Atul Anurag
PES2UG19CS075
COMPUTER NETWORKS LAB

Week #5

Simple Client-Server Application using Network Socket Programming

Objective:

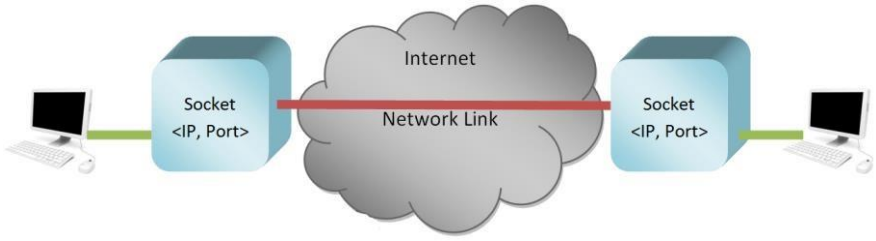
To develop a simple Client-Server application using TCP and UDP.

Pre requisites:

- Basic understanding of networking concepts and socket programming
- Knowledge of python

Sockets

Sockets are just the **endpoints of a two-way communication link** in a network. Socket helps in the communication of two processes/programs on a network (eg. Internet). The programs can communicate by reading/writing via their sockets. A socket comprises of: ***IP Address & Port number***



Task 1: (Mandatory for all students)

1. Create an application that will

a. Convert lowercase letters to uppercase

- e.g. [a…z] to [A…Z]
- code will not change any special characters, e.g. &*!

b. If the character is in uppercase, the program must not alter

2. Create Socket API both for client and server.
3. Must take the server address and port from the Command Line Interface (CLI).

Socket Programming with UDP *UDPClient.py*

UDPClient.py

```
from socket import *
serverName = '10.0.2.15'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message,(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print(modifiedMessage)
clientSocket.close()
```

UDPServer.py

UDPServer.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print("The server is ready to receive")
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

Socket Programming with TCP *TCPClient.py*

TCPClient.py

```
from socket import *
serverName = '10.0.2.15'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

TCPServer.py

TCPServer.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print 'The server is ready to receive'
while 1:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

UDP:

```
itsatul@itsatul-VirtualBox:~$ gedit UDPServer.py
itsatul@itsatul-VirtualBox:~$ python UDPServer.py
The server is ready to receive
```

```
itsatul@itsatul-VirtualBox:~$ gedit UDPClient.py
itsatul@itsatul-VirtualBox:~$ python UDPClient.py
Input lowercase sentence:networks lab week5
NETWORKS LAB WEEK5
itsatul@itsatul-VirtualBox:~$
```

TCP:

```
itsatul@itsatul-VirtualBox:~$ gedit TCPServer.py
itsatul@itsatul-VirtualBox:~$ python TCPServer.py
The server is ready to receive
```

```
itsatul@itsatul-VirtualBox:~$ gedit TCPClient.py
itsatul@itsatul-VirtualBox:~$ python TCPClient.py
Input lowercase sentence:socket programming
From Server: SOCKET PROGRAMMING
itsatul@itsatul-VirtualBox:~$
```

Problems:

Install and compile the Python programs TCPClient and UDPClient on one host and TCPServer and UDPServer on another host.

1. Suppose you run TCPClient before you run TCPServer. What happens? Why?

   In case of TCP the client side throws the following Connection refused error if TCPClient is run before TCPServer:

```
itsatul@itsatul-VirtualBox:~$ python TCPClient.py
Traceback (most recent call last):
  File "TCPClient.py", line 5, in <module>
    clientSocket.connect((serverName,serverPort))
  File "/usr/lib/python2.7/socket.py", line 228, in meth
    return getattr(self._sock,name)(*args)
socket.error: [Errno 111] Connection refused
itsatul@itsatul-VirtualBox:~$
```

   This is because in TCP a connection is required between the server and the client.

2. Suppose you run UDPClient before you run UDPServer. What happens? Why?

   In case of UDP, the client side shows no error nor any other significant changes except that the cursor stays on the terminal.

   This is because in UDP connection is not required to run the UDPClient

```
itsatul@itsatul-VirtualBox:~$ python UDPClient.py
Input lowercase sentence:hello
```

3. What happens if you use different port numbers for the client and server sides?

   In this case a connection refused message is displayed

```
itsatul@itsatul-VirtualBox:~$ python TCPClient.py
Traceback (most recent call last):
  File "TCPClient.py", line 5, in <module>
    clientSocket.connect((serverName,serverPort))
  File "/usr/lib/python2.7/socket.py", line 228, in meth
    return getattr(self._sock,name)(*args)
socket.error: [Errno 111] Connection refused
itsatul@itsatul-VirtualBox:~$
```

Task 2: Web Server

In this assignment, you will develop a simple Web server in Python that is capable of processing only one request. Specifically, your Web server will

a) create a connection socket when contacted by a client (browser);
b) receive the HTTP request from this connection;
c) parse the request to determine the specific file being requested;
d) get the requested file from the server's file system;
e) create an HTTP response message consisting of the requested file preceded by header lines; and
f) send the response over the TCP connection to the requesting browser.

If a browser requests a file that is not present in your server, your server should return a "404 Not Found" error message.

For this assignment, the companion Web site provides the skeleton code for your server. Your job is to complete the code, run your server, and then test your server by sending requests from browsers running on different hosts. If you run your server on a host that already has a Web server running on it, then you should use a different port than port 80 for your Web server.

When File is present:

When File is not present:

WIRESHARK SCREENSHOT: