L. D. College of Engineering

Ahmedabad – 380015

Lab Manual

Subject Name: Computer Networking

(629403)

MCA Semester – 2

Academic year: 2020-21

Certificate

This is to certify that **Mr. Parth Kukadiya** having enrolment no. **205160694013** of **MCA Semester – 2** has satisfactorily completed course in **Computer Networking** at L. D. College of Engineering, Ahmedabad – 380015.

Date of Submission: 30/07/2021

Staff in-charge:

Head of Department:

Index

Tasks	Topics (Programs) to be Completed	Date	Faculty Sign
1.	Implement a Python Program to print host name and IP address of local host.		
2.	Implement a Python Program to print host name and IP address of remote host where IP address of remote host is available.		
3.	Implement a Python Program to print host name and IP address of remote host where hostname of remote host is available.		
4.	Implement a TCP port scanner program in python for local host. (Note: Do not try this program for a remote host, especially outside your domain. It could cause legal problems)		
5.	Implement a UDP port scanner program in python for local host. (Note: Do not try this program for a remote host, especially outside your domain. It could Cause Legal problems)		
6.	Implement a TCP based client server program in python using TCP sockets where server displays the following: a) Host Name, IP address and Port Number on which it is hosted b) IP address and port number of a client requesting connection. Server sends the message "Thanks for Connecting!" back to client. Client displays this message on screen.		
7.	Implement a UDP based client server program in python using UDP sockets where Server displays the following: a) Host Name, IP address and Port Number on which it is hosted b) IP address and port number of a client sending some dummy message. Server displays the dummy message on screen. Server sends the message "Thanks for Message!" back to client. Client displays this message on screen.		
8.	Implement a TCP based echo client server program in python.		
9.	Implement a UDP based echo client server program in python.		
10.	Implement a TCP based daytime client server program in python.		
11.	Implement a UDP based daytime client server program in python.		
12.	Implement a TCP based client server text chat program in python.		
13.	Implement a UDP based client server text chat program in python.		
14.	Implement a TCP based echo client server program in python with a multi-threaded server.		
15.	Implement a TCP based daytime client server program in python with a multi-threaded server.		
16.	Implement a web client using urllib to: a) Display the html source of a given URL on screen b) Display the URL visited c) Display the header information transmitted in the http response sent by the contacted web-site/web-server. d) Display the http server status code		

17.	Implement an ftp client using ftplib which connects to an ftp server, takes login/password from user, displays directory list and permits upload and download of files.		
18.	Write a Python program that makes a connection to a web server and retrieve/display a document using the HTTP protocol.		
19.	Write a Python program that makes a connection to a web server and retrieve an image using the HTTP protocol.		
20.	Write a python program to implement a simple server-client program.		
21.	Write a python program to implement socket programming using multi-threading.		

1.Implement a Python Program to print host name and IP address of local host.

Source Code:

```
import socket;
host_name=socket.gethostname()
host_ip=socket.gethostbyname(host_name)
print("Hostname : ",host_name)
print("IP : ",host_ip)
```

Output:

```
Hostname :  LAPTOP-FHOFG2OR
IP :  192.168.0.104
```

2. Implement a Python Program to print host name and IP address of remote host where IP
address of remote host is available.

Source Code:

```
import socket
def check_ip():
    try:
        hname = socket.gethostname()
        hip = socket.gethostbyname( hname )
        print( "Host name : ", hname )
        print( 'IP Address : ', hip )
    except:
        print( 'unable to get IP address' )
        check_ip()
```

Output:

```
Hostname :  LAPTOP-FHOFG2OR
IP :  192.168.0.104
```

3. Implement a Python Program to print host name and IP address of remote host where

hostname of remote host is available.

Source Code:

```
import socket
def check_hostname():
    try:
        hname=socket.gethostname()
```

```
        hip=socket.gethostbyname(hname)
        print("Host name : ",hname)
        print('IP Address : ',hip)
    except:
        print('unable to get IP address')
        check_hostname()
```

Output:

```
Hostname :  LAPTOP-FHOFG2OR
IP :  192.168.0.104
```

4. Implement a TCP port scanner program in python for local host.

Source Code:

```
from socket import *
import time
startTime = time.time()
if __name__ == '__main__':
    target = input('Enter the host to be scanned: ')
    t_IP = gethostbyname(target)
    print ('Starting scan on host: ', t_IP)
    for i in range(50, 500):
        s = socket(AF_INET, SOCK_STREAM)
        conn = s.connect_ex((t_IP, i))
if(conn == 0) :
    print( 'Port %d: OPEN' % (i,) )
    s.close()
    print( 'Time taken:', time.time() - startTime )
```

