Computer Networking

Project 2 : Battleship

BERNARD Simon (s161519)
KLAPKA Ivan (s165345)

3rd year in civil engineering
Academic year 2018-2019

1 Software architecture

First there is the **WebServer** class, that is responsible for handling new connections and launching a **Worker** to process the request. **WebServer** makes sure to delete expired game every 100 call to a worker.

The **Worker** is only in charge of a single request. It first process the HTTP request by calling the **ParsedHTTPRequest** class, it then knows which HTTP response to send. It uses the help of the class **HTMLGenerator** to make the appropriate HTML page if required and send the appropriate response. It also in charge of updating the corresponding object **Game**, the hall of fame and the list of game currently being played.

The **ParsedHTTPRequest** will parse the input stream of the worker and store the important information such as the method, URL, HTTP version, etc... making it easier to access.

The **HTMLGenerator** is in charge of generating the HTML page and will be called to process chunkEncoding and gzip. It returns the string of character that compose the part of the message responsible for the HTML page. **HTMLGenerator** calls multiples files in order to function properly, like the images, the default.css, script.js or the noscript.html.

The **myHTTPException** class is used to process errors detected by the **Worker** and **ParsedHTTPRequest**.

2 Multi-thread coordination

This time we have multiple **Worker***s* that can try to access the game list, the hall of fame (possibly by a call of the **HTMLGenerator**) or a specific game (latter require very close request from the same cookie which is quite unlikely in standard use). To do so, **Worker***s* use the "synchronized" keyword on the conflicting object when needed :
For the game list : when they try to search a specific game or when they want to remove a game from the game list.
For the hall of fame : when they try to read it or when they want to put their game in it.
For a specific game : when they try to fire at a position.

The deletion of cookie/game is done at least 10 minutes after the last request of the user of this cookie, due to this high delay, synchronization is not needed on the reading/writing of the expiration date of a game. Two very close writing will write nearly the same things, so the one that will be kept is not important. If a reading by the **WebServer** is close to a writing, there can be a conflict only if the 10 minutes delay is finished, so the request will maybe work if the game is find by the **Worker** (but the **Worker** has not finished his work) before it is deleted by the **WebServer**. The game will become completely unavailable after

the request ends. So the next request will lead to a new game, there is no real issue about that.

The **WebServer** also accesses to the game list every 100 requests (so every 100 **Worker***s* created) to remove those which were used for the last time at least 10 minutes earlier. The **WebServer** also uses the "synchronized" keyword on the game list when this event occurs.

3 Limits

The use of a thread pool allows the server not to crash on a DDoS attack, but users will not be able to use the server during the attack (nearly all threads would be used by the attackers), it is better than noting but it is not the best.

The HTML is not always well displayed on mobile device, the HTML could be more responsive to the device connected.

4 Possible Improvements

- The game could allow 1v1 match.
- This game could have different level of difficulty (more ships, bigger grid,...)
- The game could make a sound when a ship is hit.
- The game could allow to customize your cookie or ask the name of the player for display in the Hall Of Fame.
- The game could tell you how many players are currently playing.