|  |  |  |
| --- | --- | --- |
|  | **Namal Institute Mianwali** |  |
| Computer Networks Laboratory Manual #5 |
|  |
| **Network Programming using Sockets API: Echo Server** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Course Title | Computer Networks | Course Number | CS – 270 L |
| Instructor | Shahzad Arif  shahzad.arif@namal.edu.pk | Lab Engineer | Asad Majeed  [asad.majeed@namal.edu.pk](mailto:asad.majeed@namal.edu.pk) |

# 1. Introduction:

A set of applications prevailing today, are network. For example, using MS Word on local host is shifting to Cloud based MS Office365; where documents are written, formatted over the network. In the coming years, most of the applications shall have an instance running on the web. In the previous Laboratory exercises, we learnt how the network is implemented for a few services. What if we want to write our own application? In this Lab and subsequent labs, we shall develop our knowledge of the Application Programming Interface (API) used to interact with the Operating System, to pass messages over the internet i.e. SOCKETS.

# 2. Objective:

* To learn the Client – Server paradigm of programming used in networks.
* To learn the fundamental issues arising in establishing a communication.
* To learn the use of SOCKETs API to write the first code to pass messages over the network.

# 3. Methodology (Software / Tools):

Although Socket API is written in C but it has been translated (ported) to other languages such as “C++”, “Python”, “Java”, “ Bash” etc. Even programming environment such as MATLAB or OCTAVE provide wrapper functions for Sockets.

We shall encourage the students to learn the Socket programming by programming in C, and its equivalent wrapper functions in Python 3+.

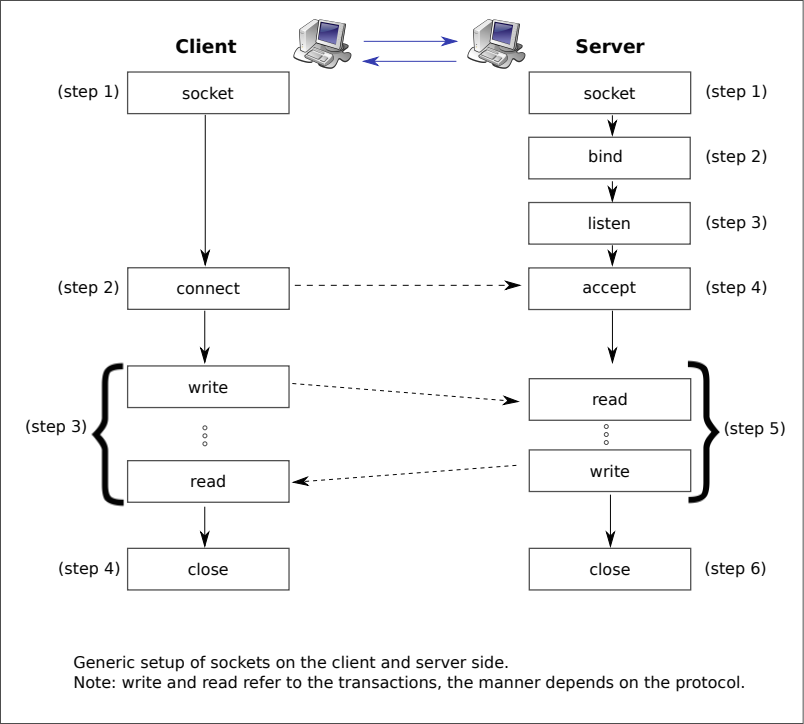
Since Cisco Packet Tracer allows a programming interface for Python and Java, only python-based code can be used in testing applications in Packet Tracer.

# 4. Background:

An application and the transport layer exchange messages via special interfaces called sockets. To ease software development, applications use well defined APIs (Application programming interfaces), which hide away intricate details of the actual process from the application developer. Therefore, for an operating system, an application to access data over a network is one process and sending data over the network interface is another process, the application can pass its data via sockets which are bound to specific ports. In this exercise, TCP/IP/Ethernet protocols are used for network data communications, whereas sockets are used by the software you write to pass data to them (application).

