NETWORK LAB REPORT

NAME: ANURAN CHAKRABORTY
ROLL NO.: 20
CLASS: BCSE-III
SECTION: A1

ASSIGNMENT NUMBER: 6

PROBLEM STATEMENT:
1. Implement a file transfer application using TCP socket.
2. Implement a DNS server using UDP socket.

DEADLINE: 11TH APRIL, 2019
SUBMITTED ON: 4TH APRIL, 2019
REPORT SUBMITTED ON: 11TH APRIL, 2019

FTP:

DESIGN

The program implements an FTP application using TCP socket. The program consists of two modules.

1. **ftp_server.py**
   This module is responsible to select a file and send it to the client. It opens up a TCP socket through which it can transfer the file 1024 bytes at a time.
2. **ftp_client.py**
   This module is responsible for receiving the file sent by the server. It opens up a TCP socket through which it can receive the file 1024 bytes at a time and then reconstruct it.

Some important parameters for the design of the program are:

Input format: The input for the program is the name of the file to transfer and to specify which port the application will run in.

Output format: The program transfers the file from one address to another.

IMPLEMENTATION

The assignment has been implemented in Python3. The detailed description is given below.

ftp_server.py

This module is the ftp server. It opens up a socket and transfers the file whose name is given as input. It then transfers the file 1024 bytes at a time.

```
import socket

# Accept ip address
port=int(input("Enter port of server machine "))

sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

sock.bind(('', port))
sock.listen(5)
c, addr=sock.accept()

filename=c.recv(1024).decode()

with open(filename,'rb') as f:

        print('Sending file..')
```

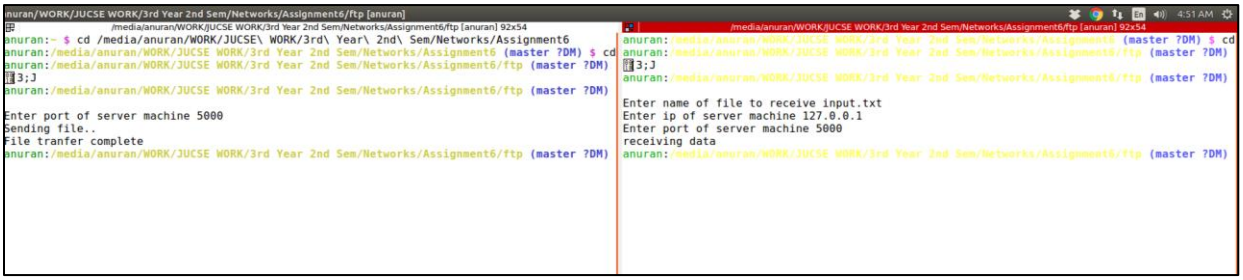
```
        # Sending file line by line
        lines=f.read(1024)
        while(lines):
                c.send(lines)
                lines=f.read(1024)


# c.send('#'.encode())
print('File tranfer complete')
c.shutdown(socket.SHUT_WR)
```

ftp_client.py

This module is the ftp client. It opens up a socket to accept the file sent by the server. It then reconstructs the file 1024 bytes at a time.

```
import socket

# Accept ip address
filename=str(input("Enter name of file to receive "))
ipaddr=input("Enter ip of server machine ")
port=int(input("Enter port of server machine "))

sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

sock.connect((ipaddr,port))

sock.send(filename.encode())

with open(filename,'wb') as f:
        print('receiving data')
        while(True):
                lines=sock.recv(1024)
                if not lines:
                        break
                f.write(lines)

sock.shutdown(socket.SHUT_WR)
```

OUTPUTS



RESULTS

The file was successfully transferred.

The TCP protocol is found appropriate to be used for the implementation of a file transfer application. This is because TCP allows the sequential transfer of messages. Thus, the order of bits in a file is not lost or altered in any means, which is extremely important for transferring files.

ANALYSIS

Overall the implementation of the assignment is more or less correct. However, provisions may be made for a full duplex transfer.

COMMENTS

Overall the lab assignment was a great learning experience as we got to implement a file transfer application. The assignment can be rated as easy.

DNS:

DESIGN

The program implements a DNS application using UDP socket. The program consists of two modules.

1. **dns_server.py**
   This module is responsible to receive the domain name which the client is requesting and accordingly return the IP address by checking from a lookup table and vice versa.
2. **dns_client.py**
   This module is the client which requests the server for the IP address of a domain name and displays the returned IP or vice versa.

Some important parameters for the design of the program are:

Input format: The input for the program is the domain name and to specify which port the application will run in.

Output format: The program returns the IP.

IMPLEMENTATION

The assignment has been implemented in Python3. The detailed description is given below.

dns_server.py

This module is the ftp server. It opens up a socket, receives the domain name from the client and accordingly return the IP address corresponding to the domain name using a dictionary.

```
import socket

# Define the mapping
dntoip={'www.abc.com':'123.90.0.1',
                'www.abcd.com':'123.91.23.1',
                'www.gfh.com':'123.98.56.1'}

iptodn={'123.90.0.1':'www.abc.com',
                '123.91.23.1':'www.abcd.com',
                '123.98.56.1':'www.gfh.com'}

# Accept ip address
port=int(input("Enter port of server machine "))

sock=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
```

```
sock.bind(('', port))

# Server mus always run
while(True):

        iporname, addr=sock.recvfrom(1024)
        iporname=iporname.decode()

        if(iporname in dntoip): # If the dn is given return ip
                dataToSend='Required IP is: '+dntoip[iporname]
        elif(iporname in iptodn):
                dataToSend='Required DN is: '+iptodn[iporname]
        else:
                dataToSend='Invalid request'

        sock.sendto(dataToSend.encode(),0,addr)

sock.close()
```

dns_client.py

This module is the dns client. It opens up a socket requests for the IP address of the domain name given as input and then displays it.

```
import socket

# Accept ip address
name=str(input("Enter domain name or ip "))
ipaddr=input("Enter ip of dns server machine ")
port=int(input("Enter port of server machine "))

sock=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

sock.connect((ipaddr,port))

# Send the domain name or ip
sock.sendto(name.encode(),0,(ipaddr,port))

# Wait for the server to send
iporname, addr=sock.recvfrom(1024)
iporname=iporname.decode()
print(iporname)

sock.close()
```

OUTPUTS

RESULTS

The requests were successfully answered. For invalid request an error message was displayed. The UDP protocol is found appropriate for implementing a DNS server. This is because the UDP protocol follows the "single request, single reply" principle. Thus, a client cannot make a query before the previous query is answered by the server, which is the criteria for the proper working of a DNS server. It also ensures that the query of one client is not answered to another.

ANALYSIS

Overall the implementation of the assignment is more or less correct. However, provisions may be made for a full duplex transfer.

COMMENTS

Overall the lab assignment was a great learning experience as we got to implement a file transfer application. The assignment can be rated as easy.