# Python Socket Programming

## Objectives

After completing this lab, students should be able to

- Familiarize with the concept of client server paradigm to implement network applications.
- Distinguish characteristics of UDP and TCP sockets.
- Explain how a TCP server handles concurrent connections.

## Equipment List

| Equipment | Quantity |
|---|---|
| RPi computers | 3 |
| Ethernet cables | 3 |
| Cisco switch | 1 |

# Lab Procedures

## Part I

## Simple TCP Socket Program

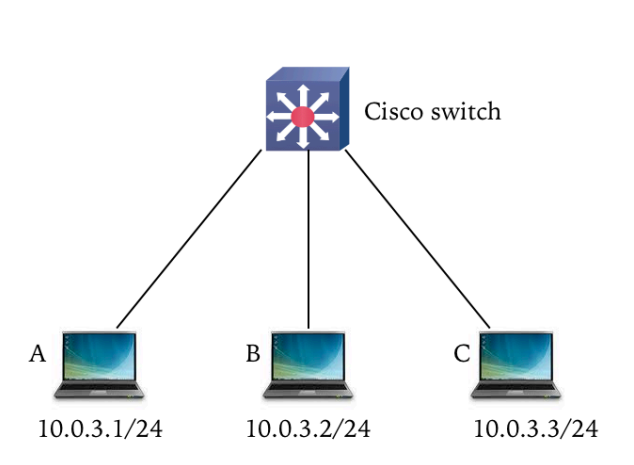


**Figure 1:** Network topology setup

Basic Python 3 client and server codes are provided for your use.

1. Turn on three RPi computers, say A, B, C, and connect their Ethernet interface to a switch by using Ethernet cables.

2. Copy the python codes from your flash drive to each computer.
3. **IP address assignment:** Assign a fixed IP address as shown in Fig. 1. For example, at Computer A, open the Terminal program and type the following commands:

   ```
   ifconfig eth0 10.0.3.1/24 up
   ```

   Repeat similar commands to configure the IP addresses of computers B and C.
4. Test the connectivity among the computers by using Ping command. At computer A, verify if it can reach B by typing the following command:

   ```
   ping 10.0.3.2
   ```

   Press `Ctrl-C` to stop the command execution. Repeat the similar command to verify the connectivity between A and C.
5. At Computer B, start Wireshark by typing the following command in the terminal:

   ```
   wireshark &
   ```

   Or you can invoke the program from the menu *Applications* → *Internet* → *Wireshark*
6. **Starting TCP server:** At Computer A, appropriately set the server socket address in `tcp_server.py`. Open Terminal and go to the directory where you save the python files. Then, start the TCP server with

   ```
   python3 tcp_server.py
   ```

7. **Starting Wireshark capture:** At Computer B,
   (a) Start traffic capture by double-clicking the interface name in the Start window. You should see a list of packets showing up in the capture window as shown in Fig. 2.
   (b) Enter `ip.addr==10.0.3.1 and tcp` in the display filter and press Enter.
8. **Running TCP client:** At Computer B,
   (a) Appropriately set the server socket address in `tcp_client.py` to match that of the server.
   (b) In the terminal, start the TCP client with

   ```
   python3 tcp_client.py
   ```

   (c) Send a few lines of messages to the server and leave the connection open. The texts should appear on the server screen.
   (d) Stop the Wireshark capture and save the capture in a pcapng file by clicking *File* → *Save As* to a flash drive.
9. At Computer A,
   (a) Open another Terminal and run the TCP server program.
   (b) Observe what happens and save the screenshot to a file.
10. At Computer C,

