Computer Networks Lab Report - Assignment 8

Name: Bikash Sah

Class: BCSE-3

Group: A1

Assignment Number: 8

Problem Statement:

Assignment 8: Application and Transport layer protocols
Submission due: 7th - 11th November 2022

Implement any two protocols using TCP/UDP Socket as suitable.

1. FTP
2. DNS
3. Telnet

Submission Date: **21 November, 2022**

Deadline: 11th November, 2022

Implementation

FTP:

Server

```
package FTP;
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;

 class FileServer {
    public static void main(String[] args) throws Exception {
        // Initialize Sockets
        ServerSocket ssock = new ServerSocket(5000);
        Socket socket = ssock.accept();
        // The InetAddress specification
        InetAddress IA = InetAddress.getByName("localhost");
        // Specify the file
        // Input the file location
        String file_location = "input.txt";
        File file = new File(file_location);
        FileInputStream fis = new FileInputStream(file);
        BufferedInputStream bis = new BufferedInputStream(fis);
        // Get socket's output stream
        OutputStream os = socket.getOutputStream();
        // Read File Contents into contents array
        byte[] contents;
        long fileLength = file.length();
        long current = 0;
        long start = System.nanoTime();
        while (current != fileLength) {
            int size = 10000;
            if (fileLength - current >= size)
                current += size;
            else {
                size = (int) (fileLength - current);
                current = fileLength;
            }
            contents = new byte[size];
            bis.read(contents, 0, size);
            os.write(contents);
            System.out.print("Sending file ... " + (current * 100) / fileLength + "% c
omplete!");
        }
        os.flush();
        // File transfer done. Close the socket connection!
        socket.close();
        ssock.close();
        System.out.println("File sent succesfully!");
    }
}
```

Client

```
package FTP;
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.net.InetAddress;
import java.net.Socket;

class FileClient {
    public static void main(String[] args) throws Exception {
        // Initialize socket
        Socket socket = new Socket(InetAddress.getByName("localhost"), 5000);
        byte[] contents = new byte[10000];
        // Initialize the FileOutputStream to the output file's full path.
        FileOutputStream fos = new FileOutputStream("output.txt");
        // FileOutputStream fos = new FileOutputStream("e:\\Bookmarks1.html");
        BufferedOutputStream bos = new BufferedOutputStream(fos);
        InputStream is = socket.getInputStream();
        // No of bytes read in one read() call
        int bytesRead = 0;
        while ((bytesRead = is.read(contents)) != -1)
            bos.write(contents, 0, bytesRead);
        bos.flush();
        socket.close();
        System.out.println("File saved successfully!");
    }
}
```

DNS:

Local Server

```
# Implementation of DNS Local Server

# The local server accepts connection from the client and sends the IP address of the
 domain name to the client.

# The local server will have its own cache in which it will store the < domain name, I
P address > pairs. This cache will be used to resolve the domain names.

# If the domain name is not present in the cache, then the local server will send a qu
ery to the root server to get the IP address of the domain name.

# The root server will send the IP Address of the .com server or the .in server to the
local server, according to the domain name.

# The local server will then send a query to the .com server or the .in server to get
 the IP address of the domain name.

# The .com server or the .in server will send the IP address of the domain name to the
```

```
local server.

# The local server will then send the IP address of the domain name to the client.

# The local server will store the < domain name, IP address > pair in its cache.


# Importing the socket library and the time library
import socket
import time

IP = "127.0.0.1"
PORT = 9999

# IP address of the root server and the port number which will be used to connect to t
he root server
ROOT_IP ="127.0.0.3"
ROOT_PORT=9998

# Create the cache dictionary
cache = {}

Found=False

# Dictionary of domain names with their IP addresses in format < domain name, IP addre
ss >
# google.com, 100.100.100.1
# facebook.com, 100.100.100.2
# youtube.com, 100.100.100.3

# instagram.in, 99.99.99.1
# twitter.in, 99.99.99.2

dict_domain={}
dict_domain["google.com"]="100.100.100.1"
dict_domain["facebook.com"]="100.100.100.2"
dict_domain["youtube.com"]="100.100.100.3"
dict_domain["instagram.in"]="99.99.99.1"
dict_domain["twitter.in"]="99.99.99.2"



# Create the socket object
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# print ("Socket successfully created")
print("[ Local Server ]: Socket successfully created...")
# Bind the socket to the port
s.bind((IP, PORT))
# print ("socket binded to %s" %(PORT))
print(f"[ Local Server ]: Socket binded to {PORT}...")
# Put the socket into listening mode
s.listen(5)
# print ("socket is listening")
print("[ Local Server ]: Socket is listening...")

# Connect to the client
c, addr = s.accept()
# print ('Got connection from', addr)
```

```
print(f"[ Local Server ]: Got connection from {addr}...")
# Receive the domain name from the client
domain_name = c.recv(1024).decode()
# print("The domain name is: ",domain_name)
print(f"[ Local Server ]: The domain name is: {domain_name}...")

# Check if the domain name is present in the cache
for i in cache:
    if i==domain_name:
        Found=True
        # print("The IP address is: ",cache[i])
        print(f"[ Local Server ]: The IP address is: {cache[i]}...")
        # Send the IP address to the client
        c.send(cache[i].encode())
        # Close the connection
        c.close()
        # Close the socket
        s.close()
        break


# Function to send the query to the root server


def send_query(domain_name):
    isReal=False
    Inter_IP=""
    # The root server will send the IP Address along with a flag to the local server,
 the flag will be 1 if the domain name is present in the root server and 0 if the doma
in name is not present in the root server.
    # If the flag is 1, then the local server will send the IP address of the domain n
ame to the client.
    # If the flag is 0, then the local server will send a query to the Inter_IP that i
t will receive from the root server. The Inter_IP will be the IP address of the .com s
erver or the .in server.
    TEMP_IP ="localhost"
    TEMP_PORT=9998
    while isReal==True:
        # Connect to the root server
        s.connect((TEMP_IP,TEMP_PORT))
        # Send the domain name to the root server
        s.send(domain_name.encode())
        # Receive the IP address and the flag from the root server
        Inter_IP=s.recv(1024).decode()
        # Received format: < IP address, port , flag >

        # Split the received string to get the IP address and the flag

        # Get the IP address
        TEMP_IP=Inter_IP.split(",")[0]
        # Get the port number
        TEMP_PORT=Inter_IP.split(",")[1]
        # Get the flag

        Real=Inter_IP.split(",")[2]
        isReal=int(Real)
    # Search in the dictionary of domain names and IP addresses
    for i in dict_domain:
```

```
        if i==domain_name:
            # print("The IP address is: ",dict_domain[i])
            print(f"[ Local Server ]: The IP address is: {dict_domain[i]}...")
            # Send the IP address to the client
            TEMP_IP=dict_domain[i]
            print("[ Local Server ]: IP address sent to the client...")
            # Close the connection

            # Close the socket

            break
    # Now the IP address of the domain name is present in the ROOT_IP and the ROOT_POR
T
    # Now send the IP address of the domain name to the client
    # c.send(TEMP_IP.encode())

    # Check if the domain is .com or .in

    # If the domain is .com, then send the query to the .com server
    # if domain_name.split(".")[1]=="com":
    #     TEMP_IP="192.00.00.00"
    # Close the connection

    # Close the socket

    # Store the < domain name, IP address > pair in the cache
    cache[domain_name]=TEMP_IP

    return TEMP_IP




IP_Address=""

if Found==False:
    IP_Address=send_query(domain_name)
    if IP_Address!="":
        # Store the < domain name, IP address > pair in the cache
        cache[domain_name]=IP_Address
        # Send the IP address to the client
        c.send(IP_Address.encode())
        # Close the connection
        c.close()
        # Close the socket
        s.close()
```

