Pontifícia Universidade Católica do Paraná
Escola Politécnica
Bacharelado em Engenharia de Software

Disciplina: Conectividade de Sistemas Ciberfísicos
Professor: Guilherme Schnirmann
Curso: BES
Nome Estudante: Diogo Bonet Sobezak e Vittorio Caprioli

Atividade Prática / Relatório

Socket API (UDP)

Descrição da Atividade:

Esta atividade consiste em implementar em Python a comunicação UDP utilizando a interface socket. O intuito desta atividade é demostrar o funcionamento do protocolo UDP, bem como as portas são mapeadas aos processos por meio da interface sockets.

Entrega:

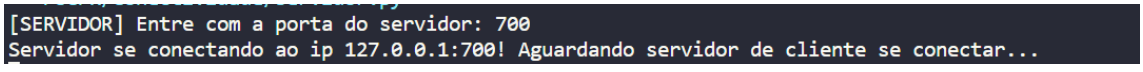
Esta atividade deverá ser entregue no AVA

O estudante deverá entregar um arquivo ".pdf" contendo as respostas da atividade proposta no item especificação.

Especificação:

Exercício 1:

1. Crie um programa servidor.py
    a. A porta deve ser pedida para o usuário (lembre-se de converter para int)
    b. Inicialize o socket UDP (IPv4 , UDP)
    c. Crie um try-except para o bind
    d. Faça um While True para leitura dos dados e da info da origem (recvfrom(1024)
    e. No mesmo While printe o endereço da origem e a mensagem recebida
2. Execute o programa (servidor.py
    a. Forneça uma porta para o servidor;
    b. Execute o comando "netstat -o -n -a" e localize a porta que você forneceu para servidor;
    c. Efetue um printscreen do comando anterior (item b);
    R: Dediquei a porta 700 para o servidor



```
UDP    0.0.0.0:5353           *:*                                   6504
UDP    0.0.0.0:5353           *:*                                   6504
UDP    0.0.0.0:5353           *:*                                   6504
UDP    0.0.0.0:5353           *:*                                   6504
UDP    0.0.0.0:5353           *:*                                   6504
UDP    0.0.0.0:5353           *:*                                   6504
UDP    0.0.0.0:5355           *:*                                   2956
UDP    0.0.0.0:53474          *:*                                   6504
UDP    0.0.0.0:53475          *:*                                   6504
UDP    0.0.0.0:57621          *:*                                   6504
UDP    0.0.0.0:61009          *:*                                   8672
UDP    0.0.0.0:63786          *:*                                   6740
UDP    26.248.248.147:137     *:*                                   4
UDP    26.248.248.147:138     *:*                                   4
UDP    26.248.248.147:1900    *:*                                   6732
UDP    26.248.248.147:49491   *:*                                   6732
UDP    127.0.0.1:700          *:*                                   9068
UDP    127.0.0.1:1900         *:*                                   6732
UDP    127.0.0.1:49493        *:*                                   6732
UDP    127.0.0.1:49664        *:*                                   4312
```

3. Crie o programa cliente.py
    a. Inicialize o socket (mesma forma do item 1)
    b. Peça para o usuário o IP do destino
    c. Peça a porta para o usuário
    d. Peça a mensagem

servidor.py | cliente.py

cliente.py > ...

```python
import socket

HOST = "127.0.0.1"
PORTA = int(input('[CLIENTE] Entre com a porta do servidor: '))
msg = input("[CLIENTE] Digite a mensagem: ")
```

    e. Envie a mensagem

```
[CLIENTE] Entre com a porta do servidor: 700
[CLIENTE] Digite a mensagem: Olá mundo! Estou enviando uma mensagem para o servidor!!!!
```

