



**POLYTECH<sup>®</sup>**  
SORBONNE



**SORBONNE**  
**UNIVERSITÉ**

POLYTECH SORBONNE UNIVERSITÉ

OS : USER

IMPLEMENTATION DU JEU ONLINE

RAPPORT

---

## Sherlock13 sur réseau en C

---

*Élève(s) :*

Daniel FERREIRA LARA

*Enseignant(s) :*

François PECHEUX

21 avril 2025

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Lancement du Projet</b>	<b>2</b>
2.1	Compilation . . . . .	2
2.2	Lancement . . . . .	2
<b>3</b>	<b>Le jeu et L'interface</b>	<b>3</b>
3.1	Overview . . . . .	3
3.2	Connection . . . . .	4
3.3	Gameplay . . . . .	5
3.3.1	Replay . . . . .	10
<b>4</b>	<b>Définition de l'Architecture</b>	<b>11</b>
4.1	Protocol TCP . . . . .	11
4.2	Messagerie . . . . .	12
4.2.1	Client vers Server . . . . .	13
4.2.2	Serveur vers Client . . . . .	13
4.2.3	Communication Sequence . . . . .	13
<b>5</b>	<b>Implementation du Code</b>	<b>15</b>
5.1	Serveur . . . . .	15
5.1.1	Variables Globales . . . . .	15
5.1.2	Fonctions Clés . . . . .	15
5.1.3	Fonction Main . . . . .	15
5.2	Architecture du Client . . . . .	15
5.2.1	Variables Globales . . . . .	15
5.2.2	Fonctions Principales . . . . .	16
5.2.3	Flux de la Fonction Principale . . . . .	16
5.3	Système de Logs . . . . .	16
5.3.1	Serveur . . . . .	17
5.3.2	Client . . . . .	17
<b>6</b>	<b>Conclusion</b>	<b>18</b>

# 1 Introduction

Ce rapport décrit le projet développé pour le cours de OS User dans le cadre du programme EISE4, et est une implémentation du jeu Sherlock 13 en code pour un système Linux. Sherlock 13 est un jeu de déduction pour 2 à 4 joueurs.

Dans ce jeu, 13 personnages sont suspects, mais chaque joueur reçoit 3 cartes représentant des innocents. Par élimination, le personnage qui n'est dans la main d'aucun joueur est le coupable. À tour de rôle, les joueurs posent des questions sur les symboles associés aux personnages pour recueillir des indices. Le premier à identifier correctement le coupable remporte la partie, mais une accusation erronée élimine le joueur.

L'objectif de ce projet était de recréer ce jeu en tant qu'application en réseau pour 4 joueurs avec une interface graphique en C. La mise en œuvre a nécessité le développement de composants serveur et client, ainsi qu'une interface graphique pour le client.

Ce rapport explique comment lancer le serveur et les clients, comment utiliser l'interface graphique du client, et détaille l'implémentation technique en se concentrant sur la communication entre le serveur et les clients.

Ce développement a été documenté et est disponible sur github, à partir du lien ci-dessous :

## 1. Repo GitHub : [Project Repo](#)

Il convient de noter que le code développé est basé sur les références suivantes : [4], [1], [2], [3]. Pour corriger les bogues et les erreurs de compilation, j'ai utilisé des forums pour les résoudre.

# 2 Lancement du Projet

## 2.1 Compilation

Le projet consiste en deux fichiers de code source : `server_v1.c` et `sh13_v1.c`, correspondant aux composants serveur et client du jeu. Pour compiler ces fichiers, il faut exécuter le script `cmd.sh` à partir du répertoire du projet :

```
1 ./cmd.sh
2 # ou
3 make # Executer le Makefile
```

## 2.2 Lancement

En conséquence, nous avons le dossier de construction avec les fichiers exécutables, de sorte que nous pouvons exécuter chacun d'eux dans un terminal, le serveur et les 4 clients. Nous pourrions également tester entre les ordinateurs connectés au réseau de la même manière, en remplaçant localhost par l'IP correcte.

```
1 ./server <Port_server> # ou make run_server <Port_server>
2
3 # Run clients
4 ./client <IP_address_server> <Port_server> <IP_address_clientA> <
  Port_clientA> <NameA>
```

```
5 ./client <IP_address_server> <Port_server> <IP_address_clientB> <  
   Port_clientB> <NameB>  
6 ./client <IP_address_server> <Port_server> <IP_address_clientC> <  
   Port_clientC> <NameC>  
7 ./client <IP_address_server> <Port_server> <IP_address_clientD> <  
   Port_clientD> <NameD>
```

## 3 Le jeu et L'interface

### 3.1 Overview

L'interface du jeu est implémentée en utilisant la bibliothèque graphique SDL2 pour le langage C. Elle se compose de plusieurs éléments clés :

- **Table des symboles** (en haut) : Une table résumant les symboles possédés par chaque joueur, qui se remplit au fur et à mesure de la partie.
- **Bouton Go** (en haut à gauche) : Utilisé pour se connecter la première fois.
- **Bouton Replay** (en haut à gauche) : Utilisé pour demander une nouvelle partie.
- **Cartes de personnages possédées** (à droite) : Ces personnages ne peuvent pas être le coupable.
- **Table des suspects** (à gauche) : Une liste de tous les suspects avec leurs symboles. Vous pouvez ajouter une croix sur les personnages que vous pensez innocents en cliquant sur la case correspondante.
- **Bannière d'information** (en bas) : Fournit des informations sur la partie.
- **Bouton Jouer** (en bas à droite) : Utilisé pour jouer lorsque c'est votre tour.

