# Networks Lab
## Assignment 7

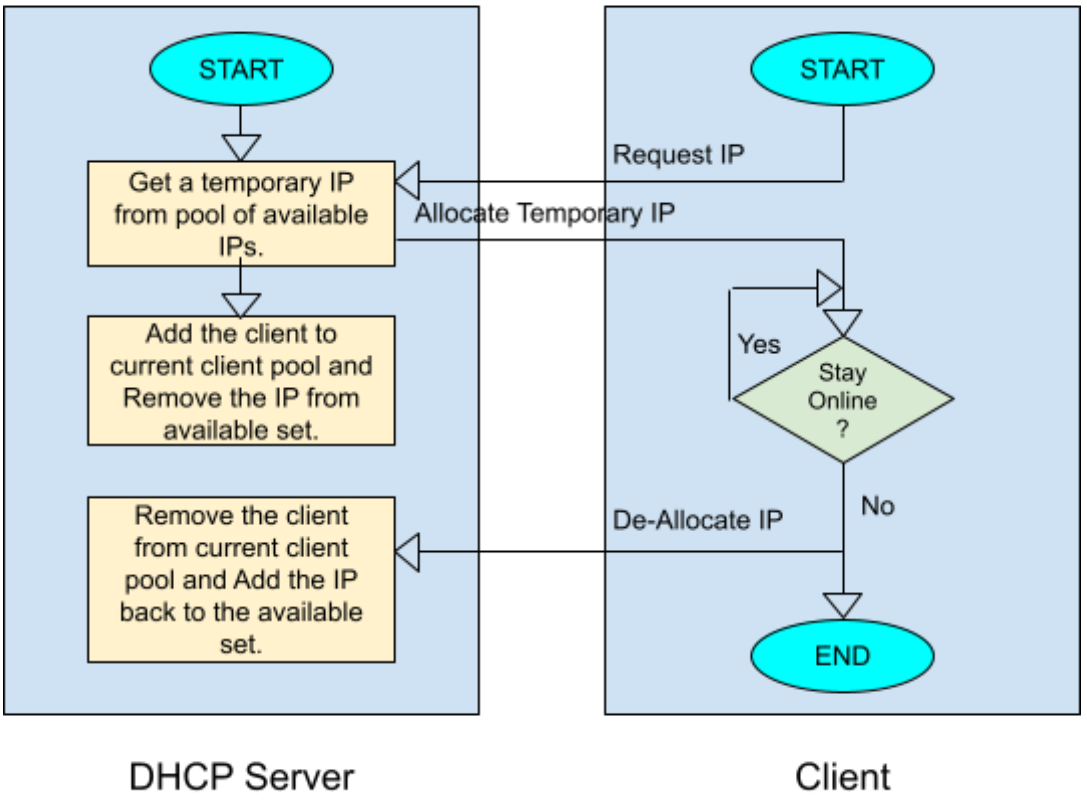
**Name:** Koustav Dhar **Class:** BCSE UG-III **Group:** A1 **Roll:** 001910501022

