Sharif University of Technology

Mohammad Amin Molaei Arpanahi

Student ID: 402012018

Data Networks

Project: FTP

June 2024

Table of Contents

Part 1: FTP protocol deployment.

Introduction

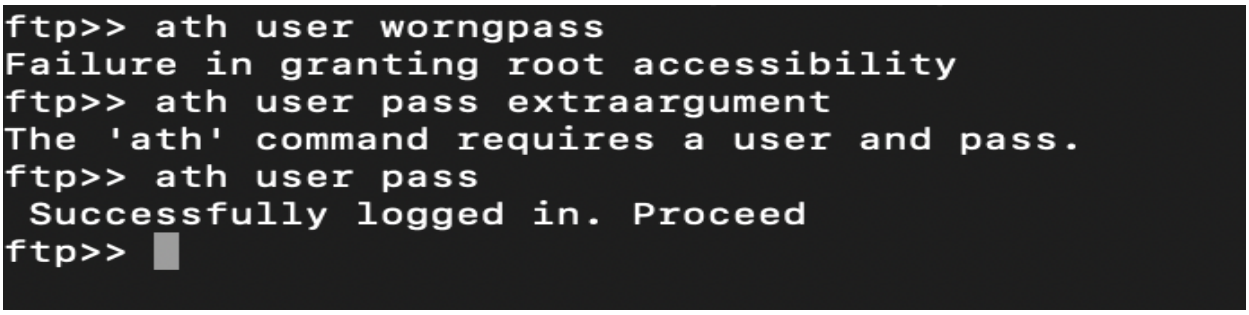
In this project, I aim to develop an FTP server using Python. The primary goal is to process commands received from the client and execute them on the server. The server will respond to the client based on the specific commands issued. Below, each command is succinctly explained. Notably, this project relies exclusively on Python's built-in libraries.

Command Implementation

1. Authentication Command

This implementation assumes that any user within the network has access to our server and can execute 'get' and 'ls' commands. However, for commands that modify server files, we have established root access. Users with specific credentials are permitted to execute commands such as 'put', 'delete', and 'mput'. The process is as follows:

- The client sends a JSON file containing the 'auth' command along with user credentials.
- The server receives this command and verifies if the user-pass pair matches an entry in its 'users.txt' file.
- If the credentials are valid, the server responds with a successful login message; otherwise, it denies access and root permissions.
- The client, upon receiving the appropriate response, can determine whether they have root access.
- The implementation supports insufficient argument scenarios by providing appropriate responses on the client side.



2. List Command

This command involves data transfer, although the data volume is minimal. The process is detailed below:

- Upon receiving the ‘ls’ command, the client checks for any arguments. If present, it informs that only the ‘ls’ command is valid.
- The client then creates a simple JSON from the command, encodes it, and sends it to the server, awaiting a random port allocation.
- The server decodes the request, which asks for a list of all items in the directory, along with their details.
- The server selects a port for data transfer and communicates it to the client for connection.
- The client, listening on the data port, receives and prints the data from the server.
- The data socket and connections are then closed on both ends.

```
Last login: Mon Jul  1 18:44:06 on ttys010
[aminmola@MacBook-Pro-3 FTP-Python % python3 client.py
ftp>> ls
Connected to data port 28378
total 169312
-rw-r--r--@ 1 aminmola  staff   6852575 Jun 30 22:57 Linear_algebra.pdf
-rw-r--r--  1 aminmola  staff  56376255 Jun 30 22:57 TA_Tutorial.mp4
-rw-r--r--@ 1 aminmola  staff  22052045 Jun 30 18:46 Tango.mp3
-rw-r--r--  1 aminmola  staff    349558 Jun 30 18:56 The_Death_of_Ivan_Ilyich.epub
-rw-r--r--  1 aminmola  staff   1048618 Jun 30 18:45 test.txt

Server response: Directory send OK
ftp>> ls extraargument
The 'ls' command does not take any arguments.
ftp>>
```

3. Get Command

The ‘get’ command process mirrors the ‘ls’ command, with the distinction that ‘get’ requires the server to open and read the file, sending data in segments. On the client side, a new file is created to write the received data from the allocated data port.

```
[aminmola@MacBook-Pro-3 FTP-Python % python3 client.py
ftp>> ls
Connected to data port 47515
total 169312
-rw-r--r--@ 1 aminmola  staff   6852575 Jun 30 22:57 Linear_algebra.pdf
-rw-r--r--  1 aminmola  staff  56376255 Jun 30 22:57 TA_Tutorial.mp4
-rw-r--r--@ 1 aminmola  staff  22052045 Jun 30 18:46 Tango.mp3
-rw-r--r--  1 aminmola  staff    349558 Jun 30 18:56 The_Death_of_Ivan_Ilyich.epub
-rw-r--r--  1 aminmola  staff   1048618 Jun 30 18:45 test.txt

Server response: Directory send OK
ftp>> get test.txt
Receiving file test.txt of size 1048618 bytes
Connected to data port 45318
Received file and saved as test.txt
Server response: Transfer complete
ftp>>
```

4. Put and Mput Command

For these commands, the client needs root access. Upon receiving a ‘put’ file request, the server verifies user authentication. If authenticated, the server opens a port to receive data from the client. Once the data transfer is complete, the server sends a final response to confirm file transfer completion. The ‘mput’ command follows a similar process, but the client sends a JSON containing multiple filenames, prompting the server to open a random data port for the transfer

5. Delete Command

Like the previous commands, ‘delete’ requires root access. The client sends the filename to the server, which, upon authenticating the client, deletes the file from the server directory.

