实验报告

1611736 钟腾

本次计算机网络实验要求完成一次关于UDP连接的服务器socket的python编程，完成编程后运行代码，先执行服务器代码使服务器处于等待接收状态，然后执行客户端代码向服务器发送10个udp包，并接收服务器返回的数据，输出报文类型序号发送时间以及各项RTT及其相关统计。根据要求，完成代码如下:

实验给出的UDPPingerServer.py:

```python
# 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(('10.41.51.25', 12000))

while True:
    print('listening..')
    # 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)

```

此代码创建了一个serverSocket来监听来自客户端的收到的信息，当收到后产生0-10间的随机数，随机数小于4的不再发送到客户端，以此模拟丢包情况。

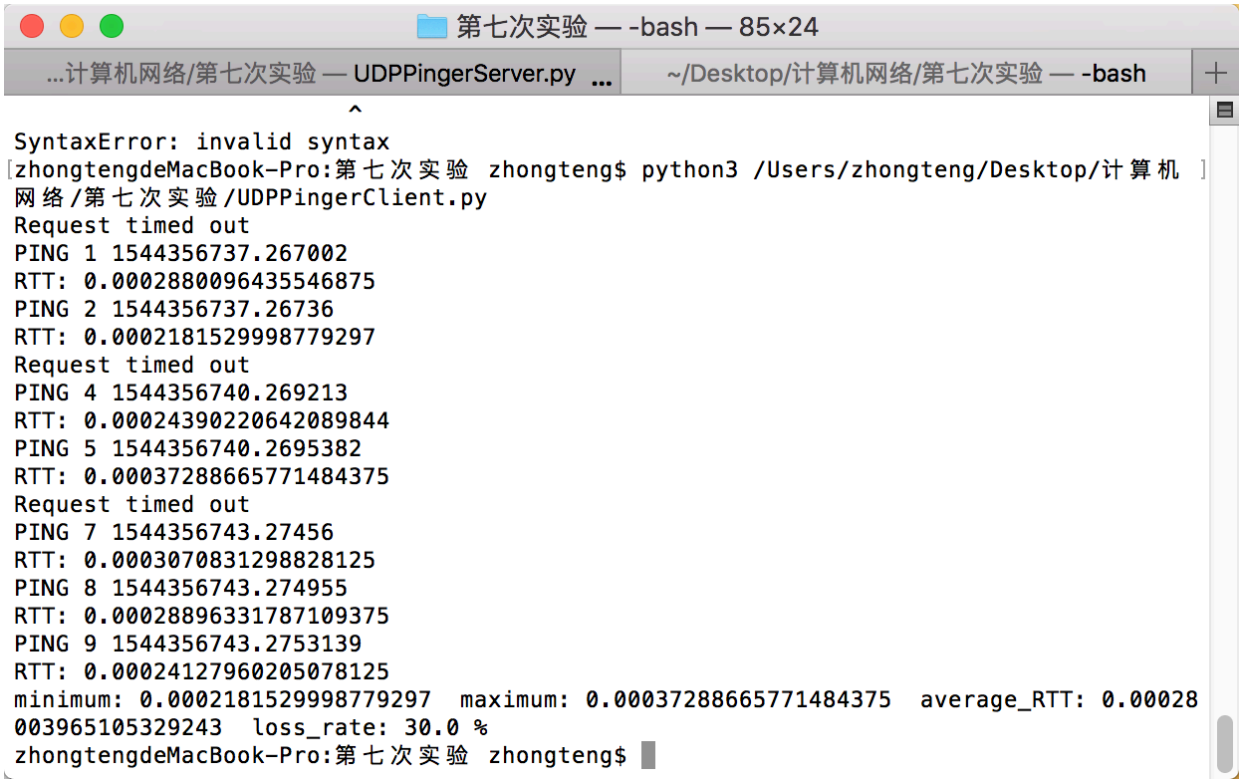
UDPPingerClient.py:

```python
from socket import *
import time

clientSocket = socket(AF_INET, SOCK_DGRAM)
clientSocket.settimeout(3) #设置1秒的延迟

minimum = 10 #RTT最小值
maximum = 0 #RTT最大值
sum_RTT = 0 #RTT总值
average_RTT = 0#平均RTT
loss_count = 0 #丢包数量

for i in range(0,10):
    send_time = time.time() #记录发送时间
    message = 'Ping '+str(i)+' '+str(send_time) #发送报文格式
    clientSocket.sendto(message.encode('utf-8'),('10.41.51.25',12000)) #发送报文

    try:
        count = 0
        RTT = 0
        received_message = clientSocket.recv(1024).decode('utf-8') #接收返回信息
        received_type,received_num= received_message.split()[0],received_message.split()[1] #截取报文类型和报文序号
        received_time = time.time() #记录接收时间
        RTT = received_time - send_time #计算RTT
        sum_RTT = sum_RTT+RTT #计算RTT总值
        if maximum<RTT: #计算RTT最大值
            maximum = RTT

        if minimum>RTT: #计算RTT最小值
            minimum = RTT
        print(received_type,received_num,send_time) #打印报文
        print("RTT:",RTT) #打印RTT

    except:
        print('Request timed out')
        loss_count = loss_count+1 #统计丢包数量
        continue

average_RTT = sum_RTT/(10-loss_count) #计算平均RTT
print("minimum:",minimum," maximum:",maximum," average_RTT:",average_RTT," loss_rate:",float(loss_count/10)*100,"%") #打印统计数据
```

首先引入socket包，建立clinetSocket，确定好ip地址和端口号，然后开始发送十条消息到服务器，记录好发送消息的时间。

接收返回消息，收到对应序号的消息计算好RTT，把类型和序号及发送时间输出，丢包的输出Request timed out提示丢包。最后输出RTT的最小值、最大值、平均值和丢包率。

实验截图：



至此顺利完成本次实验要求，通过这次实验学会了python的socket网络编程和服务器客户端利用socket完成的UDP传输数据包的步骤，受益匪浅。