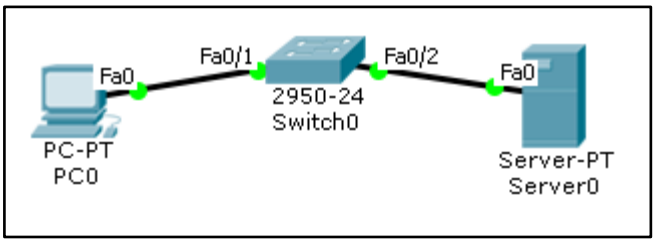
Laboratório 2 – Socket

OBJETIVO

Analisar os sockets TCP e UDP.

CENÁRIO

Composto de um cliente e um servidor.



RECURSOS

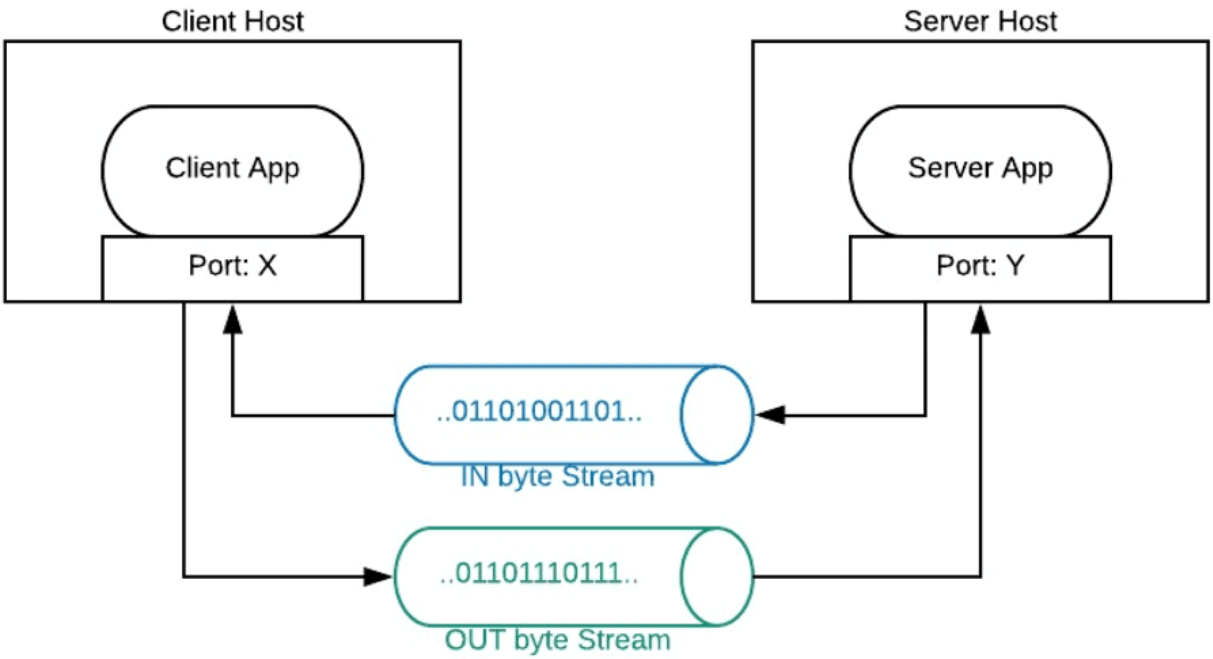
Arquivos de programa Client e Server escritos em Python e/ou Java.

INTRODUÇÃO

Socket é uma interface de programação que fornece as rotinas necessárias para comunicação dos processos de um aplicativo, seja em um sistema local ou em um ambiente de rede distribuído baseado em TCP/IP.

Depois que uma conexão ponto a ponto é estabelecida, um descritor de soquete é usado para identificar exclusivamente a conexão.

O próprio descritor de soquete é um valor numérico específico da tarefa, também conhecido como porta.



Socket

EXERCÍCIOS

PARTE I

Para testar a comunicação de processos por meio de *sockets*[1], executar o seguinte procedimento:

- Executar a IDLE do Python e abrir os arquivos (Client e Server) do socket UDP;
- Cada arquivo deve ser aberto em uma IDLE diferente ou em computadores diferentes;
- Fazer o mesmo procedimento para os arquivos socket TCP.