6. Quit Command

When the client issues a ‘quit’ command, it sends a corresponding JSON, and the server interprets this as a signal to close the socket and its connection to the client on both sides.

```
aminmola@MacBook-Pro-3 FTP-Python % python3 client.py
ftp>> ls
Connected to data port 30966
total 124184
-rw-r--r--@ 1 aminmola  staff   6852575 Jun 30 22:57 Linear_algebra.pdf
-rw-r--r--  1 aminmola  staff  56376255 Jun 30 22:57 TA_Tutorial.mp4
-rw-r--r--  1 aminmola  staff    349558 Jun 30 18:56 The_Death_of_Ivan_Ilyich.epub

Server response: Directory send OK
ftp>> put test.txt
The client doesn't have the root access. File transfer aborted.
ftp>> ath user pass
 Successfully logged in. Proceed
ftp>> put test.txt
Sending file test.txt of size 1048618 bytes
Connected to data port 35582
Sent file test.txt
Server response: Transfer complete
ftp>> ls
Connected to data port 18871
total 126240
-rw-r--r--@ 1 aminmola  staff   6852575 Jun 30 22:57 Linear_algebra.pdf
-rw-r--r--  1 aminmola  staff  56376255 Jun 30 22:57 TA_Tutorial.mp4
-rw-r--r--  1 aminmola  staff    349558 Jun 30 18:56 The_Death_of_Ivan_Ilyich.epub
-rw-r--r--  1 aminmola  staff   1048618 Jul  3 11:20 test.txt

Server response: Directory send OK
ftp>> delete test.txt
Successfully deleted.
ftp>> ls
Connected to data port 34447
total 124184
-rw-r--r--@ 1 aminmola  staff   6852575 Jun 30 22:57 Linear_algebra.pdf
-rw-r--r--  1 aminmola  staff  56376255 Jun 30 22:57 TA_Tutorial.mp4
-rw-r--r--  1 aminmola  staff    349558 Jun 30 18:56 The_Death_of_Ivan_Ilyich.epub

Server response: Directory send OK
ftp>> mput test.txt,Tango.mp3
Connected to data port 34447
Sending file test.txt of size 1048618 bytes
Server response: Transfer complete for test.txt
Sending file Tango.mp3 of size 22052045 bytes
Server response: Transfer complete for Tango.mp3
ftp>> ls
Connected to data port 19807
total 169312
-rw-r--r--@ 1 aminmola  staff   6852575 Jun 30 22:57 Linear_algebra.pdf
-rw-r--r--  1 aminmola  staff  56376255 Jun 30 22:57 TA_Tutorial.mp4
-rw-r--r--  1 aminmola  staff  22052045 Jul  3 11:21 Tango.mp3
-rw-r--r--  1 aminmola  staff    349558 Jun 30 18:56 The_Death_of_Ivan_Ilyich.epub
-rw-r--r--  1 aminmola  staff   1048618 Jul  3 11:21 test.txt

Server response: Directory send OK
ftp>> quit
Exiting...
aminmola@MacBook-Pro-3 FTP-Python %
```

Part 2: FTP Protocol Understanding

1. Investigate other protocols that are used for file transfer and compare them with FTP?

Other file transfer protocol:

1- SFTP (File Transfer using SSH)

- SFTP is a secure file transfer protocol that operates over the SSH (Secure Shell) protocol.
- It provides encryption for both the command and data channels, making it more secure than standard FTP.

Pros:

- Strong security through SSH, including encryption and authentication.
- Single connection port (typically port 22), making it easier to manage firewall rules.
- Supports additional functionalities such as file access, file locking, and directory listings.

cons:

- Generally slower than FTP due to the overhead of encryption and decryption.
- Requires SSH access, which may not be available on all servers.

2. FTPS (FTP Secure)

- FTPS is an extension to FTP that adds support for TLS (Transport Layer Security) and SSL (Secure Sockets Layer) encryption.
- It can operate in two modes: explicit (the client requests security) and implicit (the client connects to a secure port).

Pros:

- Encryption of data and commands provides strong security.
- Can be easier to integrate into existing FTP infrastructure with minimal changes.

Cons:

- Uses multiple ports, which can complicate firewall configurations.
- Not as widely supported as SFTP.

3. SCP (Secure Copy Protocol)

- SCP is a simple protocol for securely transferring files between hosts on a network.
- It also relies on SSH for security.

Pros:

- Strong security provided by SSH.
- Fast and straightforward for single file transfers or directories.

Cons:

- Limited functionality compared to FTP/SFTP (e.g., no support for resume, directory listing, or file management commands).
- Not suitable for complex file transfer operations.

4. HTTP/HTTPS

- HTTP (Hypertext Transfer Protocol) and HTTPS (HTTP Secure) are protocols primarily used for web browsing but can also be used for file transfers.
- HTTPS adds encryption using TLS.

Pros:

- Ubiquitous and well-supported by all modern web browsers and clients.
- Strong security through HTTPS.
- Easily passes through firewalls and proxies.

Cons:

- Not optimized for large file transfers or directory management.
- Lack of file transfer-specific features (e.g., resume, restart, and file permissions).

5. WebDAV (Web Distributed Authoring and Versioning)

- WebDAV is an extension of HTTP that allows clients to perform remote web content authoring operations.
- It provides a framework for users to create, change, and move documents on a server.

Pros:

- Built on HTTP/HTTPS, so it benefits from widespread support and security.
- Supports collaborative editing and version control.
- Works well with modern web applications and platforms.
-

Cons:

- Can be complex to set up and manage.
- Performance can be an issue with large file transfers.

6. SMB/CIFS (Server Message Block/Common Internet File System)

- SMB/CIFS is a network file sharing protocol used primarily in Windows environments.
- Allows shared access to files, printers, and serial ports among nodes on a network.

Pros:

- Native support in Windows environments.
- Supports detailed file permissions and access control.
- Provides robust file and print sharing capabilities.

Cons:

- Less common in Unix/Linux environments without additional configuration.
- Security concerns if not properly configured (e.g., encryption and authentication)

2. What is the commonly used transport protocol for file transfers? Is it possible to use UDP as the transport layer protocol?

The most commonly used transport protocol for file transfers is **TCP (Transmission Control Protocol)**. TCP is preferred for several reasons:

1. **Reliability**: TCP provides reliable data transfer, ensuring that all packets are received in the correct order and without corruption. This is crucial for file transfers where data integrity is paramount.
2. **Connection-Oriented**: TCP establishes a connection between the sender and receiver before data transfer begins, which ensures that both parties are ready for the transfer and can manage the data flow effectively.
3. **Error Checking and Correction**: TCP includes mechanisms for error checking and correction, which helps to ensure that the data received is exactly the same as the data sent.
4. **Flow Control**: TCP implements flow control to manage the rate of data transmission between the sender and receiver, preventing the receiver's buffer from being overwhelmed.
5. **Congestion Control**: TCP has built-in congestion control mechanisms to reduce the rate of data transmission in case of network congestion, thus maintaining network stability.

Examples of Protocols Using TCP for File Transfers

- **FTP (File Transfer Protocol)**: Uses TCP for reliable and ordered data transfer.
- **SFTP (SSH File Transfer Protocol)**: Uses TCP as it operates over SSH.

- **FTPS (FTP Secure)**: Uses TCP for encrypted file transfer.
- **HTTP/HTTPS**: Though primarily used for web content, it can also be used for file transfers, relying on TCP for transport.

UDP (User Datagram Protocol) can technically be used for file transfers, but it is not commonly used for several reasons:

1. **Lack of Reliability**: UDP is a connectionless protocol that does not guarantee the delivery, order, or integrity of packets. This means packets can be lost, duplicated, or received out of order, which is problematic for file transfers.
2. **No Error Checking or Correction**: UDP does not provide built-in error checking or correction, meaning any errors in transmission must be handled by the application layer, adding complexity to the implementation.
3. **No Flow Control or Congestion Control**: UDP lacks flow control and congestion control mechanisms, which can lead to network congestion and packet loss during high data transmission rates.