Client:

```
# Implementation of DNS client

# Client will have its own local cache in which it will store the
# < domain name, IP address > pairs. This cache will be used to
# resolve the domain names.
```

```
# If the domain name is not present in the cache, then the client will send a query to
the local server.

# When the client receives the response from the server, it will store the < domain na
me, IP address > pair in its cache.


# The client will only contact the local server. It will not contact any other server.


# Importing the socket library and the time library
import socket
import time

# Store the IP address of the local server and the port number
IP="localhost"
PORT=9999

# Create the cache dictionary
cache={}
# Create the socket object
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)

# Take the domain name as input from the user
domain_name=input("Enter the domain name: ")

# Check if the domain name is present in the cache

isFound=False
for i in cache:
    if i==domain_name:
        isFound=True
        print("The IP address is: ",cache[i])
        break

IP_ADDRESS=""

# Function to send the query to the server
def send_query(domain_name):
    # Connect to the server
    s.connect((IP,PORT))
    # Send the domain name to the server
    s.send(domain_name.encode())
    # Receive the IP address from the server
    IP_ADDRESS=s.recv(1024).decode()
    # Close the connection
    # s.close()
    # Return the IP address
    return IP_ADDRESS

if not isFound:
    # If the domain is not found call a function to send the query to the server
    IP_ADDRESS=send_query(domain_name)

    # If the IP address is not empty, then store the < domain name, IP address > pair
 in the cache

    if IP_ADDRESS!="":
```

```
        cache[domain_name]=IP_ADDRESS
        print("The IP address is::",IP_ADDRESS)
    else:
        print("The domain name is not present in the DNS server")
```

Root Server:

```
# Implementation of DNS Root Server

# The root server accepts connection from the local server and sends the IP address of
the .com server or the .in server to the local server, according to the domain name.

import socket

# IP address of the root server and the port number which will be used to connect to t
he root server

ROOT_IP ="127.0.0.1"
ROOT_PORT=9998


IP_COM="127.0.0.4"
PORT_COM=9997

IP_IN="10.10.10.4"
PORT_IN=9996



# It will receive the domain name from the local server and send the IP address of the
.com server or the .in server to the local server, according to the domain name.

# Socket object
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)

# Bind the socket to the port
s.bind((ROOT_IP,ROOT_PORT))

# Put the socket into listening mode
s.listen(5)
print("[ Root Server ]: Socket is listening...")

# Connect to the local server
c,addr=s.accept()

# Receive the domain name from the local server
domain_name=c.recv(1024).decode()

print(f"[ Root Server ]: The domain name is: {domain_name}...")
# Extract the extension of the domain name
extension=domain_name.split(".")[-1]



# Send the IP address of the .com server or the .in server to the local server, accord
```

```
ing to the domain name.
# It will send in the format < IP address port number 0 >
if extension=="com":
    print(f"[ Root Server ]: Sending IP address of .com server to the local serve
r...")
    c.send((IP_COM+" "+str(PORT_COM)+" 0").encode())
elif extension=="in":
    print(f"[ Root Server ]: Sending IP address of .in server to the local server...")
    c.send((IP_IN+" "+str(PORT_IN)+" 0").encode())

# Close the connection
c.close()
# Close the socket
s.close()

# Write in the file pychache.txt the IP address of the .com server and the .in server
```

