

Atul Anurag
PES2UG19CS075
COMPUTER NETWORKS LAB

Week #5

Simple Client-Server Application using Network Socket Programming

Objective:

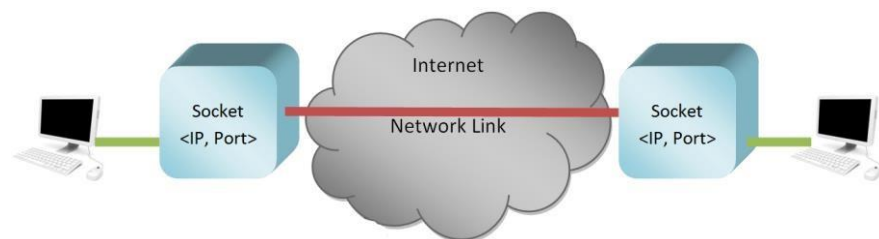
To develop a simple Client-Server application using TCP and UDP.

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 x
1 from socket import *
2 serverName = '10.0.2.15'
3 serverPort = 12000
4 clientSocket = socket(AF_INET, SOCK_DGRAM)
5 message = raw_input('Input lowercase sentence:')
6 clientSocket.sendto(message, (serverName, serverPort))
7 modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
8 print(modifiedMessage)
9 clientSocket.close()
10
```

UDPServer.py

```
UDPServer.py x
1 from socket import *
2 serverPort = 12000
3 serverSocket = socket(AF_INET, SOCK_DGRAM)
4 serverSocket.bind(('', serverPort))
5 print("The server is ready to receive")
6 while 1:
7     message, clientAddress = serverSocket.recvfrom(2048)
8     modifiedMessage = message.upper()
9     serverSocket.sendto(modifiedMessage, clientAddress)
10
```

Socket Programming with TCP *TCPClient.py*

```
TCPClient.py x
1 from socket import *
2 serverName = '10.0.2.15'
3 serverPort = 12000
4 clientSocket = socket(AF_INET, SOCK_STREAM)
5 clientSocket.connect((serverName, serverPort))
6 sentence = raw_input('Input lowercase sentence:')
7 clientSocket.send(sentence)
8 modifiedSentence = clientSocket.recv(1024)
9 print 'From Server:', modifiedSentence
10 clientSocket.close()
11
```

TCPServer.py