**Problem Statement:**

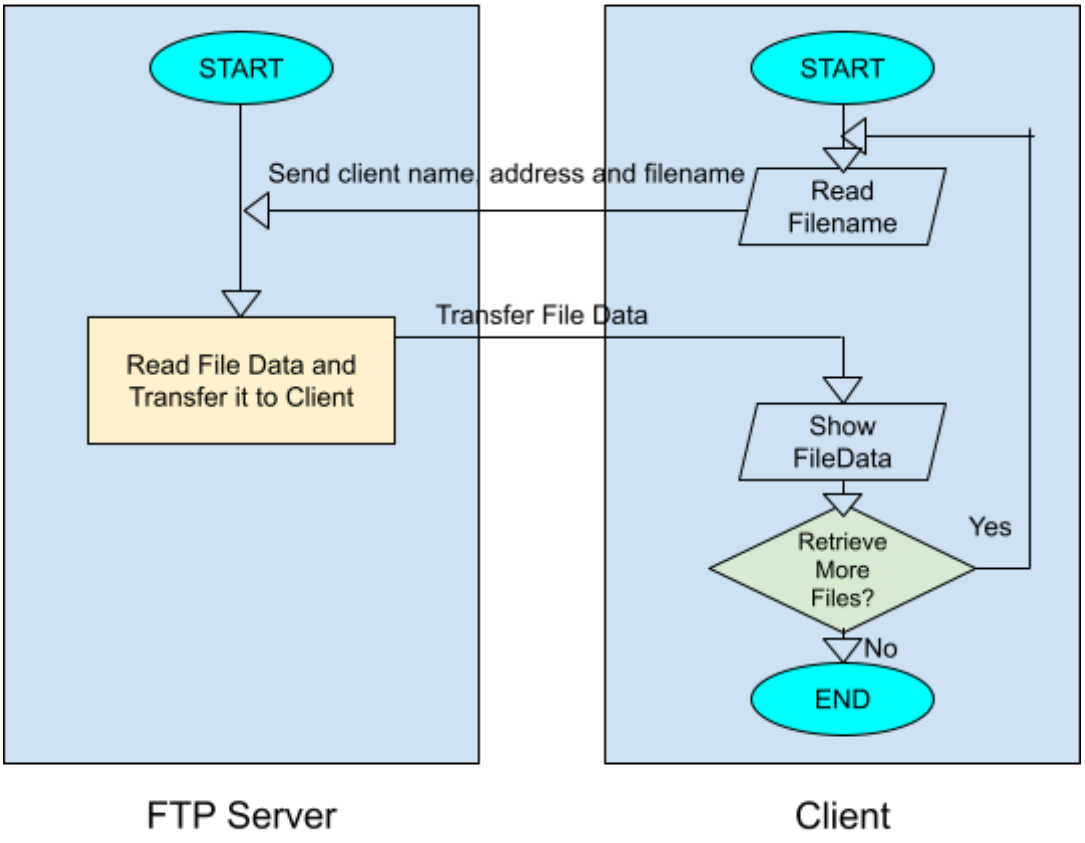
Implement any two protocols using TCP/UDP Socket as suitable.
1. BOOTP 2. FTP 3. DHCP 4. BGP 5. RIP

**Design:**

**DHCP (Dynamic Host Control Protocol):** DHCP is implemented using UDP sockets here. Whenever a client connects to the DHCP Server, it requests a temporary IP from DHCP. The DHCP maintains a set of available IPs, from which it pops an IP, and allocates that for the requesting client.
Now, whenever the client disconnects, it sends its temporary IP as a message to the DHCP Server, which then removes it from current clients, and adds the IP back to the set of available IPs, making it available for allocation for some different clients.




**FTP (File Transfer Protocol):** FTP is implemented using TCP sockets here. Whenever a client connects to the FTP Server, it requests a file by sending the filename. The FTP Server reads the file and transfers the contents of the file to the client. The client can ask for more file data or quit.




**Implementation:**

**dhcpserver.py**

```
import socket

HOST = '127.0.0.1'      # Standard loopback interface address (localhost)
OWNport = 65100         # Port to listen on (non-privileged ports are > 1023)

# Create a UDP Socket
server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server.bind((HOST, OWNport))
print("DHCP Server Started!")

startAddress = '127.0.0.3'
prefixAddress = '127.0.0.'
Range = 250             # at max 250 clients can connect

# 250 temporary ip's starting from 127.0.0.3 to 127.0.0.252
l = [i for i in range(3, 3 + Range)]
availablePool = set(l)
# dictionary to map current client's address with
# temporary IP address generated by DHCP
currentClients = dict()
cnt = 0

while True:
   message, address = server.recvfrom(1024)
   message = message.decode()
```

```
    if message == 'Request IP':
        print("Temporary IP Requested...")
        temp = availablePool.pop()  # getting an available IP
        tempAddress = prefixAddress + str(temp) # generating temporary IP
        currentClients[tempAddress] = temp  # updating the current client dict
        cnt += 1
        print("Temporary IP [" + tempAddress + "] allocated!")
        print("Currently " + str(cnt) + " clients are online\n")
        server.sendto(str.encode(tempAddress), address)
    else:
        print("IP Deallocation Requested...")
        temp = currentClients[message]
        print("Deallocating IP [" + message + "]...")
        currentClients.pop(message) # deleting this from current client dict
        availablePool.add(temp) # making this IP available again
        cnt -= 1
        print("Deallocated IP [" + message + "]!")
        print("Currently " + str(cnt) + " clients are online\n")
```