Sockets allow you to exchange information between processes on the same machine or across a network. A wide range of operating systems support socket APIs.

Figure 1: Client-Server paradigm, implemented in Sockets



Before sockets could be programmed, client-server model session initiation must be understood. A simple session initiation is depicted in Fig 1. The server is programmed to be in wait state. The client sends a request to establish a connection. The server acknowledges, and the available port numbers are exchanged, so that the data knows which port to use for communication; in this manner, multiple services can be run in parallel over the network using other ports e.g., data sharing using peer to peer networks. In addition, a basic knowledge of the network in terms of hostname, or the IP address of the server, and the port numbers available for use must be known to the programmer. The ports are generally managed by the OS.

In the client-server model, the client should know the hostname and the port number on which the server is listening, while trying to fix the port number for incoming acknowledgements from server. These ports remain fixed for the duration of communication. If the server accepts the connection request, the server keep a note of the assigned ports, both at the client side as well as the server side. The server needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client. On the client side, if the connection is accepted, a socket is successfully created, and the client can use the socket to communicate with the server.

The client and server can now communicate by writing to or reading from their sockets.

# 5. Lab Activities:

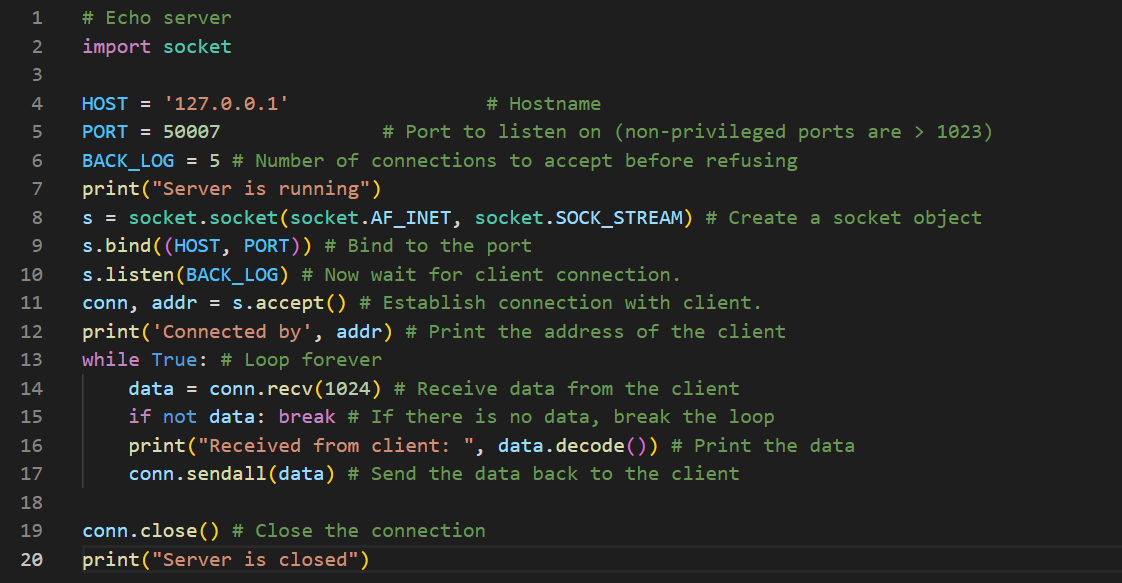
This laboratory session and subsequent sessions are divided into tasks. Each tasks requires its own setup, configuration and associated learning. You are going to be assessed on the performance in implementing the lab tasks under supervision of lab-engineer. Without further ado, let’s get our hands dirty.

