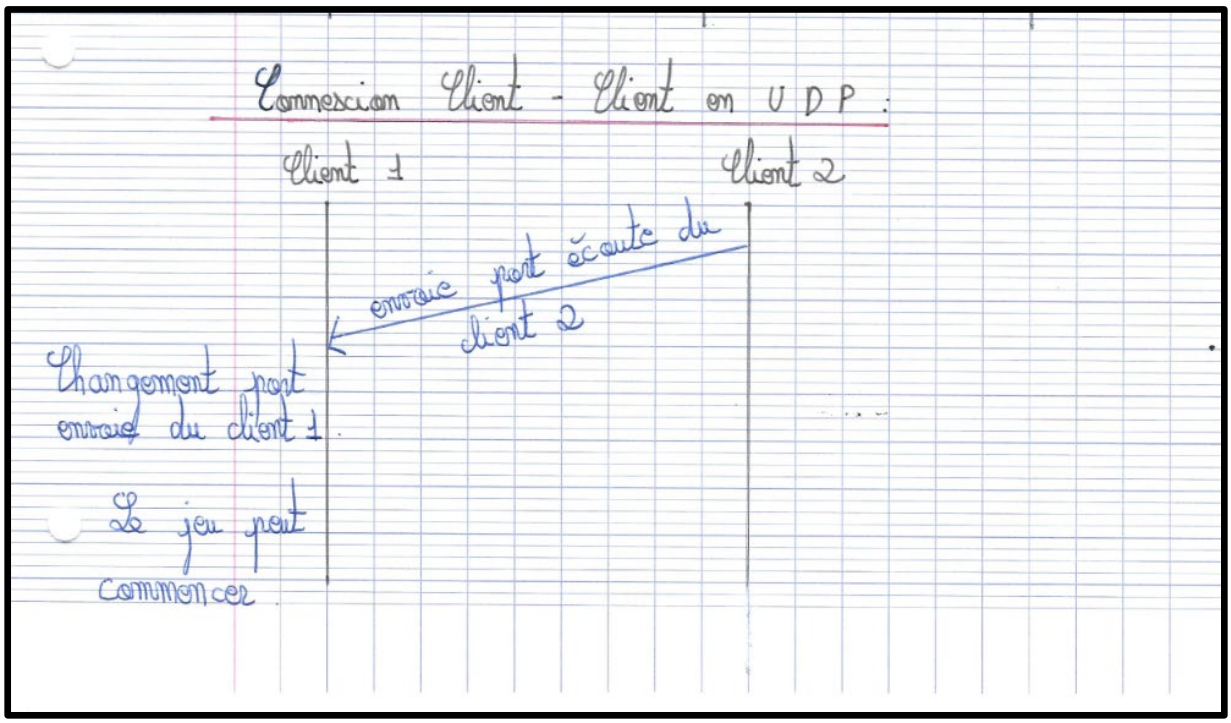
Pour commencer, la connexion client – serveur en UDP

Le client 1 commence par envoyer son adresse IP et son port d'écoute au serveur. Vu qu'il s'agit du premier client, le serveur enregistre les informations de ce dernier. Et lui renvoie le code d'enregistrement "ok".

Ensuite, le client 2 envoie son adresse IP et son port d'écoute au serveur. Le serveur enregistre le client 2. Vu qu'il ne s'agit pas du premier client, le serveur envoie au client 2 l'adresse IP et le port d'écoute du client 1.

