PROGRAMMATION SYSTÈME

PROJET SH13

Enseignant : **François PECHEUX**

Unité : **Programmation Système**

Apprentis : **BARKOUDEH Julian & DOUZET Camille & ESSAID Oumaima**

Année académique : **2020/2021**

Formation : **Electronique et Informatique parcours Informatique Industrielle**

Table des matières

1. Introduction

Dans le cadre du cours “ Programmation Système”, nous avons développé un projet visant à mettre en pratique les connaissances acquises au cours de ce semestre. Le but de ce projet est de programmer une application réseau “un jeu de cartes” en langage C en utilisant la programmation graphique UML et le protocole TCP ainsi que des threads.

2. Principe du jeu sh13

Nous avons treize personnages issus de l’univers de Sir Arthur Conan Doyle, dont un sera le criminel. Chaque personnage est identifié par son illustration et une série d’icônes. Sherlock, par exemple, c'est une Pipe, une Ampoule et un Poing.

Le principe est de tenter de déduire les personnages que chaque joueur a en main pour en arriver à connaître le coupable. Pour cela, à son tour de jeu, chaque joueur peut :

- choisir un symbole et demander qui en possède
- cibler un joueur et lui demander combien il possède d'un symbole en particulier
- porter une accusation.

Une des spécificités de notre jeu est que si le joueur se trompe lors de l’accusation, il est éliminé. Chaque joueur n’a donc qu’une chance pour trouver le coupable !

3. Objectifs du projet

Le but est de compléter les lignes de codes qui manquaient afin de créer une partie de jeu en réseau local du jeu Sherlock13 avec 4 joueurs.

Pour ce faire, on a besoin de comprendre le principe du jeu et de vérifier les codes fournis dans les deux fichiers sh13.c et server.c.

Afin de construire le jeu de cartes, il faut envoyer au joueur un paquet d'informations telles que : son Id, le nom des autres joueurs et ses cartes. Ces 3 messages contiennent des informations différentes. Ils sont toutefois tous envoyés sous la forme d'une chaîne de caractère. C'est la première lettre de cette chaîne qui détermine quel type de message le serveur vient d'envoyer. Ainsi pour décoder, un switch case a été mis en place.

4. Protocole TCP/UDP

Pour réaliser ce projet, nous avons d'abord effectué des recherches sur le protocoles TCP et UDP en nous basant sur les cours fournis dans le site et aussi sur diverses recherches réalisées sur internet à propos des deux protocoles.

Le protocole de communication UDP (User Datagram Protocol) est utilisé pour transmettre des données très rapidement, ou en petites quantités en accordant peu d'importance à la perte d'une partie de ces données.Cela réduit le coût en ressources et de plus, il existe des méthodes de substitution des données manquantes. La transmission s'effectue de manière simple entre deux entités définies par une adresse IP et un numéro de port. De plus aucune communication préalable n'est requise pour connexion, il s'agit du mode de transmission sans connexion. Ce protocole est donc employé pour sa rapidité de transmission et peut également transmettre à plusieurs récepteurs en simultanés.

Le protocole TCP (Transmission Control Protocol) est utilisé pour un transport fiable des données et en mode connecté, c'est-à-dire qu'une communication au préalable est nécessaire pour établir une connexion. L'échange de données se fait en trois temps, une première communication puis une réponse qui reconnaît la connexion et enfin la transmission des données.

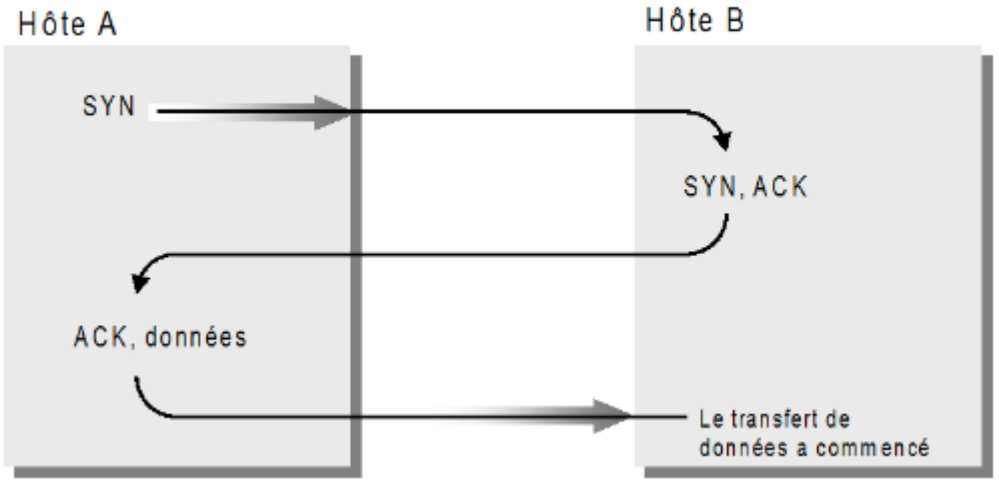


Fig.1 – Modèle d'échange de donnée TCP

a. Gestion des fonctions TCP

Nous avons dans un premier temps repris un code source fourni dans l'énoncé pour établir un serveur et un client qui communiquent en utilisant les sockets TCP.

