Rapport de TP (IoT)
The Mind

Romain Bohbot
Elodie Difonzo
Mathieu Gourichon
Roqyun Ko

Introduction

L'objectif de ce module est de mettre en place une application utilisant le protocole de communication MQTT.
Nous allons développer un jeu nommé "The Mind", celui-ci mettra en communication quatre joueurs sur quatre machines distinctes.

Objectif du jeu

The Mind est un jeu coopératif qui se joue en plusieurs manches à 4 joueurs.
Lors de la première manche, chaque joueur reçoit des carte dont le nombre est égal au niveau actuel (les cartes sont numérotées de 1 à 100).
Le but du jeu est de poser les cartes dans l'ordre croissant sans que les joueurs ne communiquent entre eux.
S'ils gagnent alors ils ont une vie supplémentaire ou un bonus shuriken mais s'ils échouent ils perdent une vie.
La deuxième manche se joue à 2 cartes par joueurs et ainsi de suite.
Le bonus shuriken permet, si l'ensemble des joueurs souhaite l'utiliser, de défausser face visible la plus petite carte de la main de chaque joueur.
Une partie s'arrête lorsque les joueurs ont réussi la dernière manche ou qu'ils n'ont plus de vie. Le jeu continue jusqu'au niveau 8 pour quatre joueurs.

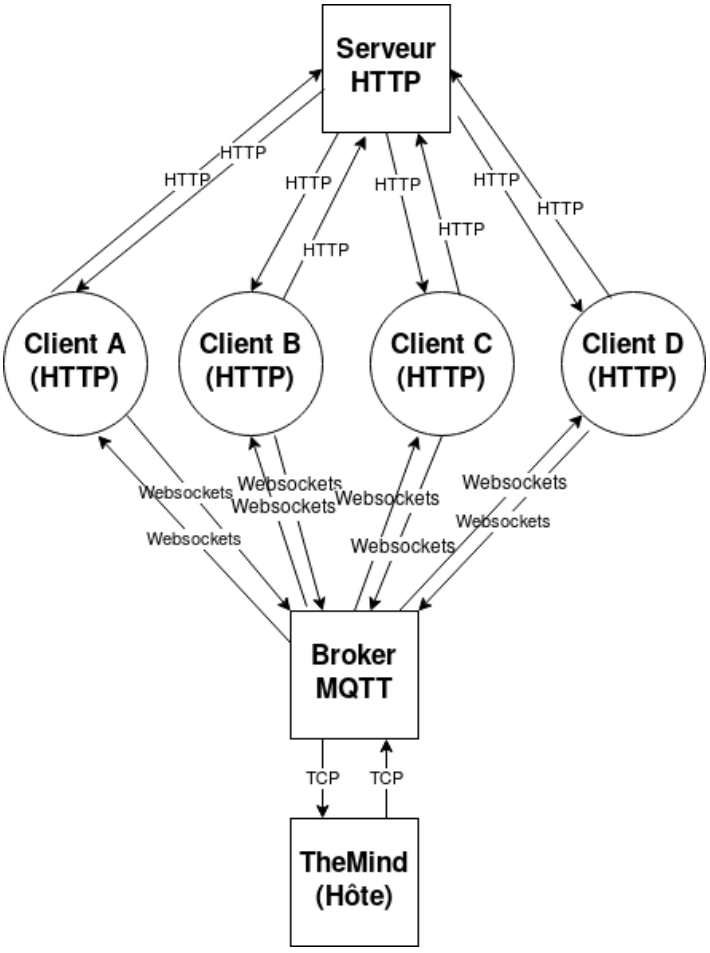
Installation du jeu

Implémentation du jeu

L'implémentation du jeu est en 3 partie :

- Serveur (codé en ***NodeJs***)
    - HTTP
    - Broker MQTT
- Hôte de jeu (codé en ***langage C***)
- Client (codé en ***HTML / JavaScript***)
    - HTTP

