Задача по программированию сокетов 2: Эхо-запросы через UDP

В этой лабораторной работе вы познакомитесь с основами программирования сокетов для UDP-соединений на языке Python. Вы узнаете, как отправлять и получать пакеты датаграмм с помощью UDP-сокетов, как задавать значение тайм-аута сокета. Познакомившись с приложением для эхо-запросов, вы поймете пользу его применения для анализа эффективности работы компьютерных сетей при помощи такого показателя, как коэффициент потери пакетов.
Сначала вам предстоит изучить простой пинг-сервер, написанный на языке Python, а затем реализовать соответствующую клиентскую программу. Функциональные возможности, предоставляемые этими программами, будут аналогичны тем, что предлагают стандартные программы «пингования», доступные в современных операционных системах. Тем не менее, наши с вами программы для общения друг с другом будут использовать более простой протокол UDP, а не стандартный протокол ICMP. Протокол пингования позволяет отправить пакет данных с клиентской машины на удаленный хост, и вернуть данные обратно клиенту без изменений (эхо-запрос и эхо-ответ). Кроме этого, протокол пингования позволяет хосту определять время прохождения сигнала (время оборота, RTT) на другие узлы сети.
Ниже представлен готовый код для серверной части, а ваша задача – написать код клиентской части программы.

Код серверной части

Следующий код реализует готовый сервер для пингования. Вы должны его скомпилировать и запустить перед тем, как запускать код клиентской части. *Что-либо изменять в нем не нужно.*
Серверный код построен таким образом, что моделирует 30-процентную потерю пакетов. Изучите его внимательно, и это поможет вам в написании клиентской части.

```
# UDPPingerServer.py
# Следующий модуль используется для генерирования случайных потерь пакетов
import random
from socket import *
# Создаем UDP-сокет
# Для UDP используем SOCK_DGRAM
serverSocket = socket(AF_INET, SOCK_DGRAM)
# Связываем порт 12000 с сокетом сервера
serverSocket.bind(('', 12000))
while True:
      # Генерируем случайное число от 0 до 10
      rand = random.randint(0, 10)
      # Получаем пакеты от клиента с адресом address
      message, address = serverSocket.recvfrom(1024)
      # Делаем символы клиентского сообщения заглавными
      message = message.upper()
      # Если rand меньше 4, считаем пакет потерянным и не выдаем ответ
      if rand < 4:
      continue
      # В противном случае сервер дает ответ
      serverSocket.sendto(message, address)
```

Сервер находится в бесконечном цикле, ожидая приходящие UDP-пакеты. Если пакет прибывает, и случайное число больше либо равно 4, сервер просто изменяет символы входящего сообщения на заглавные и отправляет их обратно клиенту.

Потеря пакетов

UDP предоставляет приложениям ненадежную транспортную службу – в сети могут происходить потери сообщений из-за переполнения очереди маршрутизатора, неисправного оборудования или по другим причинам. Но так как в современных сетях потери пакетов – достаточно редкая вещь, в серверном коде мы искусственно моделируем ситуацию потерь сетевых пакетов. Сервер создает переменную, являющуюся случайным целым числом, которое определяет, является ли входящий пакет потерянным или нет.

Код клиентской части

Вы должны реализовать следующую программу клиентской части.
Клиент должен отправить 10 эхо-запросов серверу. Поскольку UDP является ненадежным с точки зрения доставки протоколом, пакет, отправленный от клиента к серверу или наоборот, может быть потерян в сети. Так как клиент не может бесконечно ждать ответа на запрос, нужно задать период ожидания ответа (тайм-аут), равный одной секунде – если ответ не будет получен в течение одной секунды, клиентская программа должна предполагать, что пакет потерян. Чтобы узнать, как устанавливать значение тайм-аута сокета, вам следует обратиться к документации по языку Python.
В частности, ваша клиентская программа должна

(1) отправить эхо-запрос, используя UDP (Примечание: в отличие от TCP, вам не нужно сначала устанавливать соединение, так как UDP является протоколом без установления соединения.)
(2) распечатать ответное сообщение от сервера, если таковое имеется
(3) вычислить и вывести на печать время оборота (RTT) в секундах для каждого пакета при ответе сервера
(4) в противном случае, вывести сообщение «Request timed out» (Время запроса истекло)

В процессе разработки вы будете запускать `UDPPingerServer.py` на вашем компьютере и тестировать работу клиента, отправляя пакеты на *localhost* (или 127.0.0.1). После того как вы полностью отладите свой код, вы должны увидеть, как ваши клиентская и серверная части приложения, установленные на разных компьютерах, взаимодействуют по сети.

Формат сообщения

Сообщения в данной лабораторной работе имеют достаточно простой формат. Клиентское сообщение – это одна строка, состоящая из символов ASCII в следующем формате:

Ping *номер_последовательности время*

где *номер_последовательности* начинается с 1 и увеличивается до 10 для каждого последующего сообщения, отправленного клиентом, а *время* – это момент времени, когда клиент отправляет сообщение.

Задание

Завершить код клиентской части и сделать скриншоты, подтверждающие корректную работу вашей программы пингования.

Дополнительные задания

1. Данный вариант программы вычисляет время оборота для каждого пакета и выводит его отдельно. Измените вывод таким образом, чтобы он соответствовал тому, как это делается в стандартной утилите ping. Для этого вам нужно будет сообщить минимальное, максимальное и среднее значение RTT в конце каждого ответа от сервера. Дополнительно вычислите коэффициент потери пакетов (в процентах).
2. Другим приложением, подобным программе пингования, будет UDP Heartbeat (монитор доступности), который может быть использован для проверки, работает ли приложение, и вывода сообщения об односторонней потере пакетов. Клиент отправляет порядковый номер и текущую временную метку в пакете UDP на сервер, который слушает «сердцебиение» (т.е. ожидает UDP-пакеты) клиента. После получения пакетов сервер вычисляет разницу во времени и сообщает о потерях. Если пакеты отсутствуют какой-то определенный период времени, можно предположить, что клиентское приложение остановлено.
   Реализуйте UDP Heartbeat (обе части – клиент и сервер), доработав обе ваши части программы пингования.