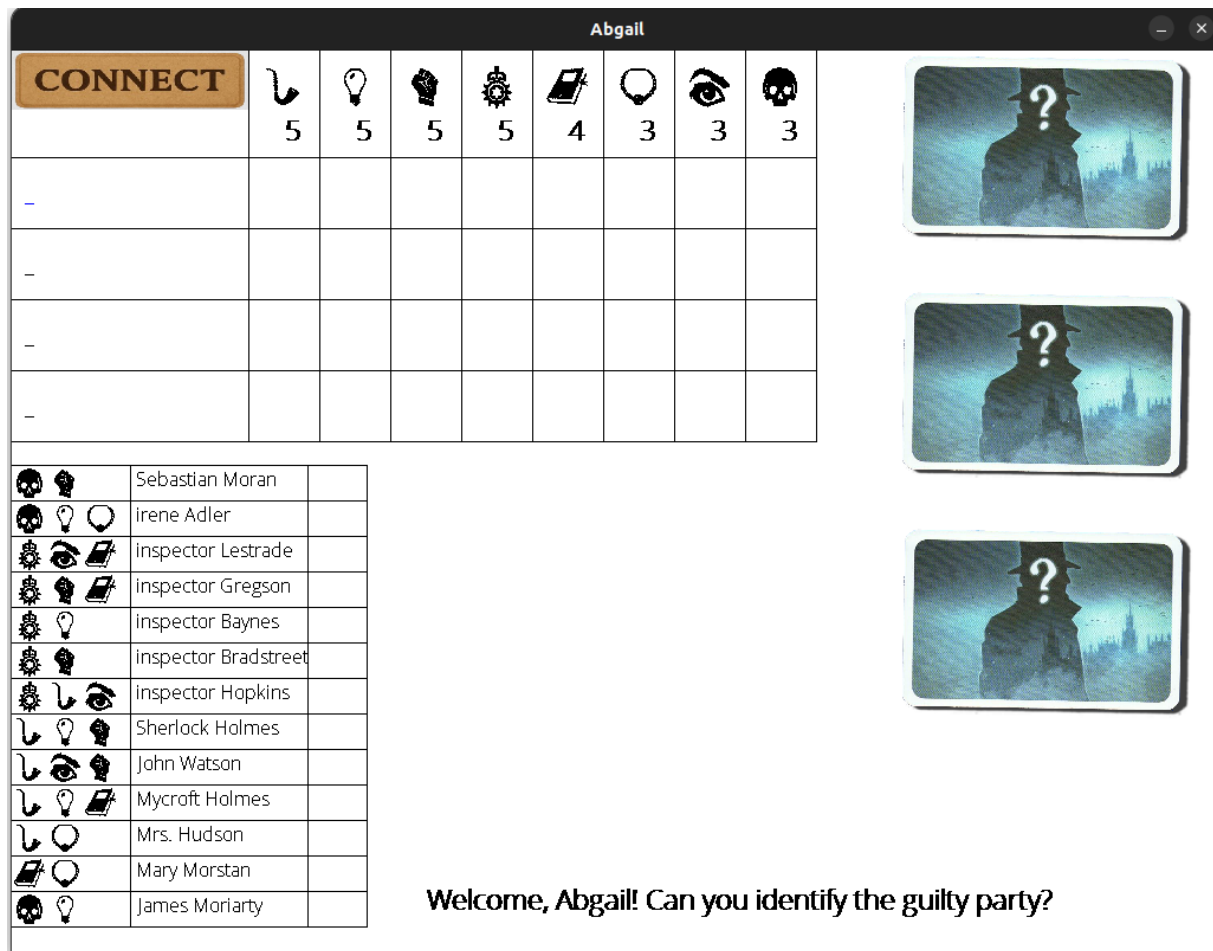


FIGURE 1 – Écran Initiale

### 3.2 Connection

Pour vous connecter au serveur, cliquez sur le bouton 'Connecter' en haut à gauche. Une fois la connexion établie, vous verrez apparaître le message "Welcome, Player! Waiting..." dans la bannière d'information et les noms des joueurs actuellement connectés dans le tableau des symboles. Le jeu commence lorsque 4 joueurs sont connectés.