Output:

```
Enter the IP address: 192.168.0.102
Enter the Starting Number: 1
Enter the Last Number: 5
Scanning completed in:  0:00:06.089865
```

5. Implement a UDP port scanner program in python for local host. (Note: Do not try this program

for a remote host, especially outside your domain. It could Cause Legal problems)

Source Code:

```
import pyfiglet
import sys
import socket
from datetime import datetime

ascii_banner = pyfiglet.figlet_format("PORT SCANNER")
print(ascii_banner)

# Defining a target
if len(sys.argv) == 2:

    # translate hostname to IPv4
    target = socket.gethostbyname(sys.argv[1])
else:
    print("Invalid ammount of Argument")

# Add Banner
print("-" * 50)
print("Scanning Target: " + target)
print("Scanning started at:" + str(datetime.now()))
print("-" * 50)

try:

    # will scan ports between 1 to 65,535
    for port in range(1, 65535):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        socket.setdefaulttimeout(1)

        # returns an error indicator
```

```
        result = s.connect_ex((target, port))
        if result == 0:
            print("Port {} is open".format(port))
        s.close()

except KeyboardInterrupt:
    print("\n Exitting Program !!!!")
    sys.exit()
except socket.gaierror:
    print("\n Hostname Could Not Be Resolved !!!!")
    sys.exit()
except socket.error:
    print("\ Server not responding !!!!")
    sys.exit()
```

Output:

```
 ____   ___  ____ _____   ____   ____    _    _   _ _   _ _____ ____
|  _ \ / _ \|  _ \_   _| / ___| / ___|  / \  | \ | | \ | | ____|  _ \
| |_) | | | | |_) || |   \___ \| |     / _ \ |  \| |  \| |  _| | |_) |
|  __/| |_| |  _ < | |    ___) | |___ / ___ \| |\  | |\  | |___|  _ <
|_|    \___/|_| \_\|_|   |____/ \____/_/   \_\_| \_|_| \_|_____|_| \_\
```

6. Implement a TCP based client server program in python using TCP sockets where Server displays the following: a) Host Name, IP address and Port Number on which it is hosted b) IP address and port number of a client requesting connection. Server sends the message "Thanks for Connecting!" back to client. Client displays this message on screen.

Source Code:

Server Side :

```
import socket
server_socket=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
port=12345
server_socket.bind(("127.0.0.1",port))
server_socket.listen(5)
```

```
while True:
print("Server waiting for connection")
client_socket,addr=server_socket.accept()
data=client_socket.recv(1024)
if not data or data.decode("utf-8")=="END":
break
print("received from client client: ",data.decode("utf-8"))
try:
hname=socket.gethostname()
hip=socket.gethostbyname(hname)
print("host-name: ",hname)
print("IP address: ",hip)
print("port number: ",port)
print("client connected from",addr)
client_socket.send(bytes("Thanks for connecting",'utf-8'))
except:
print("Exited by the user")
client_socket.close()
server_socket.close()
```

Client Side :

```
import socket
client_socket=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
client_socket.connect(("127.0.0.1",12345))
payload="hey server"
try:
client_socket.send(payload.encode('utf-8'))
msg=client_socket.recv(1024)
print(msg)
except KeyboardInterrupt:
print("exit by user")

client_socket.close()
```

Output:

```
You pressed Ctrl+C
PS D:\MCA LDCE\Sem 2\Computer Network> python 6Serve
r.py
Server waiting for connection
received from client client:  hey server
host-name:  ZeelChauhan
IP address:  127.0.0.1
port number:  12345
client connected from ('127.0.0.1', 1359)
Server waiting for connection
received from client client:  hey server
host-name:  ZeelChauhan
IP address:  127.0.0.1
port number:  12345
client connected from ('127.0.0.1', 1360)
Server waiting for connection
```

```
PS D:\MCA LDCE\Sem 2\Computer Network> python 6client.py
b'Thanks for connecting'
```

7. Implement a UDP based client server program in python using UDP sockets where Server displays the following:

a) Host Name, IP address and Port Number on which it is hosted

b) IP address and port number of a client sending some dummy message. Server displays the dummy message on screen. Server sends the message "Thanks for Message!" back to client. Client displays this message on screen.

