套接字编程作业 5-1：ICMP ping 程序

本作业要求理解 ICMP 协议，掌握使用 ICMP 请求和响应消息实现 Ping 程序。

Ping 是一个流行的网络应用程序，用于测试 IP 网络中的某个特定主机是否可达。它也可用于测量客户主机和目标主机之间的网络延迟。它的工作过程是：向目标主机发送 ICMP“回显（Echo）”报文（即 ping 分组），并监听 ICMP“回显响应（Echo reply）”应答（有时也称为 pong 分组）。ping 程序测量往返时间（RTT），记录分组丢失，并计算多个 ping-pong 交换的统计汇总（往返时间的最小值、最大值、平均值和标准差）。

在这个编程作业中，你的任务是使用原始套接字（RAW Socket）开发自己的 Ping 程序。你的程序将使用 ICMP，但为了保持简单，将不完全遵循 RFC 1739 中的官方规范。请注意，你只需要编写 ping 客户端程序，因为服务器侧所需的功能内置于所有的操作系统中。

任务 1、编写 ping 客户端代码

请在“ping 客户端框架代码.py”中标有#Fill-in-start 和#Fill-in-end 的地方填写代码，每个地方都可能需要不止一行代码。

你的 ping 程序需具有如下功能：

1）ping 程序的命令格式：

client.py destination_host [–n count]

其中：***client.py*** 是客户端程序文件名；***destination_host*** 是目标主机的域名地址；**–n *count*** 选项用来指定发送的 ping 分组个数，如不使用该选项则默认发送 4 个 ping 分组。

“ping 客户端框架代码.py”框架代码已给出了完整的 ping 命令解析代码。

2）Ping 程序发送的 ping 分组之间间隔大约一秒钟，每个 ping 分组的 ICMP 数据字段携带一个时间戳。每个 ping 分组发送完后，程序最多等待一秒，用于接收响应。

如果一秒后没有收到响应，那么应假设 ping 分组或 pong 分组在网络中丢失（或者目的主机已关闭），并显示如下信息：

Request timed out

如果收到目的主机返回的 pong 分组，则显示如下信息：

Reply from ***ip_address***: bytes=***data_bytes*** time=***rtt***ms TTL=***TTL_value***

其中：***ip_address*** 是发送响应报文的主机 IP 地址；***data_bytes*** 是响应报文中 ICMP 数据字段的字节长度；***rtt*** 是以毫秒（ms）为单位的往返时间（RTT），***TTL_value*** 是 pong 分组的 IP 首部中的生存时间（TTL）字段值。

框架代码中提供了发送一个 ping 分组的完整代码——“sendOnePing”方法，以及

计算 ICMP 报文校验和的完整代码——“checksum”方法。在接收一个 pong 分组的“receiveOnePong”方法中，你需要提取 ICMP 分组首部中的报文类型、校验和、ID 等信息来检查接收的 ICMP 分组是否有效；此外还需要提取 IP 分组首部中的 TTL、长度以及 ICMP 分组数据字段，以生成 ping 程序要显示的信息。

请在尝试完成“receiveOnePong”方法之前，先仔细研究“sendOnePing”方法和“checksum”方法。特别注意网络字节序问题！

3）ping 程序发送和接收完所有 ping-pong 分组后，要统计并显示（格式自定义）以下信息：

- 已发送的 ping 分组数量、已接收的 pong 分组数量、丢包数量和丢包率；
- 以毫秒（ms）为单位的最小、最大和平均 RTT。

任务 2、测试 ping 客户端程序

首先，通过发送分组到本地主机来测试你的客户端，目的主机地址是 127.0.0.1。

然后，分别 Ping 位于国内的 2 个目的主机和位于国外的 2 个目的主机，以测试你的 ping 程序。

这个作业要求使用原始套接字，因此在某些操作系统中，可能需要管理员/root 权限才能运行你的 Ping 程序。

任务 3*、解析并显示 ICMP 响应错误（选做）

任务 1 编写的 Ping 程序只能检测收到 ICMP 回显响应报文或没有收到响应。请修改 Ping 程序，解析 ICMP 的目的不可达错误代码，并向用户显示相应的错误结果。教材图 5-19 给出了常见的 ICMP 响应错误码。

