Задача по программированию сокетов 5: эхо-запросы через ICMP

В этой работе вы ближе познакомитесь с протоколом межсетевых управляющих сообщений (ICMP) и узнаете, как реализовать приложение для эхо-запросов (пингования) с помощью ICMP-запросов и ответов.
Данное приложение используется, в основном, для проверки доступности определенного хоста в IP-сети, а также для самотестирования платы сетевого интерфейса компьютера или в качестве теста латентности сети. Оно отправляет ICMP-пакеты («эхо-запрос» или «пинг») на целевой хост и слушает ICMP-ответы («эхо-ответ» или «понг») от него. При получении ответов измеряется время оборота (RTT), фиксируются потери пакетов и выводится статистическая сводка о полученных ответах (минимальное, максимальное и среднее значение времени оборота, а в некоторых версиях еще и стандартное отклонение от среднего).
Ваша задача заключается в разработке собственного приложения пингования на языке Python. Оно будет использовать протокол ICMP, но для некоторого упрощения не будет в точности следовать спецификации этого протокола, описанной в документе RFC 1739.
Отметим, что вам потребуется написать только клиентскую часть программы, а функциональные возможности, необходимые на стороне сервера, встроены почти во все операционные системы.
Ваше готовое приложение должно отправлять эхо-запросы на определенный хост приблизительно через одну секунду. Каждое такое сообщение-запрос содержит данные, которые включает в себя штамп времени. После отправки каждого пакета приложение ждет получения ответного сообщения в течение одной секунды. Если ответа нет, то клиент предполагает, что либо пакет запроса, либо ответный пакет были потеряны в сети (или целевой хост недоступен)

Код

Ниже представлена заготовка для клиентской программы. Вам нужно ее завершить. Места, которые вы должны заполнить своим кодом, помечены как ***#Начало вставки*** и ***#Конец вставки***. В каждом таком месте может содержаться одна или несколько строк вашего кода.

Дополнительные замечания

1. В методе `receiveOnePing()` вам необходимо получить структуру ICMP_ECHO_REPLY и извлечь нужную вам информацию – контрольную сумму, номер последовательности, время жизни TTL и т.д. Изучите перед этим метод `sendOnePing()`
2. Учет контрольной суммы уже заложен в коде. Вам не нужно об этом заботиться.
3. В этой лабораторной используются сырые сокеты. В некоторых операционных системах вам может потребоваться учетная запись администратора/суперпользователя, чтобы запустить свою программу пингования.
4. Для получения более подробной информации о протоколе ICMP смотрите заключительную часть лабораторной.

Тестирование программы

Для начала проверьте отправку пакетов на локальный хост, то есть на IP-адрес 127.0.0.1. Затем посмотрите, как ваше приложение взаимодействует по сети, отправляя эхо-запросы серверам в разных точках планеты.

Задание

Завершить код вашей программы и сделать скриншоты результатов пингования четырех различных хостов назначения, находящихся на разных континентах.