Source Code:

```
import socket
localIP = "127.0.0.1"
localPort = 20001
bufferSize = 1024
msgFromServer = "Thanks for message"
bytesToSend = str.encode(msgFromServer)
# Create a datagram socket

UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
# Bind to address and ip
UDPServerSocket.bind((localIP, localPort))
print("UDP server up and listening")
# Listen for incoming datagrams
while(True):
bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)
message = bytesAddressPair[0]
```

```
address = bytesAddressPair[1]
clientMsg = "Message from Client:{}".format(message)
clientIP = "Client IP Address:{}".format(address)
hname=socket.gethostname()
print("server host: ",hname)
print("server IP: ",localIP)
print("port: ",localPort)
print(clientMsg)
print(clientIP)
# Sending a reply to client
UDPServerSocket.sendto(bytesToSend, address)
```

Client :

```
import socket
msgFromClient = "Hello UDP Server"
bytesToSend = str.encode(msgFromClient)
serverAddressPort = ("127.0.0.1", 20001)
bufferSize = 1024
# Create a UDP socket at client side
UDPClientSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
# Send to server using created UDP socketUDPClientSocket.sendto(bytesToSend,
serverAddressPort)
msgFromServer = UDPClientSocket.recvfrom(bufferSize)
msg = "Message from Server {}".format(msgFromServer[0])
print(msg)
```

Output:

```
PS D:\MCA LDCE\Sem 2\Computer Network> python 7Server.py
UDP server up and listening
server host:  ZeelChauhan
server IP:  127.0.0.1
port:  20001
Message from Client:b'Hello UDP Server'
Client IP Address:('127.0.0.1', 64224)
```

```
PS D:\MCA LDCE\Sem 2\Computer Network> python 7client.py
Message from Server b'Thanks for message'
PS D:\MCA LDCE\Sem 2\Computer Network>
```

8. Implement a TCP based echo client server program in python

<u>CODE:</u>

<u>Python-TCP-Server.py</u>

```
import socketserver


class Handler_TCPServer(socketserver.BaseRequestHandler):
```

```
    def handle(self):
        # self.request - TCP socket connected to the client
        self.data = self.request.recv(1024).strip()
        print("{} sent:".format(self.client_address[0]))
        print(self.data)
        # just send back ACK for data arrival confirmation
        self.request.sendall("ACK from TCP Server".encode())

if __name__ == "__main__":
    HOST, PORT = "localhost", 9999

    # Init the TCP server object, bind it to the localhost on 9999 port
    tcp_server = socketserver.TCPServer((HOST, PORT), Handler_TCPServer)

    # Activate the TCP server.
    # To abort the TCP server, press Ctrl-C.
    tcp_server.serve_forever()
```

Python-TCP-Client.py

```
import socket

host_ip, server_port = "127.0.0.1", 9999
data = " Hello how are you?\n"

# Initialize a TCP client socket using SOCK_STREAM
tcp_client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
    # Establish connection to TCP server and exchange data
    tcp_client.connect((host_ip, server_port))
```

```
    tcp_client.sendall(data.encode())


    # Read data from the TCP server and close the connection
    received = tcp_client.recv(1024)
finally:
    tcp_client.close()


print ("Bytes Sent:     {}".format(data))
print ("Bytes Received: {}".format(received.decode()))
```

OUTPUT:

Python-TCP-Server.py

```
Hello How are you??
127.0.0.1 sent:
b'Hello how are you?'
```

Python-TCP-Client.py

```
Bytes Sent:      Hello how are you?

Bytes Received: ACK from TCP Server

Process finished with exit code 0
```

9. Implement a UDP based echo client server program in python.

CODE:

Python-UDP-Server.py

```
import socket

localIP = "127.0.0.1"

localPort = 20001

bufferSize = 1024

msgFromServer = "Hello UDP Client"

bytesToSend = str.encode(msgFromServer)

# Create a datagram socket
```

```
UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)

# Bind to address and ip

UDPServerSocket.bind((localIP, localPort))

print("UDP server up and listening")

# Listen for incoming datagrams

while (True):
    bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)

    message = bytesAddressPair[0]

    address = bytesAddressPair[1]

    clientMsg = "Message from Client:{}".format(message)
    clientIP = "Client IP Address:{}".format(address)

    print(clientMsg)
    print(clientIP)

    # Sending a reply to client

    UDPServerSocket.sendto(bytesToSend, address)
```

Python-UDP-Client.py

```
import socket

msgFromClient = "Hello UDP Server"

bytesToSend = str.encode(msgFromClient)

serverAddressPort = ("127.0.0.1", 20001)

bufferSize = 1024

# Create a UDP socket at client side

UDPClientSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)

# Send to server using created UDP socket
```

```
UDPClientSocket.sendto(bytesToSend, serverAddressPort)

msgFromServer = UDPClientSocket.recvfrom(bufferSize)

msg = "Message from Server {}".format(msgFromServer[0])

print(msg)
```

OUTPUT:

Python-UDP-Server.py

```
UDP server up and listening

Hey UDP Server
Message from Client:b'Hello UDP Server'
Client IP Address:('127.0.0.1', 55788)
```

Python-UDP-Client.py

```
Message from Server b'Hello UDP Client'

Process finished with exit code 0
```

10. Implement a TCP based daytime client server program in python

CODE:

DayTime-Server.py

```
# server.py
import socket
import time

# create a socket object
serversocket = socket.socket(
                socket.AF_INET, socket.SOCK_STREAM)

# get local machine name
host = socket.gethostname()

port = 9999
```

```
# bind to the port
serversocket.bind((host, port))

# queue up to 5 requests
serversocket.listen(5)

while True:
    # establish a connection
    clientsocket,addr = serversocket.accept()

    print("Got a connection from %s" % str(addr))
    currentTime = time.ctime(time.time()) + "\r\n"
    clientsocket.send(currentTime.encode('ascii'))
    clientsocket.close()
```

DayTime-Client.py

```
# client.py
import socket

# create a socket object
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# get local machine name
host = socket.gethostname()

port = 9999

# connection to hostname on the port.
s.connect((host, port))
```

```
# Receive no more than 1024 bytes

tm = s.recv(1024)


s.close()


print("The time got from the server is %s" % tm.decode('ascii'))
```

OUTPUT:

DayTime-Server.py

```
Got a connection from ('192.168.25.107', 52123)
```

DayTime-Client.py

```
The time got from the server is Tue Jul 27 23:01:44 2021
```

11. Implement a UDP based daytime client server program in python.

Source Code:

Server Side :

```
from datetime import datetime
import socket
localIP = "127.0.0.1"
localPort = 20001
bufferSize = 1024
# Create a datagram socket
UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
# Bind to address and ip
UDPServerSocket.bind((localIP, localPort))
bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)
message = bytesAddressPair[0]
address = bytesAddressPair[1]
time=datetime.now()
time=str(time)
time=str.encode(time)
UDPServerSocket.sendto(time, address)
```

Client Side:

```
import socket
msgFromClient = "hello from client"
bytesToSend = str.encode(msgFromClient)
serverAddressPort = ("127.0.0.1", 20001)
bufferSize = 1024
# Create a UDP socket at client side
```

```
UDPClientSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
# Send to server using created UDP socket
UDPClientSocket.sendto(bytesToSend, serverAddressPort)
timeFromServer = UDPClientSocket.recvfrom(bufferSize)
time=format(timeFromServer[0])
print(time)
```

Output:

```
ld be made because the target machine actively refused it
PS D:\MCA LDCE\Sem 2\Computer Network> python 11client.py
b'2021-07-28 14:54:49.997421'
PS D:\MCA LDCE\Sem 2\Computer Network>
```

12. Implement a TCP based client server text chat program in python.

<u>CODE:</u>

<u>Chat-Server.py</u>

```
# server.py

import time, socket, sys


print("\nWelcome to Chat Room\n")

print("Initialising....\n")

time.sleep(1)


s = socket.socket()

host = socket.gethostname()

ip = socket.gethostbyname(host)

port = 1234

s.bind((host, port))

print(host, "(", ip, ")\n")

name = input(str("Enter your name: "))


s.listen(1)

print("\nWaiting for incoming connections...\n")

conn, addr = s.accept()
```

```
print("Received connection from ", addr[0], "(", addr[1], ")\n")

s_name = conn.recv(1024)
s_name = s_name.decode()
print(s_name, "has connected to the chat room\nEnter [e] to exit chat room\n")
conn.send(name.encode())

while True:
    message = input(str("Me : "))
    if message == "[e]":
        message = "Left chat room!"
        conn.send(message.encode())
        print("\n")
        break
    conn.send(message.encode())
    message = conn.recv(1024)
    message = message.decode()
    print(s_name, ":", message)
```

Chat-Client.py

```
# client.py
import time, socket, sys

print("\nWelcome to Chat Room\n")
print("Initialising....\n")
time.sleep(1)

s = socket.socket()
shost = socket.gethostname()
```

```
ip = socket.gethostbyname(shost)
print(shost, "(", ip, ")\n")
host = input(str("Enter server address: "))
name = input(str("\nEnter your name: "))
port = 1234
print("\nTrying to connect to ", host, "(", port, ")\n")
time.sleep(1)
s.connect((host, port))
print("Connected...\n")

s.send(name.encode())
s_name = s.recv(1024)
s_name = s_name.decode()
print(s_name, "has joined the chat room\nEnter [e] to exit chat room\n")

while True:
    message = s.recv(1024)
    message = message.decode()
    print(s_name, ":", message)
    message = input(str("Me : "))
    if message == "[e]":
        message = "Left chat room!"
        s.send(message.encode())
        print("\n")
        break
    s.send(message.encode())
```