La connexion client – client en UDP

Cette connexion fait suite à la connexion client – serveur. Juste après avoir reçu les informations du client 1, le client 2 va lui envoyer son adresse IP et son port, comme le montre le digramme d'échange suivant :



En détail, la connexion client – client en UDP

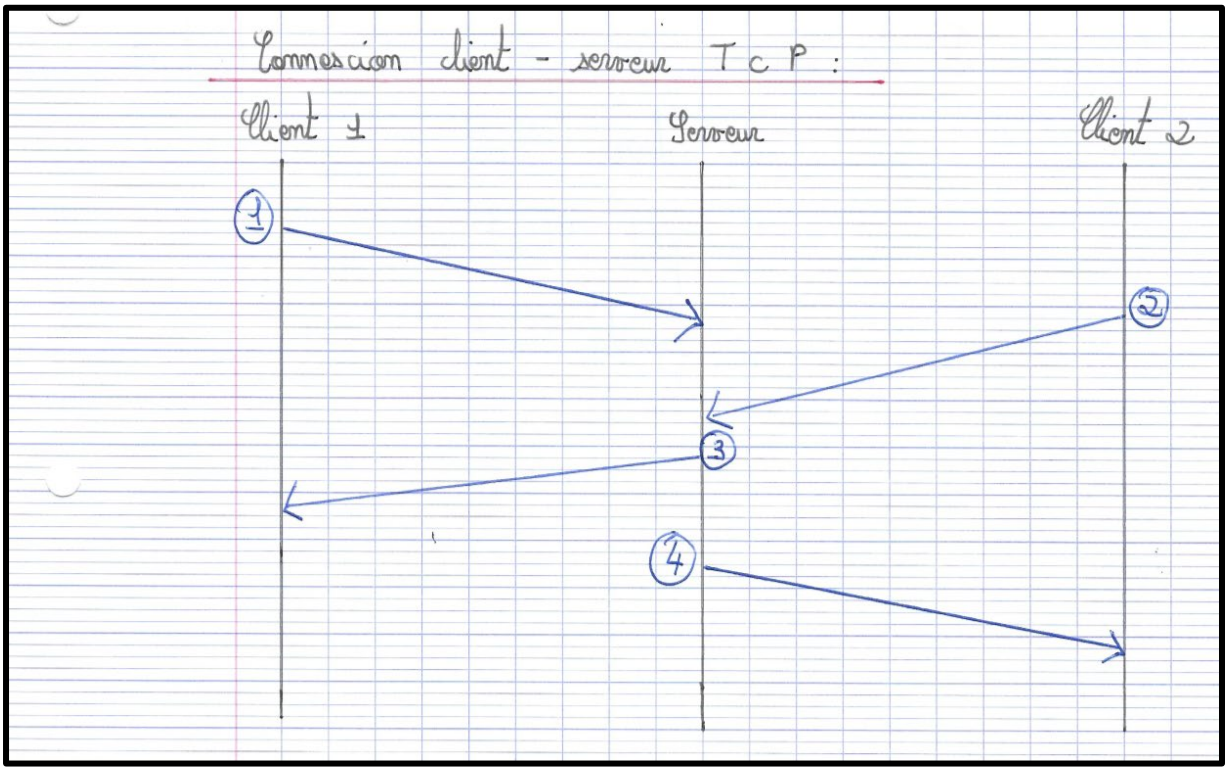
Le client 2, qui a reçu les informations de la part du serveur sur le client 1, va changer son port et son adresse IP d'envoi par ceux du client 1, puis il va envoyer son adresse IP et son port d'écoute au client 1.

Le client 1 va changer son port et son adresse IP d'envoi par ceux du client 2.

Une fois cela fait, le jeu peut commencer.

La connexion client – serveur en TCP

Comme pour le diagramme de connexion client – serveur en UDP, nous avons 2 clients voulant jouer ensemble, ainsi nous obtenons le diagramme d'échange suivant :



En détail, le fonctionnement de la connexion client – serveur en TCP

Étape n° 1 : tout d'abord, le client 1 envoie son pseudo sous format JSON au serveur, celui-ci l'enregistre.

Format JSON : {"pseudo":pseudo du joueur}

Étape n° 2 : le client 2 envoie lui aussi son pseudo, sous le même format JSON, au serveur, qui l'enregistre également.