Use Cases for UDP in File Transfers

Despite its limitations, UDP can be used in certain file transfer scenarios where speed is more critical than reliability, and the application can tolerate some level of data loss or handle error correction itself. Some examples include:

- **Real-Time Data Transfer**: Applications such as live video streaming, online gaming, and VoIP (Voice over IP) can use UDP to achieve lower latency, accepting occasional packet loss as a trade-off for real-time performance.
- **Trivial File Transfer Protocol (TFTP)**: A simplified version of FTP that uses UDP for smaller, less critical file transfers. TFTP is often used in situations like booting diskless workstations or network devices.

3- Drawbacks of FTP and Its Obsolescence on the Modern Web

FTP (File Transfer Protocol) was once a standard protocol for transferring files over the Internet. However, several drawbacks have made it increasingly obsolete on the modern web. Here are the key reasons:

1. **Lack of Security**:
    - **Plaintext Transmission**: FTP transmits data, including usernames and passwords, in plaintext. This makes it vulnerable to interception and eavesdropping by attackers using tools like packet sniffers.
    - **No Encryption**: FTP does not encrypt data during transfer, which exposes sensitive information to potential threats.
2. **Complex Firewall Configuration**:
    - **Multiple Ports**: FTP uses multiple ports (port 21 for control and a range of ports for data transfer). This can complicate firewall configurations and make it difficult to manage securely.

    - **Active vs. Passive Mode**: Differences between active and passive modes require different firewall rules, adding to the complexity of setup and maintenance.
3. **Authentication Vulnerabilities**:
    - **Weak Authentication**: FTP typically uses basic authentication mechanisms, which are not as secure as modern methods like multi-factor authentication (MFA).
    - **Brute Force Attacks**: The lack of robust authentication makes FTP susceptible to brute force attacks.
4. **Performance Issues**:
    - **Inefficient for Large Transfers**: FTP can be inefficient for transferring large files or large numbers of files, leading to slow transfer speeds and increased overhead.
    - **No Compression**: FTP does not natively support data compression, which can result in larger data transfers and longer transfer times.
5. **Limited Support for Modern Features**:
    - **No Integrity Checking**: FTP lacks built-in mechanisms for verifying the integrity of transferred files (e.g., checksums or hash verification).
    - **No Resumption of Interrupted Transfers**: Although some FTP clients support resuming interrupted transfers, it is not a standard feature, leading to potential data loss or the need to restart transfers.
6. **Obsolescence and Lack of Support**:
    - **Deprecated in Many Applications**: Many modern applications and services have deprecated FTP in favor of more secure and efficient protocols.
    - **Limited Development**: FTP has seen limited development and innovation compared to newer protocols designed with security and performance in mind.

Modern Alternatives to FTP

Several modern protocols address the shortcomings of FTP, providing more secure and efficient file transfer options:

1. **SFTP (SSH File Transfer Protocol)**
2. **FTPS (FTP Secure)**
3. **HTTPS (Hypertext Transfer Protocol Secure)**
4. **WebDAV (Web Distributed Authoring and Versioning)**
5. **SMB/CIFS (Server Message Block/Common Internet File System)**

4- Disadvantages of Using Active Mode FTP

Active mode FTP has several drawbacks, especially related to network configuration and security:

1. **Firewall and NAT Issues**:
    - **Client-Side Firewall Problems**: In active mode, the client opens a random port and informs the server, which then tries to connect back to the client on this port. If the client is behind a firewall or NAT (Network Address Translation), this incoming connection from the server can be blocked, preventing the data transfer.

    - **Port Range**: Active mode requires the client to open a wide range of ports to accommodate incoming connections from the server, complicating firewall configurations.
2. **Security Concerns**:
    - **Exposing Client Ports**: The client must open ports to allow incoming connections from the server, which can be exploited by attackers to gain unauthorized access.
    - **Data Channel Hijacking**: Since the data channel is established by the server connecting to the client, it can be intercepted or hijacked by a malicious actor.
3. **Configuration Complexity**:
    - **Dynamic Ports**: The requirement for dynamic port ranges for incoming connections on the client side complicates both client and network firewall settings.
    - **Passive FTP as a Default**: Modern firewalls and NAT devices are usually configured with passive FTP in mind, making active FTP less compatible with default network setups.

How Passive Mode Can Handle These Problems?

Passive mode FTP addresses the issues of active mode by changing the way data connections are established:

1. **Firewall and NAT Friendliness**:
    - **Server-Side Connections**: In passive mode, the client initiates both the control and data connections to the server. This avoids the problem of incoming connections to the client, which can be blocked by firewalls or NAT.
    - **Single-Sided Firewall Configuration**: Only the server needs to have open ports for data connections, simplifying the configuration and avoiding the need to open multiple client-side ports.
2. **Enhanced Security**:
    - **No Incoming Connections to Client**: Since the client initiates all connections, there are no open ports on the client side that can be exploited by attackers.
    - **Easier to Secure**: It's easier to secure the server side by controlling which ports are open for passive connections.
3. **Simpler Configuration**:
    - **Fixed Port Range on Server**: The server can be configured to use a specific range of ports for passive mode data connections, making firewall configuration more straightforward.
    - **Compatibility**: Passive mode is more compatible with modern network setups, making it the preferred mode for many FTP clients and servers.

5- Data Transfer Security in FTP and other file transfer protocols

FTP (File Transfer Protocol) does not provide any built-in security measures by default. Here are the key points regarding the security of FTP:

1. **Plaintext Transmission**: FTP transmits data, including usernames, passwords, and files, in plaintext. This means that anyone who can intercept the data can read it, making FTP inherently insecure.
2. **No Encryption**: FTP does not encrypt data during transfer, leaving it vulnerable to interception and eavesdropping.
3. **No Integrity Checking**: FTP lacks mechanisms to verify the integrity of transferred files, making it possible for files to be tampered with during transmission.

Due to these significant security drawbacks, FTP is considered insecure for transferring sensitive or confidential data over the internet.

Secure Alternatives to FTP

To address the security shortcomings of FTP, several secure alternatives have been developed:

1. **SFTP (SSH File Transfer Protocol)**

- SFTP is a secure file transfer protocol that operates over the SSH (Secure Shell) protocol.
- It provides secure authentication and encryption for both the command and data channels.

Security Features:

- **Encryption**: All data transferred over SFTP is encrypted using SSH, ensuring that data is not readable if intercepted.
- **Authentication**: SFTP uses SSH for authentication, supporting strong methods such as public key authentication and password authentication.
- **Data Integrity**: SSH ensures data integrity by using cryptographic hash functions to detect any tampering during transmission.
- **Single Port**: SFTP uses a single port (typically port 22), simplifying firewall configuration and reducing the attack surface.

2. **FTPS (FTP Secure)**

- FTPS is an extension of FTP that adds support for TLS (Transport Layer Security) and SSL (Secure Sockets Layer) encryption.
- FTPS can operate in two modes: explicit (where the client requests security) and implicit (where the client connects to a secure port).

