

Winter 2015-16 CSC 450: Computer Networks

Assignment 2: Socket Programming Assignment using UDP

In this assignment, you will learn the basics of socket programming for UDP in Python. You will learn how to send and receive datagram packets using UDP sockets and also how to set a proper socket timeout.

You will first study a simple Internet ping server written in python, and implement a corresponding client. The functionality provided by these programs is similar to the functionality provided by standard trace route (ping) programs available in modern operating systems.

The trace route allows for a client machine to send a packet of data to a remote machine, and have the remote machine return the data back to the client unchanged (an action referred to as echoing). Among other uses, trace route allows hosts to determine the round-trip time (RTT) to other machines.

You are given the complete code for the “Ping server” below. Your task is to write the “Ping client”.

Server code:

The following code fully implements a ping server. You need to compile and run this code before running your client program. You do not need to modify this code.

In this server code, 30% of the client’s packets are simulated to be lost. You should study this code carefully, as it will help you write your ping client.

```
-----  
  
# CSC450UDPPingServer.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)
```

Winter 2015-16 CSC 450: Computer Networks

Assignment 2: Socket Programming Assignment using UDP

```
# 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)
```

The server sits in an infinite loop listening for incoming UDP packets. When a packet comes in and if a randomized integer is greater than or equal to 4 is received, the server simply capitalizes the encapsulated data and sends it back to the client.

Packet loss:

UDP provides applications with an unreliable transport service. Messages may get lost in the network due to router overflows, faulty hardware or several other reasons. Because packet loss is rare or even non-existent in LANs, the server in this assignment injects artificial loss to simulate the effects of network packet loss. The server creates a variable randomized integer which determines whether a particular incoming packet is lost or not.

Client Code:

You need to implement the following client program.

The client should send 10 pings to the server.

As UDP is an unreliable protocol, a packet sent from the client to the server may be lost in the network. For this reason, the client cannot wait indefinitely for a reply to the ping message.

You should get the client to wait for no more than one second for a reply; if no reply is received within one second, your client program should assume that the packet was lost during transmission across the network.

You will need to look up the Python documentation to find out how to set the timeout value on a datagram socket.

Specifically, your client program should:

- (1) Send the ping message using UDP (Note: Unlike TCP, you do not need to establish a connection first)
- (2) Print the response message from server, if any

Winter 2015-16 CSC 450: Computer Networks
Assignment 2: Socket Programming Assignment using UDP

- (3) Calculate and print the round trip time (RTT), in seconds, of each packet, if server responds
- (4) Otherwise, print "Request timed out"

During development, you should run the **CSC450UDPPingServer.py** on your machine, and test your client by sending packets to localhost (or, 127.0.0.1).

NOTE: After you have fully debugged your code, you should see how your application communicates across the network with the ping server and ping client running on different machines.

Message Format

The ping message in this assignment is formatted in a simple way.

The client message is one line, consisting of ASCII characters in the following format:

Ping *sequence_number* *time*

Where *sequence_number* starts at 1 and progresses to 10 for each successive ping message sent by the client, and *time* is the time when the client sends the message.

What to submit via Moodle:

You will turn in the complete client code and screenshots at the client verifying that your ping program works as required.