**dhcpclient.py**

```
import socket

HOST = '127.0.0.1'  # The server's hostname or IP address
DHCPport = 65100        # The port used by the server

# Create a UDP Socket
client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
print("Client Started...")

address = (HOST, DHCPport)

print("Requesting for temporary IP from DHCP Server...")
client.sendto(str.encode("Request IP"), address)
ip, server = client.recvfrom(1024)
print("Temporary IP Allocated! [" + ip.decode() + "]")

while True:
    print("Press q to quit, anything else to stay online.")
    choice = input()
    if choice == "q" or choice == "Q":
        break

print("Terminating...")
client.sendto(ip, address)
```

**ftpserver.py**

```
import socket

HOST = '127.0.0.2'  # Standard loopback interface address (localhost)
PORT = 65300        # Port to listen on (non-privileged ports are > 1023)
```

```
print("FTP Server started!!")
while True:
   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
   s.bind((HOST, PORT))
   print("Listening for a connection on its own port....")
   s.listen()
   conn, addr = s.accept()
   name = conn.recv(1024).decode()
   filename = conn.recv(1024).decode()
   print(name, " with address " , addr , " is requesting file: ", filename)
   file = open(filename, 'r')
   data = file.read()
   conn.send(bytes(data, "utf-8"))
   print("Data sent!")
   s.close()
   print("FTP Server still running!")
```

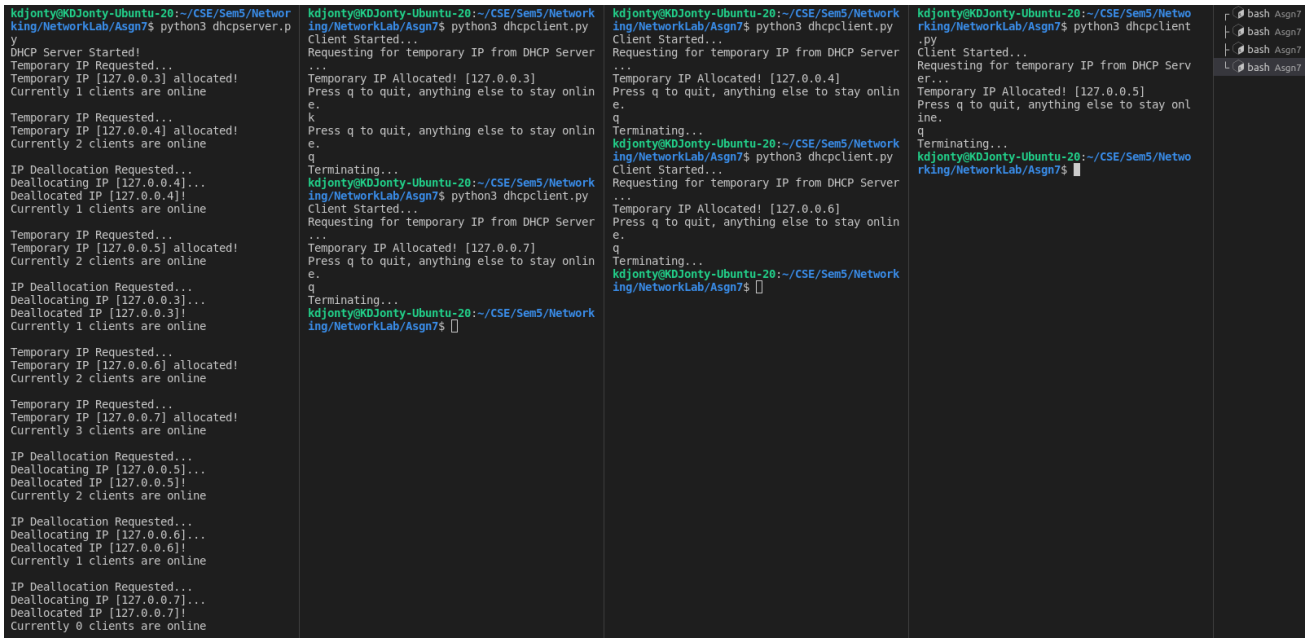
**ftpclient.py**

```
import socket

Host = '127.0.0.2'  # The server's hostname or IP address
FTPport = 65300        # The port used by the server

name = input("Enter name of the client: ")

while True:
   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
   choice = int(input("Press 1 to retrieve file.\nPress 2 to quit.\n"))
   if choice == 1:
       s.connect((Host, FTPport))
       s.send(bytes(name, "utf-8"))
       filename = input("Enter filename to be searched: ")
       s.send(bytes(filename, "utf-8"))
       data = s.recv(1024).decode()
       print("The contents of the file: \n" + data + "\n\n")
       s.close()
   elif choice == 2:
       break
   else:
       print("Invalid Choice. Try Again.")
```