Une socket est un point d'accès aux couches réseau TCP/UDP liée localement à un port et qui permet la communication avec un port distant sur une machine distante.

Pour ce faire nous devons ouvrir deux terminaux et exécuter le code serveur qui va attendre qu'une connexion s'effectue. Ensuite, nous lançons le code sh13.c c à partir des 4 terminaux. Nous verrons alors que la communication est effectuée.

5. Description protocole communication server-client

a. Côté client

Le client envoie différents types de demandes au serveur tel que la demande à qui appartient un objet et l'occurrence autrement dit le nombre de fois où le participant a eu une telle carte. Dans le cas où le joueur gagne une partie, le serveur broadcaste son nom aux autres joueurs.

On peut décrire le processus en détails comme indiqué ci-dessous:

1. On déclare et on initialise les variables graphiques telles que : "winner", "game over", "fail", "connect"...
2. Dans la boucle graphique, on démarre le jeu en appuyant sur le bouton "connect" pour la variable connectEnabled==1.
3. Une fois le bouton est appuyé, on crée le message d'envoi du client au serveur contenant les informations suivantes : l'adresse IP du serveur, le numéro de port du serveur.
4. On sélectionne le joueur et on garde les coordonnées graphiques des joueurs.
5. On sélectionne un objet puis on récupère ses coordonnées pour savoir s'il a été sélectionné
6. On sélectionne un coupable puis on récupère ses coordonnées pour savoir s'il a été sélectionné
7. On commence le traitement fait à travers les appuies de la souris, on appuie sur go pour jouer : soit une accusation un des joueurs est coupable, soit on demande un objet à un autre joueur, soit on demande d'objet à un joueur spécifique à travers les trois variables : joueurSel, objetSel et guilSel.
8. On met Synchro à 1 pour recevoir un message du serveur sur le thread créé et on reçoit l'Id, si la chaîne de caractère commence par 'I', s'il s'agit d'un 'L' le joueur reçoit la liste des joueurs, si c'est un 'D' le joueur reçoit ses trois cartes, s'il s'agit d'un message 'M' le joueur reçoit le numéro du participant courant, le joueur reçoit une valeur de table carte si le message est un 'V', un des joueurs est gagné s'il s'agit d'un 'W' et d'un 'E'v si le joueur a échoué.
9. On finit le traitement avec la variable synchro à 0.

b. Côté Server

Voici une description du déroulé de la fonction principale avec les étapes principales pour gérer le serveur :

1. On commence par créer les sockets TCP pour établir la connexion
2. On déclare ensuite quel est le joueur courant et le premier joueur
3. On entre dans une boucle d'initialisation du tableau des structures des clients clients contenants toutes ses coordonnées
4. Puis, on entre dans une boucle infinie avec deux cas possibles :
    - Si la variable fsmServer vaut 0 : Phase de connexion des joueurs :