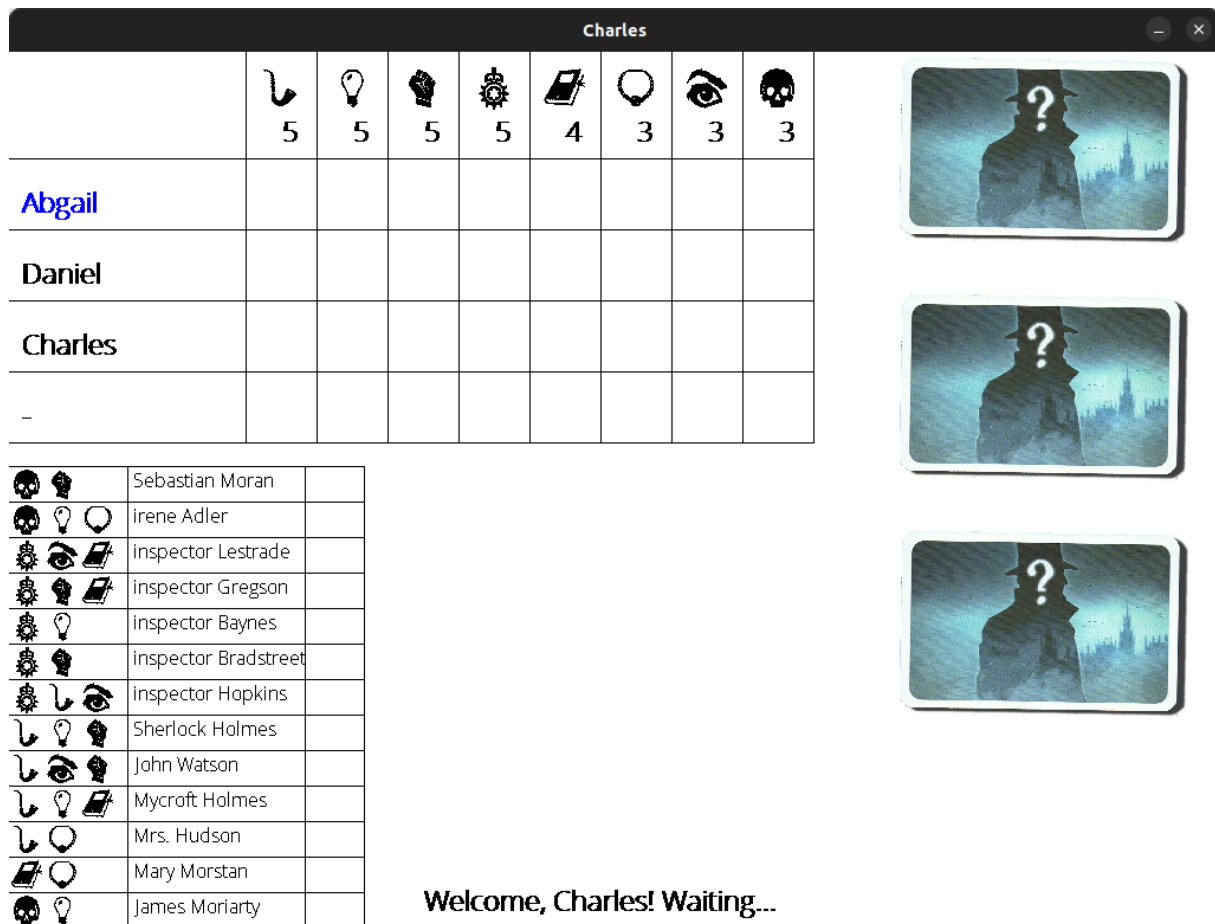













FIGURE 2 – Waiting other players














### 3.3 Gameplay


Lorsque le jeu commence, vous verrez vos 3 cartes de personnage sur la droite. La ligne du tableau des symboles correspondant à votre nom est remplie avec les symboles de vos cartes.

Le premier joueur (joueur 0) commence la partie. Le nom du joueur actuel est surligné en vert dans le tableau des symboles. Lorsque c'est votre tour, le bouton 'Go' apparaît en bas à droite de l'écran.

Abgail								
								
	5	5	5	5	4	3	3	3
Abgail	1	1	2	0	0	1	1	2
Daniel								
Charles								
François								

	Sebastian Moran	<div style="border: 1px solid red; width: 20px; height: 20px; transform: rotate(45deg);"></div>
	Irene Adler	<div style="border: 1px solid red; width: 20px; height: 20px; transform: rotate(45deg);"></div>
	Inspector Lestrade	
	Inspector Gregson	
	Inspector Baynes	
	Inspector Bradstreet	
	Inspector Hopkins	
	Sherlock Holmes	
	John Watson	<div style="border: 1px solid red; width: 20px; height: 20px; transform: rotate(45deg);"></div>
	Mycroft Holmes	
	Mrs. Hudson	
	Mary Morstan	
	James Moriarty	














You have your cards!














FIGURE 3 – Prêt à jouer

Vous avez trois actions possibles :

1. **Investigation 1** : Sélectionnez un symbole en cliquant sur son image en haut du tableau des symboles et demandez aux autres joueurs s'ils possèdent ce symbole (sans en préciser le nombre) en cliquant sur le bouton Go. Si un joueur possède le symbole, une étoile apparaît dans le tableau dans la case correspondante. Sinon, c'est 0 qui est affiché.

								
	5	5	5	5	4	3	3	3
Abgail			*					
Daniel	2	1	0	1	0	1	1	1
Charles			*					
François			*					

	Sebastian Moran	
	Irene Adler	
	Inspector Lestrade	
	Inspector Gregson	
	Inspector Baynes	
	Inspector Bradstreet	
	Inspector Hopkins	✗
	Sherlock Holmes	
	John Watson	
	Mycroft Holmes	
	Mrs. Hudson	✗
	Mary Morstan	
	James Moriarty	✗

**You have your cards!**

FIGURE 4 – Inverstigation à tous

2. **Investigation 2** : Sélectionner un symbole comme dans l'action précédente et un joueur en cliquant sur son nom à gauche du tableau des symboles, puis valider avec le bouton Go. Le joueur désigné doit indiquer le nombre exact de symboles de ce type qu'il possède. Le nombre exact de symboles est alors enregistré dans le tableau.



