UNIVERSIDADE FEDERAL DE ALAGOAS
CAMPUS A. C. SIMÕES
INSTITUTO DE COMPUTAÇÃO

Thyago Viana Pereira
Andrey Noewertton Ferreira da Silva
Felipe Gabriel Marques dos Santos
Manasés Lindolfo Ferreira Barros

PROJETO DA DISCIPLINA REDES DE COMPUTADORES

MACEIÓ - AL
2024

Thyago Viana Pereira
Andrey Noewertton Ferreira da Silva
Felipe Gabriel Marques dos Santos
Manasés Lindolfo Ferreira Barros

CHAT DE BATE-PAPO: SIMULAÇÃO

Trabalho de Redes de Computadores apresentado no Instituto de Computação da Universidade Federal de Alagoas, como requisito da AV2 para a disciplina supracitada.

Responsável pela disciplina:
Prof. Dr. Almir Pereira Guimarães.

MACEIÓ - AL
2024

SUMÁRIO

1 INTRODUÇÃO

O projeto visa demonstrar a implementação e aplicação de um sistema de rede usando *socket* e *thread*. Neste projeto foi desenvolvido um servidor-cliente TCP/IP que tem por objetivo permitir a troca de mensagens entre diferentes usuários, em diferentes dispositivos.

2 FUNCIONALIDADES

As funcionalidades da aplicação giram em torno dos conceitos de **socket** e ***thread*:** com o primeiro (socket) viabilizamos ponto de comunicação entre diferentes dispositivos através de conexão socket e com o segundo (thread) possibilitasse que novas conexões sejam estabelecidas sem interromper o fluxo de dados entre os usuários(clientes).

- O servidor se instala numa porta e espera alguém se conectar;
- O cliente se conecta usando IP + Porta;
- Conexão estabelecida, é solicitado que o usuário digite seu username;
- Uma vez escolhido um username válido a aplicação de mensageria é iniciada;
- Encerra-se o chat digitando */close* no terminal.

A aplicação simula um sistema de mensageria onde diferentes usuários podem se comunicar de forma ininterrupta, simultânea e sem perda de dados.

3 O QUE PODERIA SER IMPLEMENTADO A MAIS?

- Troca de mensagens entre usuários específicos de modo privado;
- Apagar mensagens enviadas;
- Remover usuário do servidor;
- Criação de uma interface.

4 DIFICULDADES ENCONTRADAS

A principal dificuldade foi na conexão entre máquinas diferentes na mesma rede e em redes diferentes; posteriormente foi possível fazer essa integração configurando um encaminhamento de porta no roteador e uma regra de entrada no firewall com a porta sendo usada. Um outro problema também foi o gerenciamento das conexões por parte do cliente que iam se estabelecendo no servidor, em manter essas conexões ativas e encerrá-las sem afetar as demais conexões.

5 CÓDIGO FONTE

CLIENTE

```
#!/usr/bin/env python3

import socket
import threading
import time
import sys
import os

#########################################################################
#                     Defining functions

import os

def clear_terminal():
    os.system('cls' if os.name == 'nt' else 'clear')

def get_nickname():
    """
            gets into a loop until it gets a valid nickname and then
returns it
    """
    while True:
        nickname = input("nickname: ")
        if (3 <= len(nickname) <= 10):
            return nickname
        else:
            clear_terminal()
            print("The nickname is invalid. try again.\n")
```

```
            time.sleep(2)
            clear_terminal()


def nickname_setter():
    global stopping_thread
    global client_socket



    while not stopping_thread:
              nickname = print("Enter a nicknmame between 3 and 10
characters:\n")
        try:
            nickname = get_nickname()
            client_socket.send(f"{nickname}".encode("utf-8"))

            response = client_socket.recv(1024).decode("utf-8")

            if (response == "FALSE"):
                clear_terminal()
                print("This nickname already exists. Try another one!")
                time.sleep(2)
                clear_terminal()
            else:
                clear_terminal()
                break
        except:
                  print("An unexpected error has occurred. Connection
closed!")
            client_socket.close()
            stopping_thread = True
            break


def message_send():
    global stopping_thread
    while not (stopping_thread):
        print("\n")
        message = input()
        try:
            client_socket.send(message.encode("utf-8"))
```

```
                                if (message.lower() == r"/close" or
message.lower().startswith(r"/close")):
                    stopping_thread = True
                    break
            except:
                            print("An unexpect error has occured. Closing
connection!\n")
                client_socket.close()
                stopping_thread = True


def message_recv():
    global stopping_thread
    while not (stopping_thread):
        try:
            message = client_socket.recv(2048).decode("utf-8")

            print(f"{message}\n")
        except:
                            print("An unexpect error has occured. Closing
connection!\n")
                client_socket.close()
                stopping_thread = True

#########################################################################
#########################################################################









"""
    set the port number and the server address
"""
server_address = "192.168.0.110"
server_port = 12345
stopping_thread = False

"""
    Create the client socket and connects it to the server
"""
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
try:
    client_socket.connect((server_address, server_port))
    print(f"Connected to {server_address} : {server_port}\n")
except socket.error as msg:
    print(f"Connection error. {msg}\n")
    sys.exit()

nickname_setter()
msg = client_socket.recv(1024).decode("utf-8")
print(f"{msg} Type '/close' to leave the chat.\n")
rcv_thread = threading.Thread(target=message_recv)
send_thread = threading.Thread(target=message_send)

rcv_thread.start()
send_thread.start()


while True:
    if not (stopping_thread):
        pass
    else:
        client_socket.close()
        sys.exit(
            "Connection closed!\n"
        )
```