Étape n° 3 : une fois que le serveur a récupéré les pseudos des 2 clients, il envoie au client 1 les informations du client 2, à savoir : son adresse IP, son port ainsi que son pseudo.

Étape n° 4 : le serveur envoie les données du client 1 au client 2, l'adresse IP, le port et le pseudo du client 1.

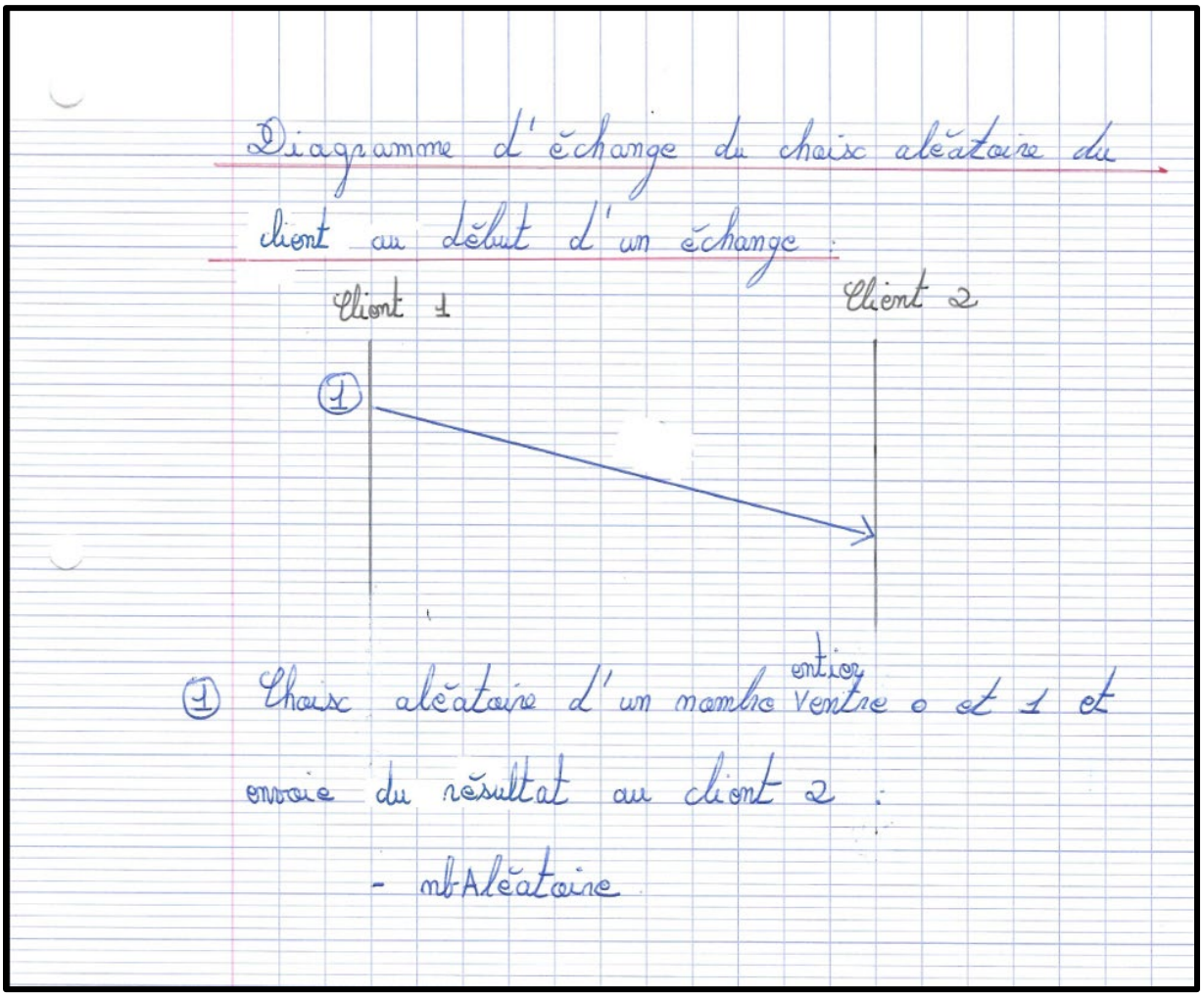
Dans les étapes 3 et 4, les envois de données se font sous format JSON :

