Computer Networking

Project 1 : Battleship

KLAPKA Ivan (s165345)

3rd year in civil engineering
Academic year 2018-2019

Table des matières

1 Software architecture

1.1 Battleship

The implementation of the game is running thanks to tree main class : *Game*, *Boat* and *Coord*.

1.1.1 Game

Game is responsible for processing fire request, ships random positioning and provides a list of coordinates which have been fired at. It also register the number of shots fired and number of hits (without counting for strikes at the same position).

Choosing random positions for the ships is done by taking a random existing coordinate, choosing randomly through a set of available positions of the end of the ship and check that theses positions and the one between them are not already taken.

1.1.2 Boat

Boat contains the list of his coordinates and his type. It can check whether it is placed at a certain position or not.

1.1.3 Coord

Coord is responsible for the coordinates, it can translate from trees different conventions : Traditional (B3), Order out off 100 (12) and classical (1,2). It can verify whether coordinates exists, are aligned or not and return the coordinates between them. It can also look for existing coordinates that are aligned and at a certain distance.

1.2 TCP Communication

The TCP communication is done thanks to tree main class : *BattleshipClient*, *BattleshipServer* and *Worker*.

1.2.1 BattleshipClient

BattleshipClient is responsible for sending the request and interpreting the response. It translates both the actions of the player into request and the responses of the server into readable information. It works by sending a request and reading a two bytes header. If the

header is the one expected, the process continue by reading more data (if necessary) and display the information. If the header is not the expected process then the process alert the user, abandon the instruction and wait for the next one. A time out occurs after 10 seconds if the response isn't received.

1.2.2 BattleshipServer

BattleshipServer is responsible for intercepting new connections and creating new workers to handle them.

1.2.3 Worker

Worker is responsible for answering request from the client. It waits for a two bytes header to arrive, when it does, the worker processed to retrieve more data (if necessary) and sends the corresponding response. It is also the one manipulating the Battleship game, it applies request and checks the game state.

2 Multi-thread coordination

Multi-threading is only used on the server side. The server is constantly looking for new connections and when one is found it launches a dedicated thread ("worker") to handle the new player. Each worker get their own socket and answer the request sent through it. A time out system is also implemented, after 5 minutes if the player has not sent a request the worker will close the connection.

This allows for multiples players to play at the same time and avoids the server from running useless workers.

3 Limits

3.1 Request error will not shut down client

Since the implementation of a game over is done by the server (i.e. When a game is over, the server responds with "1 4" for each guess until it receives "1 0" again), the client has to client has to communicate the request errors and not shut down. This choice leads to request error no shutting down the client and can become inconvenient. For example if the server is running an other version, the client will successfully connect but will not be able do to anything and will re-transmit errors to the player without explanations.

3.2 Incorrect message of size different from 2 will corrupt communication

For example, if the client sends "1 0 52" instead of "1 1 52" the server will start a new game but it will also, due to the stream oriented communication, read the first byte of the next header as 52. This problem will carry on and make communication impossible.

This problem is also true for the server, example : if the server tries to send the confirmation of a hit "1 2 3" but the client receives "1 1 3".

Since the TCP protocol is lossless and won't suffer from any packet re-ordering, this error can't happen in practice but still worth mentioning in the case of a wrong implementation of the BP protocol.

3.3 The server is vulnerable

The server does not have a limit for the number of running workers (player playing simultaneously). The server can easily be crashed by bombarding the server with connection requests.

4 Possible Improvements

4.1 A better interface

- The game could be improved by displaying a visual representation of the game.
- An external program with buttons instead of keyboard coordinates.
- Display the number of shots remaining

4.2 A new type of response to implement a game over

The BP protocol can be improved by adding a new type of response to handle a game over. It would contains a traditional header and a additional byte of data indicating if the player has won or lost. Example : "1 5 0" would mean the player has lost and "1 5 1" that the player has won.