Lab II: Python Socket Programming

Objectives

- Learn the concept of client server paradigm to implement network applications
- Learn how to use UDP and TCP sockets to create a simple network application.
- Learn how to create a TCP server that handles concurrent connections.

Equipment List

Equipment	Quantity
Laptop computers	3
Ethernet cables	3
Cisco switch	1

Lab Procedures

Part I: Simple TCP Socket Program

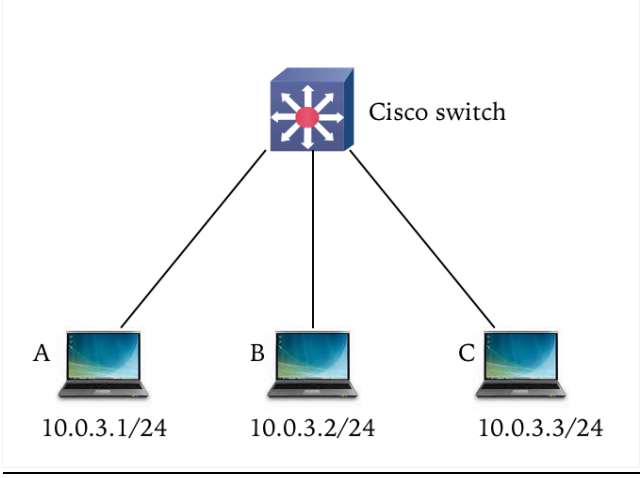

Figure 1. Network topology

Basic Python 3 client and server codes are provided for your use. Download the code files from myLE to all your laptops. If you do not have enough labtop computers, use RPi computers in the lab.

1. Connect three computers A, B, C via the Ethernet switch and assign a fixed IP address as shown in Fig.1. Test the connectivity among the computers by using Ping command.

2. In each laptop, go to Start menu and select Anaconda3 (64-bit) -> Anaconda Prompt. Then, go to the directory where you save the files. Let us assume here that the directory is `C:\Users\yourname\code-py3>`

3. Start the TCP server program at Computer A with

```
c:\>Users\yourname\code-py3>python tcp_server.py
```

4. At Computer B,
    a. Open Wireshark to capture traffic with the display filter set to '`ip.addr==10.0.3.1 and tcp`'.
    b. Appropriately set the IP address in tcp_client.py. Then, open Anaconda Prompt and start the TCP client program with

```
c:\>Users\yourname\code-py3>python tcp_client.py
```

    Send a few lines of messages to the server and leave the connection open. The texts should appear on the server screen.
5. At Computer A,
    a. Open another Anaconda Prompt.
    b. Run the TCP server program and observe what happens.

6. At Computer C, start the TCP client program and try to send a line of message to the server. Observe what happens.

7. Keep the TCP server running but quit the terminals in Computers B and C (either type 'quit' or try Ctrl+Fn+P or Ctrl+Fn+B). Repeat Step 3 to 6 but with UDP server program and UDP client program but set display filter set to '`ip.addr==10.0.3.1 and udp`'.

Lab Questions:

1.1. In Step 4,
    a. Did you see any frames captured by Wireshark? What are they?
    b. Look in the packet contents and list all unique socket addresses for the ***data frames/messages*** exchanged between the client and the server. Explain how the client

socket addresses and the server socket address are used in the data frames/messages exchanged between the client and the server.

1.2. In Step 5, Did the server run successfully? If not, explain why.

1.3. In Step 6, Does the client successfully communicate with the server? If not, explain why.

1.4. For the UDP server and clients, Did you see the same results in Steps 4 – 6 as before ? Explain.

Part II: Threaded TCP Server

1. Run the threaded TCP server at Computer A.
2. At Computer B,
    a. Open Wireshark to capture traffic with the display filter set to '`ip.addr==10.0.3.1 and tcp`'.
    b. Run the TCP client program at Computer B with

    ```
    c:\Users\yourname\code-py3>python tcp_client.py
    ```

    Then, send a few lines of messages to the server and leave the connection open. The texts should appear on the server screen.
3. At Computer C,
    a. Open Wireshark to capture traffic with the display filter set to '`ip.addr==10.0.3.1 and tcp`'.
    b. Run the TCP client program and try to send a line of message to the server. Observe what happens.

Lab Questions:

2.1. In Step 3, Does the client successfully communicate with the server? Were the results different from those you see in Part I? Explain.

2.2. In Steps 2 and 3, look in the frame contents and list all the unique socket addresses in frames exchanged between the client and the server. How many unique socket addresses of the server did you see?