{"address":adressClient,"port":portClient,"pseudo":pseudoClient}

Le choix aléatoire du premier joueur servant

Lors du début d'un match, un des deux clients est choisi aléatoirement et c'est celui qui a été choisi qui va servir en premier. Ce choix est fait, par le client 1, après qu'il ait reçu les informations pour se connecter au client 2.

En voici le diagramme d'échange :



Comme vous pouvez le voir, le client 1 va "choisir", aléatoirement, un entier parmi 0 et 1. Puis va l'envoyer au client 2, sous forme de nombre entier.

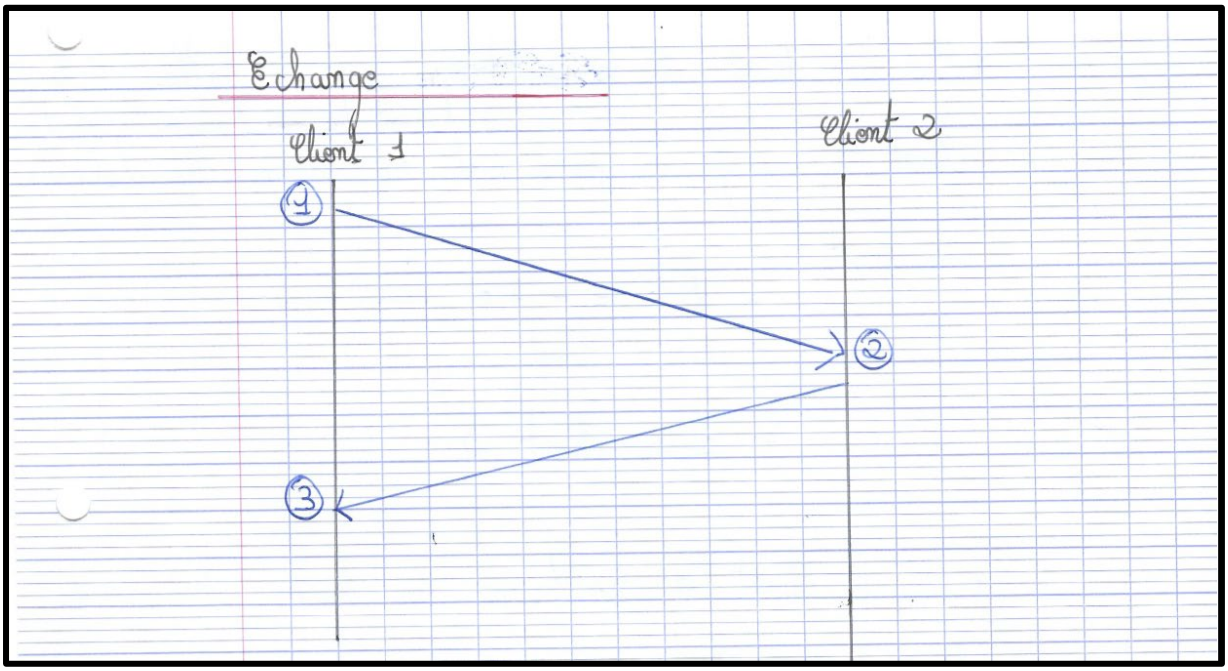
Voici les interprétations du choix aléatoire :

- Si c'est le 0 qui sort, alors c'est le client 1 qui sert en premier.
- Si c'est le 1 qui sort, alors c'est donc le client 2 qui sert en premier.

Les échanges de la partie

Nous allons maintenant vous présenter comment fonctionne le diagramme présentant les échanges qui ont lieu durant la partie.