Security Features:

- **Encryption**: FTPS uses TLS/SSL to encrypt the control and data channels, protecting data from interception.
- **Authentication**: FTPS supports certificate-based authentication, providing an additional layer of security.
- **Compatibility**: FTPS can be integrated with existing FTP infrastructures, making it easier to transition from insecure FTP.

- **Multiple Ports**: FTPS may use multiple ports for data transfer, which can complicate firewall configuration.

3. FTP over SSH (FTP via SSH Tunnel)

- FTP over SSH involves tunneling the FTP control and data connections through an SSH tunnel, providing a secure way to use FTP.
- It combines the ease of use of FTP with the security of SSH.

Security Features:

- **Encryption**: The SSH tunnel encrypts all FTP traffic, ensuring that data is not readable if intercepted.
- **Authentication**: SSH provides strong authentication methods, such as public key authentication.
- **Data Integrity**: SSH ensures data integrity by using cryptographic hash functions.
- **Existing FTP Clients**: FTP over SSH allows the use of existing FTP clients, with the added benefit of SSH security.
- **Complexity**: Setting up an SSH tunnel for FTP can be more complex than using SFTP or FTPS.

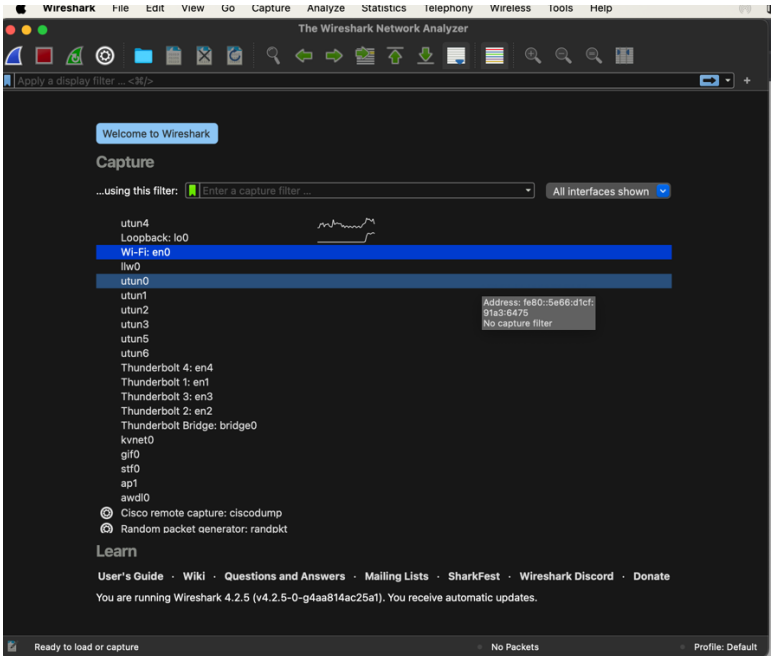
6- Wire Shark for FTP Service

Using **Wireshark**, we can monitor the packets exchanged between the client and server. By generating a file.txt of size **10GB**, we can observe that the GET command has successfully transferred the data from our server's directory to the client's directory. This process allows us to verify the integrity and efficiency of data transmission within the network.

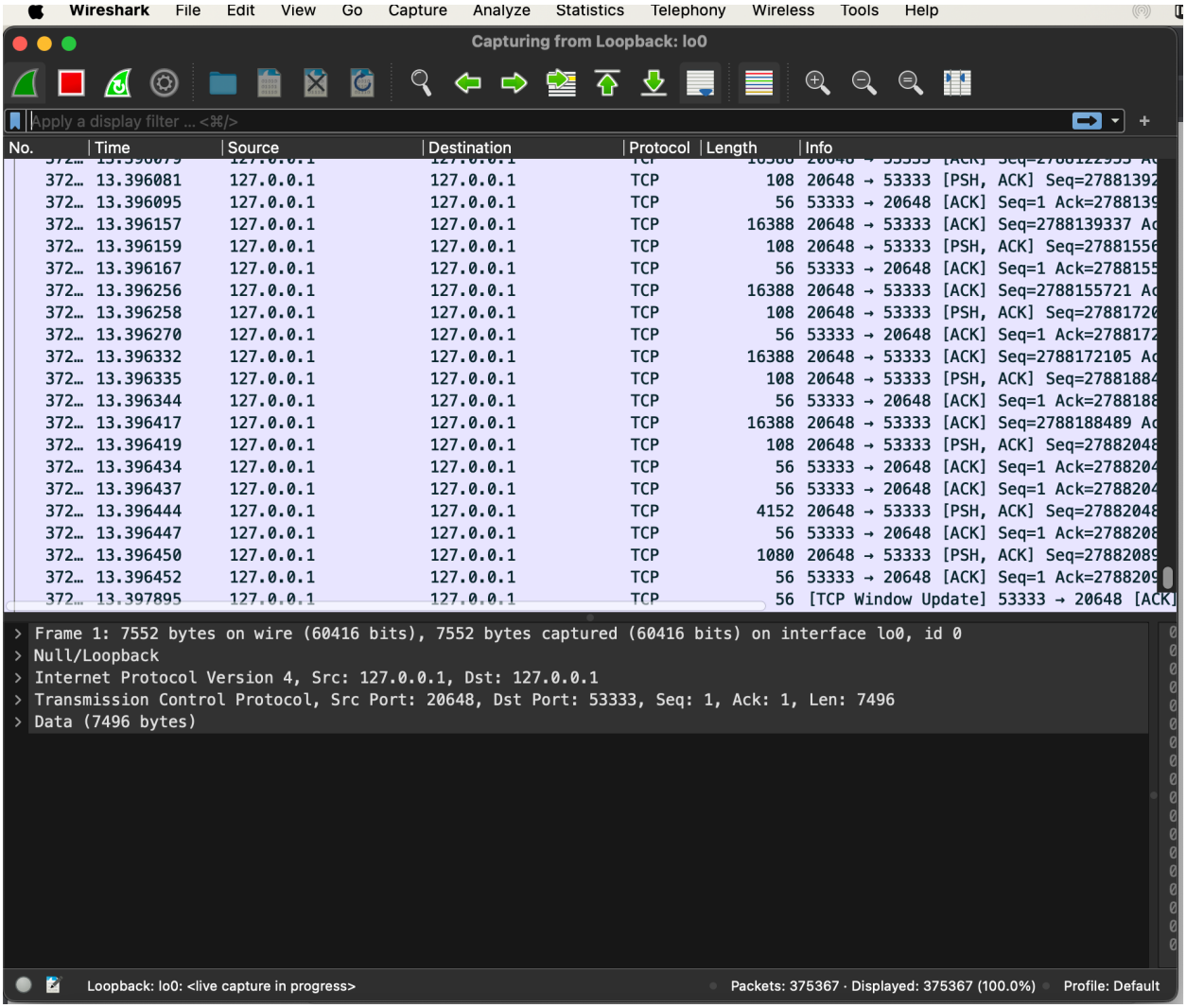
```
ftp>> ls
Connected to data port 15459
total 169320
-rw-r--r--@ 1 aminmola  staff      6852575 Jun 30 22:57 Linear_algebra.pdf
-rw-r--r--  1 aminmola  staff     56376255 Jun 30 22:57 TA_Tutorial.mp4
-rw-r--r--  1 aminmola  staff     22052045 Jul  3 11:21 Tango.mp3
-rw-r--r--  1 aminmola  staff       349558 Jun 30 18:56 The_Death_of_Ivan_Ilyich.epub
-rw-------  1 aminmola  staff  10737418240 Jul  3 13:11 file.txt
-rw-r--r--  1 aminmola  staff      1048618 Jul  3 11:21 test.txt

Server response: Directory send OK
ftp>> get file.txt
Receiving file file.txt of size 10737418240 bytes
Connected to data port 20648
Received file and saved as file.txt
Server response: Transfer complete
ftp>>
```

Here is the start of transferring file.