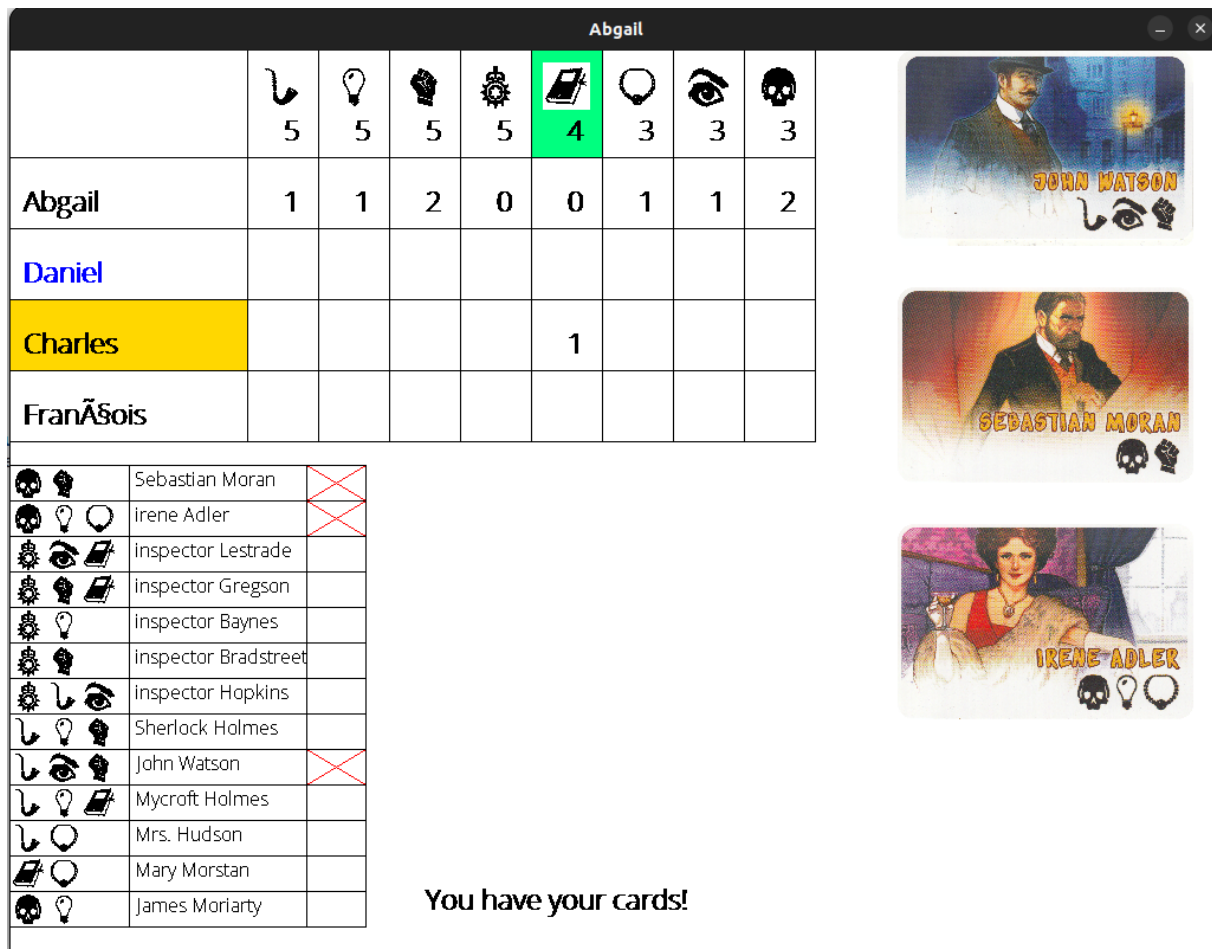


FIGURE 5 – Investigation Solo

3. **Accusation** : Sélectionnez un personnage en cliquant sur son nom dans le tableau des suspects et cliquez sur Go pour l'accuser. Deux scénarios sont possibles :
  - Si vous avez trouvé le bon coupable, vous gagnez et le jeu se termine. Tous les joueurs sont informés du gagnant et du coupable par la bannière d'information. Le nom du coupable apparaît en vert dans le tableau des suspects.



FIGURE 6 – You Win (Victoire)

- Si le personnage désigné est innocent, vous êtes éliminé du jeu. Votre nom apparaît en rouge dans le tableau des symboles et vous passez votre tour pour le reste de la partie. Les autres joueurs sont informés du personnage que vous avez accusé à tort. Une croix correspondant à ce personnage est automatiquement tirée pour tous les joueurs dans le tableau des suspects.