Voici la photographie du diagramme des échanges :



En détail, le fonctionnement des échanges

Tout d'abord, en tout début de partie, les joueurs 1 et 2 sont positionnés sur la case n° 3 (au milieu), avec les joueurs 1 et 2 correspondants respectivement et virtuellement, aux clients 1 et 2.

Supposons que le joueur 1 soit choisi aléatoirement pour servir en premier.

Étape n° 1 : le joueur 1 commence par choisir son tir. Ensuite, on construit la chaine d'envoi de l'échange, contenant le tir du joueur ainsi que le score des 2 joueurs de la partie, puis on envoie dette chaine au client 2.

Étape n° 2 : le client 2 récupère la chaine envoyée par le client 1 et en effectue le traitement pour mettre les attributs de son objet Échange à jour. Ensuite, le joueur 2 va devoir choisir entre rester à sa position actuelle ou se déplacer d'une case en plus ou moins par rapport à celle sur laquelle il est. On regarde ensuite si le joueur 2 attrape la balle en comparant le tir du joueur 1 avec la position du joueur 2.

Si le joueur 2 ne rattrape pas la balle, le joueur 1 gagne le point (à ce moment le score du joueur 1 augmente de 1). On envoie donc au client 1 les données suivantes : score joueur 1 = 1, score joueur 2 = 0, et tir = -10. Le client 1 récupère et effectue le traitement pour mettre les attributs de son objet Échange à jour. Et peut démarrer un nouvel échange.

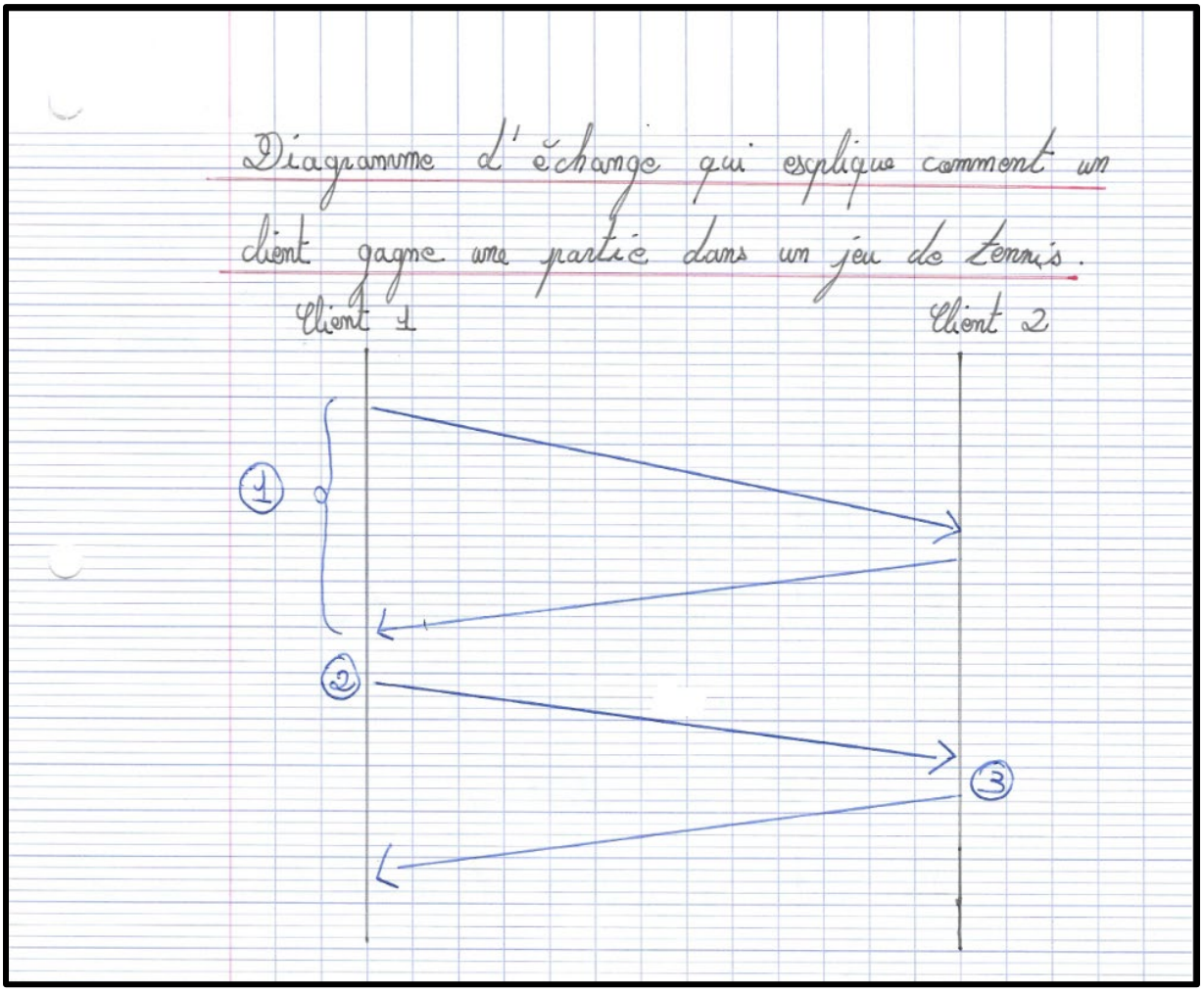
Si le joueur 2 rattrape la balle, il peut choisir où positionner son tir, on construit une chaine d'envoi contenant les scores des 2 joueurs, ainsi que le tir du joueur 2, et le client 2 l'envoie au client 1 qui en fait le traitement.