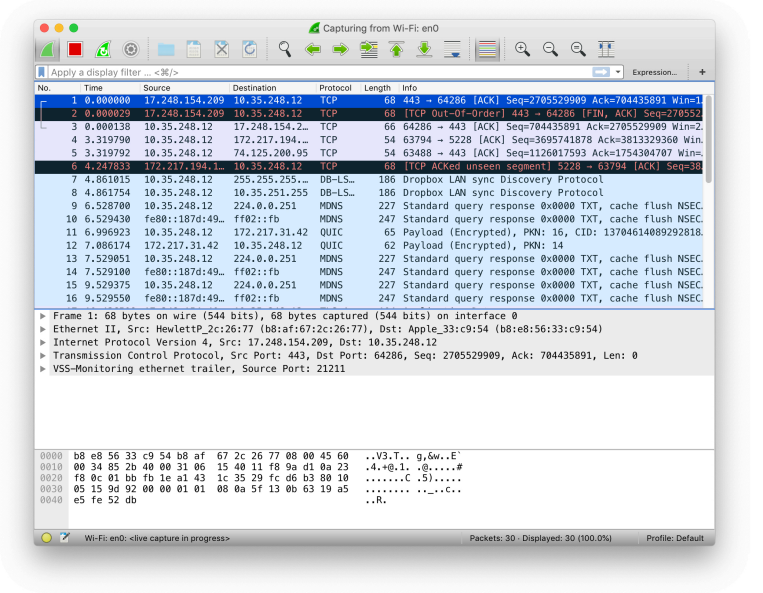


**Figure 2:** Wireshark caption option window

(a) Start the TCP client program and try sending a line of message to the server. Observe what happens and save the screenshot to a file.

(b) Press `Ctrl+C` to terminate the client.

11. Keep the TCP server at Computer A running but quit the TCP client at Computer B.
12. Repeat Steps 6 to 10 but with UDP server and UDP client.
    Change the Wireshark display filter in Step 7(b) to `ip.addr==10.0.3.1 and udp`.
13. At Computer A, stop the TCP and UDP servers by pressing `Ctrl+C`.

## Part II

## Threaded TCP Server

---

1. At Computer A, run the threaded TCP server `tcp-threaded-server.py`
2. At Computer B,

   (a) Open Wireshark to capture traffic with the display filter `ip.addr==10.0.3.1 and tcp`.

   (b) Open two terminals and run the TCP client program in each terminal with

   ```
   python3 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 at Computer A.

   (c) Stop the Wireshark capture and save the capture in a pcapng file by clicking *File* → *Save As* to a thumb drive.

3. At Computer C,

   (a) Open Wireshark to capture traffic with the display filter `ip.addr==10.0.3.1 and tcp`.

   (b) Open two terminals and run the TCP client program in each terminal. Try sending a line of message to the server. Save the screenshot of the terminal running the TCP server at Computer A to

a file.

(c) Stop the Wireshark capture and save the capture in a pcapng file by clicking *File* → *Save As* to a thumb drive.

4. Shutdown RPi computers, turn-off the power switch, and roll up the cables.

---

# Part I

## Simple TCP Socket Program

### Lab Questions:

From the Wireshark captured files in Step 8,

I.1 (5 pts) In Step 8 when running the TCP client,

(a) (2 pts) What frames are captured by Wireshark in Steps 8(b) and 8(c)? Explain.

(b) (3 pts) 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.

I.2 (2 pts) In Step 9, Did the TCP server run successfully? If not, explain why.

I.3 (2 pts) In Step 10, Did the client successfully connect to the server? If not, explain why.

I.4 (4 pts) When running the UDP server and clients in Step 12,

(a) (2 pts) Are the frames captured in Steps 8(b) and 8(c) different from those captured when running the TCP server? Explain.

(b) (2 pts) When you ran the UDP client at Computer C in Step 10, did you see the same result like when running the TCP client? Explain.

# Part II

## Threaded TCP Server

### Lab Questions:

II.1 (2 pts) In Step 3(b), Did the clients successfully connect to the server? Were the results different from those you see in Part I? Explain.

Based on the Wireshark captures in Steps 2(c) and 3(c)

II.2 (3 pts) How many socket addresses belong to the server? What are they? How many sockets did the server create?

II.3 (3 pts) How many socket addresses belong to the clients? What are they? How many sockets were created in each client computer?