OUTPUT:

Chat-Server.py

```
Welcome to Chat Room

Initialising....

DESKTOP-16NSNIA ( 192.168.25.107 )

Enter your name: Ankita

Waiting for incoming connections...

Received connection from  192.168.25.107 ( 50910 )

Sagar has connected to the chat room
Enter [e] to exit chat room

Me : Hello
Sagar : Hey
Me : How are you?
Sagar : I'm Fine!!How about you?
Me : All good! Thank You!
Sagar : What are you doing?
Me : I.m busy with office work
Sagar : Sorry for disturbing you. Take care. Bye
Me : [e]
```

Chat-Client.py

```
Welcome to Chat Room

Initialising....

DESKTOP-16NSNIA ( 192.168.25.107 )

Enter server address: 192.168.25.107

Enter your name: Sagar

Trying to connect to  192.168.25.107 ( 1234 )

Connected...

Ankita has joined the chat room
Enter [e] to exit chat room

Ankita : Hello
Me : Ankita : How are you?
Me : I'm Fine!!How about you?
Ankita : All good! Thank You!
Me : What are you doing?
Ankita : I.m busy with office work
Me : Sorry for disturbing you. Take care. Bye
Ankita : e
```

13. Implement a UDP based client server text chat program in python.

Source Code:

Server Side :

```
import socket
localIP = "127.0.0.1"
localPort = 20001
bufferSize = 1024
msgFromServer = "Start chat"
bytesToSend = str.encode(msgFromServer)
# Create a datagram socket
import socket
localIP = "127.0.0.1"
localPort = 20001
bufferSize = 1024
msgFromServer = "Start chat"
bytesToSend = str.encode(msgFromServer)
Create a datagram socket
UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
# Bind to address and ip
UDPServerSocket.bind((localIP, localPort))
print("UDP server up and listening")
# Listen for incoming datagrams
while True:
bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)
address = bytesAddressPair[1]
clientMsg = format(bytesAddressPair[0])
```

```
print(clientMsg)
message=input(' -> ')
bytesToSend = str.encode(message)
UDPServerSocket.sendto(bytesToSend, address)
```

Client Side :

```
import socket
serverAddressPort = ("127.0.0.1", 20001)
bufferSize = 1024
# Create a UDP socket at client side
UDPClientSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
message = input(' -> ')
bytesToSend = str.encode(message)
while message.lower().strip() != 'bye':
UDPClientSocket.sendto(bytesToSend, serverAddressPort)
msgFromServer = UDPClientSocket.recvfrom(bufferSize)
msg = format(msgFromServer[0])
print(msg) # show in terminal
message = input(" -> ") # again take input
bytesToSend = str.encode(message)
```

Output:

```
PS D:\MCA LDCE\Sem 2\Computer Network> python 13server.py
UDP server up and listening
b'hii server'
 -> hii client
b'how are you'
 -> i am fine you ??
b'i am also fine'
 -> byy
```

```
S D:\MCA LDCE\Sem 2\Computer Network> python 13client.py
-> hii server
'hii client'
-> how are you
'i am fine you ??'
-> i am also fine
'byy'
-> bye
S D:\MCA LDCE\Sem 2\Computer Network>
```

14. Implement a TCP based echo client server program in python with a multi-threaded server.

Source Code:

Server Side :

```
# import socket programming library
import socket
# import thread module
from _thread import *
import threading
```

```
print_lock = threading.Lock()
# thread function
def threaded(c):
while True:
# data received from client
data = c.recv(1024)
if not data:
print('Bye')
# lock released on exit
print_lock.release()
break
# reverse the given string from client
data = data[::-1]
# send back reversed string to client
c.send(data)
# connection closed
c.close()
def Main():
host = ""
# reverse a port on your computer
# in our case it is 12345 but it
# can be anything
port = 12345
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((host, port))
print("socket binded to port", port)
# put the socket into listening mode
s.listen(5)
print("socket is listening")
# a forever loop until client wants to exit
while True:
# establish connection with client
c, addr = s.accept()
# lock acquired by client

print_lock.acquire()
print('Connected to :', addr[0], ':', addr[1])
# Start a new thread and return its identifier
start_new_thread(threaded, (c,))
s.close()
if __name__ == '__main__':
Main()
```

