
UNIVERSIDADE FEDERAL
DE ALAGOAS

**UNIVERSIDADE FEDERAL DE ALAGOAS - UFAL**

**INSTITUTO DE COMPUTAÇÃO - IC**

**CURSO DE CIÊNCIAS DA COMPUTAÇÃO**

**GUSTAVO RIBEIRO FRANCO GAIA**

**KAROLAINE LIMA SANTOS**

**LARA VITÓRIA SILVA SANTOS BARROS**

# REDES DE COMPUTADORES

**MACEIÓ - AL**

**2022**

GUSTAVO RIBEIRO FRANCO GAIA
KAROLAINE LIMA SANTOS
LARA VITÓRIA SILVA SANTOS BARROS

# PROJETO REDES DE COMPUTADORES
# APLICAÇÃO DE SOCKET

Trabalho apresentado para a Disciplina Redes de computadores, pelo Curso de Ciências da computação da Universidade Federal de Alagoas (UFAL), ministrada pelo Prof. ALMIR PEREIRA GUIMARÃES

Maceió
2022

**SUMÁRIO**

## 1 INTRODUÇÃO

A seguir será apresentado um relatório abordando as principais funcionalidades da aplicação, construída em python, de um sistema de rede com a implementação de SOCKET e THREAD.

O sistema se utiliza de uma conexão Servidor-Cliente do tipo TCP/IP, e tem como propósito simular a aplicação de um grupo de mensagens que facilitem a comunicação de diversos usuários simultaneamente, como grupos em redes sociais.

## 2 DESENVOLVIMENTO

### 1.Funcionalidades

As principais funcionalidade desenvolvidas na aplicação giram em torno da implementação de um grupo de conexões SOCKET, de diversos usuários(clientes), a um servidor especifico, que, através da implementação de THREAD, é capaz de realizar novas conexões enquanto recebe e transmite dados entre os usuários.

A aplicação simula um grupo de conversas, e permite aos usuários trocar mensagens initerruptamente de forma simultânea, realizando a conexão do tipo TCP/IP de forma a garantir a entrega de todos os dados integralmente.

Uma vez realizada a conexão com o servidor, este armazena as informações de cada cliente em um mapa de conexões, que é utilizado para transmitir os dados enviados por um cliente a todos os outros presentes no suposto grupo. Todos os usuários conectados são informados quando uma mensagem é enviada, quem a enviou, quem saiu ou quem entrou no grupo, ou seja, quem realizou uma nova conexão ou desfez uma já existente. Quando um usuário opta por abandonar o grupo seus dados são removidos do mapa de conexões existentes no servidor, e a aplicação daquele cliente específica e encerrada.

Dados importantes como a PORTA de conexão para o socket, IPV4 do servidor, tamanho de cada mensagem e endereço de cada cliente é utilizada para identificar e realizar troca de dados com eficiência. A função THREAD não é apenas utilizada pelo servidor, sendo também implementada na aplicação do cliente para permitir que este seja capaz de transmitir mensagens enquanto, simultaneamente, as recebe.

### 2.O que poderia ser implementado?

Após considerar diversos fatores, deixando de lado a interface da aplicação e observando funcionalidades comuns em aplicações de troca de mensagens, chegamos à conclusão que implementações como mensagens privadas, de cliente a cliente específicos a partir da utilização de endereços, poderiam ser opções interessantes a se adicionar na aplicação.

## 3.Dificuldades encontradas

As principais dificuldades que tivemos no decorrer da aplicação foi como realizar um controle dos clientes que se mantinham ou não conectados ao servidor, e como lidar com os dados que eles deixariam registrados ao abandonar a “conversa”, assim como a forma de gerenciamento dos usuários e assegurar que novas conexões estavam de fato sendo realizadas. A solução para os dois problemas foi a implementação do mapa de conexões que nos permitiu manter um certo controle sobre dados de usuários que estavam ou não ativos no servidor.

## CÓDIGO SERVIDOR

servidor.py > end_conn

```python
from ctypes import sizeof
import socket
import threading
import time
import sys

SERVER_IP = socket.gethostbyname(socket.gethostname())
PORT = 5050
ADDR = (SERVER_IP, PORT)
FORMATO = 'utf-8'

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDR)

conexoes = []

def enviar_mensagem_individual(conexao, mensagem):
    conexao['conn'].send(mensagem.encode())
    time.sleep(0.2)

def end_conn(addr):
    global conexoes
    print(f"[DESCONECTANDO] PORT_SERVER: {PORT}, PORT_CLIENT: {addr[1]}")
    for conexao in conexoes:
        if(addr[1] == conexao["addr"][1]):
            enviar_mensagem_individual(conexao, "Você saiu!")
            conexoes.remove(conexao)


def enviar_mensagem_todos(addr, mensagem, size):
    global conexoes
    print(f"[ENVIANDO] Enviando mensagens. Emissor : {{[CLIENT_IP : {addr[0]}] [CLIENT_PORT : {addr[1]}]}} Bytes: {size}")
    for conexao in conexoes:
        if(addr[1] != conexao["addr"][1]):
            enviar_mensagem_individual(conexao, mensagem)

def handle_clientes(conn, addr):
    print(f"[NOVA CONEXAO] PORT_SERVER: {PORT}, PORT_CLIENT: {addr[1]}")
    global conexoes
    nome = False

    while(True):
        msg = conn.recv(1024).decode(FORMATO)
        if(msg):
            if(msg.startswith("nome=")):
                mensagem_separada = msg.split("=")
                nome = mensagem_separada[1]
                mapa_da_conexao = {
                    "conn": conn,
                    "addr": addr,
                    "nome": nome,
                }
                conexoes.append(mapa_da_conexao)
```

```
54
55              elif(msg.startswith("msg=")):
56                  mensagem_separada = msg.split("=")
57                  mensagem = nome + "=" + mensagem_separada[1]  # type: ignore
58                  size = sys.getsizeof(mensagem_separada[1])
59                  enviar_mensagem_todos(addr, mensagem, size)
60
61              elif (msg == "close"):
62                  end_conn(addr)
63                  break
64
65
66
67
68  def start():
69      print("
70      server.listen()
71      while(True):
72          conn, addr = server.accept()
73          thread = threading.Thread(target=handle_clientes, args=(conn, addr))
74          thread.start()
75
76  start()
```

(method) listen: (__backlog: int = ..., /) -> None

**CÓDIGO CLIENTE**

cliente.py > handle_mensagens

```python
from glob import glob
from pyclbr import Class
import socket
import threading
import time

PORT = 5050
FORMATO = 'utf-8'
SERVER = "192.168.0.2"
ADDR = (SERVER, PORT)

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(ADDR)

def handle_mensagens():
    while(True):
        msg = client.recv(1024).decode()
        if(msg == "Você saiu!"):
            print(msg)
            break
        else:
            mensagem_splitada = msg.split("=")
            print(mensagem_splitada[0] + ": " + mensagem_splitada[1])

def enviar_mensagem(mensagem):
        client.send(mensagem.encode(FORMATO))

def iniciar():
    thread1 = threading.Thread(target=handle_mensagens)
    thread1.start()
    while(True):
        mensagem = input()
        if(mensagem == ""):
            enviar_mensagem("msg=" + "Saiu!")
            enviar_mensagem("close")
            break
        enviar_mensagem("msg=" + mensagem)


nome = input('Digite seu nome: ')
enviar_mensagem("nome=" + nome)
enviar_mensagem("msg=" + "Entrou no grupo!")


iniciar()
client.close
```