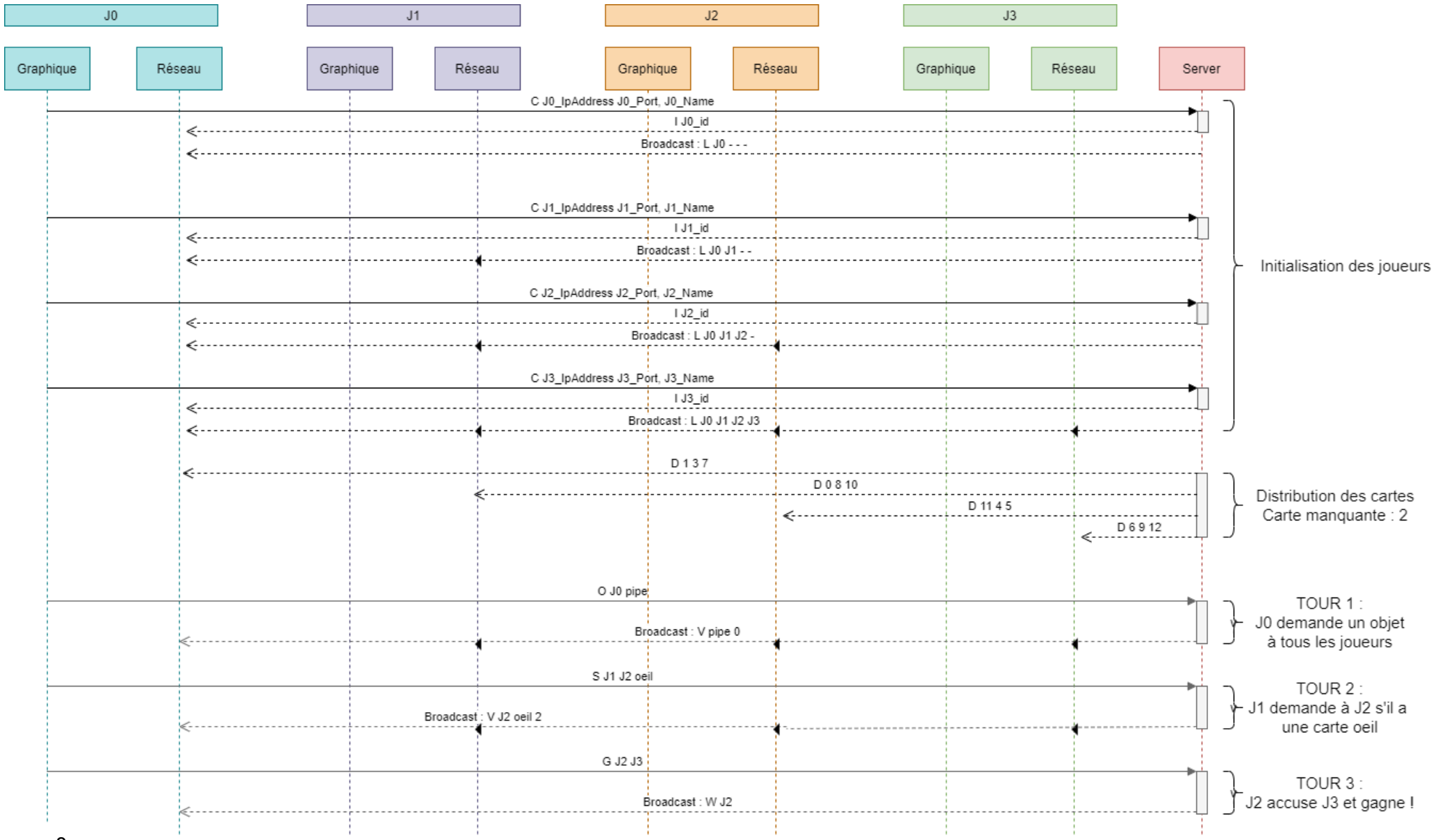
      On entre dans le case C : On envoie les informations du client qui se connecte (id etc)

      Mais lorsque le nombre de clients est égale à 4, on passe à la distribution des cartes avec leurs valeurs. Finalement, on informe les joueurs de celui qui va commencer la partie

    - La variable fsmServer vaut 1 : Phase d'échange avec le client : 3 actions possibles du jeu :
        - case O : choisir un symbole et demander qui en possède
        - case S : cibler un joueur et lui demander combien il possède d'un symbole en particulier
        - case G : porter une accusation.
5. On passe au tour suivant en regardant si la personne suivante n'a pas fait de fausses accusations plus tôt dans la partie.
6. Si tous les joueurs ont fait des fausses accusations, le jeu se termine
7. Lorsqu'une partie est finie, le jeu et les clients sont réinitialisés

c. Diagramme séquentiel UML

Voici le déroulement d'une partie :


J0
J1
J2
J3
Graphique
Réseau
Graphique
Réseau
Graphique
Réseau
Graphique
Réseau
Server
C J0_IpAddress J0_Port, J0_Name
I J0_id
Broadcast : L J0 - - -
C J1_IpAddress J1_Port, J1_Name
I J1_id
Broadcast : L J0 J1 - -
C J2_IpAddress J2_Port, J2_Name
I J2_id
Broadcast : L J0 J1 J2 -
C J3_IpAddress J3_Port, J3_Name
I J3_id
Broadcast : L J0 J1 J2 J3
Initialisation des joueurs
D 1 3 7
D 0 8 10
D 11 4 5
D 6 9 12
Distribution des cartes
Carte manquante : 2
O J0 pipe
Broadcast : V pipe 0
TOUR 1 :
J0 demande un objet
à tous les joueurs
S J1 J2 oeil
Broadcast : V J2 oeil 2
TOUR 2 :
J1 demande à J2 s'il a
une carte oeil
G J2 J3
Broadcast : W J2
TOUR 3 :
J2 accuse J3 et gagne !

6. Conclusion

En conclusion, ce projet fut très intéressant au niveau technique. En effet le projet étant très complet, nous avons pu mettre en pratique nos connaissances des cours de Programmation Système, Réseaux et aussi améliorer notre niveau de programmation en C. Le projet étant “pré-rempli” a permis de nous focaliser sur le développement des parties plus critiques. Ainsi, nous avons pu développer une application réseau complète en peu de temps tout en apprenant des nouvelles notions comme les threads, les appels systèmes et la gestion d’une interface graphique.