Étape n° 3 : on fait la même chose que l'étape 2, mais cette fois-ci avec le client et le joueur 1.

La partie est finie ?

Rappelons qu'un joueur gagne une partie si son score est égal à 5 ou plus, avec 2 points d'écarts avec son adversaire.

Voici le diagramme représentant comment est déclaré un gagnant d'une partie :



En détail, le fonctionnement de ce diagramme

Étape n° 1 : la partie est en cours entre le client 1 et 2, la partie fait rage entre ces deux joueurs (nous avons déjà expliqué cette partie précédemment). On vérifie bien entendu, à chaque échange, le score des deux joueurs.

Si le joueur 1 obtient un score égal à 5 avec 2 points d'écarts, on passe à l'étape 2. À l'inverse si c'est le client 2 qui obtient un score égal à 5 avec 2 points d'écarts, on passe à l'étape 3.

Étape n° 2 : le joueur 1 a obtenu un score égal à 5 points ou plus, avec 2 points d'écarts avec son adversaire. Nous informons donc le client 2 en lui envoyant le score des joueurs 1 et 2. Une fois que le client 2 aura reçu les données, la partie s'achève et nous aurons l'affichage du score final avec le nom du gagnant.

Étape n° 3 : le joueur 2 a obtenu un score égal à 5 points ou plus, avec 2 points d'écarts avec son adversaire. Nous informons donc le client 1 en lui envoyant le score des 2 joueurs. Une fois que le client 1 aura reçu les données, la partie s'achève et nous aurons l'affichage du score final avec le nom du gagnant.

Format des fichiers JSON

Dans ce projet, nous n'envoyons qu'une seule donnée par messages sous format JSON, ce sont les informations concernant un échange du jeu de tennis.