Após analisar o comportamento da comunicação de processos, responda às seguintes perguntas:

1. Execute o cliente TCP antes de executar o servidor TCP. O que acontece? Por quê? [1,0 ponto]

Resposta:

2. Faça o mesmo procedimento para o cliente e servidor UDP. O resultado foi similar ao socket TCP? Compare os resultados e justifique. [2,0 pontos]

Resposta:

3. O que acontece se o número da porta que o cliente tentar se conectar for diferente da porta disponibilizada pelo servidor? [1,0 ponto]

Resposta:

PARTE II

4. Faça um chat entre cliente e 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 cinco números do TIA do primeiro aluno do grupo (em ordem alfabética). [6,0 pontos]

Código do servidor [3,0 pontos]:

[1] Requer Apêndice ou Arquivos de programa Client e Server.

Código do cliente [3,0 pontos]:

APÊNDICE

Arquivo TCP Server em Python:

```
import socket #importa modulo socket

TCP_IP = '192.168.0.3' # endereço IP do servidor
TCP_PORTA = 24000       # porta disponibilizada pelo servidor
TAMANHO_BUFFER = 1024     # definição do tamanho do buffer

# Criação de socket TCP
# SOCK_STREAM, indica que será TCP.
servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# IP e porta que o servidor deve aguardar a conexão
servidor.bind((TCP_IP, TCP_PORTA))

#Define o limite de conexões.
servidor.listen(1)

print("Servidor dispoivel na porta 5005 e escutando.....")
# Aceita conexão
conn, addr = servidor.accept()
print ('Endereço conectado:', addr)
while 1:
    #dados retidados da mensagem recebida
    data = conn.recv(TAMANHO_BUFFER)
    if data:
        print ("Mensagem recebida:", data)
        conn.send(data.upper())  # envia dados recebidos em letra maiuscula
```

Arquivo TCP Client em Python:

```
import socket #importa modulo socket

TCP_IP = '192.168.0.3' # endereço IP do servidor
TCP_PORTA = 24000      # porta disponibilizada pelo servidor
TAMANHO_BUFFER = 1024

MENSAGEM  = input("Digite sua mensagem para o servidor: ")

# Criação de socket TCP do cliente
cliente = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Conecta ao servidor em IP e porta especifica
cliente.connect((TCP_IP, TCP_PORTA))

# envia mensagem para servidor
cliente.send(MENSAGEM.encode('UTF-8'))

# recebe dados do servidor
data, addr = cliente.recvfrom(1024)

# fecha conexão com servidor
cliente.close()

print ("received data:", data)
```

Arquivo UDP Server em Python:

```
import socket

IP_servidor = "192.168.0.3" #endereço onde o Server será executado
PORTA_servidor = 5005       #porta aberta pelo Server para conexão

# Criação de socket UDP
# Argumentos, AF_INET que declara a família do protocolo; se fosse um envio
via Bluetooth usariamos AF_BLUETOOTH
# SOCK_DGRAM, indica que será UDP.
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)

# IP e porta que o servidor deve aguardar a conexão
sock.bind((IP_servidor, PORTA_servidor))

while True:
    # Recebe mensagem via socket sock.recvform
    # aloca 1024 bytes
    #separa dados e armazena em data e o endereço de origem e guarda em addr
    data, addr = sock.recvfrom(1024)
    #imprime endereço do cliente
    print("Mensagem recebida de : ",addr)
    #exibe texto enviado pelo cliente
    print ("Mensagem recebida:", data)
```

Arquivo UDP Client em Python:

```
import socket  #importa modulo socket

IP_destino = "192.168.0.3"  #Endereço IP do servidor
PORTA_destino = 5005          #Numero de porta do servidor
MENSAGEM = "Hello, World!"

print ("Endereço IP de destino:", IP_destino)
print ("Porta UDP de destino:", PORTA_destino)
print ("Mensagem enviada:", MENSAGEM)

#Criação de socket UDP
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)

#Envia mensagem usando socket UDP
sock.sendto(MENSAGEM.encode('UTF-8'), (IP_destino, PORTA_destino))
```