Client Side :

```
# Import socket module
import socket
def Main():
# local host IP '127.0.0.1'
host = '127.0.0.1'
# Define the port on which you want to connect
port = 12345
```

```
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
# connect to server on local computer
s.connect((host,port))
# message you send to server
message = "shaurya says geeksforgeeks"
while True:
# message sent to server
s.send(message.encode('ascii'))

# messaga received from server
data = s.recv(1024)
# print the received message
# here it would be a reverse of sent message
print('Received from the server :',str(data.decode('ascii')))
# ask the client whether he wants to continue
ans = input('\nDo you want to continue(y/n) :')
if ans == 'y':
continue
else:
break
# close the connection
s.close()
if __name__ == '__main__':
Main()
```

Output:

```
PS D:\MCA LDCE\Sem 2\Computer Network> python 14server.py
socket binded to port 12345
socket is listening
Connected to : 127.0.0.1 : 1394
```

```
PS D:\MCA LDCE\Sem 2\Computer Network> python 14client.py
Received from the server : skeegrofskeeg syas ayruahs

Do you want to continue(y/n) :y
Received from the server : skeegrofskeeg syas ayruahs

Do you want to continue(y/n) :
```

15. Implement a TCP based daytime client server program in python with a multithreaded

Source Code:

Server Side :

```
from os import times
import socket
from _thread import *
import threading
from datetime import datetime
print_lock = threading.Lock()
```

```
bufferSize = 1024
def threaded(s):
while True:
data = s.recvfrom(bufferSize)
addr=data[1]
msg=format(addr[0])
if not msg:
print('Bye')
print_lock.release()
break
rightnow=datetime.now()
rightnow=str(rightnow)
bytesToSend = str.encode(rightnow)
s.sendto(bytesToSend,addr)
s.close()

def Main():
host = ""
port = 12345
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind((host, port))
print("socket binded to port", port)
print("socket is listening")
while True:
print_lock.acquire()
start_new_thread(threaded, (s,))
s.close()
if __name__ == '__main__':
Main()
```

Client Side :

```
import socket
bufferSize=1024
def Main():
host = '127.0.0.1'
port = 12345
s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.connect((host,port))
message = "Hello"
bytesToSend = str.encode(message)
serverAddressPort=(host,port)
s.sendto(bytesToSend, serverAddressPort)
data = s.recvfrom(bufferSize)
msg = format(data[0])
print('Received from the server :',msg)
s.close()
if __name__ == '__main__':
Main()
```

Output:

```
socket binded to port 12345
```

```
Received from the server : b'2021-07-28 16:13:15.264887'
```

16. Implement a web client using urllib to:

a) Display the html source of a given URL on screen

b) Display the URL visited

c) Display the header information transmitted in the http response sent by the contacted web-site/web-server.

d) Display the http server status code

Source Code:

```
import urllib.request
from http import HTTPStatus
url="http://python.org"
with urllib.request.urlopen(url) as response:
print("HTML code: \n",response.read())

print("URL: ",url)
print("Headers: ",response.headers)
print("Status code:", HTTPStatus.OK.value)
```

Output:

```
HTML code:
 b'<!doctype html>\n<!--[if lt IE 7]>   <html class="no-js ie6 lt-ie7 lt-ie8 lt-ie9">   <![endif]-->\n<!--[if IE 7]>      <html
<!--[if IE 8]>      <html class="no-js ie8 lt-ie9">                    <![endif]-->\n<!--[if gt IE 8]><!--><html class="no-js" lan
charset="utf-8">\n    <meta http-equiv="X-UA-Compatible" content="IE=edge">\n\n    <link rel="prefetch" href="//ajax.googleapis
l="prefetch" href="//ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/jquery-ui.min.js">\n\n    <meta name="application-name" cont
 content="The official home of the Python Programming Language">\n    <meta name="apple-mobile-web-app-title" content="Python.o
nt="yes">\n    <meta name="apple-mobile-web-app-status-bar-style" content="black">\n\n    <meta name="viewport" content="width=
ldFriendly" content="True">\n    <meta name="format-detection" content="telephone=no">\n    <meta http-equiv="cleartype" conten
se">\n\n    <script src="/static/js/libs/modernizr.js"></script>\n\n    <link href="/static/stylesheets/style.15ff3dddc9c3.css"
 <link href="/static/stylesheets/mq.e887b902092b.css" rel="stylesheet" type="text/css" media="not print, braille, embossed, spe
>\n    <link href="/static/stylesheets/no-mq.bf0c425cdb73.css" rel="stylesheet" type="text/css" media="screen" />\n    \n    \n
.googleapis.com/ajax/libs/jqueryui/1.12.1/themes/smoothness/jquery-ui.css">\n\n    \n    <link rel="icon" type="image/x-icon" h
on-precomposed" sizes="144x144" href="/static/apple-touch-icon-144x144-precomposed.png">\n    <link rel="apple-touch-icon-preco
4x114-precomposed.png">\n    <link rel="apple-touch-icon-precomposed" sizes="72x72" href="/static/apple-touch-icon-72x72-precom
 href="/static/apple-touch-icon-precomposed.png">\n    <link rel="apple-touch-icon" href="/static/apple-touch-icon-precomposed.
```

```
HTML code:
Squeezed text (640 lines).
URL:  http://python.org
Headers:  Connection: close
Content-Length: 49903
Server: nginx
Content-Type: text/html; charset=utf-8
X-Frame-Options: DENY
Via: 1.1 vegur, 1.1 varnish, 1.1 varnish
Accept-Ranges: bytes
Date: Thu, 29 Jul 2021 04:12:20 GMT
Age: 1842
X-Served-By: cache-bwi5135-BWI, cache-bom4748-BOM
X-Cache: HIT, HIT
X-Cache-Hits: 2, 7
X-Timer: S1627531940.099434,VS0,VE0
Vary: Cookie
Strict-Transport-Security: max-age=63072000; includeSubDomains


Status code: 200
```

