Universidade Federal de Alagoas
Instituto da Computação
Redes de Computadores - 2020.1

Relatório do Projeto de Implementação da AB2

Dupla: Rafael Emílio Lima Alves e Wagner Anthony de Medeiros Silva.

1.0 Descrição do Projeto

A aplicação escolhida pela dupla consiste em um clássico jogo da velha, o qual requer o pareamento de dois jogadores para iniciar a partida e permite partidas simultâneas.

A proposta foi posta em prática na linguagem de programação python, utilizando as funções primitivas de rede dessa linguagem, através das bibliotecas nativas socket e threadings. Ademais, para a construção da aplicação e do layout em si foram utilizadas as bibliotecas tkinter e pygame.

2. Desenvolvimento da Aplicação

2.1 Client-side

Após a construção do layout usando as bibliotecas tkinter (janela do username) e pygame (janelas da aplicação), e a construção da lógica matemática por trás do jogo, foi iniciada a implementação da comunicação do lado cliente utilizando a biblioteca sockets.

O socket é criado e estabelece comunicação com o servidor somente quando o jogador entra na janela de espera criada pela função waiting_room. Após a criação e conexão do socket o cliente fica solicitando mensagens no servidor, e quando a primeira sentença dessa mensagem for o termo “begin”, quer dizer que há dois jogadores simultâneos na fila de espera, dessa forma a conexão entre os dois é estabelecida e a partida é iniciada.

O movimento inicial sempre será do jogador com o símbolo “X”, alternando movimentos com o jogador “O”. Para isso, a mensagem recebida do servidor passa a ser composta por 3 elementos: a frase “yourTurn”; o símbolo do movimento da vez (“X” ou “O”); e o tabuleiro do jogo, que no client-side é manipulado como uma matriz, mas é enviado e recebido no servidor como uma string de nove componentes correspondente à matriz, onde o 0 indica casa não marcada, o 2 o símbolo “O” e o 5 uma casa marcada pelo símbolo “X”.

Quando um movimento válido é feito no tabuleiro, o cliente envia uma mensagem para o servidor composta por quem fez a jogada e a nova configuração do tabuleiro, o servidor irá checar se há vencedores, caso não, o padrão de mensagem descrito no parágrafo anterior é atualizado com o tabuleiro novo e o símbolo do jogador que tem a vez.

Ao final da partida, o vitorioso recebe a mensagem “youWin” e é direcionado a janela de resultado de vitória, enquanto o jogador derrotado recebe “youLoose (string do tabuleiro)”, para que assim possa exibir na tela o movimento ganhador do seu oponente, e após isso é direcionado à tela de derrota. Quando a partida se encerra com empate a checagem é feita no próprio cliente, portanto não há mensagens no servidor para definir isso.

Em todos os casos de fim de partida, a conexão do socket é encerrada somente quando o looping da partida é finalizado, impedindo assim o fim da conexão antes do momento apropriado.

2.2 Server-side

No lado servidor da aplicação, inicialmente criamos um socket e fazemos com que ele escute na porta determinada pelo programa. O servidor vai então aceitar conexões de novos jogadores, armazenando os dados desses jogadores em um dicionário chamado “clients”, e colocando os jogadores em uma fila de espera por uma partida. Com ao menos dois jogadores na fila, o servidor pode começar uma partida removendo esses jogadores da fila, e criando uma nova thread da função “playgame()”, de modo que o servidor pode continuar a aceitar novos jogadores enquanto os jogos atuais são executados.

A função “playgame()” recebe os dados dos jogadores, cria um novo jogo e envia esse jogo para os jogadores, bem como o nome dos competidores. Então, a função “playgame()” entra em um ciclo que consiste em enviar o jogo atual para o “jogador X”, receber uma jogada, enviar o jogo atual para o “jogador O”, e receber outra jogada, até que um dos competidores ganhe o jogo, ou que seja identificado um empate no client-side. A cada nova jogada a função “playgame()” chama a função “checkgame()” que verifica o jogo atual tentando encontrar uma jogada vencedora. Quando uma jogada vencedora é identificada, o servidor manda uma mensagem para o vencedor notificando a vitória, e outra mensagem para o perdedor notificando a derrota. Finalmente o servidor encerra a conexão com os competidores e fecha a thread responsável por administrar o jogo atual.

3.0 Dificuldades na implementação

3.1. Solucionadas:

- Criação de partidas simultâneas sem encerramento da conexão anterior, solucionado através do uso de Threads.
- Obter o input do nome de usuário através de uma janela do pygame, o qual foi solucionado através do uso de uma janela de autenticação construída com a biblioteca do tkinter.
- A integração do recebimento de mensagens constantes do servidor com o looping das janelas do pygame e a captação de eventos do usuário, o qual provocava travamento na execução da aplicação. Solucionada através da configuração de timeout do socket cliente.

3.2. A serem solucionados em implementações futuras:

- Ocasionais timeoutexception que ocorrem no pareamento da partida, mas de forma ínfima.