



# **UNIVERSIDADE FEDERAL DE ALAGOAS**

## **INSTITUTO DE COMPUTAÇÃO**

Caio César Medeiros Lopes  
João Victor Gomes dos Santos  
Matheus Ryan da Silva Nascimento  
Taisa Lima e Silva

### **INTRODUÇÃO**

Este projeto tem como objetivo apresentar a funcionalidade da biblioteca socket na linguagem de programação Python através da transferência de arquivos no modelo cliente-servidor. Foi feito em conjunto pelos estudantes citados acima. A principal função de um socket é fornecer uma interface para a comunicação de dados, permitindo que programas estabeleçam conexões entre duas ou mais máquinas. Já a thread nada mais é do que o processamento paralelo de serviços com o socket, ou seja, ter um sistema que receba mais de duas instâncias simultaneamente. Por meio disso, daremos seguimento ao relatório do projeto.

### **FUNCIONALIDADES**

As threads estão sendo iniciadas no lado servidor da aplicação por meio da função “threading.Thread” que por sua vez cria uma instância da classe Thread, ao fim, é iniciado o funcionamento da thread que permite a execução de processos em paralelo.

As principais funcionalidades do programa são a listagem de arquivos presentes no servidor, além de download e upload de arquivos por meio da conexão cliente-servidor estabelecida pela biblioteca socket:

#### **1. Listagem de arquivos:**

Os clientes podem solicitar a lista de arquivos disponíveis no servidor, no intuito de visualizar os arquivos presentes no mesmo, para download futuro.

#### **2. Download de arquivos:**

No lado cliente da aplicação, é requisitado a opção de download mais o “ID” do respectivo arquivo. No lado servidor é feita a checagem da existência do arquivo, se o mesmo existir, é feita a transferência do arquivo para o cliente.

### 3. Upload de arquivos:

No lado cliente é requisitado o upload de arquivo, em seguida é feita a listagem dos arquivos presentes no cliente, de modo igual a opção de download, é feita também a escolha do “ID” do arquivo, dessa vez dos arquivos do cliente, para que seja possível o envio correto do arquivo desejado.

### 4. Sair:

Envia para o servidor que o cliente está encerrando a conexão, no lado servidor ele encerra a conexão com o cliente, já no lado cliente é feito o encerramento da conexão cliente-servidor.

## **PROTOCOLOS UTILIZADOS**

### 1.IP

O protocolo IP está sendo usado para estabelecer o endereço ipv4 do servidor e do cliente, estabelecendo a conexão.

### 2. TCP/IP

O TCP/IP foi implementado para possibilitar a transmissão dos arquivos nas funções de download e upload (cliente), `receive_files` e `send_files` (servidor).aq

## **POSSÍVEIS ADIÇÕES**

Algumas das possíveis funcionalidades que poderiam vir a ser implementadas visando otimizações seriam:

- Deleção de arquivo tanto do lado cliente como no lado servidor,
- Compactação e descompactação de arquivos para otimizar o tempo de transferência,
- Gerenciamento de armazenamento no lado servidor no intuito de que o mesmo não tenha seu limite excedido
- Encerramento do servidor
- Sistema de criptografia Ponta a Ponta.

## **DIFICULDADES**

Houve uma grande dificuldade inicial com o entendimento e aplicação das threads, juntamente com relativa dificuldade na interpretação das funções de conexão cliente-servidor, uma última dificuldade vista no final da aplicação, foi o encerramento direto do servidor, tendo em vista que na aplicação atual o servidor somente é fechado quando todas as conexões se encerram.