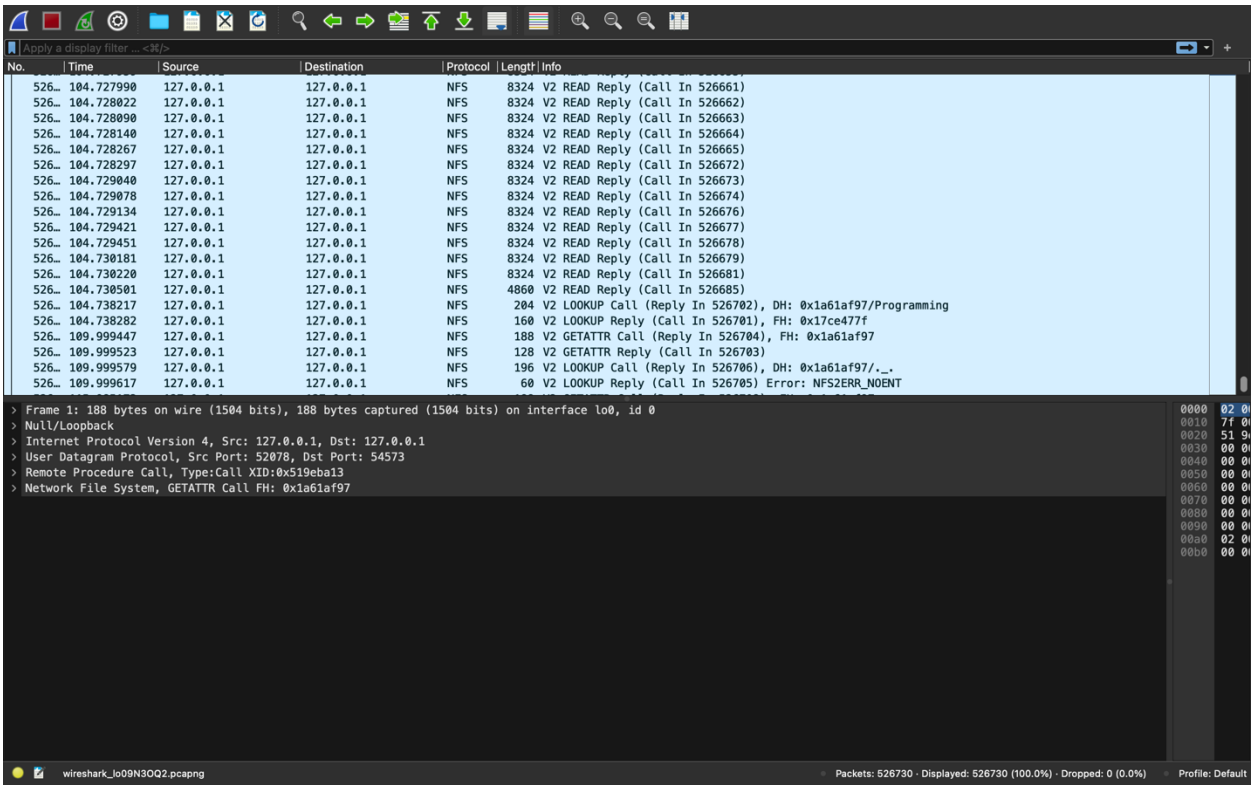


For our implementation, we observe that the packets have a maximum size of **16388 Bytes**. Additionally, there are ACK (Acknowledgment) packets with a size of **56 Bytes** and PSH (Push Function) combined with ACK packets, which are **108 Bytes** in size. These details are crucial for analyzing the network's data transmission characteristics and optimizing performance.



7- Wire Shark for FTP.Sharif.edu

Like previous part I have downloaded a heavy file from ftp sharif server and see the packets on the Wire Shark Application.

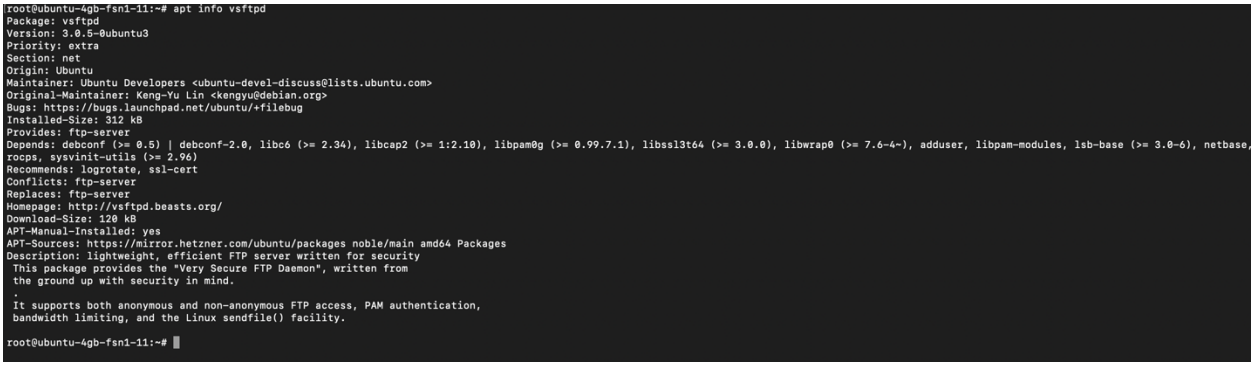


As illustrated in the screenshot, we have a read call with a size of 200 Bytes and the corresponding Read Reply with a size of 8324 Bytes**.** It is also evident that the last data packet, which represents the remainder of the file, is not 8324 Bytes but rather 4860 Bytes. This indicates the final segment of data being transmitted to complete the file transfer process.

