Centro de Ciências e Tecnologia

Ciência da Computação

UNIVERSIDADE FEDERAL DO CARIRI

TP9: SOCKET UDP

Disciplina: Lab Redes (CC0044) Professora: Camila Oliveira

Objetivos

Neste laboratório, você aprenderá os fundamentos da programação de sockets UDP em Python. Você vai aprender como enviar e receber pacotes de datagramas usando sockets UDP e também como definir um tempo limite de socket adequado.

Ao longo do laboratório, você se familiarizará com um aplicativo Ping e sua utilidade na computação estatísticas, como taxa de perda de pacotes.

Você primeiro estudará um servidor de ping simples da Internet escrito em Python e implementará um cliente correspondente. A funcionalidade fornecida por esses programas é semelhante à funcionalidade fornecidos por programas de ping padrão disponíveis em sistemas operacionais modernos. No entanto, esse programa usa um protocolo mais simples, UDP, em vez do protocolo ICMP (Internet Control Message Protocol) padrão para comunicar uns com os outros. O protocolo ping permite que uma máquina cliente envie um pacote de dados para um máquina remota e faz com que a máquina remota retorne os dados de volta ao cliente inalterados (uma ação conhecido como eco). Entre outros usos, o protocolo ping permite que os hosts determinem o round-trip time (o tempo de ida e volta) para outras máquinas.

Abaixo , você tem o código completo para o servidor Ping. Sua tarefa é escrever o cliente Ping.

Código s_ping.py

O código a seguir implementa totalmente um servidor de ping. Você precisa compilar e executar este código antes de executar seu programa cliente. Você não precisa modificar este código.

Neste código do servidor, 30% dos pacotes do cliente são simulados para serem perdidos. Você deve estudar este código com cuidado, pois isso ajudará você a escrever seu cliente de ping.

```
# s_ping.py
import random
from socket import *
# Cria o socket UDP
serverSocket = socket(AF_INET, SOCK_DGRAM)
# Define o endereço e a porta do socket
serverSocket.bind(('', 50000))
while True:
        # Gera números randômicos entre 0 to 10
        rand = random.randint(0, 10)
        # Recebe o pacote e o endereço do cliente
        message, address = serverSocket.recvfrom(1024)
        # Coloca a mensagem em maíusculo
        message = message.upper()
        # Se o número rand é menor que 4, consideramos o pacote perdido.
        if rand < 4:
                continue
        #Caso contrário, o servidor responde
        serverSocket.sendto(message, address)
```

O servidor fica em um loop infinito ouvindo se chega pacotes UDP. Quando um pacote chega, se um inteiro aleatório é maior ou igual a 4, o servidor simplesmente capitaliza os dados encapsulados e envia de volta para o cliente.

Código cliente

Você precisa implementar o seguinte programa cliente.

O cliente deve enviar 10 pings para o servidor. Como o UDP não é um protocolo confiável, um pacote enviado do cliente para o servidor podem ser perdidos na rede, ou vice-versa. Por este motivo, o cliente não pode esperar indefinidamente por uma resposta do servidor. Você deve fazer o cliente esperar até um segundo para uma resposta; se nenhuma resposta for recebida em um segundo, seu programa cliente deve assumir que o pacote foi perdido durante a transmissão pela rede. Você precisará procurar o Python documentação para descobrir como definir o valor de tempo limite em um socket UDP.

Especificamente, seu programa cliente deve

(1) enviar a mensagem de ping usando UDP (Nota: Ao contrário do TCP, você não precisa estabelecer uma conexão primeiro, já que o UDP é um protocolo sem conexão.)

(2) imprimir a mensagem de resposta do servidor, se houver

(3) calcular e imprimir o tempo de ida e volta (RTT), em segundos, de cada pacote respondido

(4) caso contrário, imprima “Request time out”

Formato da mensagem

As mensagens de ping são formatadas de maneira simples. A mensagem do cliente é uma linha, consistindo em caracteres ASCII no seguinte formato:

Ping número_sequencia tempo

onde *número_sequencia* começa em 1 e avança até 10 para cada mensagem de ping sucessiva enviada pelo cliente, e *tempo* é a hora em que o cliente envia a mensagem.

Durante o desenvolvimento, você deve executar o s_ping.py em sua máquina e testar seu cliente enviando pacotes para localhost (ou, 127.0.0.1). Depois de depurar completamente seu código, você deve ver como sua aplicação funciona com seu servidor e cliente rodando em máquinas diferentes.

Ping avançado

Atualmente, o programa calcula o tempo de ida e volta para cada pacote e o imprime individualmente. Modifique isso para corresponder à maneira como o programa de ping padrão funciona. Você terá que mostrar os RTTs mínimo, máximo e médio no final de todos os pings do cliente. Além disso, calcule a taxa de perda de pacotes (em porcentagem).

Ao terminar:

Você deve apresentar execução do seu código e explicar o seu funcionamento para a professora.