Homework 2: Multiplayer Battleship Game

Hadachi&Lind

November 28, 2016
1 INTRODUCTION

The objective of this homework is to demonstrate how to implement distributed applications using RPC, Remote Objects as well as how to organize the design of the application to follow the indirect communication paradigm.

We eventually know all the advantages and disadvantages of the indirect communication; therefore, please take them into account when designing the solution of this homework.

2 DESCRIPTION

The application that you have to implement is a multi-player network game, which is the "battleship" game. The game is well known, the rules are expressed by the link: [https://en.wikipedia.org/wiki/Battleship_%28game%29].
Traditionally, there are two players positioning their ships and then they bombard each other, in turn based (one after the other), in an attempt to sink each others ships. As you can see, there is no real-time aspect in the game. The players do commit actions in an organized manner. Hence, it might be resumed to an event driven design (and we proved in Seminar #6 the event driven design is suitable for an indirect communication paradigm).

In our implementation, we will not limit the game into two players, but we will introduce multi-player option (at least three players). And the rest remains as in the traditional game. We just assume that all the war-parties are fighting each other (aka “Deathmatch”). The winner is the last one standing on the sea battlefield. Optionally, we can score each successful hit.

### 3 Requirements

The following use cases must be supported:

- User starts the application, and can select what game server to join
  - The list of game servers is collected automatically by a client application (listening to server announces)

- User has to provide his nickname in order to join with the game
  - After nickname is specified user may select the game server to play on

- Server should not allow two users using the same nickname
  - The player should be reject from joining the game server with the message to change his nickname.

- Once user has joined the game server, it can either create a new game session or join existing one.

- In case the user wants to create a new game session, he/she has need to provide the desired size of the battle field, then the new game field is created and the user has to position his ships on flat map (no obstacles, just water). Once the ships are positioned, the creator player should wait for someone to join his game session. The creator player must be notified each time new player joined with his game session. Once the creator player is satisfied with the number of players who have joined, he may trigger the start of a battle.

- In case users want to join the existing game session, he/she has to position his ships and wait for the master player to start the battle.

- Once the game session has a battle running the game server must preserve the order of players, allowing them to shoot one after the other. Each time the shoot is committed by a player, the server must notify the next one of his turn to shoot.

- Once the game session has a battle running the game server has to check the end-game condition, which is the situation where all the ships standing on the battlefield belong to only one player.
• Once the shoot is done by player the server must check the hit-conditions
  – if any ships were hit it should be visible for the player who triggered the shoot
  – the suffering player should see his ship attacked, and he/she should see the origin
    (who did attack him)
  – the other player should not see this positive hit

• Once the shoot is done by player the server must check the sink-conditions, in other
  words: if any ships were hit and if that hit made a ship sink (completely destroyed)
  – if any ships were sunked it should be visible for everyone at moment of sinking

• Once the player is out of the game he can do one of the following:
  – leave the game session
  – wait till the remaining players finish the session
    * while waiting the player sees everything all ships of all players as well as the
      scores and the damaged ships (so called spectator mode)

• Once the game session is finished the creator-player may restart the session.

• Player should be able to leave the game session at any moment (all his/her ships have
  to be removed from the game field)
  – in case the creator-player is leaving the game-session the server selects the new
    owner among the remaining players
  – in case only one player is involved in the game and the others already left, the
    game session should end

• Player should be able to disconnect and connect again without loosing his game ses-
  sion
  – disconnecting should result in leaving the game session
  – in case the player did disconnect without leaving and he/she is not connected
    back for long time, then the creator player has liberty to withdraw the concerned
    player
  – in case the creator player did disconnect without leaving and he/she is not con-
    necting back long enough, the server selects the new owner among the remaining
    players
    * the previous creator-player can still connect back with lost ownership of the
      game session
  – in case game session was finished at the time the user is reconnecting, he/she
    should be informed and suggested to create or join another session
4 IMPLEMENTATION

You are free to use any third party libraries for GUI, Network etc. when implementing the application (as long as the indirect communication paradigm is respected). You can choose any GUI design as long as the game requirements are respected (CLI based of course allowed too). The use of threading is up to you and can be even avoided as long as other requirements are preserved. The indirect communication middleware is of a free choice as long as it supports Python 2.7.

5 DELIVERABLES

They will be two major deliverable for this Homework:

**First:** You have to submit your Team members names including you and also the public git repository link via the Moodle latest 7th of December at 12h00.

Link: https://moodle.ut.ee/mod/feedback/view.php?id=276547

**Second:** A report that includes the following:

- User manual explaining setup process: Client, Server, Middleware (other possible dependencies required).
  - As we are relying on indirect communication, we will eventually use middleware and third-party services.
- User manual explaining the user interface and how to play the game.

**Third:** The source code of your implementation.

Concerning the implementation, we will consider only the source code from your public git repo provided. Note, any modification done after the due date to the git won't be considered.

Concerning the report, it should be compressed and submitted in the course website by one member of the team.

Link: https://courses.cs.ut.ee/2016/ds/fall/Main/Kodutööd

5.1 GRADING:

Total points for this Homework are 20 pts, and its breakdown is as follows:

- 6 pts: The user manual document
- 14 pts: The implementation:
  - 2 pts: code is functional (and does run)
  - 6 pts: the indirect communication is used correctly
  - 4 pts: requirements were respected
– 2pts: code is documented

• +3pts bonus: RPC or Remote Objects were used on top of indirect communication paradigm