FTP IMPLEMENATION
PROJECT REPORT

Muhammad Raza Vasnani k163890

Mustafa Irfan k163886

ABSTRACT

The File Transfer Protocol (FTP) is a standard network protocol used to transfer computer files on a computer network between a client and server. FTP is based on a model client-server architecture with separate control and data connections between the client and the server. The command channel is used to control the communication and the data channel is used for the actual transmission of files.

In this project we implemented ftp using two different approaches first we implemented ftp using socket programming where a TCP connection was established and the file was been transferred from one host computer to another this had no involvement of a server for transmission of files while the second approach included the implementation of FTP using PYTHON build-in library named FTPlib. The FTPlib module in Python allows you to write Python programs that perform a variety of automated FTP tasks. We can connect to a FTP server to retrieve files and process them locally.

INTRODUCTION

PROJECT MOTIVATION:

Our motivation was to use and observe this widely used protocol which is becoming obsolete in modern days.We aimed to look for advantages and disadvantages of FTP, and to develop a product like teamviewer for p2p collabration.

METHODOLOGY

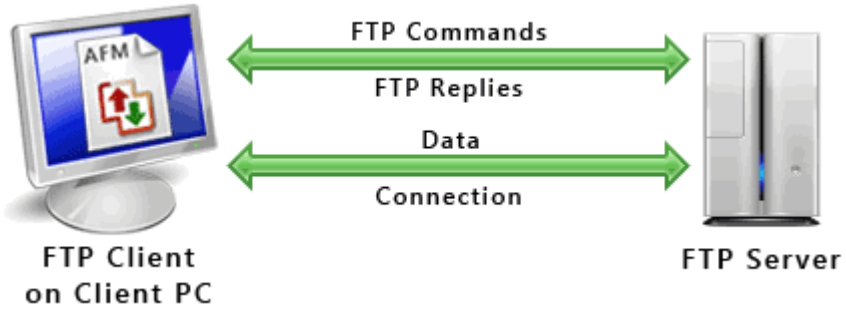
The project consist of implementation of FTP Protocol which has been executed using Python and the Python modules named as FTPlib. We first need to initiate a connection to the FTP server for that an instance is created by using port 1026

```
ftp = FTP('')

ftp.connect('localhost',1026)
```

Now in order to get access to the files on the sever we need to provide login credentials which include username and a password now after the connection is being established next the directory is to be provided from where we either need to download a file to or where the file we want to upload is placed which is done using the command of **ftp.cwd(‘’).**

After the directory has been provided now the client has multiple option that if they want to upload a file or either the requirement is to download a file.



The file is downloaded is in the blocks of 1024 bytes until the whole process of file transfer has been completed and the file is transferred successfully. Once the transfer of file is completed the ftp connection is being closed as it a crucial task which is mandatory to perform after the transfer of files.

Active and Passive Connection Mode

The FTP server may support **Active** or **Passive** connections or both. In an Active FTP connection, the client opens a port and listens and the server actively connects to it. In a Passive FTP connection, the server opens a port and listens (passively) and the client connects to it. Most FTP client programs select passive connection mode by default because server administrators prefer it as a safety measure.

FTP COMMANDS

- **USER *username***: Used to send the user identification to server.
- **PASS *password***: Used to send the user password to the server.
- **LIST**: Used to ask the server to send back a list of all the files in the current remote directory. The list of files is sent over a (new and non-persistent) data TCP connection and not over the control TCP connection.
- **RETR *filename***: Used to retrieve (i.e., get) a file from the current directory of the remote host.

- **STOR *filename***: Used to store (i.e., put) a file into the current directory of the remote host.

