Atividade de programação 2: Laboratório UDP Pinger

Neste laboratório, você aprenderá os fundamentos da programação de soquete para UDP em Python. Você aprenderá como enviar e receber pacotes de datagramas usando soquetes UDP e também como definir um tempo limite de soquete adequado. Ao longo do laboratório, você se familiarizará com um aplicativo Ping e sua utilidade na computação de estatísticas, como taxa de perda de pacotes.

Primeiro, você estudará um servidor de ping da Internet simples escrito em Python e implementará um cliente correspondente. A funcionalidade fornecida por esses programas é semelhante à funcionalidade fornecida por programas de ping padrão disponíveis em sistemas operacionais modernos. No entanto, esses programas usam um protocolo mais simples, o UDP, em vez do padrão ICMP (Internet Control Message Protocol). para se comunicarem entre si. O protocolo ping permite que uma máquina cliente envie um pacote de dados para uma máquina remota e faça com que a máquina remota retorne os dados ao cliente inalterados (uma ação conhecida como eco). Entre outros usos, o protocolo ping permite que os hosts determinem os tempos de ida e volta para outras máquinas.

O código completo para o servidor Ping está abaixo. Sua tarefa é escrever o cliente Ping.

Código do servidor

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

Nesse código do servidor, 30% dos pacotes do cliente são simulados como perdidos. Você deve estudar esse código com cuidado, pois ele o ajudará a escrever seu cliente de ping.

```
# UDPPingerServer.py
# We will need the following module to generate randomized lost packets
import random
from socket import *

# Create a UDP socket
# Notice the use of SOCK_DGRAM for UDP packets
serverSocket = socket(AF_INET, SOCK_DGRAM)
# Assign IP address and port number to socket
serverSocket.bind(('', 12000))

while True:
    # Generate random number in the range of 0 to 10
    rand = random.randint(0, 10)
    # Receive the client packet along with the address it is coming from
    message, address = serverSocket.recvfrom(1024)
    # Capitalize the message from the client
    message = message.upper()
```

```
# If rand is less is than 4, we consider the packet lost and do not respond
if rand < 4:
    continue
# Otherwise, the server responds
serverSocket.sendto(message, address)
```

O servidor fica em um loop infinito ouvindo os pacotes UDP recebidos. Quando um pacote chega e se um número inteiro aleatório for maior ou igual a 4, o servidor simplesmente passa os dados recebidos para maiúsculo e os envia de volta ao cliente.

Perda de Pacote

O UDP fornece aos aplicativos um serviço de transporte não confiável. As mensagens podem se perder na rede devido ao estouro da fila do roteador, hardware defeituoso ou outros motivos. Como a perda de pacotes é rara ou mesmo inexistente em redes típicas de campus, este servidor de exemplo injeta perda artificial para simular os efeitos da perda de pacotes de rede. O servidor cria uma variável inteira aleatória que determina se um determinado pacote de entrada é perdido ou não.

Código do 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 pode ser perdido na rede ou vice-versa. Por esse motivo, o cliente não pode esperar indefinidamente por uma resposta a uma mensagem de ping. Você deve fazer com que o cliente espere até um segundo por uma resposta; se nenhuma resposta for recebida dentro de um segundo, seu programa cliente deve assumir que o pacote foi perdido durante a transmissão pela rede. Você precisará consultar a documentação do Python para descobrir como definir o valor de tempo limite (*timeout*) em um soquete de datagrama.

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, pois 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, se as respostas do servidor
(4) caso contrário, imprimir "Request timed out"

Durante o desenvolvimento, você deve executar o `UDPingerServer.py` em sua máquina e testar seu cliente enviando pacotes para *localhost* (ou, 127.0.0.1). Depois de depurar totalmente o código, você deve ver como seu aplicativo se comunica na rede com o servidor de ping e o cliente de ping em execução em máquinas diferentes.

Formato da mensagem

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

Ping *número_de_sequência horario*

onde *número_de_sequência* começa em 1 e progride até 10 para cada mensagem de ping sucessiva enviada pelo cliente, e *horário* é o momento em que o cliente envia a mensagem.

O que entregar

Você entregará o código do cliente completo e as capturas de tela no cliente, verificando se o seu programa de ping funciona conforme necessário.

Exercícios Opcionais

1. 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 precisará relatar os RTTs mínimo, máximo e médio ao final de todos os pings do cliente. Além disso, calcule a taxa de perda de pacotes (em porcentagem).
2. Outra aplicação semelhante ao UDP Ping seria o UDP Heartbeat. O Heartbeat pode ser usado para verificar se um aplicativo está funcionando e para relatar a perda de pacotes unidirecionais. O cliente envia um número de sequência e carimbo de data/hora atual no pacote UDP para o servidor, que está ouvindo o Heartbeat (isto é, os pacotes UDP) do cliente. Ao receber os pacotes, o servidor calcula a diferença de horário e reporta a perda de pacotes. Se os pacotes Heartbeat estiverem ausentes por algum período de tempo especificado, podemos assumir que o aplicativo cliente parou.
   Implemente o UDP Heartbeat (cliente e servidor). Você precisará modificar o `UDPingerServer.py`, e seu cliente de ping UDP.