Architecture

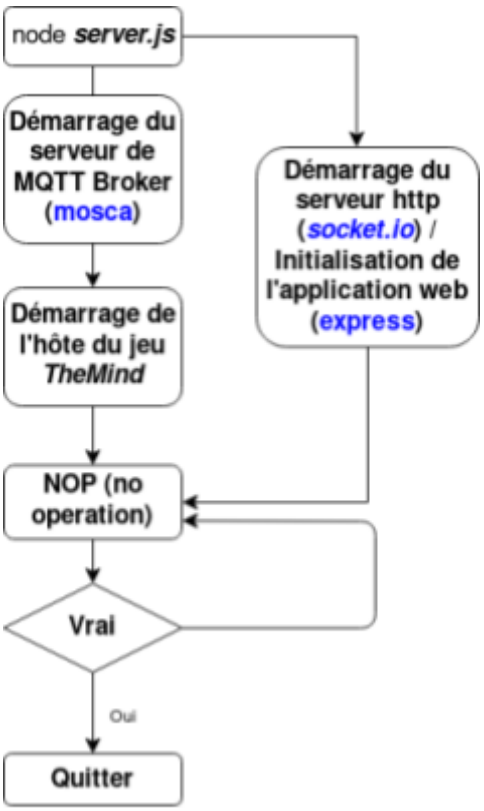


Partie Serveur

Le Node.js script ***server.js*** débute le serveur HTTP (implémenté par le module ***socket.io*** *et* ***http***) et initialise l'application web (implémenté par le module ***express***) pour charger les fichiers javascript, HTML et image. Ainsi, on peut démarrer les clients par navigateur web et se connecter au jeu à distance. Le port utilisé pour les clients est 8000.

On commence également le broker MQTT (implémenté par le module ***mosca***) pour que les clients et l'hôte puissent communiquer. Puisque le client web ne peut pas communiquer en utilisant le protocole ***TCP*** (port 1883), il faut configurer également le protocole ***Websockets sans SSL*** (port 9000). On peut choisir les autres ports lors de la configuration.

Le script lance également l'hôte, codé en langage C.



server.js : Diagramme de flux

Partie Hôte

Le logiciel, codé en langage C, gère le déroulement du jeu et communique avec les clients pour recevoir et envoyer les données. Il utilise le protocole **TCP** (port 1883) pour la communication à travers le broker MQTT.

Pour l'implémentation de la communication MQTT, on utilise le projet open-source ***mosquitto*** d'**Eclipse** (https://github.com/eclipse/mosquitto).

C'est l'hôte qui vérifie l'ordre des cartes posées par les utilisateurs, défausse les cartes, récompense, pénalise et augmente le niveau. Par conséquent, les clients ont peu d'action qu'ils peuvent exercer.

Il y a 4 types de données qu'il reçoit :

- le nom d'un joueur
- la carte joué
- l'enquête pour utiliser un shuriken
- la voix pour utiliser un shuriken