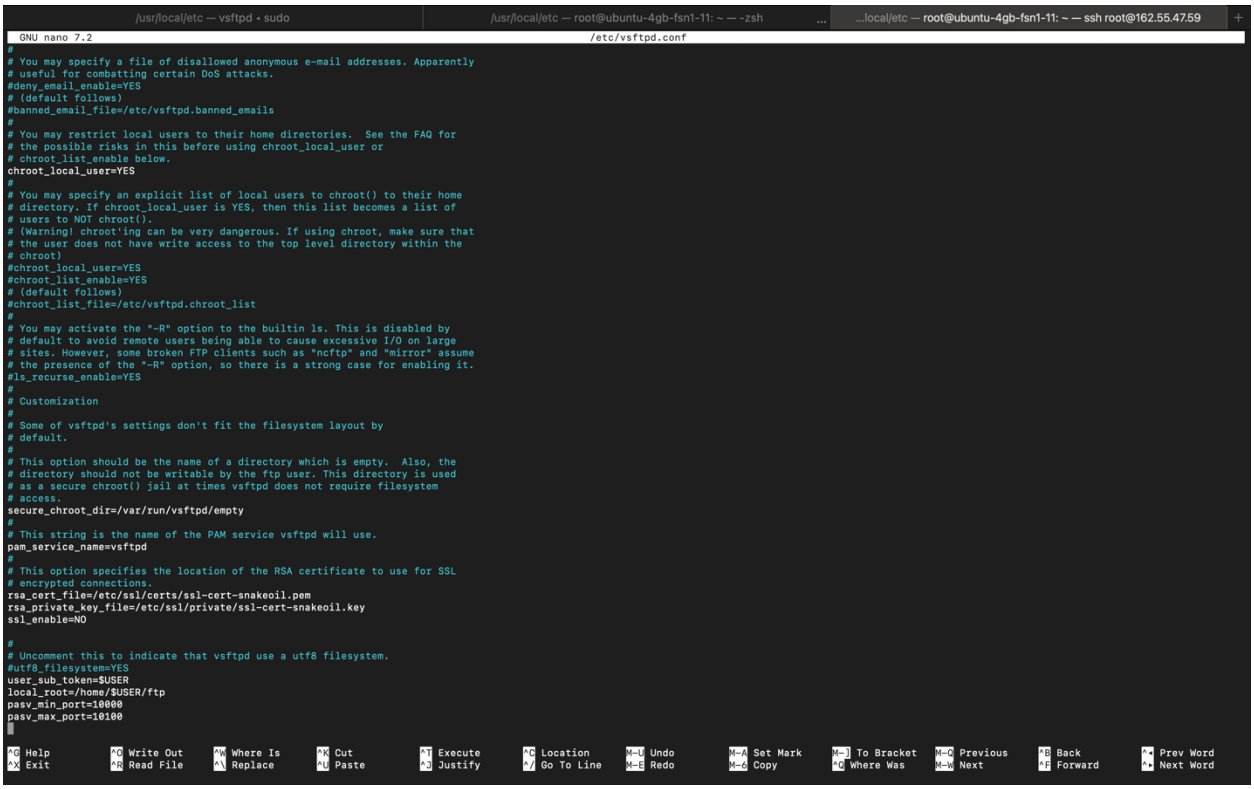
Part 3: Setting Up a Local FTP Server

Part A - Install and Configure FTP Server

I have installed the **vsftpd** package with the apt package manager as below and here is the information for this package.



After installing this package, I have change and save the config file at “/etc/vsftpd.confg” using nano.



now I have start the vsftpd service using the bash command below.

“sudo systemctl start vsftpd”

Here are the running processors with vsftpd included name.

```
root@ubuntu-4gb-fsn1-11:~# ps aux | grep vsftpd
root       18771  0.0  0.1  16728  7040 ?        S    Jul03   0:00 sudo vsftpd /etc/vsftpd.conf
root       18772  0.0  0.0  16728  2472 ?        Ss   Jul03   0:00 sudo vsftpd /etc/vsftpd.conf
root       18773  0.0  0.0   9088  3456 ?        S    Jul03   0:00 vsftpd /etc/vsftpd.conf
root       30080  0.0  0.0   6544  2304 pts/0    S+   09:38   0:00 grep --color=auto vsftpd
root@ubuntu-4gb-fsn1-11:~#
```