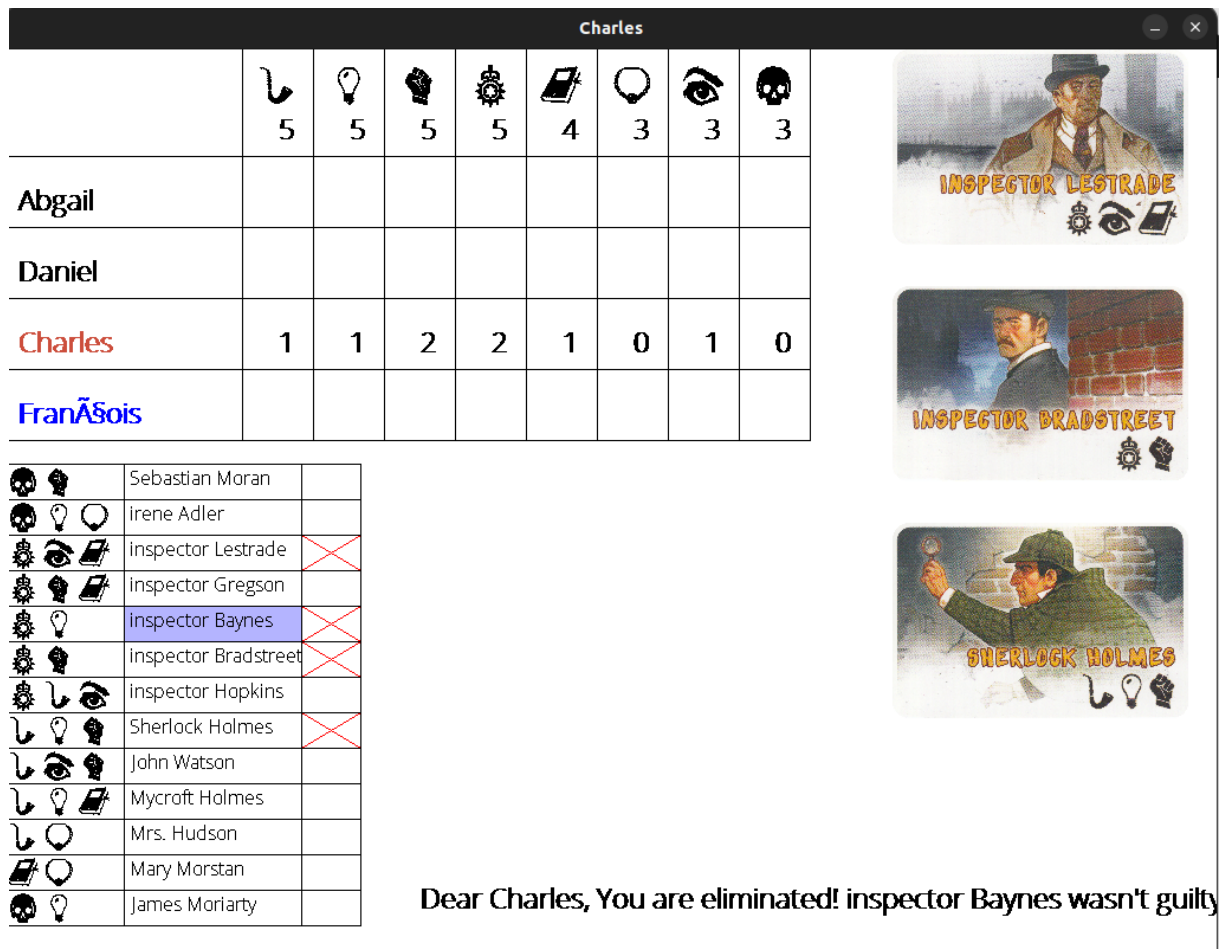


























FIGURE 7 – Looser (Défaite)

### 3.3.1 Replay

À la fin d'une partie, un bouton 'Replay' apparaît à la place du bouton 'Connect'. Il permet aux joueurs de demander une nouvelle partie. Lorsqu'un joueur clique dessus, le bouton disparaît et sa fenêtre se réinitialise. Les autres joueurs sont informés de la demande par le biais de la bannière d'information.

								
	5	5	5	5	4	3	3	3
Abigail								
Daniel								
Charles								
François								

	Sebastian Moran	
	Irene Adler	
	Inspector Lestrade	
	Inspector Gregson	
	Inspector Baynes	
	Inspector Bradstreet	
	Inspector Hopkins	
	Sherlock Holmes	
	John Watson	
	Mycroft Holmes	
	Mrs. Hudson	
	Mary Morstan	
	James Moriarty	

**Player Daniel is ready! (2/4)**

FIGURE 8 – Looser

## 4 Définition de l'Architecture

L'architecture logicielle proposée est une relation client-serveur, avec une relation N-1, c'est-à-dire plusieurs clients pour le même serveur. Les fonctionnalités proposées dans la feuille de route ont été mises en œuvre et, en plus, des mesures de sécurité ont été créées, comme l'envoi d'un message en fonction de l'utilisateur.

### 4.1 Protocol TCP

Le protocole TCP est basé sur une connexion fiable entre le client et le serveur, qui repose sur une "poignée de main" entre les deux. Dans cette optique, il est nécessaire que la connexion soit acceptée avant que les messages ne soient envoyés, comme nous le verrons plus loin. En outre, chaque client dispose de son propre socket pour communiquer avec le serveur. En fait, les sockets sont ouvertes et fermées à chaque nouveau message du serveur en raison de l'utilisation des threads.

L'image ci-dessous illustre les étapes nécessaires, ainsi que les fonctions C existantes que j'ai utilisées.

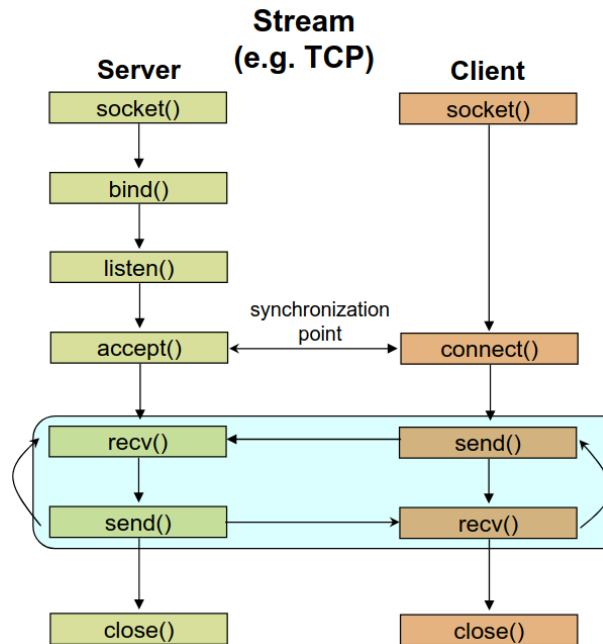


FIGURE 9 – Figure de la page 30 de [4]

#### Côté serveur :

1. Le serveur crée un socket d'écoute avec `socket()`
2. Il associe le socket à une adresse IP et un port via `bind()`
3. Il se met en écoute des connexions avec `listen()`
4. Il accepte les nouvelles connexions avec `accept()` qui crée un nouveau socket dédié à chaque client
5. Il peut alors échanger des données avec `send()` et `recv()`

#### Côté client :

1. Le client crée un socket avec `socket()`
2. Il se connecte au serveur avec `connect()` en spécifiant l'IP et le port
3. Une fois connecté, il peut échanger des données avec `send()` et `recv()`

La fermeture de connexion peut être initiée par l'un ou l'autre avec `close()`.

