Protocolo de um jogo de corrida online - Documentação de código

Willyan P. Bueno , Vitor A. B. Von Gilsa , Gabriel R. Verruck , Sofia M. Port

Instituto Federal de Educação, Ciência e Tecnologia - Campus Concórdia
Distrito Fragosos - s/n Km 8, Concórdia - SC, 89700-000 - Brasil

willyanpaproski123@gmail.com, vitoraipumirim@gmail.com,
verruckgabriel@gmail.com, sofia.manduca.port06092004@gmail.com

1. Resumo

Dada a proposta de desenvolver um sistema que utilize dos protocolos das camadas de aplicação e transporte da rede, este documento tem como objetivo mostrar como foi o processo de elaboração da atividade. Será apresentado as ferramentas utilizadas, métodos, e explicação da funcionalidade.

2. Introdução

Para a realização deste projeto, tivemos como escolha 7 propostas para serem feitas com a linguagem de programação Python:

1. Chat Seguro, utilizando criptografia simétrica;
2. Compartilhamento de Arquivos;
3. Controle de Ar Condicionado via sistema;
4. Streaming de Áudio;
5. Jogo Online de Tiro em Primeira Pessoa;
6. Jogo RPG Online;
7. Jogo Online de Corrida Multiplayer.

Em concordância com o grupo, selecionamos o Jogo Online de Corrida Multiplayer para desenvolver.

3. Ferramentas

Para realizar a proposta escolhida pesquisamos e selecionamos algumas bibliotecas do Python para utilizar no desenvolvimento:

1. **Threading:** Esta biblioteca torna possível a utilização de threads em nosso código. Segundo Developerworks (2012) Threads são fluxos de programas que executam em paralelo dentro de uma aplicação, isto é, uma ramificação de uma parte da aplicação que é executada de forma independente e escalonada independentemente do fluxo inicial da aplicação.
2. **Socket:** De acordo com Machado (2018), Sockets são usados para enviar dados através da rede. Para que isso seja feito, devemos estabelecer um servidor, e os clientes, que vão trocar mensagens através dos protocolos das camadas de aplicação e transporte, sendo eles TCP e UDP.
3. **Time:** Segundo Python (), a biblioteca Time é nativa da linguagem python, e provê várias funções relacionadas a tempo.

4. Estrutura da aplicação

Para construir a aplicação, dividimos ela em servidor e cliente, onde juntos fazem todo o sistema ser executado:

4.1. Servidor

O lado do servidor precisa sempre estar funcionando para os clientes se conectarem e jogarem o jogo de corrida, que foi feito de acordo com as regras de um DragRacing.

Para construção do código do servidor, foram utilizadas as bibliotecas socket e threading. Após importadas as bibliotecas no código, foi criada uma lista vazia que vai receber os clientes, colaborando com que a aplicação torne-se multi-client (mais de um cliente conectado ao servidor).

Depois, foram criadas outras três funções, que em conjunto fazem o servidor funcionar e comunicar-se com os clientes:

1. **Função main():** Primeiramente é definido uma variável que basicamente é o nosso servidor, onde junto da função socket, definimos ele para comunicar-se com endereços do tipo v4, e utilizar o protocolo TCP. Após isso, é criada uma condição onde o servidor tenta estabelecer sua conexão no IP da rede, na porta 10101, e definir para somente permitir a conexão de dois clientes, e por fim, mostrar no terminar a mensagem ”Servidor conectado”. Caso isto não aconteça, é mostrado no terminal a mensagem ”Não foi possível iniciar o servidor”, e a aplicação é encerrada. Então, caso a conexão seja estabelecida na rede, é feito um loop onde o servidor vai sempre estar recebendo e aceitando conexões dos clientes, e adicionando eles à lista que pertence à variável anteriormente declarada. Também é armazenada em uma variável, uma threading para fazer com que a função confirmacao() seja executada junto da função main(), e por fim essa thread é inicializada. Confira no código à seguir:

```
1
    def main():
3
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5
    try:
7
        server.bind(('192.168.0.120', 10101))
9       server.listen(2)
        print("Servidor conectado")
11
    except:
13
        print('Nao foi possivel iniciar o servidor.')
15      exit()
17  while True:
19      cliente, addr = server.accept()
        clientes.append(cliente)
21
```

```
        thread = threading.Thread(target=confirmacao, args=[
            cliente])
        thread.start()
```

2. **Função confirmacao():** Esta função é executada em conjunto da função main() através de uma thread, e recebe como parâmetro a conexão do nosso cliente. Primeiramente é feito um loop, onde o servidor tenta receber uma mensagem do cliente, e estabelece uma condição para caso a mensagem recebida do cliente for do tipo string com o valor ”y”ou ”Y”, que caso for verdadeira, é mostrado no terminal a mensagem ”Confirmação do jogador realizada”, e então o jogo se inicia no lado do cliente. Caso estas etapas não seja realizadas, a função deleteClient() é chamada, o cliente perde sua conexão e a aplicação é encerrada. Confira no código abaixo:

```
    def confirmacao(cliente):

    while True:

        try:

            msgRecv = cliente.recv(2048)
            msg = msgRecv.decode()
            print(msg)

            if(msg == 'y' or msg == 'Y'):

                print('Confirmacao do jogador realizada')

        except:

            deleteClient(cliente)
            exit()
```

