UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

Trabalho Socket

(Redes de Computadores)

Lucas Vinícius de Carvalho Ikeda

1.1 – O que é um socket

Um socket é um ponto final de comunicação bidirecional em uma rede, que pode ser usado para enviar e receber dados entre diferentes dispositivos. Ele permite a comunicação entre processos em máquinas diferentes, seja na mesma rede local ou através da internet. Um socket é caracterizado por um endereço IP, um número de porta e um protocolo de transporte, como TCP ou UDP. Ele fornece uma interface de programação para que os aplicativos possam estabelecer conexões de rede, enviar e receber dados, e encerrar as conexões quando necessário. Os sockets são amplamente utilizados em aplicações de rede, como servidores web, jogos online, chat e transferência de arquivos.

1.2 Código fonte e explicação do código

TCP Cliente

```python
import socket

# Configurações do cliente
host = socket.gethostname()
port = 12345

# Cria um socket TCP
tcp_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcp_socket.connect((host, port))

while True:
    # Solicita mensagem do usuário
    message = input('Digite uma mensagem (ou "exit" para sair): ')

    # Envia mensagem para o servidor
    tcp_socket.send(message.encode())

    if message == 'exit':
        break

    # Recebe resposta do servidor
    response = tcp_socket.recv(1024)
    print('Resposta do servidor:', response.decode())

# Fecha o socket
tcp_socket.close()
```

TCP Servidor

```python
import socket

# Configurações do servidor
host = socket.gethostname()
port = 12345

# Cria um socket TCP
tcp_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcp_socket.bind((host, port))
tcp_socket.listen(1)
print('Servidor TCP esperando por conexões...')

# Aceita a conexão do cliente TCP
conn, addr = tcp_socket.accept()
print('Conexão TCP estabelecida com o cliente:', addr)

while True:
    # Recebe mensagem do cliente
    data = conn.recv(1024).decode()
    print('Mensagem recebida do cliente:', data)

    # Responde ao cliente
    response = 'Servidor TCP recebeu a mensagem: ' + data
    conn.send(response.encode())

    # Encerra a conexão se a mensagem for 'exit'
    if data == 'exit':
        break

# Fecha o socket
tcp_socket.close()
```

UDP Cliente

```python
import socket

# Configurações do cliente
host = socket.gethostname()
port = 12345

# Cria um socket TCP
tcp_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcp_socket.connect((host, port))
```

```
while True:
    # Solicita mensagem do usuário
    message = input('Digite uma mensagem (ou "exit" para sair): ')

    # Envia mensagem para o servidor
    tcp_socket.send(message.encode())

    if message == 'exit':
        break

    # Recebe resposta do servidor
    response = tcp_socket.recv(1024)
    print('Resposta do servidor:', response.decode())

# Fecha o socket
tcp_socket.close()
```

UDP Servidor

```
import socket

# Configurações do servidor
host = 'localhost'
port = 12345

# Cria um socket UDP
udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
udp_socket.bind((host, port))
print('Servidor UDP esperando por pacotes...')

while True:
    # Recebe pacote e endereço do cliente
    data, addr = udp_socket.recvfrom(1024)
    print('Pacote recebido do cliente:', data.decode())

    # Responde ao cliente
    response = 'Servidor UDP recebeu o pacote: ' + data.decode()
    udp_socket.sendto(response.encode(), addr)

    # Encerra a conexão se o pacote for 'exit'
    if data.decode() == 'exit':
        break

# Fecha o socket
udp_socket.close()
```

Híbrido Cliente

```python
import socket

# Configurações do cliente TCP
tcp_host = socket.gethostname()
tcp_port = 12345

# Configurações do cliente UDP
udp_host = socket.gethostname()
udp_port = 54321

# Criação do socket TCP
tcp_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcp_socket.connect((tcp_host, tcp_port))

# Criação do socket UDP
udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

while True:
    # Solicita mensagem do usuário
    message = input('Digite uma mensagem (ou "exit" para sair): ')

    # Envia mensagem TCP para o servidor
    tcp_socket.send(message.encode())

    if message == 'exit':
        break

    # Recebe resposta TCP do servidor
    tcp_response = tcp_socket.recv(1024)
    print('Resposta do servidor TCP:', tcp_response.decode())

    # Envia mensagem UDP para o servidor
    udp_socket.sendto(message.encode(), (udp_host, udp_port))

    # Recebe resposta UDP do servidor
    udp_response, udp_addr = udp_socket.recvfrom(1024)
    print('Resposta do servidor UDP:', udp_response.decode())

    # Encerra a conexão se a mensagem for 'exit'
    if message == 'exit':
        break

# Fecha os sockets
tcp_socket.close()
udp_socket.close()
```

UDP Servidor

```
import socket

# Configurações do servidor TCP
tcp_host = socket.gethostname()
tcp_port = 12345

# Configurações do servidor UDP
udp_host = socket.gethostname()
udp_port = 54321

# Criação do socket TCP
tcp_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcp_socket.bind((tcp_host, tcp_port))
tcp_socket.listen(1)

print('Servidor TCP esperando por conexões...')

# Aceita a conexão do cliente TCP
conn, addr = tcp_socket.accept()
print('Conexão TCP estabelecida com o cliente:', addr)

# Criação do socket UDP
udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
udp_socket.bind((udp_host, udp_port))

while True:
    # Recebe mensagem do cliente TCP
    tcp_data = conn.recv(1024).decode()
    print('Mensagem TCP recebida do cliente:', tcp_data)

    # Responde ao cliente TCP
    tcp_response = 'Servidor TCP recebeu a mensagem: ' + tcp_data
    conn.send(tcp_response.encode())

    # Encerra a conexão se a mensagem for 'exit'
    if tcp_data == 'exit':
        break

    # Recebe mensagem do cliente UDP
    udp_data, udp_addr = udp_socket.recvfrom(1024)
    print('Mensagem UDP recebida do cliente:', udp_data.decode())

    # Responde ao cliente UDP
    udp_response = 'Servidor UDP recebeu a mensagem: ' +
udp_data.decode()
    udp_socket.sendto(udp_response.encode(), udp_addr)
```

```
    # Encerra a conexão se a mensagem for 'exit'
    if udp_data.decode() == 'exit':
        break

# Fecha os sockets
tcp_socket.close()
udp_socket.close()
```