Programming Assignment 1: UDP Echo Client and Server

NOTE: Before starting the assignment, please see *Submission Guidelines* folder in Canvas with general instructions for implementing all assignments and format of submissions.

Using UDP sockets, you will write a client and server program that enables the client to send a string of some specified length to the server over the network, and the server simply echoes back that string back to the client.

The client program should take the following command-line parameters:

- IP address of server
- UDP port of server
- Length of string to be sent

The client program will read in the above input parameters, initialize a string containing alphabetical characters of the specified length[1], and send the message using the UDP socket API to the server running at the specified IP address and port. If the client does not receive a message back from the server within a certain amount of time (one second), the client should retry up to a maximum number of tries (3) before terminating.[2] The program output should print out trace information when data is sent and received, and account for error conditions. Client trace output must include:

- A message when data is sent to the server indicating destination IP address and port and length plus content of the data sent
- A message when data is received from the server indicating source IP address and port and contents of the data received
- An error message when any error occurs such as when a time-out occurs because the server is not running

The server program should take the following command-line parameters:

- IP address that server listens on (127.0.0.1 will be used to test the program)
- UDP port that server listens on (e.g. 12000)

The server will listen on the loopback address and the given port number, and be prepared to receive data from the client up to a fixed maximum length (100 bytes). The server will wait in an infinite loop to receive data from a client, and then send the received data back to the client without modification. Server trace output must include:

- A message when data is received from the client indicating source IP address and port and contents of the data received

---

[1] The string may contain any value e.g. 'XXXXXXXXXX'

[2] The socket API supports a timeout facility. See documentation on socket library relevant to programming language you are using.

- A message when data is sent back to the client indicating destination IP address and port

Test Cases (both test cases must be demonstrated for full credit):

1. Test both client and server running on the same host, such as your computer (this should illustrate the client sending a single message and the server response)
2. Test client behavior when server is not running (this should illustrate the client sending 3 messages and timing out after each one). Note: In some operating systems, the client may exit with an error when sending to a non-existing server. To simulate this test case it may be necessary to run the server, but modify or comment out the code so that the server receives messages from the client, but does not send the messages back to the client.

Example of the client trace output (server responds):

```
Sending data to   127.0.0.1, 12000: XXXXXXXXXX (10 characters)
Receive data from 127.0.0.1, 12000: XXXXXXXXXX
```

Example of the client trace output (server does not respond):

```
Sending data to  127.0.0.1, 12000: XXXXXXXXXX (10 characters)
Message timed out
Sending data to  127.0.0.1, 12000: XXXXXXXXXX (10 characters)
Message timed out
Sending data to  127.0.0.1, 12000: XXXXXXXXXX (10 characters)
Message timed out
```

Example of server trace output:

The server is ready to receive on port: 12000

Receive data from client 127.0.0.1, 56777: XXXXXXXXXX
Sending data to client 127.0.0.1, 56777: XXXXXXXXXX

References:

- Python
  - https://docs.python.org/3/library/socket.html
- C
  - https://linux.die.net/man/7/socket
  - https://linux.die.net/man/3/byteorder

Submission Requirements:
Please submit the following five types of individual files to Canvas by due date. **Please, NO zip files.**

- ✓ Submit the client source program file (please name the file *client.py* and include name, UCID, section in comments at top of source files)
- ✓ Submit separate wireshark .pcap files for Test Case 1 and Test Case 2 (please name *test-case1.pcap* and *test-case2.pcap* respectively)
- ✓ Submit screenshots in .pdf format showing the trace output of the client and server
- ✓ Submit the README file (C programs only; No need for README for Python programs)

Program Grading Rubric: Total of 10 points
If program does not run under Python3 or command-line arguments are not supported, NO credit is awarded.

- Command-line Arguments (2)
  - Client sends Echo request to IP address and port specified on command-line
    - Format: String in encoded form
    - Length: as specified
- Test Case 1 (2)
  - Client sends single request, and Server responds
- Test Case 2 (6)
  - Client sends 3 Echo Requests with timeout

Late submissions not accepted on this assignment.

Academic Integrity
If academic integrity standards are not upheld, no credit is given. This includes copying of program, quiz or wireshark .pcap file from any source, or hard-coding of results in your program.