## 4.2 Messagerie

Le client communique avec le serveur en utilisant un protocole basé sur des messages textuels. Ci-dessous, je décrirai les deux communications possibles, leur signification, le protocole à suivre et nous aurons un diagramme qui résume le chemin des données :

### 4.2.1 Client vers Server

Type	Data	Utilisation
C : Connection	IP, port, name	Se connecter au serveur
R : Replay	ID	Demande de redémarrage
G : Guilt	ID, suspect number	Faire une accusation
O : Others	ID, symbol number	Action d'enquête 1
S : Solo	ID, player, symbol number	Action d'enquête 2

TABLE 1 – Messages depuis le client

### 4.2.2 Serveur vers Client

Type	Data	Information envoyée
I : ID	ID	Identifiant du joueur
L : List	Player names (4)	Liste des noms des joueurs
D : Deck	Suspect numbers (3)	Les 3 cartes d'un joueur
M	ID	Numéro du joueur actuel
V : Value	ID, symbol number, value	Valeur du symbole dans le tableau
E : Eliminated	ID, suspect number	ID éliminé et suspect innocent
W : Winner	ID, suspect number	ID du gagnant et coupable
R : Replay	ID	Demande de redémarrage

TABLE 2 – Messages depuis le serveur

### 4.2.3 Communication Sequence

La communication entre le serveur et les clients suit des séquences spécifiques pour les différents événements du jeu :

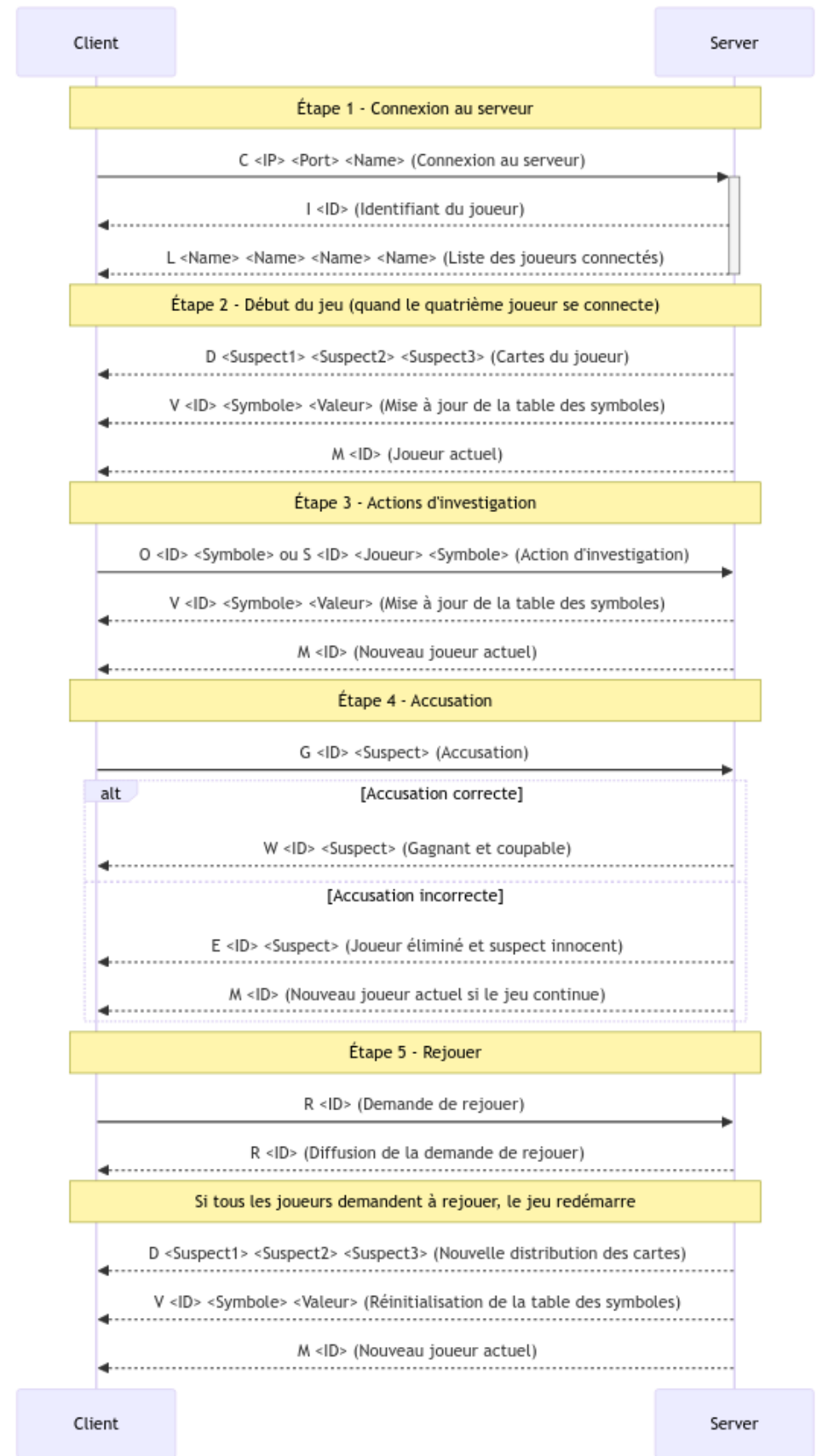


FIGURE 10 – Flou de Communication

## 5 Implementation du Code

### 5.1 Serveur

#### 5.1.1 Variables Globales

Le serveur utilise plusieurs variables globales pour gérer l'état du jeu :

- `nbClients` : Nombre de clients connectés au serveur
- `fsmServer` : Définit l'état du serveur (0 : en attente de connexions, 1 : partie en cours)
- `deck` : Liste des 13 cartes (représentées par des entiers)
- `tableCartes` : Représente la table des symboles (tableau 4×8 où les 4 lignes correspondent aux 4 joueurs et les 8 colonnes aux 8 symboles)
- `nomcartes` : Noms des suspects
- `joueurCourant` : Numéro du joueur actuel
- `eliminated` : Indique les joueurs éliminés
- `nbEliminated` : Nombre de joueurs éliminés
- `nbReplay` : Nombre de joueurs demandant une nouvelle partie