Voici, ci-dessous, comment est construit le message que nous envoyons lors d'un échange du jeu :

```
public void construireChaineEnvoie(int entityJoueur, int tirJoueur){
    if(entityJoueur == 1 || entityJoueur == 2){
        this.echangeString = "{";
        this.echangeString += "scoreClient1:" + this.scoreClient1;
        this.echangeString += "scoreClient2:" + this.scoreClient2;
        this.echangeString += "tirClient:" + tirJoueur;
        this.echangeString += "}";
    }
    else{
        System.out.println("[ERREUR FCT: CONSTRUIRE CHAINE ENVOIE] Joueur non trouvé, l'entityJoueur doit être égal à 1 ou 2.");
    }
}
```

Voici comment est construit l'envoie de l'adresse IP, du port et du pseudo aux Clients, dans TCP :

```
"{\"address\" :" + socketClient1.getInetAddress().getHostAddress() + ",\"port\" :" + socke
tClient1.getPort() + ",\"pseudo\" :" + pseudoClient1 + "}";
```

Et enfin comment est construit l'envoie du pseudo au serveur, dans TCP :

```
"{\"pseudo\" :" + Client.nomJoueur1 + "}";
```

Compilation, configuration et exécution du jeu de Tennis

Tout d'abord, il faut commencer par dézipper le fichier contenant le répertoire de chaque application : UDP ou TCP. Ensuite, il suffit d'ouvrir un terminal de commande et de se rendre dans le répertoire choisi.

Pour jouer, vous aurez besoin de 3 terminaux de commandes. Les dossiers de codes ont été disposés pour simuler l'utilisation de chacune des parties sur 3 machines différentes. Il faudra donc :

- 1 terminal pour le serveur.
- 1 second pour le client 1
- 1 troisième pour le client 2

Chaque dossier de code est composé de la manière suivante, et est accompagné de leurs documentations javadoc, dans le sous-dossier "Doc" :

- Pour le client : Echange.java, config_client.json, et Client.json
- Pour le serveur (en UDP) : config_serv.json, Serveur.java
- Pour le serveur (en TCP) : config_serv.json, Serveur.java, Thread_serv.java

Il est très important de ne pas vouloir mélanger les codes, les codes dans le dossier UDP sont pour ceux utilisant UDP et cela fonctionne également avec TCP, les fichiers sont faits que pour être exécuté avec certains autres fichiers.

Compilation

Pour la compilation rien de plus simple, dans n'importe quel dossier de code, il suffit de faire "javac *.java". Ce qui va avoir pour effet de compiler tout le code source.

Configuration

Pour la configuration, chaque dossier de code comporte 1 fichier de configuration, cependant, il y en a 2 formes :

La forme pour les clients

```
{
    "addressServeur":"127.0.0.1",
    "portServeur":"9876",
    "addressClient":"127.0.0.2",
    "portClient":"6930"
}
```

Il est important de constater que dans le fichier de config du client, il y a les données du serveur, c'est ce qui permet de les connecter ensemble. Si vous changez la configuration du serveur, il faudra également changer la configuration de "addressServeur" et de "portServeur".

La forme pour les serveurs

```
{
    "addressServeur":"127.0.0.1",
    "portServeur":"9876"
}
```

L'exécution

Elle aussi est très simple, il suffit de faire "java **x**" avec **"x"** équivalant à Client, si nous voulons lancer un client, à Serveur, si nous voulons lancer un serveur.

Les difficultés rencontrées

Dans ce projet nous avons rencontré une difficulté majeure que nous n'avons su effacer. Nous sommes restés bloquer pendant des heures sur ce bug, cette non-fonction, nous ne savons pas comment la qualifier, qui est celui de connecter 2 clients ensemble en mode TCP. Même en ayant reçu des données de connexions de la part du serveur, nous n'avons pu les connecter ensemble. Nous avons quand même construit le code l'échange qui devait avoir lieu, mais sans pouvoir le tester. Nous avons eu l'idée de faire passer les échanges via le serveur, mais cela voulait dire ne pas respecter le sujet du projet, alors nous en sommes restés sur ce bug ce qui empêche de jouer au tennis en mode TCP. Cependant, dans TCP, nous avons mis les mêmes relations d'échanges entre les clients que ceux que nous avions mis en place pour UDP.

À part cette difficulté majeure, le projet était sympathique, et plaisant à développer.