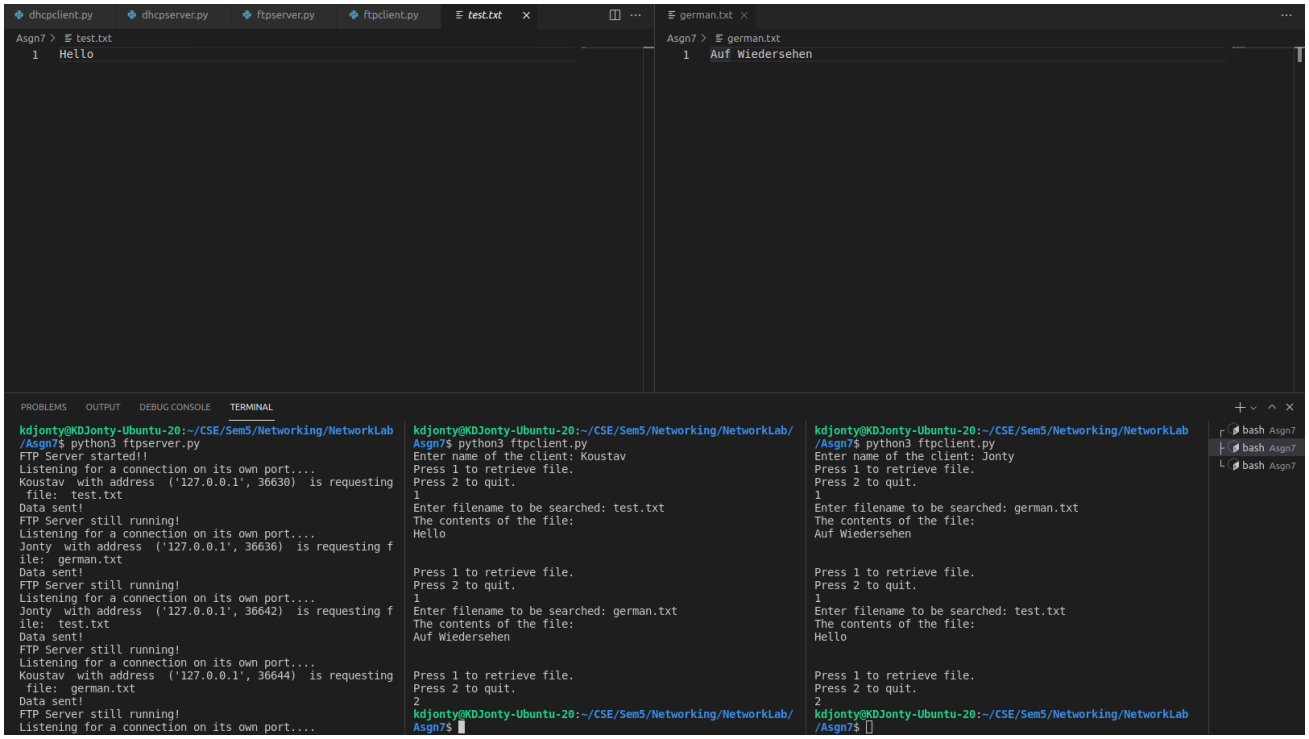
**Results:**

Terminal Output Screenshots are provided for DHCP and FTP below.

## DHCP:




## FTP:




### Comments:

Overall the lab assignment was a great learning experience as we got to implement a dynamic host configuration protocol and file transfer application. This assignment can be rated as easy.

### Improvements:

- No flow control protocol is considered, hence an error-free channel is assumed, which is not a practical scenario.
- No frame format for the packets is also considered.
- This would have been more efficient if it was implemented in a language closer to the system such as C/C++.