3. **Função deleteClient():** É uma função bem simples, que somente deleta um cliente da lista de clientes, fazendo-o perder a conexão com o servidor. Para que o cliente seja removido da lista de clientes, passamos ele como parâmetro da função, e o removemos com a função remove(). Observe no código à seguir:

```
    def deleteClient(cliente):

        clientes.remove(cliente)
```

4. Contudo, o código que faz o lado do servidor funcionar fica da seguinte forma:

```
    import threading
```

```
3     import socket
5     clientes = []
7     def main():
9         server = socket.socket(socket.AF_INET, socket.SOCK_STREAM
              )
11        try:
13            server.bind(('192.168.0.120', 10101))
              server.listen(2)
15            print("Servidor conectado")
17        except:
19            print('Nao foi possivel iniciar o servidor.')
              exit()
21
          while True:
23
              cliente, addr = server.accept()
25            clientes.append(cliente)
27            thread = threading.Thread(target=confirmacao, args=[
                  cliente])
              thread.start()
29
      def confirmacao(cliente):
31
          while True:
33
              try:
35
                  msgRecv = cliente.recv(2048)
37                msg = msgRecv.decode()
                  print(msg)
39
                  if(msg == 'y' or msg == 'Y'):
41
                      print('Confirmacao do jogador realizada')
43
              except:
45
                  deleteClient(cliente)
47                exit()
49    def deleteClient(cliente):
51        clientes.remove(cliente)
53    main()
```

4.2. Cliente

O lado do servidor precisa estar conectado ao servidor para funcionar, e ser possível a execução de um jogo de corrida DragRacing.

Para desenvolver o código do lado do cliente, foram utilizadas as bibliotecas threading, socket e time da linguagem de programação python. O código foi estruturado para funcionar com a execução de quatro funções:

1. **Função main():** Esta função vai realizar a conexão do cliente com o servidor e executar a função dragRace() através de uma thread para que o jogo seja iniciado. Primeiramente, é criada uma variável que vai basicamente ser o nosso cliente, onde junto da função socket, ele é definido para realizar conexões somente com endereços do tipo v4 e utilizar o protocolo TCP para comunicar-se. Após isso, é estabelecido uma condição onde o cliente tenta se conectar com o servidor através do endereço IP da rede e a porta onde o servidor está sendo executado, e mostra no terminal a mensagem ”Cliente conectado”. Caso não seja possível executar estas etapas, é mostrado no terminal a mensagem ”Não foi possível conectar-se ao servidor”e a aplicação do lado do cliente é encerrada. Após o cliente ser conectado, é pedido para que ele digite um nome de usuário para poder distingui-lo de outros clientes. Também é criada e iniciada uma thread para que a função dragRace() seja executada junto da função main(). Confira no código à seguir:

```
    def main():

    cliente = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    try:

        cliente.connect(('192.168.0.120', 10101))
        print("Cliente conectado")

    except:

        print('\nNao foi possivel conectar-se ao servidor.\n')
        exit()

    username = input("Digite seu nickname: ")
    print("\nConectado.")

    thread = threading.Thread(target=dragRace, args=[cliente,
        username])

    thread.start()
```

2. **Função marchas():** Esta função é basicamente o que define as regras do Drag Racing, e faz a interação com o usuário para ele jogar, e recebe como parâmetro o nome de usuário, e o cliente. Primeiramente é definido uma variável que armazena o inicio da corrida, e outra que armazena a marcha que o jogador

inicia, sendo a marcha um. Depois é feito um loop condicional, onde ele executa enquanto a variável marcha for menor do que cinco. Dentro deste loop, é mostrado no terminal a mensagem ”Acelerando”, e o código espera um tempo de o valor da marcha que o jogador está mais três segundos para ir para a próxima etapa. Após este tempo de espera, é mostrado uma mensagem para o jogador pressionar a tecla Q e pressionar Enter para passar a marcha. Quando o jogador realiza as instruções, é feito uma verificação onde é averiguado se o usuário às fez corretamente, e caso o jogador tenha feito certo, é acrescentado um valor a mais na marcha e mostrado no terminal a marcha em que o jogador está. Quando a marcha alcança o valor de cinco, é executado um bloco de código, onde acrescentamos o valor ”6”à variável marcha, e é definido o fim da corrida. Após isso é calculado o tempo que o usuário levou para finalizar a corrida (determinante para saber quem venceu em um Drag Racing), e mostrado no terminal juntamente do nome e o tempo que o jogador levou para alcançar o final da corrida, a mensagem ”Jogador ¡nome do jogador¿ alcançou a linha de chegada em ¡tempo¿ segundos. Também, através da função send(), é enviado ao servidor o resultado da corrida. Veja no código à seguir:

```
2     def marchas(cliente, username):
4     inicioCorrida = round(float(time.time()%60), 2)
6     marcha = 1
8     while marcha < 5:
10        print('Acelerando\n')
          time.sleep(marcha + 3)
12        marchaComando = input('Pressione <Q> e <ENTER> para
              passar a marcha: ')
14        if(marchaComando == 'q' or marchaComando == 'Q'):
16            marcha+=1
              print(f"Marcha {marcha}!")
18
          else:
20
              print('Game Over!')
22            exit()
24        if(marcha == 5):
26            time.sleep(8)
              marcha = 6
28
              fimCorrida = round(float(time.time()%60), 2)
30            resultado = round((fimCorrida - inicioCorrida), 2)
32            chegada  = f'Jogador <{username}> alcancou a linha de
                  chegada em {abs(resultado)} segundos'
```

```
            cliente.send(f'{chegada}'.encode('utf-8'))
34
            print(chegada)
```