Шаблон кода программы ICMP-пингования на языке Python

```
from socket import *
import os
import sys
import struct
import time
import select
import binascii

ICMP_ECHO_REQUEST = 8
def checksum(str):
     csum = 0
     countTo = (len(str) / 2) * 2
     count = 0
     while count < countTo:
          thisVal = ord(str[count+1]) * 256 + ord(str[count])
          csum = csum + thisVal
          csum = csum & 0xffffffffL
          count = count + 2
     if countTo < len(str):
          csum = csum + ord(str[len(str) - 1])
          csum = csum & 0xffffffffL
     csum = (csum >> 16) + (csum & 0xffff)
     csum = csum + (csum >> 16)
     answer = ~csum
     answer = answer & 0xffff
     answer = answer >> 8 | (answer << 8 & 0xff00)
     return answer
def receiveOnePing(mySocket, ID, timeout, destAddr):
     timeLeft = timeout

     while 1:
          startedSelect = time.time()
          whatReady = select.select([mySocket], [], [], timeLeft)
          howLongInSelect = (time.time() - startedSelect)
          if whatReady[0] == []: # Время истекло
          return "Превышен интервал ожидания запроса"
          timeReceived = time.time()
          recPacket, addr = mySocket.recvfrom(1024)

          #Начало вставки

          #Извлекаем заголовок ICMP из IP-пакета

          #Конец вставки
          timeLeft = timeLeft - howLongInSelect
          if timeLeft <= 0:
               return " Превышен интервал ожидания запроса"
```

```
def sendOnePing(mySocket, destAddr, ID):
     # Формат заголовка: type (8), code (8), checksum (16), id
     (16), sequence (16)
     myChecksum = 0
     # Делаем фиктивный заголовок с нулевой контрольной суммой.
     # struct -- Интерпретирует строки как упакованные бинарные
     данные
     header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0,
     myChecksum, ID, 1)
     data = struct.pack("d", time.time())
     # Вычисляем контрольную сумму данных и заголовка.
     myChecksum = checksum(header + data)

     # Получаем реальную контрольную сумму и помещаем в
     заголовок.
     if sys.platform == 'darwin':
          myChecksum = socket.htons(myChecksum) & 0xffff
          #Переводим 16-битное целое в порядок байтов, принятый в
          сети Интернет.
     else:
          myChecksum = socket.htons(myChecksum)

     header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0,
     myChecksum, ID, 1)
     packet = header + data

     mySocket.sendto(packet, (destAddr, 1)) # AF_INET должен
     иметь тип tuple, а не str
     #И списки, и кортежи (тип tuple) состоят из набора объектов
     #к которым можно обращаться по их порядковому номеру
def doOnePing(destAddr, timeout):
     icmp = socket.getprotobyname("icmp")
     #SOCK_RAW – достаточно мощный тип сокета. Для подробной
     информации см.: http://sock-raw.org/papers/sock_raw

     #Начало вставки

     #Здесь создаем сокет

     #Конец вставки
     myID = os.getpid() & 0xFFFF #Возвращаем идентификатор
     текущего процесса
     sendOnePing(mySocket, destAddr, myID)
     delay = receiveOnePing(mySocket, myID, timeout, destAddr)
     mySocket.close()
     return delay
def ping(host, timeout=1):
     #timeout=1 означает: Если ответа нет в течение секунды,
     клиент предполагает, что #либо пакет запроса, либо ответный
     пакет потеряны
     dest = socket.gethostbyname(host)
     print "Пингуем " + dest + " используя Python:"
     print ""
```

```
    #Отправляем эхо-запросы серверу каждую секунду
    while 1 :
        delay = doOnePing(dest, timeout)
        print delay
        time.sleep(1)# ждем одну секунду
    return delay

ping("www.poly.edu")
```

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 вывести минимальное, максимальное и среднее значение RTT в конце каждого ответа от сервера. Дополнительно вычислите коэффициент потери пакетов (в процентах).
2. Ваша программа обрабатывает только тайм-ауты в получении ICMP-ответов. Добавьте к ней функционал анализа кодов ошибок протокола ICMP и вывод соответствующих результатов для пользователя. Примерами кодов ошибок ICMP являются: 0 – сеть назначения недоступна, 1 – хост назначения недоступен.

Протокол ICMP

ICMP-заголовок

ICMP-заголовок начинается после 160-го бита IP-заголовка (если не используются параметры IP)

Биты	**160-167**	**168-175**	**176-183**	**184-191**
160	Тип	Код	Контрольная сумма	
192	Идентификатор		Номер последовательности	

- **Тип** – тип сообщения ICMP
- **Код** – подтип заданного типа
- **Контрольная сумма** – вычисляемые для всего ICMP-пакета целиком данные для проверки ошибок. При вычислении контрольной суммы значение поля полагается равным нулю
- **Идентификатор** – значение, возвращаемое в случае, когда сообщение является эхо-ответом
- **Номер последовательности** – значение, возвращаемое в случае, когда сообщение является эхо-ответом

Эхо-запрос

Эхо-запрос – это сообщение ICMP, данные которого должны быть возвращены обратно в эхо-ответе от принимающего хоста.

- Тип ICMP для эхо-запроса равен 8.
- Код устанавливается в значение 0.
- Идентификатор и номер последовательности используются клиентом, чтобы идентифицировать соответствующие друг другу пары запрос-ответ. На практике большинство систем семейства Linux используют уникальный идентификатор для

каждого процесса пингования, а номер последовательности – увеличивающееся в течение этого процесса число. Семейство ОС Windows использует фиксированный идентификатор, который меняется от версии к версии Windows, а номер последовательности сбрасывается только во время загрузки системы.

- Данные, полученные в эхо-запросе, должны полностью быть включены в эхо-ответ.

Эхо-ответ

Эхо-ответ – это сообщение ICMP, генерируемое в ответ на эхо-запрос.

- Тип ICMP и код для эхо-ответа устанавливаются в 0.
- Идентификатор и номер последовательности используются клиентом для определения соответствия пар запрос-ответ.
- Данные, полученные в эхо-запросе, должны полностью быть включены в эхо-ответ.