## 5.1. Task No. 1:

## Write a basic Server and Client Application on local host.

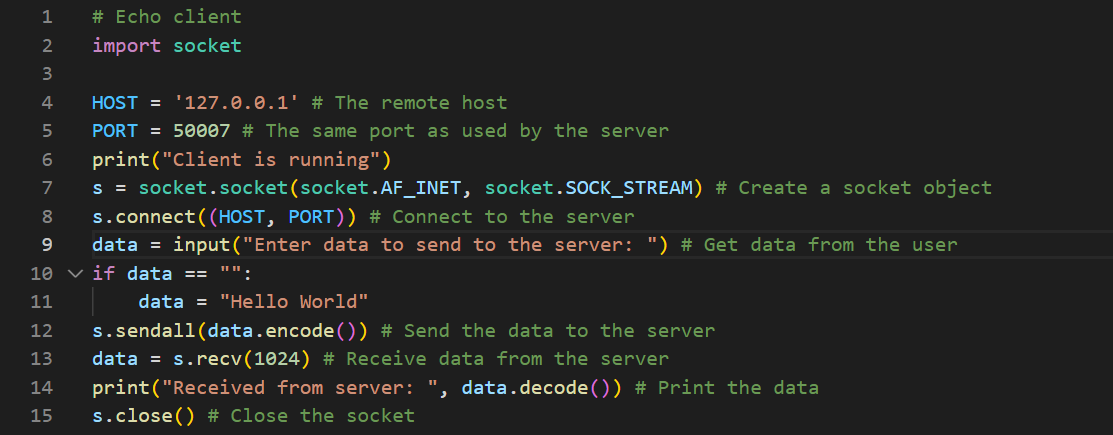
To complete this task, you need to have python installed on the system. Using your favourite editor, write the code as shown in the Figure 2, saving the file as **“server.py”**.

Figure 2: Listings for a simple server in Python



Similarly, in a separate file, type in the code listed in Figure 3 for client, and save as **“client.py”**.

Figure 3: Listings for Client in Python



Note down a major difference in the code.

***Observations / Reasons:***

### 5.1.1 Test the Client-Server communications:

To test the server, client communication, follow the following steps

* In a terminal type: *python server.py*
* In a separate terminal type: *python client.py*

1. Note down the messages shown in terminal. Write down the message which was observed in the screen.
2. Identify, which of the messages was passed over the network.

### 5.1.2. Understanding the Socket Interface:

Using online help write down the purpose of the following functions and parameters

1. Socket

A socket connecting to the network is created at each end of the communication. The socket has a specific address. This address is composed of an IP address and a port number. The socket is generally employed in client-server applications.

1. Connect

This is very simple to create a socket client using Python's socket module function. The socket. connect(hosname, port ) **opens a TCP connection to hostname on the port**. Once we have a socket open, we can read from it like any IO object.

1. Listen

Socket programming is **a way of connecting two nodes on a network to communicate with each other**. One socket(node) listens on a particular port at an IP, while another socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.

1. Bind

The bind() function **binds a unique local name to the socket with descriptor socket**.

1. Accept

The accept() function shall extract the first connection on the queue of pending connections, create a new socket with the same socket type protocol and address family as the specified socket, and allocate a new file descriptor for that socket

1. Recv

The recv function is **used to read incoming data on connection-oriented sockets, or connectionless sockets**. When using a connection-oriented protocol, the sockets must be connected before calling recv.

1. Send

The function socket send () **sends length bytes to the socket from data**.

1. AF\_INET

This address family **provides interprocess communication between processes that run on the same system or on different systems**. Addresses for AF\_INET sockets are IP addresses and port numbers.

1. SOCK\_STREAM

**Provides sequenced, two-way byte streams with a transmission mechanism for stream data**. This socket type transmits data on a reliable basis, in order, and with out-of-band capabilities.

1. PORT

A port in networking is **a software-defined number associated to a network protocol that receives or transmits communication for a specific service**.

### 5.1.3. Improving the Server with Iterative Operations:

We can make the server running continuously by placing the accept connections, and subsequent read, write transactions, in a loop. Implement the improved version of server shown in Figure 4.

Note down what happens if you remove “conn.close()”.

Figure 4: Improved version of server