作业提交要求

任务 1 和 2 必做，任务 3 选做。

将以下内容压缩打包后提交，压缩文件的命名规则为“作业 5-1-学号-姓名”：

1）任务 1 的完整 ping 客户端代码文件

2）运行结果报告文件，包含：

- Python 版本
- 任务 2 的 ping 客户端程序运行窗口截图

3）可选提交：任务 3 的 ping 客户端代码文件 1 份，以及运行结果报告 1 份（包含客户端运行窗口截图）

ping 客户端框架代码

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

ICMP_ECHO_REQUEST = 8

def checksum(string):
    csum = 0
    countTo = (len(string) // 2) * 2
    count = 0

    while count < countTo:
        thisVal = string[count] * 256 + string[count+1]
        csum = csum + thisVal
        csum = csum & 0xffffffff
        count = count + 2

    if countTo < len(string):
        csum = csum + string[len(string) - 1]
        csum = csum & 0xffffffff

    csum = (csum >> 16) + (csum & 0xffff)
    csum = csum + (csum >> 16)
    answer = ~csum
    answer = answer & 0xffff
    answer = answer >> 8 | (answer << 8 & 0xff00)
    return answer

def receiveOnePong(mySocket, destAddr, ID, sequence, timeout):
    timeLeft = timeout

    while 1:
        startedSelect = time.time()
        whatReady = select.select([mySocket], [], [], timeLeft)
        howLongInSelect = (time.time() - startedSelect)
        if whatReady[0] == []: # Timeout
            return None
        timeReceived = time.time()
        recPacket, addr = mySocket.recvfrom(1024)

        #Fill in start

        #Fetch the ICMP header from the IP packet

        #Fill in end

        timeLeft = timeLeft - howLongInSelect
        if timeLeft <= 0:
            return None
```

```
def sendOnePing(mySocket, destAddr, ID, sequence):
    # Header is type (8), code (8), checksum (16), id (16), sequence (16)

    myChecksum = 0
    # Make a dummy header with a 0 checksum
    # struct -- Interpret strings as packed binary data
    header = struct.pack("!BBHHH", ICMP_ECHO_REQUEST, 0, myChecksum, ID, sequence)
    data = struct.pack("!d", time.time())

    # Calculate the checksum on the data and the dummy header.
    myChecksum = checksum(header + data)

    # Get the right checksum, and put in the header
    if sys.platform == 'darwin':
        # Convert 16-bit integers from host to network byte order
        myChecksum = htons(myChecksum) & 0xffff
    else:
        myChecksum = htons(myChecksum)

    header = struct.pack("!BBHHH", ICMP_ECHO_REQUEST, 0, myChecksum, ID, sequence)
    packet = header + data

    mySocket.sendto(packet, (destAddr, 1)) # AF_INET address must be tuple, not str
    # Both LISTS and TUPLES consist of a number of objects
    # which can be referenced by their position number within the object.

def doOnePing(destAddr, ID, sequence, timeout):
    icmp = getprotobyname("icmp")

    # SOCK_RAW is a powerful socket type. For more details:
    #http://sock-raw.org/papers/sock_raw
    #Fill in start

    #Create Socket here

    #Fill in end

    sendOnePing(mySocket, destAddr, ID, sequence)
    rtt = receiveOnePong(mySocket, destAddr, ID, sequence, timeout)

    mySocket.close()
    return rtt

def ping(dest, count):
    # timeout=1 means: If one second goes by without a reply from the server,
    # the client assumes that either the client's ping or the server's pong is lost
    timeout = 1

    myID = os.getpid() & 0xFFFF    # Return the current process i
    loss = 0

    # Send ping requests to a server separated by approximately one second
    for i in range(count) :
        result = doOnePing(dest, myID, i, timeout)

        #Fill in start
```

```
            #Print response information of each pong packet:
            #No pong packet, then display "Request timed out."
            #Receive pong packet, then display "Reply from host_ipaddr : bytes=… time=… TTL=…"

            #Fill in end

            time.sleep(1)# one second

        #Fill in start

        #Print Ping statistics

        #Fill in end

        return

if __name__ == "__main__":
        if len(sys.argv) < 2:
                print('IcmpPing.py dest_host [-n <count>]')
                sys.exit()
        host = sys.argv[1]
        try:
                dest = gethostbyname(host)
        except:
                print('Can not find the host "%s". Please check your input, then try again.'%(host))
                exit()

        count = 4
        try:
                opts, args = getopt.getopt(sys.argv[2:], "n:")
        except getopt.GetoptError:
                print('IcmpPing.py dest_host [-n <count>]')
                sys.exit(2)
        for opt, arg in opts:
                if opt == '-n':
                        count = int(arg)

        print("Pinging " + host + " [" + dest + "] using Python:")
        print("")
        ping(dest, count)
```