Word Game

12th November 2017

INTRODUCTION

The main idea of our project is to build a network based game : Word Game. This is a simple translation of the game in which we are allowed to spell a word and our opponent can spell any word (non-repetitive) that starts with the ending character of the previous word and this goes on. This is done for some fixed number of rounds and the one who takes greatest latency in typing (hence not thinking quickly enough about short words) is the loser and ranks are given on that basis.

What makes our project stand apart is that people can create **ROOMs**. A ROOM is a private environment of 4 members that can play the game and compete with each other. Our game can concurrently run any number of rooms. People who have their name registered in the game server, can create a room and be the first participant in the word game and wait for other people to join the same room. If the room is already created (the room name has already been taken) and it's full, then he must create/join a different room. One who is not registered in the game server can get himself registered and after only that he can join ROOMs.

This is the realistic model exactly in the way in which actual gaming world works.

OBJECTIVE

1. The main idea is to build a text-based application, such that the server will be able to communicate parallely with the clients.
2. We have the feature of creating ROOMs that allow people to be grouped and a group of 4 members to play and compete in the game.
3. The game has been made fair enough for everyone. No member can add words with special characters in them or words that are repeated by anyone.
4. The game is made interesting as after each move one can view the dynamic scores of every other person in the ROOM. This instills in the contender : the blood to battle!

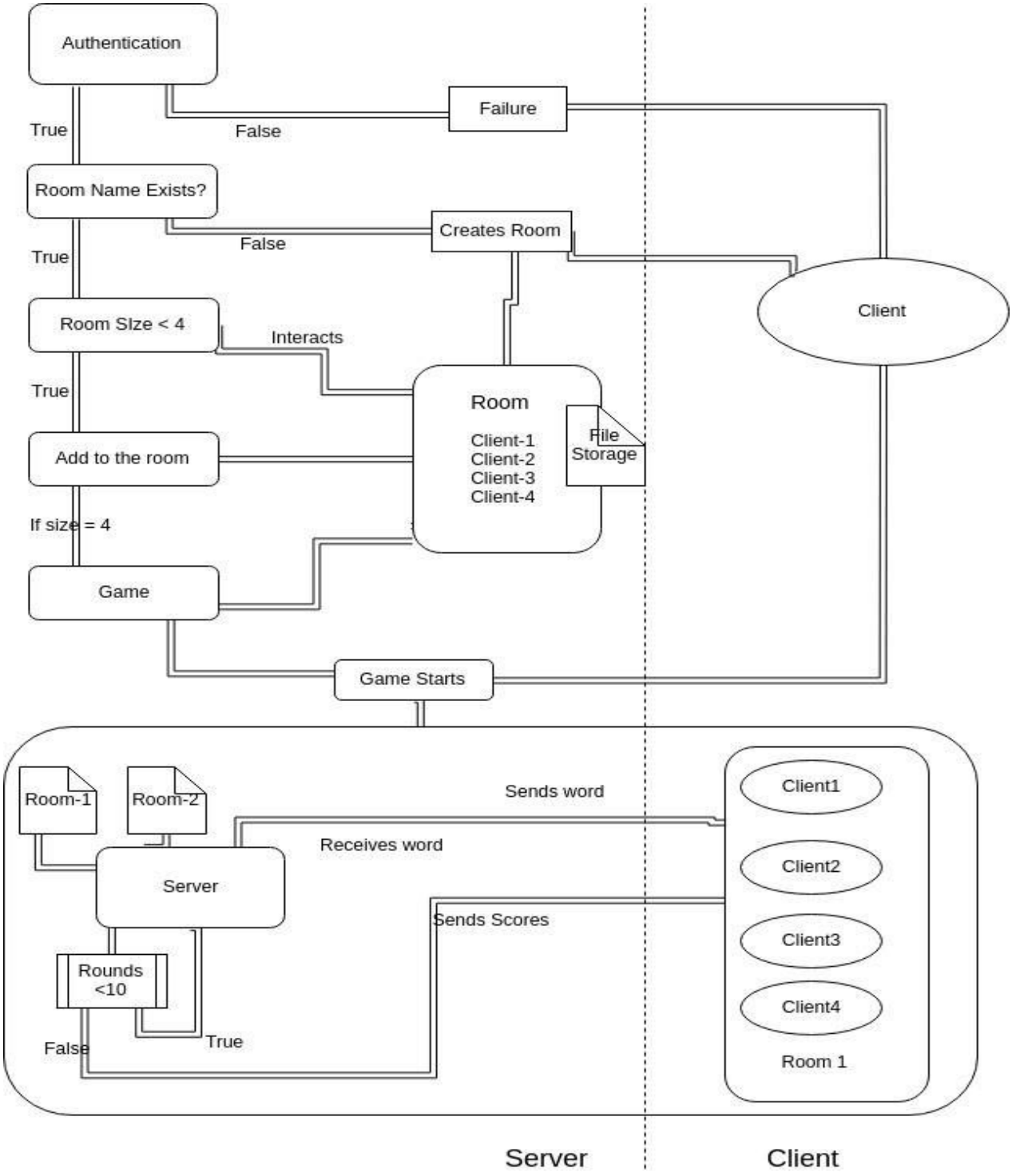
ASSUMPTIONS

1. ROOM size is kept 4 for making the game competitive and not crowded.
2. Word size that can be given as input is kept at 1024 max.

3. Rounds that can be played after which results are declared is 10.

ARCHITECTURE

We represent to you the model diagram that we have used in our game:



As evident from the diagram, we have clearly shown the distribution of the work between client and the server. The client provides a request for creation of a room together with username, password and room name and the server verifies the details as shown in the diagram. Only after 4 members have been added to the ROOM, the game is started.

After the game starts, the server might be connected to multiple ROOMs simultaneously as shown. If in a particular ROOM the number of rounds have reached 10, the server displays the score (or say penalties) of each of the participating members in the game.

The client script only manages the checking of the validity of the words. The main synchronization is carried out by the server. It manages many files for the game (a set of files per room) and cycles through the turns of each member in the sequence in which they have joined the ROOM.

IMPLEMENTATION ENVIRONMENT

Language Used

We have used C language to construct the scripts. Both executables are compiled from C code.

Connection Type

We have used a persistent TCP connection for our purpose. The client sends a connection request to the server using TCP connection.

Multiple Connection Strategy

We have used *fork* calls in the server to create parallel working of client connections.

Multiple ROOM Strategy

We have used file names according to the room names to distinguish one ROOM stats from the other. This keeps them working independently.

SUMMARY

It looks as we have achieved every milestone of what was our objective. We have implemented them and the code runs in a clean way. Our whole motivation was from the childhood game that we play with our siblings - “Say the name of a country that starts with the ending letter of the one

that I say” until we run out of country names. Our text-based word game rejuvenates those memories and is a great success in terms of application of networks on our daily life.

PROJECT MEMBERS’ DETAILS

Akash Kumar Dutta - 150071
Harsha Nalluru - 14408
Abdus Samad - 13013

The implemented code can be found here : https://github.com/HarshaNalluru/cs425projec