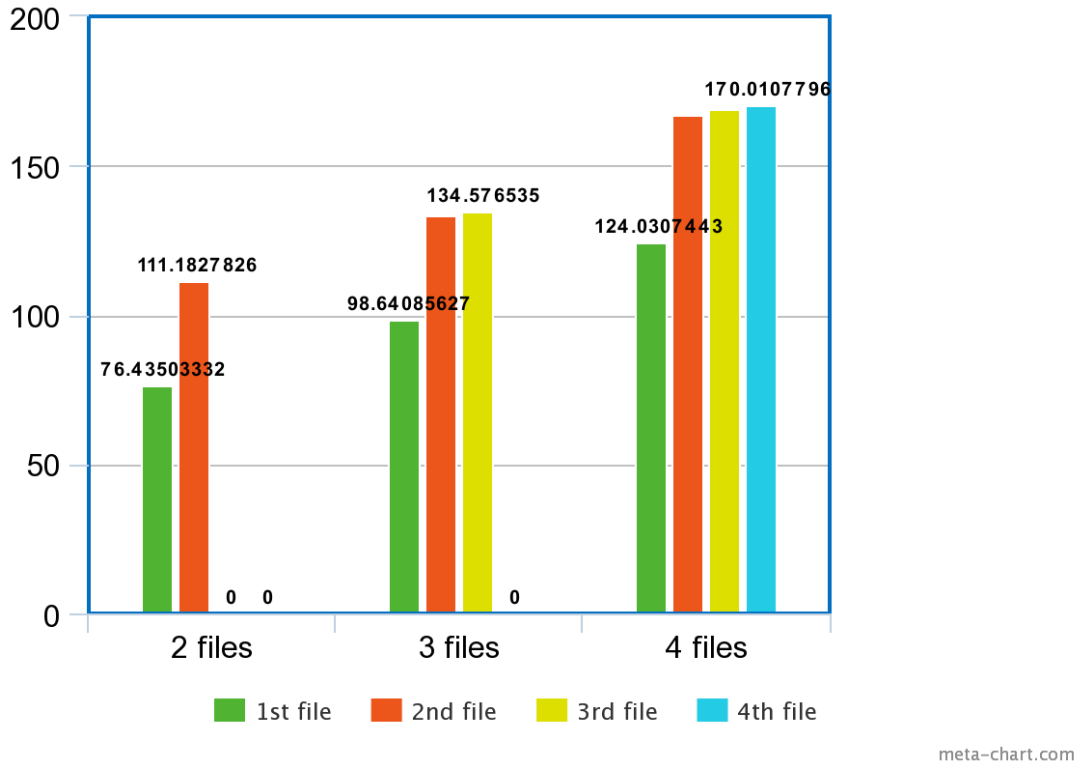
RESULT

➢ TEST CASE#1

In order to test the performance of the system we conducted a test.

1. First two files of 2.5 GB were upload and the time was measured. File1 was completely uploaded in 76.43 sec and the File2 took 111.18 sec to complete the upload process.
2. Then we uploaded 3 files the same two files with another file of 2.5 GB and the time of upload was measured. File1 completed the upload in 98.65 sec, File2 completed the upload in 134.57 sec.
3. Then we uploaded 4 files the same two files with 2 other file of 2.5 GB and the time of upload was measured. File1 completed the upload in 124.03 sec, File2 completed the upload in 170.01 sec.

It was observed that as the number of files to upload was increased the time to upload the same file had a significant increase in time. Hence number of file(upload) is directly proportional to time taken to complete upload process.

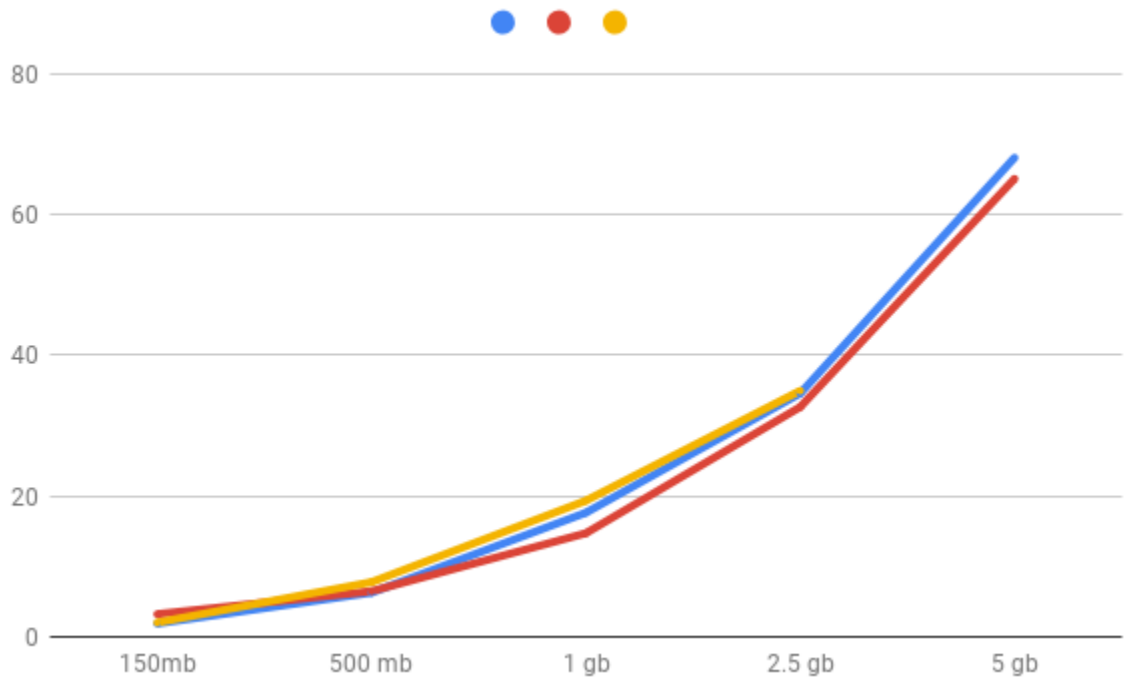


TESTCASE#2

Another test was conducted based on the size of file and upload time was measured.