## CÓDIGO FONTE DO PROGRAMA

### SERVIDOR:

```
import socket
import os
import threading

def receive_files(conn):

    # CONFIRMAR ARQUIVO ATRAVÉS DO CLIENTE
    try:
        confirm = conn.recv(4096).rstrip().decode() #RECEBE A
CONFIRMAÇÃO
        file_name = confirm.split(":")[-1] #PEGA O NOME DO
ARQUIVO
        file_size = int(confirm.split(":")[0]) #PEGA O TAMANHO DO
ARQUIVO

        except Exception as error: #EXCEÇÃO CASO O ARQUIVO NÃO
SEJA ENCONTRADO
            print("\nErro na confirmacao:", error)
            return

        aux_size = file_size
        # OCORRÊNCIA DE ARQUIVOS DE MESMO NOME
        aux_name = file_name
        i = 1
        while os.path.exists("../dados/" + aux_name):
            aux_name = '(' + str(i) + ') ' + file_name #ADICIONA
PARÊNTESES
            i += 1

        file_name = aux_name

        # REALIZANDO O DOWNLOAD DO ARQUIVO
        with open("../dados/" + file_name, 'wb') as file:
            while True: #LAÇO PARA A BAIXAR O ARQUIVO
                if aux_size <= 0:
                    break

                try:
                    byte = conn.recv(1024)
                    file.write(byte)
                    aux_size -= len(byte)
```

```

        except Exception as error:
            print("\nErro no download:", error)
            return

    print("\nArquivo adicionado com sucesso!")

def list_files(conn): #FUNÇÃO PARA LISTAR OS ARQUIVOS DENTRO
DO DIRETÓRIO

    files = os.listdir(os.path.join(os.getcwd(), "../dados"))
#LISTA OS ARQUIVOS
    options = "Opções de download:\n"
    i = 1

    for item in files:
        options += str(i) + ". " + item + "\n"
        i += 1

    try:
        conn.sendall(str.encode(options)) #ENVIA A LISTA DE
ARQUIVOS COMO UMA STRING
    except Exception as error:
        print("Erro no envio:", error)

def send_files(conn): #FUNÇÃO PARA ENVIAR OS ARQUIVOS
    files = os.listdir(os.path.join(os.getcwd(), "../dados"))
#LISTA OS ARQUIVOS EM UM VETOR

    # RECEBENDO OPCAO
    try:
        resp_option = int(conn.recv(1024).rstrip().decode())
#RECEBE A OPÇÃO ESCOLHIDA PELO CLIENTE
        file_size = os.path.getsize(
            "../dados/" + files[resp_option-1])
        confirm = str(file_size) + ":" + \
            str(files[resp_option-1])
        conn.sendall(confirm.encode())#ENVIA A CONFIRMAÇÃO DO
ARQUIVO

    except Exception as error:
        print("Erro no recebimento de dados:", error)
        return

    # ENVIANDO ARQUIVO

```

```

    try:
        choosed_file = open(
            "../dados/" + str(files[resp_option-1]), "rb")
#ESCOLHE O ARQUIVO

        while True:
            file_in_bytes = choosed_file.read(1024) #LÊ E ENVIA
CADA BYTE DOS ARQUIVOS
            if len(file_in_bytes) <= 0:
                # FINALIZOU O ARQUIVO
                break
            conn.sendall(file_in_bytes) # ENVIA O BYTE

        except Exception as error:
            print("Erro no envio de dados:", error)

def client_control(conn, client_addr): #MENU

    while True:
        option = conn.recv(1024).decode()

        if option == "upload":
            receive_files(conn)

        elif option == "to_list":
            list_files(conn)

        elif option == "download":
            send_files(conn)

        elif option == "exit":
            conn.close()
            break

def servidor():

    server_addr = ("localhost", 12345)
    server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)

    server_socket.bind(server_addr)
    server_socket.listen(5)

    print("\nEscutando em", server_addr)

```

```

while True:

    # NOVA CONEXAO
    conn, client_addr = server_socket.accept()
    print("\nNova conexao recebida de ", client_addr)

    # ADICIONANDO THREAD
    thread = threading.Thread(
        target=client_control, args=(conn, client_addr))
    thread.start()

    print("Conexoes ativas:", threading.active_count()-1)

if __name__ == "__main__":
    servidor()

```

### **CLIENTE:**

```

import socket
import os

def list_files(conn): #DISPLAY

    print()
    try:
        list_files = str(conn.recv(4096).rstrip().decode())
#LISTA OS ARQUIVOS PARA SEREM RECEBIDOS
        print(list_files)
        input("\nPressione enter...")

    except Exception as error:
        print("\nErro em obter opções:", error)
        return

def upload(conn): #UPLOAD

    # LISTANDO ARQUIVOS DISPONIVEIS PARA ENVIO
    files = os.listdir(os.path.join(os.getcwd(), "../dados
cliente"))

    options = "\nOpções de envio:\n"
    i = 1