#### 5.1.2 Fonctions Clés

Le serveur implémente plusieurs fonctions importantes :

- `melangerDeck()` : Mélange le paquet en échangeant des éléments de façon aléatoire
- `createTable()` : Initialise la table des symboles en fonction des cartes distribuées aux joueurs
- `sendMessageToClient()` : Envoie un message à un client spécifique via TCP
- `broadcastMessage()` : Envoie un message à tous les clients connectés
- `nextPlayer()` : Retourne le numéro du prochain joueur devant jouer
- `init_game()` : Initialise les données pour une nouvelle partie

#### 5.1.3 Fonction Main

La fonction principale du serveur exécute les étapes suivantes :

1. Initialise les variables et crée un socket TCP
2. Lie le socket au port spécifié et écoute les connexions
3. Initialise une partie (mélange du paquet, création de la table)
4. Entre dans une boucle infinie en attente de connexions des clients
5. Traite les messages entrants des clients selon l'état du serveur

### 5.2 Architecture du Client

#### 5.2.1 Variables Globales

Le client utilise plusieurs variables globales pour gérer l'état du jeu et l'interface utilisateur :

- Informations de connexion : adresses IP et ports du serveur et du client, nom du joueur (`gServerIpAddress`, `gServerPort`, `gClientIpAddress`, `gClientPort`, `gName`)



- Variables d'état du jeu : ID du joueur (`gId`), joueur/objet/suspect sélectionnés (`joueurSel`, `objetSel`, `guiltSel`)
- Variables d'état de l'interface : drapeaux de visibilité des boutons (`goEnabled`, `connectEnabled`, `replayEnabled`)
- Données de jeu : cartes possédées (`b[3]`), table des cartes (`tableCartes`), suppositions de culpabilité (`guiltGuess`)
- État des joueurs : joueurs éliminés (`eliminated`), vainqueur (`gWinner`)

### 5.2.2 Fonctions Principales

Le client implémente plusieurs fonctions importantes :

- `fn_serveur_tcp()` : Gère le thread serveur TCP pour recevoir les messages
- `sendMessageToServer()` : Envoie un message au serveur
- `init_game()` : Initialise une nouvelle partie
- `rst_click()` : Réinitialise les sélections du joueur
- `end_game()` : Gère la fin de la partie et affiche le résultat
- `eliminate()` : Gère l'élimination d'un joueur

### 5.2.3 Flux de la Fonction Principale

La fonction principale du client exécute les étapes suivantes :

1. Analyse les arguments de ligne de commande pour obtenir les informations de connexion
2. Initialise SDL, TTF et crée la fenêtre du jeu
3. Charge les textures et les ressources graphiques (cartes, objets, boutons)
4. Initialise le thread serveur TCP pour recevoir les messages
5. Entre dans la boucle principale d'événements pour :
  - Traiter les entrées utilisateur (clics de souris, mouvements)
  - Gérer les messages reçus du serveur (identifiants, listes de joueurs, cartes, etc.)
  - Mettre à jour l'affichage en fonction de l'état du jeu
6. Affiche l'interface graphique avec :
  - La grille de jeu montrant les objets associés à chaque joueur
  - Les cartes du joueur
  - Les boutons d'action (connexion, jouer, rejouer)
  - Les suspects et leurs objets associés
  - Les messages d'état du jeu

## 5.3 Système de Logs

Pour faciliter le débogage et le suivi du bon fonctionnement du jeu, un système de journalisation a été mis en place côté serveur et côté client. Chaque événement important est enregistré dans un fichier log correspondant, ce qui permet d'analyser les interactions entre les joueurs et le serveur après une partie.

```
[2025-04-21 02:14:35] [DEBUG] |##>M 0
[2025-04-21 02:14:35] [INFO] Game Restarted
[2025-04-21 02:14:56] [DEBUG] <--|PKG 127.0.0.1:37500 Data: [G 0 7
]
[2025-04-21 02:14:56] [DEBUG] -----New msg-----
[2025-04-21 02:14:56] [DEBUG] <--|G 0 7 (Abgail, Sherlock Holmes)
[2025-04-21 02:14:56] [INFO] Player 0 (Abgail, Sherlock Holmes) found the guilty person
[2025-04-21 02:14:57] [DEBUG] <--|PKG 127.0.0.1:37506 Data: [R 0
]
[2025-04-21 02:14:57] [DEBUG] -----New msg-----
[2025-04-21 02:14:57] [DEBUG] <--|R 0
[2025-04-21 02:14:57] [DEBUG] |##>R 0
[2025-04-21 02:14:59] [DEBUG] <--|PKG 127.0.0.1:37522 Data: [R 2
]
[2025-04-21 02:14:59] [DEBUG] -----New msg-----
[2025-04-21 02:14:59] [DEBUG] <--|R 2
[2025-04-21 02:14:59] [DEBUG] |##>R 2
[2025-04-21 02:15:01] [DEBUG] <--|PKG 127.0.0.1:37538 Data: [R 3
```

FIGURE 11 – Logs Serveur

### 5.3.1 Serveur

Le serveur écrit dans un fichier nommé `server.log`, en enregistrant notamment :