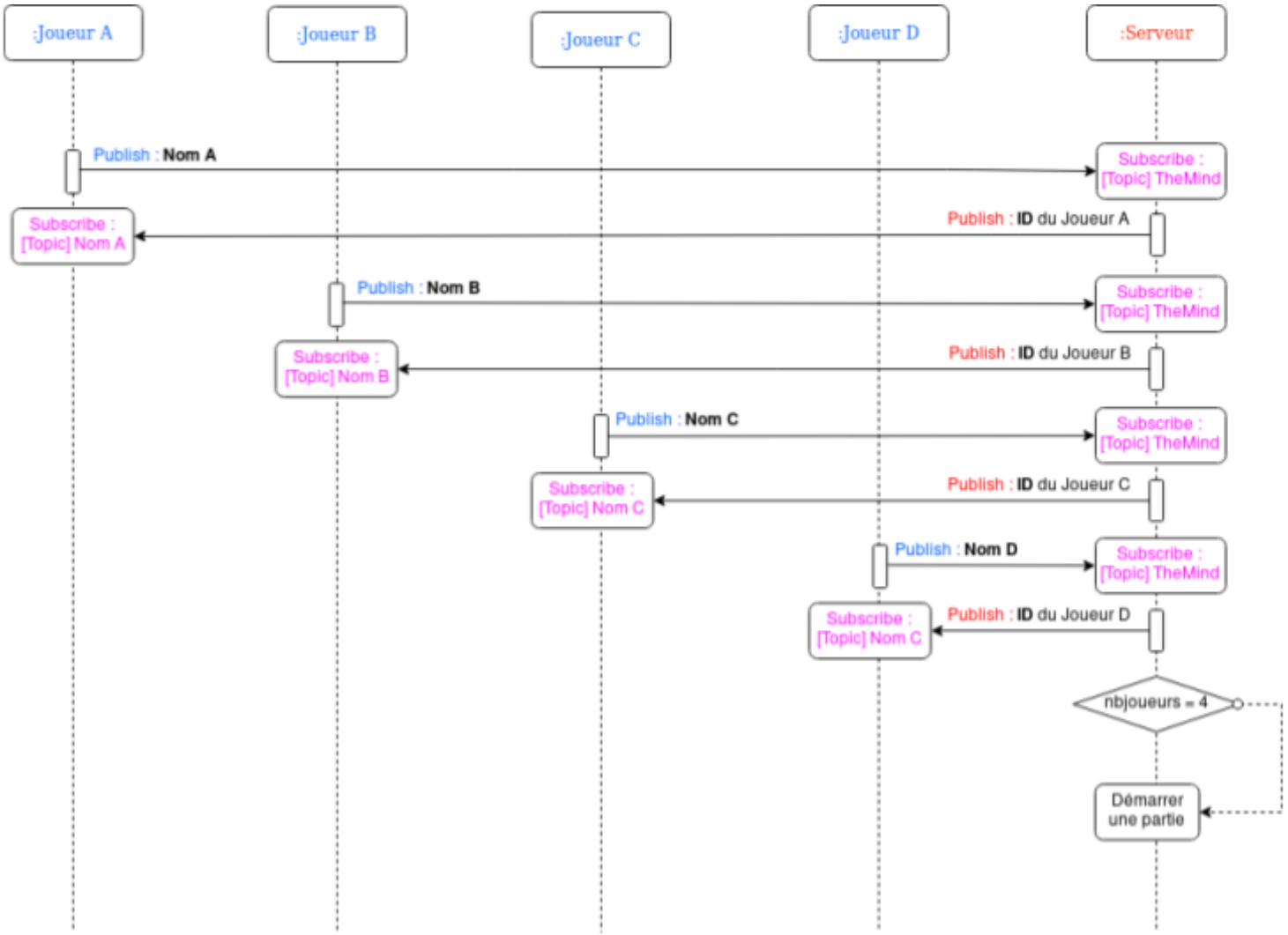
A partir de ces données, le logiciel renvoie aux clients

- l'identifiant unique du client
- les manches de chaque joueur
- autorisation d'utiliser un shuriken
- le nombre de vie
- le nombre de shuriken
- la carte récemment jouée
- la carte défaussée
- la commande pour initialiser les manche et les carte jouées (pour recommencer une partie)
- message

Le schéma ci-dessous récapitule le protocole :