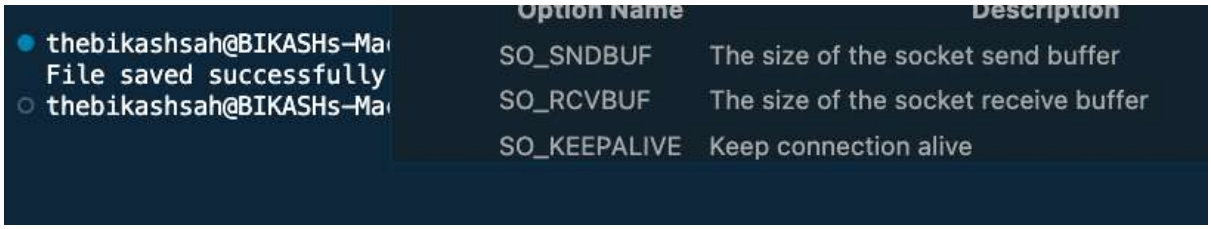
com server

```
# Implemenation of DNS .com Server
# It will store the < domain name, IP address > pairs in the cache.
# It will accept connection from the local server and send the IP address of the domai
n name to the local server.


import socket


IP_COM="127.0.0.1"
PORT_COM=9997

# Create the cache dictionary
cache={}
cache["google.com"]="199.99.99.99"
cache["facebook.com"]="200.200.200.200"

# Create the socket object
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)

# Bind the socket to the port
s.bind((IP_COM,PORT_COM))

# Put the socket into listening mode
s.listen(5)
print("[ .com Server ]: Socket is listening...")

# Connect to the local server
c,addr=s.accept()
print(f"[ .com Server ]: Got connection from {addr}...")
# Receive the domain name from the local server
domain_name=c.recv(1024).decode()
print(f"[ .com Server ]: The domain name is: {domain_name}...")

# Send the IP address of the domain name to the local server
```

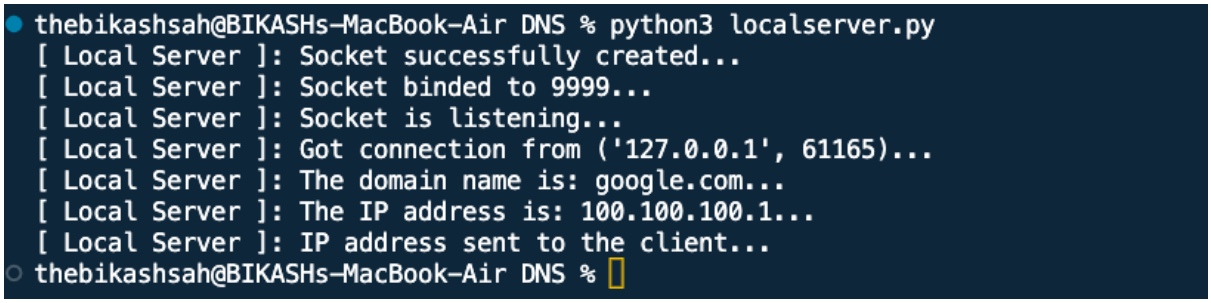
```
# Send in the format < IP address port number 1 >
c.send((cache[domain_name]+" "+str(PORT_COM)+" 1").encode())
print(f"[ .com Server ]: The IP address is: {cache[domain_name]}...")

# Close the connection
c.close()
# Close the socket
s.close()
```

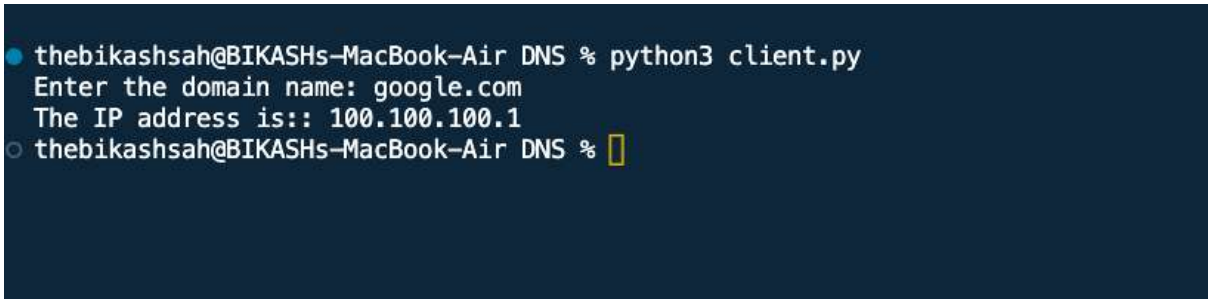
Test Cases and Results

FTP:



DNS:





Comments

> This assignment was like a real life project, it had a problem statement and I had to come up with a solution, I have learnt a lot, I learnt a lot of python in this assignment and also learnt how to implement big problems by dividing it into subproblems.