Réseaux I
projet semestre 5

MADANI Abdenour
TRIOLET Hugo

Licence 3
2021 - 2022

Table des matières

1 Introduction

L'objectif de ce projet est de réaliser un jeu de dames. On pourra consulter les règles du jeu à l'adresse suivante: `https://fr.wikipedia.org/wiki/Dames`. Le projet comportera un client et un serveur.

Le l angage de programmation utilisé est le **C**.

2 Protocole utilisé

Le protocole utilisé est le protocole `TCP`, qui permet de recevoir des accusés de réception de l abonne transmission du message. Effectivement, pour du jeu en ligne, le protocole `UDP` est normalement plus adapté du fait de sa simplicité de transmission et surtout qui permet de garder la connexion et la communication active quand il y a des paquets perdus. Cependant, dans notre cadre d'utilisation, le `TCP` convient bien afin que les coups demandés par les joueurs soient bien réceptionnés et bien pris en compte. De ce fait le joueur saura s'il doit rejouer son coup pour cause d'erreur de transmission.

3 Utilisation

Compilez les deux fichiers .c puis exécutez les chacun dans un terminal différent.

4 Partie Client

Côté client, on commence par initialiser la structure de socket d'adresse :

```
struct sockaddr_in adresse;
int sockfd;
char buffer[BUFFER_SIZE];
char *adresse_serveur = "127.0.0.1";
```

Le client sqe connectera au serveur qui possède l'adresse "127.0.01" (ce qui veut dire qu'on jouera en local sur la machine du fait que cette adresse est l'adresse par défaut du périphérique réseau d'une machine).
Ensuite, on initialise le socket :

```
// Creation de la socket
sockfd = socket(AF_INET, SOCK_STREAM, 0);

if (sockfd < 0) {
        printf("Erreur durant la creation de socket\n");
        exit(1);
} else {
printf("La socket a bien ete creee\n");
```

```
}
```

Pour que la connexion avec les erveur soit au minimum fructueuse, il faut que le serveur soit déjà actif.
de ce fait ensuite, on tente une connexion avec le serveur et si celle-ci échoue, le programme renvoie un message d'erreur. Les messages que le client envoie au serveur sont la première implémentation de l'envoie d'un coup à jouer. On tapera les messages dans le terminal et ceux-ci seront stockés dans un buffer (représentant une chaine de caractères). Puis sera vérifié à chaque envoie d'un message au serveur si celui-ci à bien reçue l'intégralité du message :

```
memset(&adresse, 0, sizeof(adresse));

adresse.sin_family = AF_INET;
adresse.sin_addr.s_addr = inet_addr(adresse_serveur);
adresse.sin_port = PORT;

if (connect(sockfd, (struct sockaddr *) &adresse, sizeof(adresse)) < 0)
        printf("Erreur durant la connexion au serveur !\n");
        exit(1);
}  else {
printf("Connect  au serveur\n");
}

memset(buffer, 0, BUFFER_SIZE);
printf("Entrez votre message : ");

while (fgets(buffer, BUFFER_SIZE, stdin) != NULL) {
        if (sendto(sockfd, buffer, BUFFER_SIZE, 0, (struct sockaddr *) &
                printf("Erreur durant l'envoi des donn es\n\t-%s", buff
        }
        if (recvfrom(sockfd, buffer, BUFFER_SIZE, 0, NULL, NULL) < 0) {
                printf("Erreur durant la r ception des donn es\n");
        } else {
                printf("Bien re u : ");
                fputs(buffer, stdout);
                printf("\n");
        }
}
```

5 Partie Serveur

Dans la partie serveur, bien que cela soit infructueux, on gérera le plateau de jeu via une matrice de taille 10*10 (ceci avec une fonction d'initialisation du plateau de jeu et une fonction d'affichage). De manière similaire, la partie

serveur commencera par créer un socket, vérifier sa validité et ensuite attendre des connexions. Une fois faite (limité à deux clients donc 2 joueurs, les deux joueurs d'une partie classique), on réceptionnera le message et on vérifiera sa conformité avec le message initial. Puis on renvoie le message à l'envoyeur pour lui communiqué ce que l'on a reçu.

```
// Cr ation du plateau de jeu
        // '.' signifie case vide et 'Px' (ou 'D' si c'est une dame) signifie ca
        char ** plateau = creation_plateau();
        afficher_plateau(plateau);

        struct sockaddr_in adresse, adresse_client;
        socklen_t len;
        int sockfd, newsockfd;
        char buffer[BUFFER_SIZE];


        // Cr ation de la socket
        sockfd = socket(AF_INET, SOCK_STREAM, 0);

    if (sockfd < 0) {
        printf("Erreur durant la cr ation de socket\n");
        exit(1);
    } else {
        printf("La socket a bien   t   cr   e\n");
    }


        // Cr ation de l'adresse du serveur
        memset(&adresse, 0, sizeof(adresse));

        adresse.sin_family = AF_INET;
        adresse.sin_addr.s_addr = INADDR_ANY;
        adresse.sin_port = PORT;

        // Connexion/Liaison
        if (bind(sockfd, (struct sockaddr *) &adresse, sizeof(adresse)) < 0) {
                printf("Erreur pendant la liaison\n");
                exit(1);
        }
        else {
        printf("Liaison effectu e avec succ s\n");
        }

        // Mise en place de l' coute  du serveur
        printf("En attente d'une connexion.\n");
```

```
// Deux joueurs maximum
    listen(sockfd, 2);

    // Tant que le serveur est en  coute :
    while (1) {
            len = sizeof(adresse_client);

            //On accepte la connexion d'un joueur
            newsockfd = accept(sockfd, (struct sockaddr *) &adresse_client, &

            if (newsockfd < 0) {
                    printf("Erreur durant l'acceptation de la connexion\n");
                    exit(1);
            } else {
        printf("Connexion accept e avec succ s\n");
            }

            while (1) {
                    memset(buffer, 0, BUFFER_SIZE);

                    // Le serveur re oit un message d'un client
                    if(recvfrom(newsockfd, buffer, BUFFER_SIZE, 0, (struct s
                            printf("Erreur durant la r ception des donn es
                            exit(1);
                    } else {
            printf("Donn es re ues: %s\n", buffer);
                    }


                    // et il renvoie le message    l'envoyeur
                    if (sendto(newsockfd, buffer, BUFFER_SIZE, 0, (struct so
                            printf("Erreur dans l'envoi des donn es\n");
                            exit(1);
                    } else {
            printf("Donn es envoy es : %s\n", buffer);
                    }

                    // traitement du coup demand


            }
    }
```