4. Execute o programa (cliente.py)
    a. Forneça um endereço IP para o servidor;
    b. Forneça uma porta para o servidor;
    c. Forneça a mensagem a ser enviada;

```
[CLIENTE] Entre com o IP HOST do servidor: 127.0.0.1
[CLIENTE] Entre com a porta do servidor: 700
[CLIENTE] Digite a mensagem: Estou enviando uma mensagem para o servidor ligado!
```

5. Com o servidor ativo, envie uma mensagem do cliente UDP;
6. Encerre o programa do servidor;
7. Envie outra mensagem do cliente para o servidor

Relatório:

Questão 1:

a) O cliente indica erro se o servidor não tiver sido iniciado primeiro?
R: Não, a aplicação de cliente continua executando, sem encerrar o processo.

b) O cliente é notificado se enviar uma mensagem para um servidor que não existe?
R: O cliente não é notificado pois o protocolo UDP não garante a entrega dos pacotes.

c) O cliente é notificado se o servidor for encerrado?
R: Não, o cliente não foi notificado quando o servidor foi encerrado.

d) Qual a maior diferença do script cliente TCP e UDP?
R: O TCP é orientado a conexão (Executa controle de fluxo, erros com retransmissão e sequenciamento;)), já o UDP não é orientado a conexão (Não executa controle de fluxo, erro, sequenciamento;)

Exercício 2:

1. Altere o código do servidor para que o IP possa ser passado como argumento do BIND.
2. Lance dois servidores na mesma porta, 9999, mas em endereços diferentes: 127.0.0.2 e 127.0.0.3:

   cmd

   python servidor.py

   python servidor.py

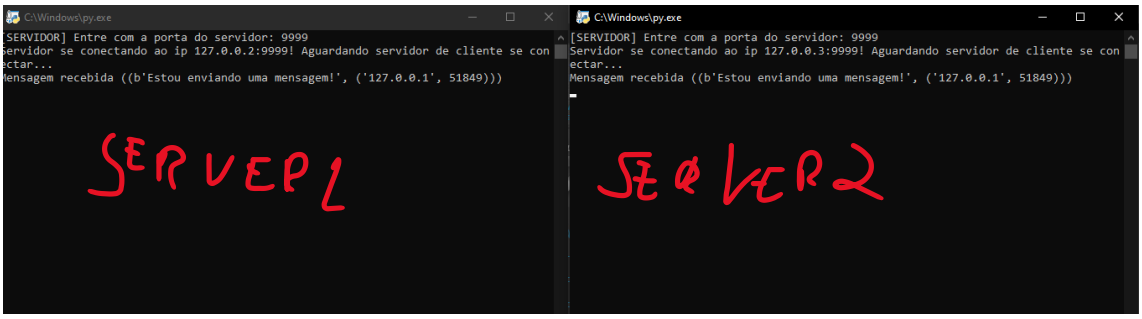
3. Com os servidores ativos, envie uma mensagem do cliente UDP;
4. Efetuar um *printscreen* dos testes realizados.

Relatório:

Questão 2:

a) O cliente pode enviar mensagens para servidores diferentes no mesmo socket?
R: Não o cliente pode se conectar em apenas um ip e uma porta simultaneamente (sem ser o de broadcast)

b) O cliente consegue enviar mensagens para o endereço de Broadcast 127.255.255.255?
R: Sim, o cliente envia a mensagem e os dois servidores recebem o pacote com as mensagens.



c) Explique a dificuldade criar um mecanismo confiável para mensagens em BROADCAST.
R: Como o UDP não é confiável isso pode ser mais complicado pois, com o broadcast ele garante que todas as mensagens enviadas cheguem no destino mesmo que algo falhe no meio do processo.

d) Pesquise o que significa transporte por datagrama e explique.
R: Também chamado de Protocolo de datagrama do usuário (UDP), é um protocolo de transmissão de dados e pacotes que é muito simples, não é orientado a conexão, não executa controle de fluxo, erro e sequenciamento. Por ser muito simples ele não é muito confiável pois ele não garante a chegada do pacote no destino, mas sim que ele chegue de forma rápida.