Networking Lab Assignment 14

FTP implementation

Albin Antony

20 April 2019

1 File Transfer Protocol

1.1 Aim

To implement a subset of File Transfer Protocol using TCP/IP.

1.2 Theory

In a typical FTP session, the user is sitting in front of one host (the local host) and wants to transfer files to or from a remote host. In order for the user to access the remote account, the user must provide a user identification and a password. After providing this authorization information, the user can transfer files from the local file system to the remote file system and vice versa. The user interacts with FTP through an FTP user agent. The user first provides the hostname of the remote host, causing the FTP client process in the local host to establish a TCP connection with the FTP server process in the remote host. The user then provides the user identification and password, which are sent over the TCP connection as part of FTP commands. Once the server has authorized the user, the user copies one or more files stored in the local file system into the remote file system (or vice versa).

Throughout a session, the FTP server must maintain state about the user. In par- ticular, the server must associate the control connection with a specific user account, and the server must keep track of the users current directory as the user wanders about the remote directory tree. Keeping track of this state information for each ongoing user session significantly constrains the total number of sessions that FTP can maintain simultaneously.

1.3 Algorithm

Algorithm 1 Algorithm for FTP server

```
START
Create TCP SOCKET
Bind SOCKET to a PORT
Start listing at the binded PORT for connection from CLIENT
WHILE TRUE:
    ACCEPT connection from CLIENT
    RECEIVE Filename from CLIENT
    IF FILE Found
        SEND FOUND
        READ FILE AND SEND IT TO CLIENT
STOP
```

Algorithm 2 Algorithm for FTP client

```
START
CREATE TCP SOCKET
Connect to server using IP and PORT
SEND Filename to server
IF response= FOUND
    CREATE A FILE
    RECEIVE packets from server until no more packets are
        send.
    WRITE packets to a FILE
STOP
```

1.4 Program

1.4.1 Server

```
import socket

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

port=8080

s.bind(('',port))

s.listen(5)

while True:
    c,addr=s.accept()
    filename=c.recv(1024)
    print "Finding file : "+filename+"......"
    print ('Got Connection from',addr)
    try:
        file = open(filename,'rb')
        c.send('Found')
        print "File Found"
        data = file.read(1024)
        print "Reading File...."
        print "Sending..."
        while(data):
            c.send(data)
            data = file.read(1024)
        file.close()
        print "File closed"
        break
```

```
    except:
        c.send('No File')
        print "No file found"
        break
c.close()
```

1.4.2 Client

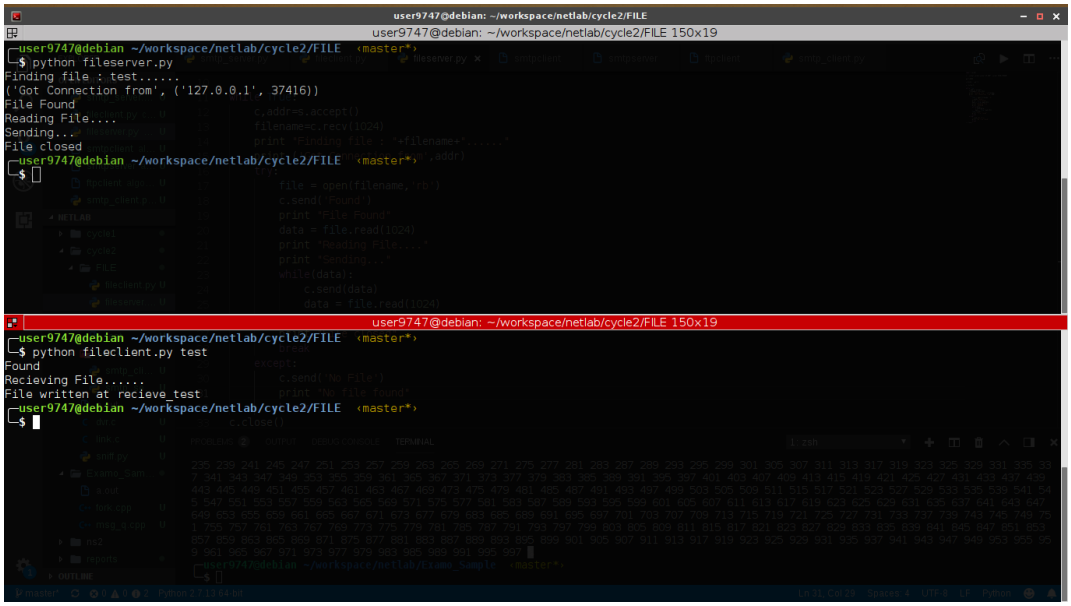
```
import socket
import sys

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

port = 8080
s.connect(('localhost',port))
s.send(sys.argv[1])
response = s.recv(1024)
print response
if(response =='Found'):
    file = open('recieve_'+ sys.argv[1] ,'wb')
    print "Recieving File....."
    while True:
        data = s.recv(1024)
        if not data:
            break
        file.write(data)
    file.close()
    print "File written at recieve_"+sys.argv[1]
else:
    print "File Not Found"

s.close()
```

1.5 Output



- Client sends filename to Server
- Server replies with file found
- Server reads file and sends it.
- Client receives data from server and writes it to a file

1.6 Result

Implemented FTP using TCP in Python2.7 and executed on Debian 9.4 Kernel 4.9 and outputs were verified.