Protocole du jeu

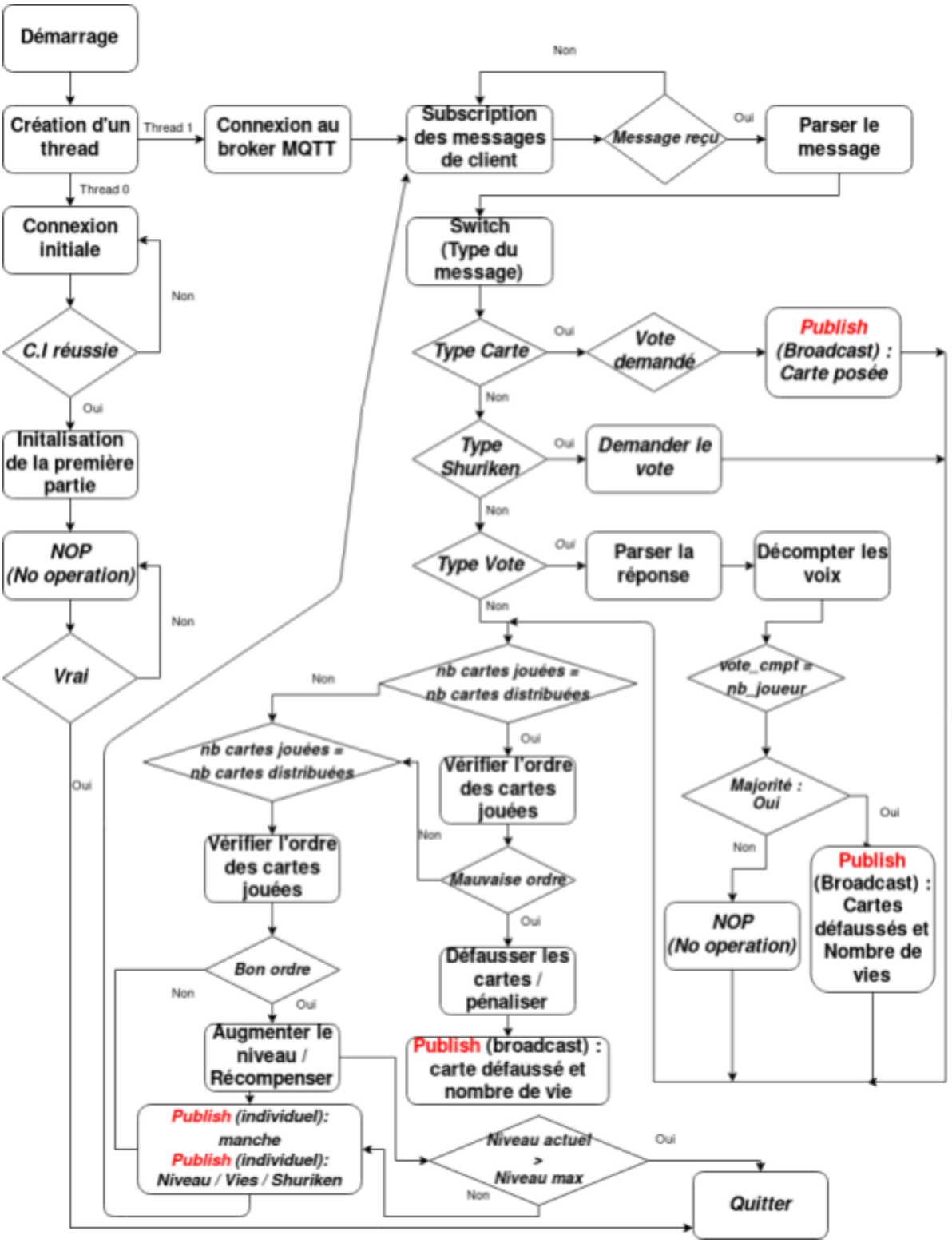
Architecture logiciel



Connexion initiale

Quand le client envoie le nom, l'hôte attribue un identifiant unique pour le client. Ainsi, même s'il y a des conflits de nom, ceux-ci ne posent pas de problème pour la communication.

Diagramme de flux



Partie Client

La partie client reçoit les entrées d'utilisateur et donne l'affichage graphique. Elle a été réalisé en HTML5 ainsi qu'en javascript. Elle communique également avec l'hôte du jeu à travers le broker MQTT. En revanche, puisqu'on ne peut pas utiliser directement le protocole **TCP**, il faut utiliser le protocole **Websockets sans SSL**. Pour l'implémentation de communication MQTT, on utilise le projet open-source **Paho Javascript Client** d'**Eclipse** (https://github.com/eclipse/paho.mqtt.javascript).

Pour attribuer le nom, on entre le nom du client dans l'URL en tant que *query*, par exemple, http://localhost:8000/index.html?name=NOM. Si le serveur est lancé, le jeu commencera lorsque quatre client seront connectés.
Comme le jeu est majoritairement géré au niveau du serveur, le nombre d'action exécutée par le côté client est limité :

- Poser la carte de son deck ayant la plus petite valeur.
- Shuriken
    - Utiliser
    - Voter pour en utiliser un.

Consulter la ***partie hôte*** pour consulter le protocole de jeu utilisé.