3. **Função contagemRegressiva():** Como o próprio nome sugere, esta função realiza uma contagem regressiva. É utilizada no código para fazer a contagem antes do jogo ser iniciado, sendo de cinco segundos. Para que a contagem seja realizada, é definida uma variável que recebe o valor ”6”, e depois é feito um loop condicional onde enquanto a variável anteriormente declarada for maior que um, é mostrado no terminal o valor desta menos 1, o código aguarda um segundo para realizar a próxima etapa, e decresce o valor da variável em um. Confira abaixo:

```
2   def contagemRegressiva():

4   contagem = 6

6   while contagem > 1:

8       print(contagem-1)
        time.sleep(1)
10      contagem-=1
```

4. **Função dragRace():** Esta função executa o nosso jogo junto das outras funções necessárias para que ele funcione, e recebe como parâmetro o cliente e seu nome de usuário. É estabelecido um loop, onde temos uma condicional que tenta executar as etapas do jogo. Primeiramente aparece uma mensagem perguntando se o usuário está pronto. Esta mensagem aguarda receber os valores ”y”e ”Y”, onde se receber o valor esperado, o envia para o servidor, que realiza a confirmação do jogador, em seguida começa a contagem para o jogo funcionar, chamando a função contagemRegressiva(), e por fim executa a função marchas, assim, fazendo o jogo funcionar. Caso isso não aconteça, é mostrado no terminal a mensagem ”Game over!”, e o jogo é encerrado. Observe no código à seguir:

```
1   def dragRace(cliente, username):
3   while True:
5       try:
7           msg = input('Pronto? ')
9           if (msg == 'y') or (msg == 'Y'):
11              cliente.send(f'{msg}'.encode('utf-8'))
13              contagemRegressiva()
                marchas(cliente, username)
15
```

```
            else:
17
                exit()
19
        except:
21
            print('Game Over!')
23          exit()
```

Contudo, o código para o lado do cliente executar o jogo fica da seguinte forma:

```
2  import threading
   import socket
4  import time

6  def main():

8      cliente = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

10     try:

12         cliente.connect(('192.168.0.120', 10101))
           print("Cliente conectado")
14
       except:
16
           print('\nNao foi possivel conectar-se ao servidor.\n')
18         exit()

20     username = input("Digite seu nickname: ")
       print("\nConectado.")
22
       thread = threading.Thread(target=dragRace, args=[cliente, username
           ])
24
       thread.start()
26
   def marchas(cliente, username):
28
       inicioCorrida = round(float(time.time()%60), 2)
30
       marcha = 1
32
       while marcha < 5:
34
           print('Acelerando\n')
36         time.sleep(marcha + 3)
           marchaComando = input('Pressione <Q> e <ENTER> para passar a
               marcha: ')
38
           if(marchaComando == 'q' or marchaComando == 'Q'):
40
```

```
            marcha+=1
            print(f"Marcha {marcha}!")

        else:

            print('Game Over!')
            exit()

        if(marcha == 5):

            time.sleep(8)
            marcha = 6

            fimCorrida = round(float(time.time()%60), 2)
            resultado = round((fimCorrida - inicioCorrida), 2)

            chegada  = f'Jogador <{username}> alcancou a linha de
                chegada em {abs(resultado)} segundos'
            cliente.send(f'{chegada}'.encode('utf-8'))

            print(chegada)

def contagemRegressiva():

    contagem = 6

    while contagem > 1:

        print(contagem-1)
        time.sleep(1)
        contagem-=1

def dragRace(cliente, username):

    while True:

        try:

            msg = input('Pronto? ')

            if (msg == 'y') or (msg == 'Y'):

                cliente.send(f'{msg}'.encode('utf-8'))
                contagemRegressiva()
                marchas(cliente, username)

            else:

                exit()

        except:

            print('Game Over!')
            exit()

main()
```

Referências

DEVELOPERWORKS, B. **Threads em Python**. 2012. Url https://imasters.com.br/back-end/threads-em-python. Acessado em: 2023/05/21.

MACHADO, A. **Socket em Python**. 2018. Url https://blog.4linux.com.br/socket-em-python/. Acessado em: 2023/05/21.

PYTHON, O. **time - Acesso ao horário e conversões**. Url https://docs.python.org/pt-br/3.9/library/time.html. Acessado em: 2023/05/21.