SERVIDOR

```
#!/usr/bin/env python3

import socket
import threading
import os

#####################################################################
#                     Defining functions

def clear_terminal():
    os.system('cls' if os.name == 'nt' else 'clear')

def broadcast_msg(message, clientSender):
    """
         send the message received from clientSender to all the clients
on the server
        except for the clientSender himself
    """
    for client in clients_sockets:
        if (client != clientSender):
            client.send(message.encode("utf-8"))


def recv_message(client):

    """
        recieve the message from anyone who is in the chat,
          verify if it's the key message to exit the chat and if not,
send it to everybody
        if it is, remove the user from the chat group
    """
    global stop_threading
```

```
    while (not stop_threading):
        # client_indexing = clients_sockets.index(client)
        # nickname = clients_nicknames[client_indexing]
        try:
            message = client.recv(2048).decode("utf-8")

            client_indexing = clients_sockets.index(client)
            nickname = clients_nicknames[client_indexing]

            if message == "/close":

                # client_indexing = clients_sockets.index(client)
                # nickname = clients_nicknames[client_indexing]

                   print(f"{clients_addresses[client_indexing]} has been
disconnected\n")

clients_addresses.remove(clients_addresses[client_indexing])
                clients_sockets.remove(client)
                clients_nicknames.remove(nickname)

                msg = (f"{nickname} disconnected!")

                broadcast_msg(msg, client)
                client.close()
                stop_threading = True

                break
            else:
                msg = f"{nickname}: {message}\n"
                broadcast_msg(msg, client)
        except:

            client_indexing = clients_sockets.index(client)
            nickname = clients_nicknames[client_indexing]

                print(f"An unexpect error has occured at the rcv msg!
Closing connection at {clients_addresses[client_indexing]}")
            clients_sockets.remove(client)
            clients_nicknames.remove(nickname)

clients_addresses.remove(clients_addresses[client_indexing])
            broadcast_msg(f"{nickname} has been disconnected")
```

```
            client.close()
            stop_threading = True


#########################################################################




"""
     define the port and gthe host ip to send it to any connection in
the network
"""

port = 12345
host = "0.0.0.0"

"""
     Defnine 4 lists to be managed by the threading and a flag to stop
some processes
"""
clients_sockets = []
clients_nicknames = []
clients_addresses = []
nicknames = []

stop_threading = False


"""
     creates the server socket and binds it to the host and the port
previously defined
"""
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind((host, port))
server_socket.listen()
print(f"server listening at port: {port}...")


while True:
    client, address = server_socket.accept()
    print(f"A connection has been established with {str(address)}\n\n")
```

```
    while True:
        try:
            nickname = client.recv(1024).decode("utf-8")

            if (nickname in clients_nicknames):
                client.send("FALSE".encode("utf-8"))
            else:
                client.send("TRUE".encode("utf-8"))
                break
        except:
            print("An unexpect error has occured! Closing connection")
            client.close()
    if (clients_sockets):
        msg = f"{nickname} joined the chat group!\n"
        broadcast_msg(msg, client)
                client.send(f"Welcome     to     the     group     chat
{nickname}!".encode("utf-8"))



    clients_nicknames.append(nickname)
    clients_addresses.append(address)
    clients_sockets.append(client)

              client_thread    =    threading.Thread(target=recv_message,
args=[client])
    client_thread.start()
```