```

```

for item in files:
    options += str(i) + ". " + item + "\n"
    i += 1
print(options)

# ESCOLHENDO ARQUIVO A SER ENVIADO
choosed_id = int(input("Digite o ID do arquivo a ser
enviado: "))

# CONFIRMANDO ARQUIVO COM O SERVIDOR
try:
    file_name = files[choosed_id-1] #SELECIONA O ARQUIVO NO
ARRAY DE ARQUIVOS
    file_size = os.path.getsize("../dados cliente/" +
files[choosed_id-1]) #SELECIONA O CAMINHO NO DIRETÓRIO

    confirm = str(file_size) + ":" + str(file_name) #CONFIRMA
O ARQUIVO
    conn.sendall(confirm.encode()) #ENVIA O ARQUIVO

    print("\nEnviando o arquivo:", file_name)

except Exception as error:
    print("Erro na escolha do arquivo:", error)
    return

# ENVIANDO ARQUIVO
try:
    choosed_file = open("../dados cliente/" + file_name,
"rb") #SELECIONA O ARQUIVO PARA SER ENVIADO

    while True:
        file_in_bytes = choosed_file.read(1024) #Lê O
ARQUIVO
        if len(file_in_bytes) <= 0:
            # FINALIZOU O ARQUIVO
            break
        conn.sendall(file_in_bytes) #ENVIA OS BYTES DO
ARQUIVO

    except Exception as error:
        print("\nErro de dados:", error)
        return

    print("\nArquivo enviado!")

```

```

input("Pressione enter para continuar...")

def download(conn):

    # COMANDO
    print()
    choosed_id = input("Digite o ID do arquivo escolhido
(disponiveis na listagem): ")

    # ENVIANDO option
    try:
        conn.send(choosed_id.encode())

    except Exception as error:
        print("\nErro:", error)
        return

    # confirm ARQUIVO
    try:
        confirm = conn.recv(4096).rstrip().decode() #RECEBE A
CONFIRMAÇÃO DE RECEBIMENTO
        file_name = confirm.split(":")[-1] #LÊ O NOME DO ARQUIVO
        file_size = int(confirm.split(":")[0]) #PEGA O TAMANHO DO
ARQUIVO

        print(file_name)

    except Exception as error:
        print("\nErro na confirmação:", error)
        return

    # OCORRÊNCIA DE ARQUIVOS DE MESMO NOME
    aux_name = file_name
    i = 1
    while os.path.exists("../dados cliente/" + aux_name):
        aux_name = '(' + str(i) + ')' + file_name
        i += 1

    file_name = aux_name
    aux_size = file_size

    # BAIXANDO ARQUIVO
    try:
        file = open("../dados cliente/" + file_name, 'wb') #ENTRA
NA PASTA E USA O WB PARA ESCREVER

```



```

except Exception as error:
    print("Erro na obtenção do arquivo:", error)
    return

while True:
    if aux_size <= 0: #ENQUANTO A QUANTIDADE DE BYTES
RECEBIDOS NÃO CHEGAR EM ZERO
        break

    try:
        byte = conn.recv(1024) #RECEBE O BYTE DO SERVIDOR
        file.write(byte) #ESCREVE O ARQUIVO NO DIRETÓRIO
ESCOLHIDO
        aux_size -= len(byte) #DIMINUI A QUANTIDADE DE BYTES

    except Exception as error:
        print("Erro no download:", error)
        return

    print("Arquivo baixado!")

def client():

    addr = ("localhost", 12345)

    server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    server_socket.connect(addr)

    while True:

        # MENU
        print('\n[1] Listar arquivos\n[2] Enviar arquivo\n[3]
Receber arquivo\n[4] Sair\n')
        option = input("Opção: ")

        if option == '1':
            try:
                server_socket.send("to_list".encode())
            except Exception as error:
                print("Erro:", error)
                server_socket.close()
                break

            error = list_files(server_socket)
            if error:
                server_socket.close()

```

```

        break

elif option == '2':
    try:
        server_socket.send("upload".encode())
    except Exception as error:
        print("Erro:", error)
        server_socket.close()
        break

    error = upload(server_socket)
    if error:
        server_socket.close()
        break

elif option == '3':
    try:
        server_socket.send("download".encode())
    except Exception as error:
        print("Erro:", error)
        server_socket.close()
        break

    error = download(server_socket)
    if error:
        server_socket.close()
        break

elif option == '4':
    try:
        server_socket.send("exit".encode())
    except Exception as error:
        print("Erro:", error)

    server_socket.close()
    break
else:
    print('Opção invalida!')

if __name__ == "__main__":
    client()

```

**Link para o repositório no github:**

<https://github.com/TaisaLima/Projeto-de-redes-2023.1>