17. Implement an ftp client using ftplib which connects to an ftp server, takes login/password from user, displays directory list and permits upload and download of files.

Source Code:

```
import ftplib
hostname='ftp.dlptest.com'
username='dlpuser'
password='rNrKYTX9g7z3RgJRmxWuGHbeu'
ftp_server=ftplib.FTP(hostname,username,password)
ftp_server.encoding='utf-8'
filename="demoDev.txt"
with open(filename,'rb') as file:
ftp_server.storbinary(f"STOR {filename}",file)
ftp_server.dir()
with open(filename,'wb') as file:
ftp_server.retrbinary(f"RETR {filename}", file.write)
```

Output:

```
=============== RESTART: D:/MCA LDCE/Sem 2/Computer Network/17
-rw-r--r--   1 1001    1001      567128 Jul 29 04:14 10MB.zip
-rw-r--r--   1 1001    1001        1391 Jul 29 04:14 15server.py
>>>
```

18. Write a Python program that makes a connection to a web server and retrieve/display a document using the HTTP protocol.

CODE:

```
import socket


mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

mysock.connect(('data.pr4e.org', 80))

cmd = 'GET http://data.pr4e.org/romeo.txt HTTP/1.0\r\n\r\n'.encode()

mysock.send(cmd)


while True:

   data = mysock.recv(512)

   if len(data) < 1:

      break

   print(data.decode(),end='')

mysock.close()
```

OUTPUT:

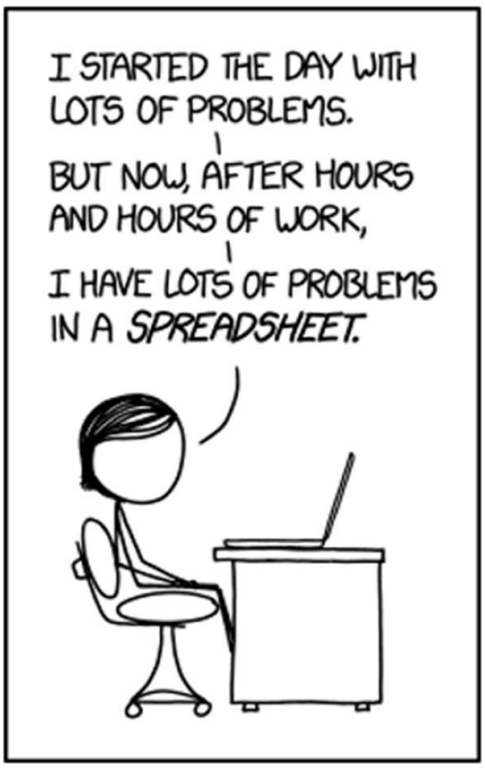
```
HTTP/1.1 200 OK
Date: Tue, 27 Jul 2021 18:16:09 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sat, 13 May 2017 11:22:22 GMT
ETag: "a7-54f6609245537"
Accept-Ranges: bytes
Content-Length: 167
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Connection: close
Content-Type: text/plain

But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief
```

19. Write a Python program that makes a connection to a web server and retrieve an image using the HTTP protocol.

Source Code:

```
import requests
recv= requests.get('https://imgs.xkcd.com/comics/making_progress.png')
with open(r'C:\Users\devpa\OneDrive\Documents\sem 2\python\image.png','wb') as f:
f.write(recv.content)
```

Output:



20. Write a python program to implement a simple server-client program .

CODE:

