UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO

RELATÓRIO DE PROJETO - AB2

Guilherme de Oliveira Monteiro Peixoto e Artur Cavalcante de Jesus

Pedra Papel Tesoura Multiplayer

1. Descrição

O projeto consiste na implementação de um jogo de Pedra Papel Tesoura multiplayer e utiliza conceitos de sockets, protocolo de transporte e outros assuntos referentes à disciplina. Foi inteiramente construído utilizando Python como linguagem.

2. Como executar a aplicação

● É necessário obter a biblioteca pygame, que pode ser instalada com o seguinte comando em distribuições Linux: `python3 -m pip install -U pygame --user`

- Se preferir, instale as bibliotecas citadas no arquivo `requirements.txt` usando o comando: `pip install -r requirements.txt`

● Após a instalação, o server.py deve ser executado. Isso pode ser feito com o seguinte comando no terminal: `python3 server.py`

● Em seguida, é necessário executar o client.py duas vezes para entrar com os dois jogadores. O jogo só inicia após a entrada do segundo jogador. Para isso, execute o código abaixo: `python3 client.py`

3. Desenvolvimento

O jogo foi desenvolvido na linguagem de programação Python utilizando a biblioteca Pygame.Foram implementadas as funções Sockets e Threads, ambas presentes no arquivo server.py. A função Sockets também implementada no arquivo network.py permite a conexão entre dois processos rodando na mesma máquina;

Analisando os comandos utilizados:

- Socket: neste ponto temos um comando para dizer ao S.O. que desejamos criar um socket com determinadas características. Em geral, as características são relacionadas a família do protocolo que estamos usando. No arquivo server.py podemos identificar o uso de AF_INET para endereços de rede IPv4 e SOCKET_STREAM protocolo TCP na camada de transporte.

```
socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

- Bind: realiza a associação entre a estrutura socket e o endereço/porta do servidor.

```
server = "localhost"
port = 8080
s.bind((server, port))
```

- Listen: coloca o socket em operação, neste caso, o servidor entra no estado de aguardar por alguma solicitação de clientes que desejam se comunicar. O método listen geralmente recebe um número como parâmetro, indicando a quantidade de conexões que serão enfileiradas pelo protocolo TCP até que o servidor realize o próximo passo, ou seja o comando accept.

```
s.listen(2)
```

- Accept: esse comando aceita conexões de clientes, retornando uma tupla da conexão realizada. Neste caso a tupla contém o endereço do cliente que solicitou a comunicação, bem como um *File Descriptor* usado para receber e enviar dados para este cliente.

```
conn, addr = s.accept()
```

- Recv, Send: estes comandos são usados para receber e enviar dados, respectivamente. No cliente, na parte que recebe a mensagem do servidor colocamos o recebimento (recv) dentro de uma estrutura de repetição para os dados recebidos do servidor sejam entregues de **4096** bytes.

```
data = conn.recv(4096).decode()

conn.send(str.encode(str(p)))
```

- Close: Este comando encerra uma dada conexão entre o cliente e o servidor.

```
conn.close()
```

Enquanto o servidor atende uma conexão ele fica dedicado a ela. Para evitar isso é possível fazer um passo intermediário para criar um novo processo ou thread para tratar da nova conexão que está chegando. Com isso o processo/thread pai fica somente recebendo as conexões e o processo/thread filho trata das requisições dos clientes. A função Threads permite que a aplicação execute tarefas de forma assíncrona, utilizando da biblioteca nativa do Python (_thread);

4. Maiores dificuldades

Implementar a interface do jogo com pygame; Fazer o cliente interpretar as respostas do servidor de maneira adequada; Garantir que o servidor gerencie de forma correta e diferente para cada cliente; Criar um código de comunicação cliente servidor que possa suprir as necessidades da aplicação; Implementar o Sistema de Logs; Fazer a interface cliente interagir com o servidor;

5. Futuras implementações:

Melhorar a interface do jogo; Adicionar botões para sair e reiniciar o jogo; Adicionar placar; Melhorar o sistema de logs.