- Les connexions et déconnexions des clients avec leur adresse IP et port
- Les messages reçus des clients et les réponses envoyées
- Les changements d'état du jeu (début de partie, tour du joueur, élimination, victoire)
- Les erreurs de communication ou d'exécution

```
[2025-04-21 02:14:35] [DEBUG] |##>M 0
[2025-04-21 02:14:35] [INFO] Game Restarted
[2025-04-21 02:14:56] [DEBUG] <--|PKG 127.0.0.1:37500 Data: [G 0 7
]
[2025-04-21 02:14:56] [DEBUG] -----New msg-----
[2025-04-21 02:14:56] [DEBUG] <--|G 0 7 (Abgail, Sherlock Holmes)
[2025-04-21 02:14:56] [INFO] Player 0 (Abgail, Sherlock Holmes) found the guilty person
[2025-04-21 02:14:57] [DEBUG] <--|PKG 127.0.0.1:37506 Data: [R 0
]
[2025-04-21 02:14:57] [DEBUG] -----New msg-----
[2025-04-21 02:14:57] [DEBUG] <--|R 0
[2025-04-21 02:14:57] [DEBUG] |##>R 0
[2025-04-21 02:14:59] [DEBUG] <--|PKG 127.0.0.1:37522 Data: [R 2
]
[2025-04-21 02:14:59] [DEBUG] -----New msg-----
[2025-04-21 02:14:59] [DEBUG] <--|R 2
[2025-04-21 02:14:59] [DEBUG] |##>R 2
[2025-04-21 02:15:01] [DEBUG] <--|PKG 127.0.0.1:37538 Data: [R 3
```

FIGURE 12 – Logs Serveur

### 5.3.2 Client

Chaque client génère son propre fichier de log, par exemple `client_Daniel.log`, qui contient :

- Les tentatives de connexion au serveur

- Les messages envoyés et reçus
- Les actions effectuées par l'utilisateur (sélections, suppositions, éliminations)
- Les erreurs ou comportements inattendus dans l'interface utilisateur

```
[INFO] Creation du thread serveur tcp !      You, 3 hours ago • Add gitignore and some example
[DEBUG] |-->C localhost 40004 Daniel
[DEBUG] <--|I 1 (Daniel)
[INFO] The Daniel user ID is :1
[DEBUG] <--|L Abgail Daniel - -
[DEBUG] <--|L Abgail Daniel Charles -
[DEBUG] <--|L Abgail Daniel Charles François
[DEBUG] <--|D 10 12 6 (Mrs. Hudson, James Moriarty, inspector Hopkins)
[DEBUG] <--|V 1 0 2 (Daniel, Sebastian Moran)
[DEBUG] <--|V 1 1 1 (Daniel, irene Adler)
[DEBUG] <--|V 1 2 0 (Daniel, inspector Lestrade)
[DEBUG] <--|V 1 3 1 (Daniel, inspector Gregson)
[DEBUG] <--|V 1 4 0 (Daniel, inspector Baynes)
[DEBUG] <--|V 1 5 1 (Daniel, inspector Bradstreet)
[DEBUG] <--|V 1 6 1 (Daniel, inspector Hopkins)
[DEBUG] <--|V 1 7 1 (Daniel, Sherlock Holmes)
[DEBUG] <--|M 0 (Abgail)
[DEBUG] <--|M 1 (Daniel)
[INFO] Played! joueur=-1 objet=2 guilt=-1
[DEBUG] |-->O 1 2 (Daniel, inspector Lestrade)
[DEBUG] <--|V 0 2 100 (Abgail, inspector Lestrade)
[DEBUG] <--|V 2 2 100 (Charles, inspector Lestrade)
[DEBUG] <--|V 3 2 100 (François, inspector Lestrade)
[DEBUG] <--|M 2 (Charles)
[DEBUG] <--|E 2 4 (Charles, inspector Baynes)
[INFO] Charles is eliminated! inspector Baynes wasn't guilty!
[DEBUG] <--|M 3 (François)
[DEBUG] <--|W 3 9 (François, Mycroft Holmes)
[INFO] Dear Daniel, You lose!      The guilty person was Mycroft Holmes!      Winner : François
[INFO] =====End of the game=====
[DEBUG] <--|R 3 (François)
```

FIGURE 13 – Logs Client Daniel

Ces fichiers sont précieux pour rejouer une session, identifier les erreurs ou comportements anormaux et améliorer la robustesse du système. Les fichiers complets se trouvent dans le dossier doc/logs/ de ce repo.

## 6 Conclusion

Ce projet a été une expérience très enrichissante pour moi. J'ai pu renforcer mes connaissances en C, tout en découvrant concrètement comment fonctionnent les sockets et les communications réseau. Travailler avec les bibliothèques C et développer l'interface graphique du jeu m'a donné des compétences pratiques que je trouve vraiment utiles. Pour les prochaines versions, je pense à plusieurs améliorations :

Revoir la structure de l'interface pour la rendre plus agréable visuellement Séparer les fichiers et créer des bibliothèques pour mieux organiser le code Permettre aux joueurs d'interagir entre eux avec des réactions Ajouter des sons pour rendre le jeu plus vivant quand on fait des actions

Merci pour ce projet qui m'a permis d'appliquer ce qu'on a vu en cours à quelque chose de concret et amusant.

## Références

- [1] Bogotobogo. Sockets serveur et client, 2020. Consulté le 04/04/2025.
- [2] Broux. Sockets en c. <https://broux.developpez.com/articles/c/sockets/#LV>, 2007. Consulté le 09/04/2025.
- [3] Emmanuel Delahaye. La gestion du temps en c. <https://emmanuel-delahaye.developpez.com/tutoriels/c/notes-langage-c/?page=lttime-hgt-la-gestion-du-temps>, 2009. Consulté le 12/04/2025.
- [4] Eleftherios Kosmas. Presentation programmation de socket en c. Technical report, Université de Crète. Consulté le 04/04/2025.