ELEN 4017 – Network Fundamentals

LAB # 2 - Socket Programming

Objective:

1. Get introduced to network/socket programming.
2. Get familiarized with the classes and methods in Python language which are used for basic socket programming.
3. Create a basic Client and a Server based on TCP and UDP sockets in Python language.

Instructions:

1. This lab must be completed individually by each student.
2. An individual lab report, consisting of responses to the tasks given in this handout, must be submitted to the demonstrators for marking.
3. In addition to the report, demonstrators may ask questions to test your understanding of the material and marks for the lab will be allocated based on your ability to answer the questions.
4. Consult the references in the end to perform your tasks. Section 2.7 of Kurose book (6^{th} edition) has information on how to create a basic client and server in Python.
5. You MUST run the server first before running the client.
6. Please do not use port numbers below 1024 as they may be reserved for some existing networking service/application.
7. The usage of different python classes/methods might vary based on the version of Python being used.

Introduction:

As per our discussions from the class, a networking application consists of a client process and a server process which communicate with each other via *sockets*. *Sockets* are interfaces/ APIs with which the application layer/ network application developer interacts with the rest of the network. Recall that there are two types of sockets 1) TCP sockets and 2) UDP sockets.
TCP is connection oriented and provides a reliable byte-stream channel through which data flows between two end systems. UDP is connectionless and sends independent packets of data from one end system to the other, without any guarantees about delivery [1].

Below is a sequence of actions that need to be implemented by a TCP server and client in order to establish a client server application. Python class/method corresponding to each action is also mentioned alongside each action in parentheses.

TCP Server:

1. Server creates a TCP socket. (socket.socket())
2. It then chooses a port number and IP address and attaches the socket with this port number and IP address. (bind())

3. Listens for an incoming TCP connection request from the client
4. If a connection request is received from a client, it creates a new socket for the connection with that particular client. (accept())
5. Starts sending and receiving data from the client. (send() and recv())
6. Closes all the sockets, if no further communication with clients is intended. (close())

TCP Client:

1. If server is already running:
    a. Get IP address and port number of server
    b. Create a TCP socket (socket.socket())
    c. Send a TCP connection request from the client socket to the server socket (connect()). If success
        i. Send and receive data with the server
        ii. Close sockets when no more communication with the server is intended

In this lab we aim to create basic client and server processes for an 'echo' application by making use of relevant classes and methods in Python language.

Tasks:

1. Based on the sequence of actions mentioned above for TCP client and server, create a basic TCP server and a TCP client. For simplicity, both the client and the server can run on the same PC. The client process requires input from the user, and then sends it to the server process. The server process prints the sentence upon receiving it.
.
2. Modify the code so that the server echoes back whatever it listens from the client
3. Modify the code so that data could be sent more than once from client to server and the server only closes the connection upon receiving a specific command e.g. stop, exit etc.
4. Modify the code so that the client and the server code makes use of UDP sockets instead of TCP sockets. Note that no connection is required for UDP transfers. You can make use of the arguments of the socket.socket() method to make either a TCP or a UDP socket.

References:

1. J. Kurose and K. Ross, "Computer Networking: A Top Down Approach-(Section 2.7)", 6th Edition. Pearson Education. 2012
2. https://docs.python.org/3/library/socket.html