**ReversiS**

Bădărău Dragoș (2A1)

Faculty of Computer Science „Alexandru Ioan Cuza” Iași

1. **Introduction**

ReversiS is the server implementation that allows two clients to play the strategy board game Reversi. The game is algorithmically completed by the server that executes every command given by each client. This paper illustrates the server functionality which is based on establishing a connection with the clients. The report provides information about the application architecture, the used technologies, the implementation details, and overall improvements that could be considered in the future.

1. **Technologies**

The communication between the server and the clients will be made using the **Transmission Control Protocol** (TCP) - the original transport-layer protocol within the Internet protocol suite. This connection-oriented method is characterized by providing a connection-oriented service that guarantees that each packet is received intact and in the correct order. In addition, this very formal process has built-in error checking therefore it will retransmit missing packets and eliminate duplicates because every byte of information is numbered.

The application will be written in **C** and based on the **Client-Server** model with a concurrent server using sockets to provide concurrent services to multiple clients. When we will want to create a connection the client will initiate the connection request. After the connection between the server and each client is successfully done, there will be created a **child process** for each of the users. Besides, the clients will request to start a match and the server will create a separate session for both clients through a function that will take the descriptors of each client. The server will always use port 3061. All the clients will have the same IP address, so there will be a port number for each game. The port numbers are for communication, not security.

1. **Program Architecture**

The entire program is based upon communication between a server and multiple clients who will be paired to play the game. The entire communication will occur between the client and the server.

The entire project is based on the functionality of the server. It will start, wait for the clients, group them into sets of two and create matches. Each client will be capable to connect, disconnect and request to join a match with the client specified to the server.

1. **Implementation Details**

Reversi is a strategy board game for two players, played on an 8x8 board. The game starts with four pieces of two different colors placed in the middle of the board. Each player tries to capture the opponent pieces by surrounding the other player’s pieces horizontally, vertically, or diagonally. The game ends when the table is completed with pieces and the winner is the one with more of its pieces on it.

Most communication (the entire game-related) will happen in the matches. The clients will have unique names and passwords and they will have to log in before each game. The 8x8 board will be shown as an 8x8 matrix number composed of zeros, ones, and twos (zero for the space where there is no piece, one for the spaces where the player one’s pieces are, two for the spaces where the player two’s pieces are). The client will be able to send a number (from 0 to 63) that will be the space where he wants to put his piece (for example, the number 20 represents the 20/8=2nd line and 20%8=4th column; the matrix is numbered from 0 on column and line). There will be functions verifying the correctness of the board, analyzing the table after each move, being able to tell the client to move again (if its move doesn’t respect the rules). After each correct move, the table will change correspondingly according to the game rules (by changing the pieces surrounded correctly). Besides, each player can be identified by its name, because there will be a vector keeping track of each current clients’ file descriptor and its account. The idea of the algorithm that will sustain the match-flow is summarized in the following code:

The commands used:

1. Username:password ( for example Dragos:qwe )
2. Start:descriptor/quit (for example start:0 if the client wants to wait, start:4 if the client wants to play against the client with the descriptor 4)
3. The number representing the position where the player wants to move if it’s the turn for the client to move or 0 if he has to wait and it’s not his turn
4. **Conclusions and Improvements**

This application represents the server part of the board game Reversi. The most important improvements that could be done to the code may refer to bug-fixing. Its client will have the responsibility to design its graphic interface. Overall, the application could be improved by the following ideas:

* The user being able to have an account (a name and a password) and to log in
* Each round has a limited amount of time
* The player being able to pause the game
* A set of statistics about the game and each user
* The user being able to add other users as “friends” and chat with them
* Different modes of play, including the one against the computer (with different levels of difficulty)
* The server being able to remove players due to inactivity
* The player being suggested moves
* The capacity of the server to store any game

1. **Bibliography**

* Project information

[**https://profs.info.uaic.ro/~computernetworks/ProiecteNet2021.php**](https://profs.info.uaic.ro/~computernetworks/ProiecteNet2021.php)

* TCP Concurrent Server and Client using socket

For the server : <https://profs.info.uaic.ro/~computernetworks/files/NetEx/S9/servTcpCSel.c>

For the client : <https://profs.info.uaic.ro/~computernetworks/files/NetEx/S9/cliTcp.c>

* Reversi Game (game functionality and inspiration)

<https://cardgames.io/reversi/>