```
TCPServer.py x
1 from socket import *
2 serverPort = 12000
3 serverSocket = socket(AF_INET, SOCK_STREAM)
4 serverSocket.bind(('', serverPort))
5 serverSocket.listen(1)
6 print 'The server is ready to receive'
7 while 1:
8     connectionSocket, addr = serverSocket.accept()
9     sentence = connectionSocket.recv(1024)
10    capitalizedSentence = sentence.upper()
11    connectionSocket.send(capitalizedSentence)
12    connectionSocket.close()
13
```

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

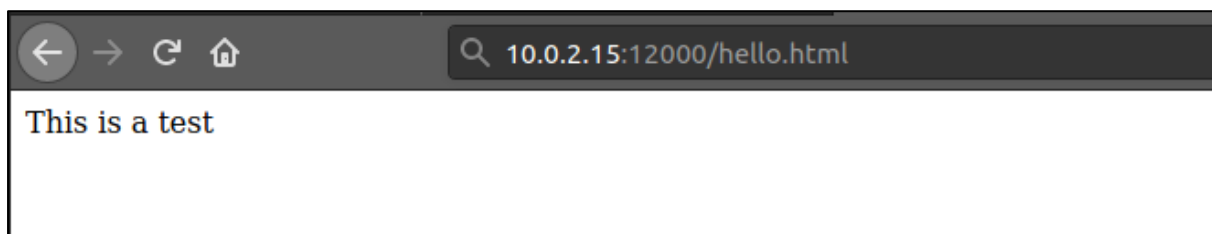
In this assignment, you will develop a simple Web server in Python that is capable of processing only one request. Specifically, your Web server will

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

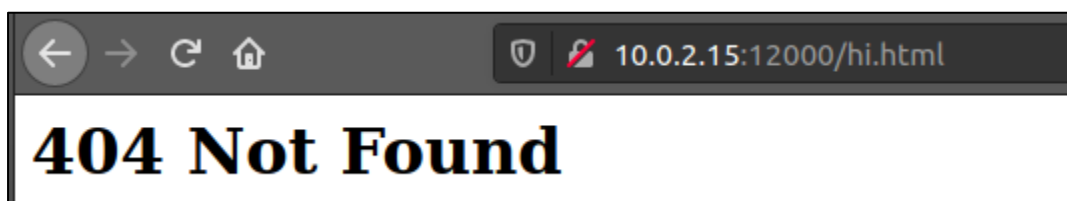
If a browser requests a file that is not present in your server, your server should return a “404 Not Found” error message.

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:

http

No.	Time	Source	Destination	Protocol	Length	Info
14	11.027394361	10.0.2.4	10.0.2.15	HTTP	438	GET /hello.html HTTP/1.1
19	11.027661659	10.0.2.15	10.0.2.4	HTTP	68	HTTP/1.1 200 OK
80	69.360898728	10.0.2.4	10.0.2.15	HTTP	409	GET /hi.html HTTP/1.1
83	69.361054937	10.0.2.15	10.0.2.4	HTTP	131	HTTP/1.1 404 Not Found

Frame 14: 438 bytes on wire (3504 bits), 438 bytes captured (3504 bits) on interface any, id 0

Linux cooked capture

Internet Protocol Version 4, Src: 10.0.2.4, Dst: 10.0.2.15

Transmission Control Protocol, Src Port: 60542, Dst Port: 12000, Seq: 1, Ack: 1, Len: 370

Hypertext Transfer Protocol

0000 00 00 00 01 00 06 08 00 27 bb 1d 9e 00 00 08 00
0010 45 00 01 a6 31 1e 40 00 40 06 f0 21 0a 00 02 04 E...1@_@:!
0020 0a 00 02 0f ec 7e 2e e0 f0 dc 53 5d 3e 99 db a3S]>
0030 80 18 01 f6 39 09 00 00 01 01 08 0a 0a 44 3f 949.....D?
0040 85 46 37 ca 47 45 54 20 2f 68 65 6c 6c 6f 2e 68 .F7 GET /hello.h
0050 74 6d 6c 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f tml HTTP /1.1 Ho
0060 73 74 3a 20 31 30 2e 30 2e 32 2e 31 35 3a 31 32 st: 10.0 .2.15:12
0070 30 30 30 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 000 Use r-Agent:
0080 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 28 58 31 Mozilla /5.0 (X1
0090 31 3b 20 55 62 75 6e 74 75 3b 20 4c 69 6e 75 78 1; Ubuntu; Linux
00a0 20 78 38 36 5f 36 34 3b 20 72 76 3a 38 35 2e 30 x86_64; rv:85.0
00b0 29 20 47 65 63 6b 6f 2f 32 30 31 30 30 31 30 31) Gecko/20100101
00c0 20 46 69 72 65 66 6f 78 2f 38 35 2e 30 0d 0a 41 Firefox/85.0 A
00d0 63 63 65 70 74 3a 20 74 65 78 74 2f 68 74 6d 6c ccept: text/html
00e0 2c 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 68 74 ,applicati on/xht
00f0 6d 6c 2b 78 6d 6c 2c 61 70 70 6c 69 63 61 74 69 ml+xml, a pplicati
0100 6f 6e 2f 78 6d 6c 3b 71 3d 30 2e 39 2c 69 6d 61 on/xml;q =0.9, ima
0110 67 65 2f 77 65 62 70 2c 2a 2f 2a 3b 71 3d 30 2e ge/webp, */*;q=0.

Hypertext Transfer Protocol (http), 370 bytes

Packets: 123 - Displayed: 4 (3.3%) - Dropped: 0 (0.0%)

Profile: Default