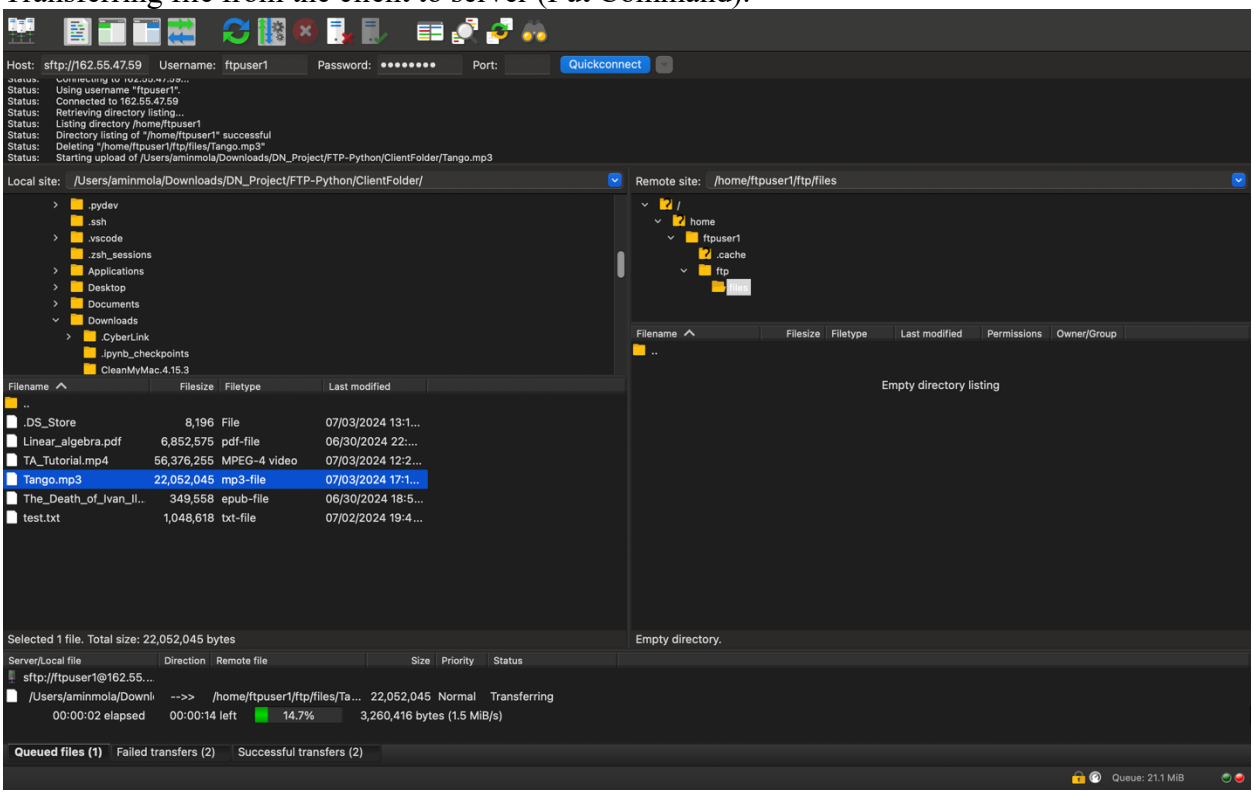
Adding FTP users

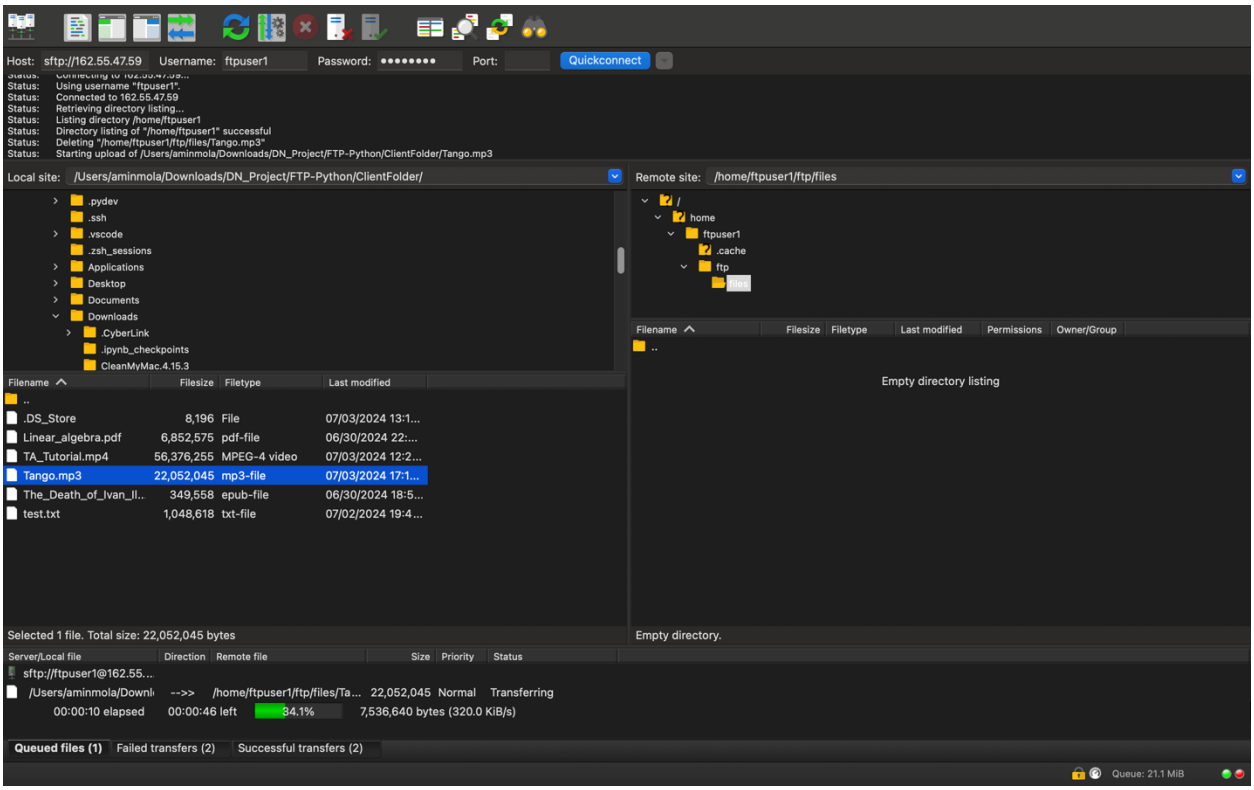
With following commands, I just added three users and added a directory for each of the users.

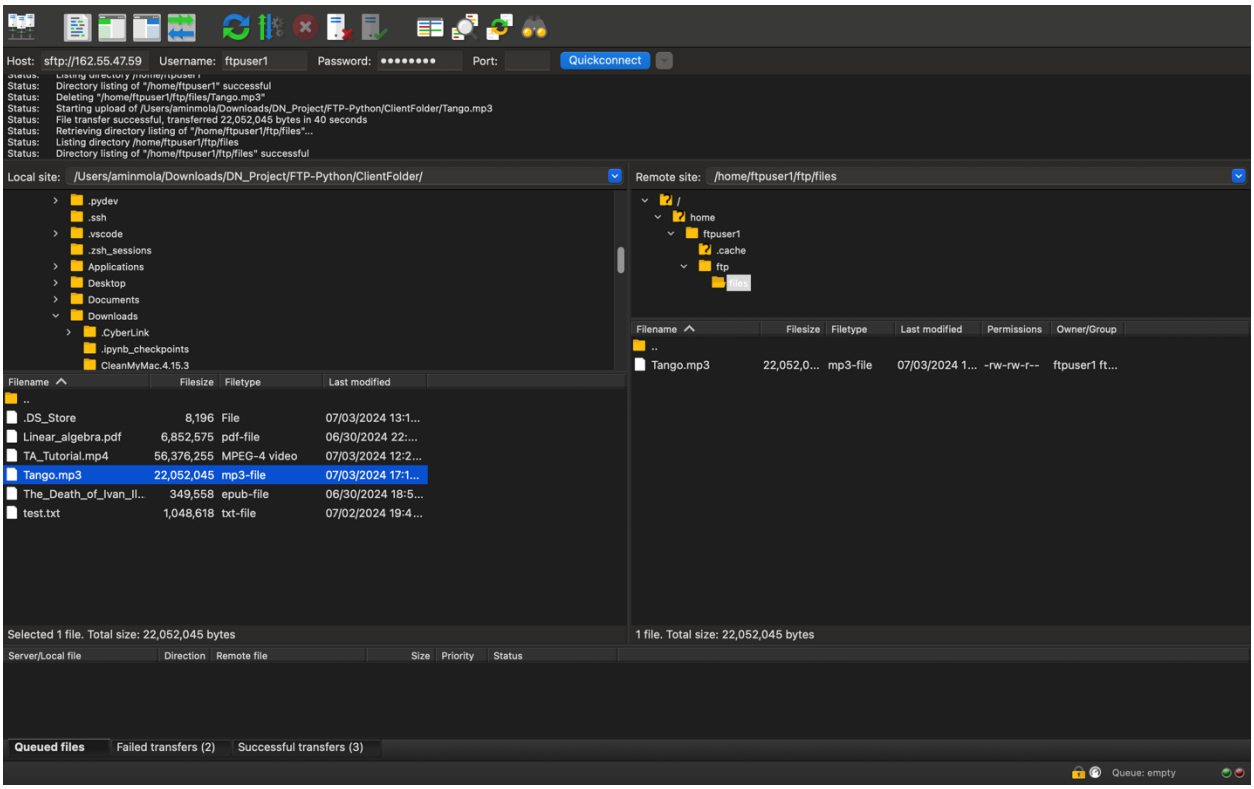
sudo adduser ftpuser1
sudo adduser ftpuser2
sudo adduser ftpuser3