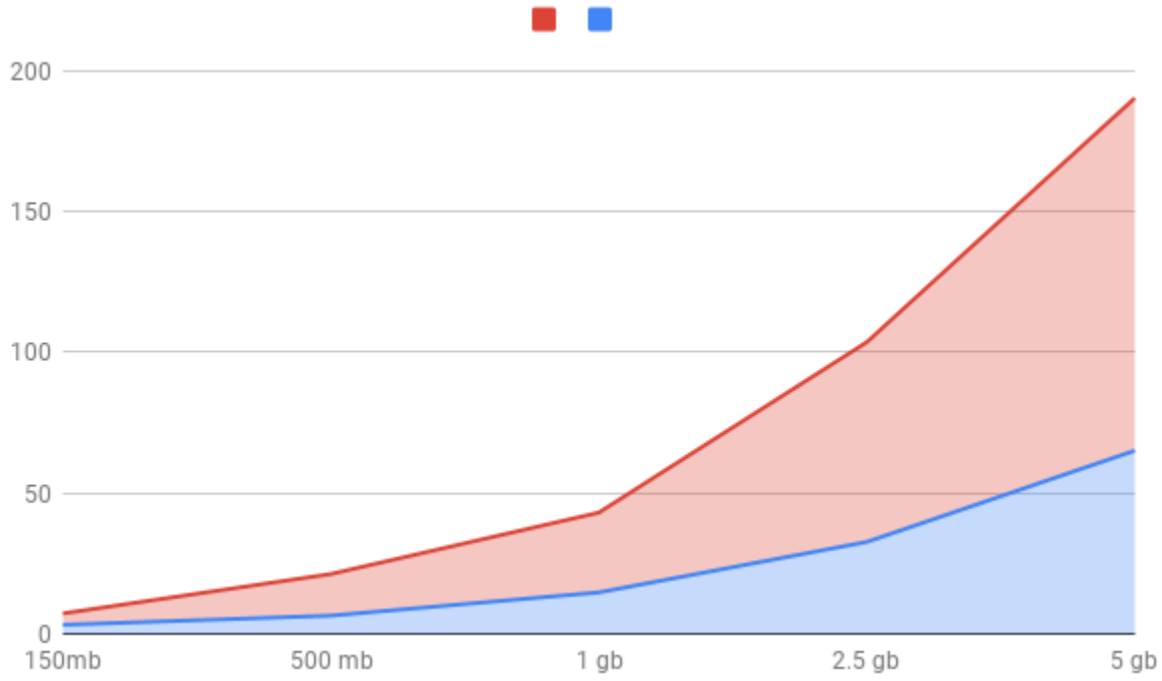
The observation was measured on files of 5 different sizes and the upload process was conducted 3 times.

It was observed that as the file size was increased the time taken for upload also increased exponentialy.



➢ TEST CASE#3

The third test case includes, we first used FTP implementation for file transfer then we used normal tcp socket programming communication to send file from a pc to another PC.It is clearly shown that ftp takes almost double time as we first uploaded to server then downloaded that is 2 transactions but in tcp communication only single transaction cycle is used.



Conclusion:

After handful test cases we concluded that FTP is less effective interim of performance as compared to TCP file share. But by the help of FTP we can save files in server and take advantage of Client server mechanism which can provide features such as version control also. Lastly if goal is to simple enable file sharing between 2 hosts then, FTP may not be the best option due to its extra overhead.

Appendix:

CLIENT SIDE CODE:

```
from ftplib import FTP
import os
ftp = FTP('')
ftp.connect('localhost',1026)
ftp.login(user="user",passwd="12345")
#print(ftp.pwd())
ftp.cwd('\\')
ftp.retrlines('LIST')

def uploadFile():
    filename = 'myfirstfile.doc'
    ftp.storbinary('STOR '+filename
,open(filename,'rb'))
    ftp.quit()

def downloadfile():
    filename= 'testfile.txt'
    localfile=open(filename,'wb')
    ftp.retrbinary('RETR ' + filename, localfile.write,
1024)

    ftp.quit()
    localfile.close()
```

SERVER SIDE CODE

```
from pyftpdlib.authorizers import DummyAuthorizer
from pyftpdlib.handlers import FTPHandler
from pyftpdlib.servers import FTPServer

authorizer=DummyAuthorizer()
authorizer.add_user("user","12345","C:\\Users\\th\\Down
loads\\FTPserver",perm="elradfmw")
authorizer.add_anonymous("C:\\Users\\th\\Downloads\\FTP
server",perm="elradfmw")

handler= FTPHandler
handler.authorizer=authorizer

server=FTPServer(("127.0.0.1", 1026), handler)
server.serve_forever()
```