CN_20-Server.py

```
import socket
def server_program():
    # get the hostname
    host = socket.gethostname()
    port = 5000  # initiate port no above 1024

    server_socket = socket.socket()  # get instance
    # look closely. The bind() function takes tuple as argument
    server_socket.bind((host, port))  # bind host address and port together

    # configure how many client the server can listen simultaneously
    server_socket.listen(2)
    conn, address = server_socket.accept()  # accept new connection
    print("Connection from: " + str(address))
```

```
    while True:
        # receive data stream. it won't accept data packet greater than 1024 bytes
        data = conn.recv(1024).decode()
        if not data:
            # if data is not received break
            break
        print("from connected user: " + str(data))
        data = input(' -> ')
        conn.send(data.encode())  # send data to the client
    conn.close()  # close the connection

if __name__ == '__main__':
    server_program()
```

CN_20-Client.py

```
import socket

def client_program():
    host = socket.gethostname()  # as both code is running on same pc
    port = 5000  # socket server port number

    client_socket = socket.socket()  # instantiate
    client_socket.connect((host, port))  # connect to the server

    message = input(" -> ")  # take input

    while message.lower().strip() != 'bye':
        client_socket.send(message.encode())  # send message
        data = client_socket.recv(1024).decode()  # receive response
```

```
        print('Received from server: ' + data)  # show in terminal

        message = input(" -> ")  # again take input

    client_socket.close()  # close the connection

if __name__ == '__main__':
    client_program()
```

OUTPUT:

CN_20-Server.py

```
Connection from: ('192.168.25.107', 64645)
from connected user: Hello
 -> Hii
from connected user: How are you??
 -> I'M Fine!How are you?
from connected user: I.m also good. Thank you!!
 -> Okay,then Bye!!!
```

CN_20-Client.py

```
 -> Hello
Received from server: Hii
 -> How are you??
Received from server: I'M FineHow are you?
 -> I.m also good. Thank you!!
Received from server: Okay,then Bye!!!
 -> Bye!!
```

21. Write a python program to implement socket programming using multi-threading

Source Code:

```
# import socket programming library
import socket

# import thread module
from _thread import *
```

```
import threading
print_lock = threading.Lock()
# thread function
def threaded(c):
while True:
# data received from client
data = c.recv(1024)
if not data:
print('Bye')
# lock released on exit
print_lock.release()
break
# reverse the given string from client
data = data[::-1]
# send back reversed string to client
c.send(data)
# connection closed
c.close()def Main():
host = ""
# reverse a port on your computer
# in our case it is 12345 but it
# can be anything
port = 12345
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((host, port))
print("socket binded to port", port)
# put the socket into listening mode
s.listen(5)
print("socket is listening")
# a forever loop until client wants to exit
while True:
# data received from client
data = c.recv(1024)
if not data:
print('Bye')
# lock released on exit
print_lock.release()
break
# reverse the given string from client
data = data[::-1]
# send back reversed string to client
c.send(data)
# connection closed
c.close()
def Main():
host = ""
# reverse a port on your computer
# in our case it is 12345 but it
# can be anything
```

```
port = 12345
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((host, port))
print("socket binded to port", port)
# put the socket into listening mode
s.listen(5)
print("socket is listening")
# a forever loop until client wants to exit
while True:
# establish connection with client
c, addr = s.accept()
# lock acquired by client
print_lock.acquire()
print('Connected to :', addr[0], ':', addr[1]) # Start a new thread and return its identifier
start_new_thread(threaded, (c,))
s.close()
if __name__ == '__main__':
Main()
```

Client Side :

```
# import socket programming library
import socket
# import thread module
from _thread import *
import threading
print_lock = threading.Lock()
# thread function
def threaded(c):
while True:
# Import socket module
import socket
def Main():
# local host IP '127.0.0.1'
host = '127.0.0.1'
# Define the port on which you want to connect
port = 12345s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
# connect to server on local computer
s.connect((host,port))
# message you send to server
message = "shaurya says geeksforgeeks"
while True:
# message sent to server
s.send(message.encode('ascii'))
# messaga received from server
data = s.recv(1024)
# print the received message
# here it would be a reverse of sent message
print('Received from the server :',str(data.decode('ascii')))
# ask the client whether he wants to continue
ans = input('\nDo you want to continue(y/n) :')
```

```
if ans == 'y':
continue
else:
break
# close the connection
s.close()
if __name__ == '__main__':
Main()
```

Output:

```
socket binded to port 12345
socket is listening
Connected to : 127.0.0.1 : 59678
Bye
```

```
Received from the server : skeegrofskeeg syas ayruahs

Do you want to continue(y/n) :n
PS C:\Users\devpa\OneDrive\Documents\sem 2\python>
```