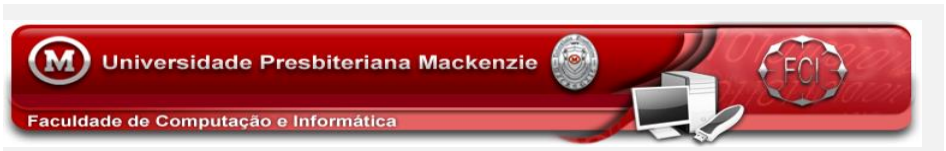

Redes de computares

Prof. Dr. Bruno da Silva Rodrigues

Bruno.rodrigues@mackenzie.br

Analise Programação de Socket UDP e TCP

Procedimento

- *Execute a IDLE do Python e abra os arquivos (Cliente e servidor) do socket UDP*
- *Cada arquivo deve ser aberto em uma IDLE diferente ou em computadores diferentes.*
- *Faça o mesmo procedimento para os arquivos socket TCP*

Após abrir o arquivo analise os pacotes e responda:

Questão 1*. Compile dos programas (cliente e servidor) TCP e UDP:*

a) *Execute o cliente TCP antes de executar o servidor TCP. O que acontece? Por quê?*

Ao executar o cliente TCP antes do servidor, um erro é mostrado pois o cliente não consegue abrir uma conexão.

b) *Faça o mesmo procedimento para o cliente e servidor UDP. O resultado foi similar ao socket TCP? Compare os resultados e justifique.*

Já o cliente UDP não mostra nenhum erro pois ele não abre uma conexão com o servidor, somente envia a mensagem.

c) *O que acontece se o número da porta que o cliente tentar se conectar for diferente da porta disponibilizada pelo servidor?*

Para o cliente TCP será mostrado um erro pois não será possível conectar ou criar conexão. Já para o cliente UDP não será mostrado nenhum erro, pois esse só envia, sem precisar de conexão.

Objetivos da atividade:

-Aprender a programar e Socket UDP e TCP assim como analisar e comparar o funcionamento de ambos protocolos da camada de transporte

Bibliografias

KUROSE, J. F. e ROSS, K. W. Redes de Computadores e a Internet – Uma Nova Abordagem – Pearson

Deitel, Harvey M.; Deitel, Paul J. - Java: como programar - 6ª edição - Pearson

Internet Engineering Task Force. Disponível em: https://www.ietf.org/t

Questão 2. *Faça um chat entre cliente servidor (UDP ou TCP) onde ambos os lados trocam mensagens até uma das partes enviar o comando QUIT.* ***A porta do socket deve ser os primeiros 5 números do seu TIA.***

Servidor:

servidor.py > ...

```python
import socket

host = '127.0.0.1'
port = 32141

server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind((host, port))

print("Servidor iniciado.")

while True:
    data, addr = server_socket.recvfrom(1024)
    message = data.decode()
    print("Mensagem recebida de {}: {}".format(addr, message))

    if message.upper() == "QUIT":
        print("Encerrando servidor.")
        break

    response = input("Digite sua mensagem: ")
    server_socket.sendto(response.encode(), addr)

server_socket.close()
```

Cliente:

```python
# cliente.py
import socket

host = '127.0.0.1'
port = 32141

client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

while True:
    message = input("Digite sua mensagem: ")
    client_socket.sendto(message.encode(), (host, port))

    if message.upper() == "QUIT":
        print("Encerrando cliente.")
        break

    data, _ = client_socket.recvfrom(1024)
    response = data.decode()
    print("Resposta do servidor: {}".format(response))

client_socket.close()
```

Questão 3. *Faça uma aplicação qualquer usando o socket. Essa aplicação pode ser para enviar arquivos ou controlar algum objeto em uma das pontas da conexão ou gerenciar diversas conexões usado threads. Após elaboração do projeto, um vídeo deve ser gravado mostrando o funcionamento da aplicação e explicando o código.*

Tratar arquivos

Se a opção escolhida for manipular arquivos, você deverá fazer um chat e oferecer um menu para transferência de arquivos. O programa deverá dar ao usuário a opção de escolher qual arquivo será enviado e transmitir o arquivo via socket.

****será necessário estudar manipulação de arquivos. Não serão aceitos projetos onde o arquivo a ser enviado já é predefinido no código.***

Tratar várias conexões

Neste desafio, o programa será capaz de tratar e responder diversas conexões usando threads

*será necessário estudar threads.

Outras interações

*Você pode usar seu projeto ou conceitos de **Pygame** usado em semestres anteriores e controlar objetos usando um computador remoto através do socket.*

O projeto em questão foi desenvolvido em JavaScript utilizando socket.io

O Socket.io é uma biblioteca para comunicação em tempo real entre cliente e servidor. Ele usa principalmente WebSockets para comunicação, mas também pode usar outras técnicas de transporte como Long Polling, Flash Sockets etc. como fallback, caso WebSockets não estejam disponíveis.

O Socket.io fornece uma API de alto nível e abstrai muitos detalhes da comunicação em tempo real, como reconexão automática, fallback para outros métodos de transporte, tratamento de erros etc. Isso torna mais fácil para os desenvolvedores se concentrarem na lógica da aplicação.

A ideia de utilizar JavaScript é uma boa para que ninguém precise instalar nada uma vez que esse é interpretado pelo próprio navegador. O projeto é desenvolvido em torno de um servidor que gerencia os usuários, estados e salas além de fornecer o html e js necessário quando os clientes conectam no root do website.

A ideia é que diversas pessoas se juntem em uma "sala do game" com os amigos através de um código que é fornecido pelo servidor ao criar uma sala. Com esse código os amigos podem entrar na sala e assim começar o jogo! O último que sobreviver vence.

Vídeo demonstração do jogo:

https://drive.google.com/file/d/1G1PeO9VG7bsjj9QfMho4BWcjRmJsXwcV/view?usp=share_link

Código fonte:

https://github.com/Schwarzam/js-game

Notas

Questões 1 e 2 da atividade (CHAT simples) – 4,0 pontos

Questão 3 Desafio – 5,0 pontos

Apresentação do projeto – 1,0 ponto

Obs. Mesmo você não conseguindo fazer o exercício 3, você deverá apresentar a ideia para os colegas contando as dificuldades enfrentadas no desenvolvimento da atividade.