sudo mkdir /home /ftpuser1
sudo mkdir /home/ftpuser2
sudo mkdir /home/ftpuser3

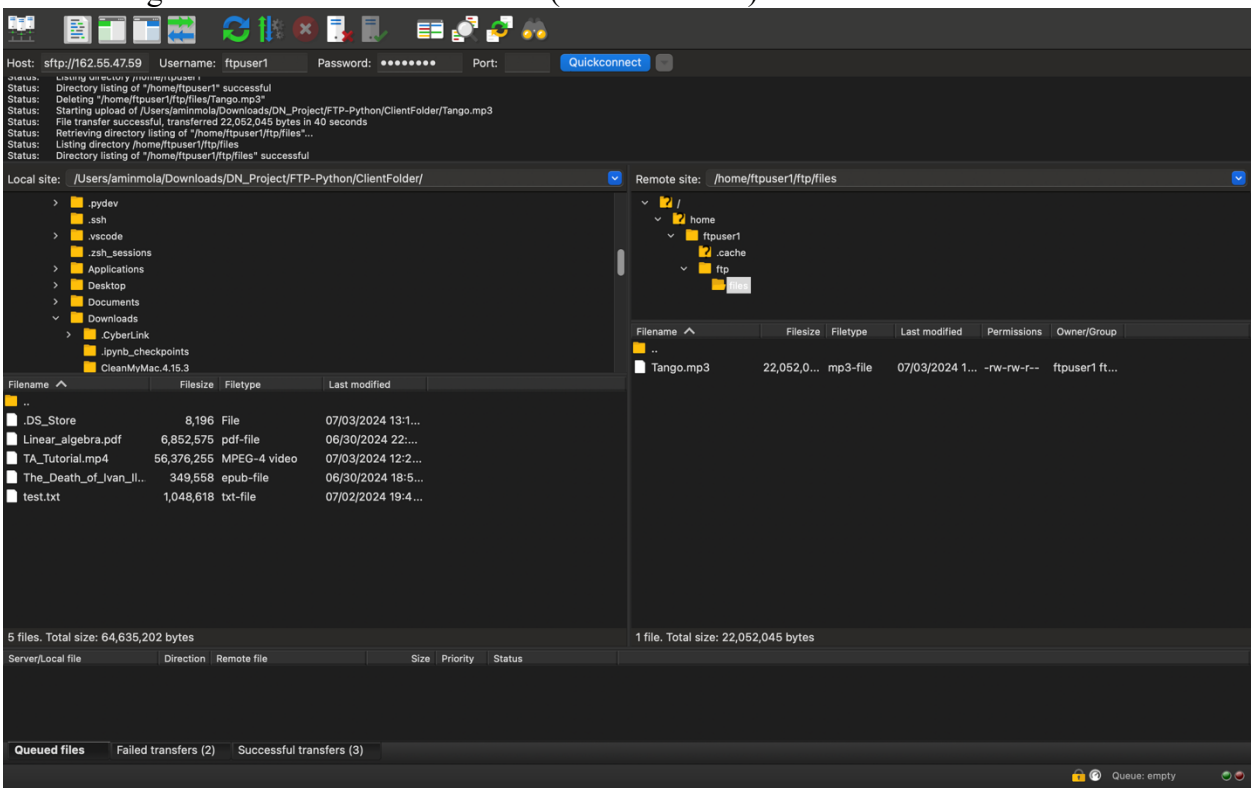
Now I have used a FileZilla as a FTP client UI to get and put the files from my server.
Here are my transmissions between the client and server.
Transferring file from the client to server (Put Command):

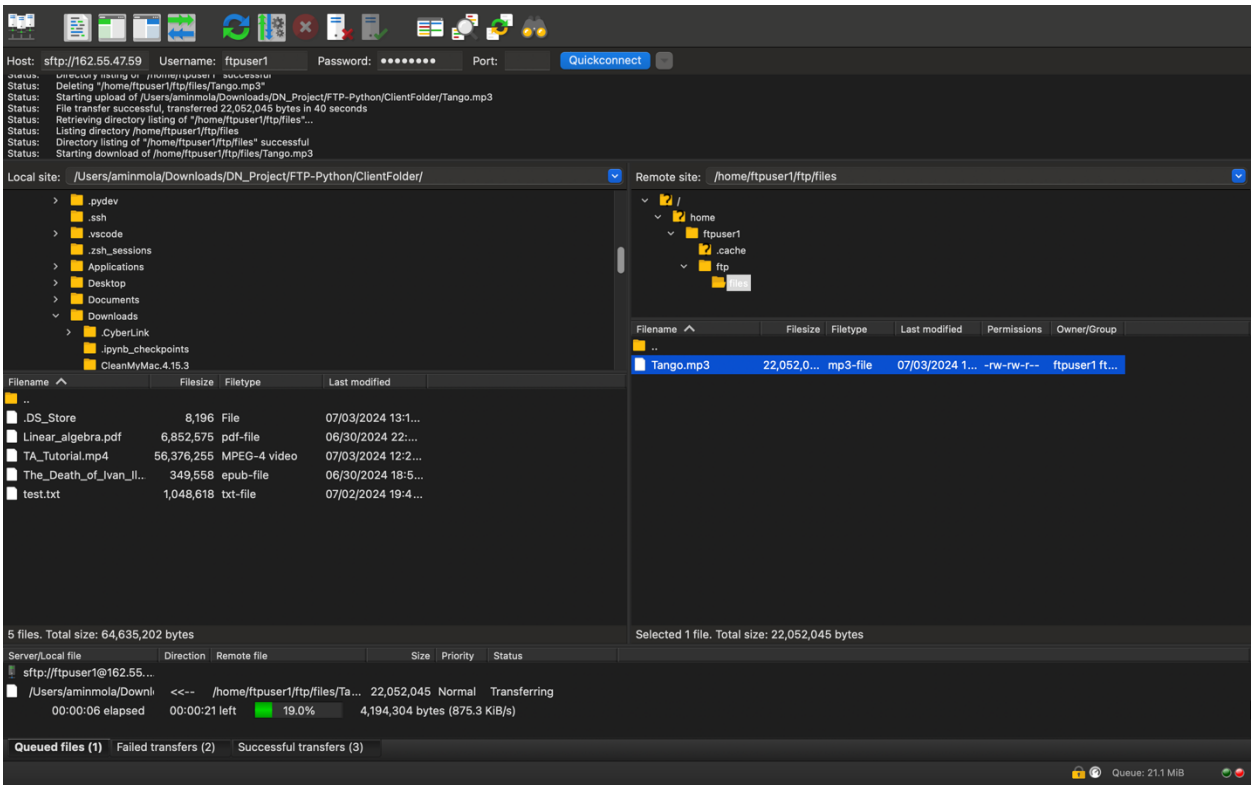


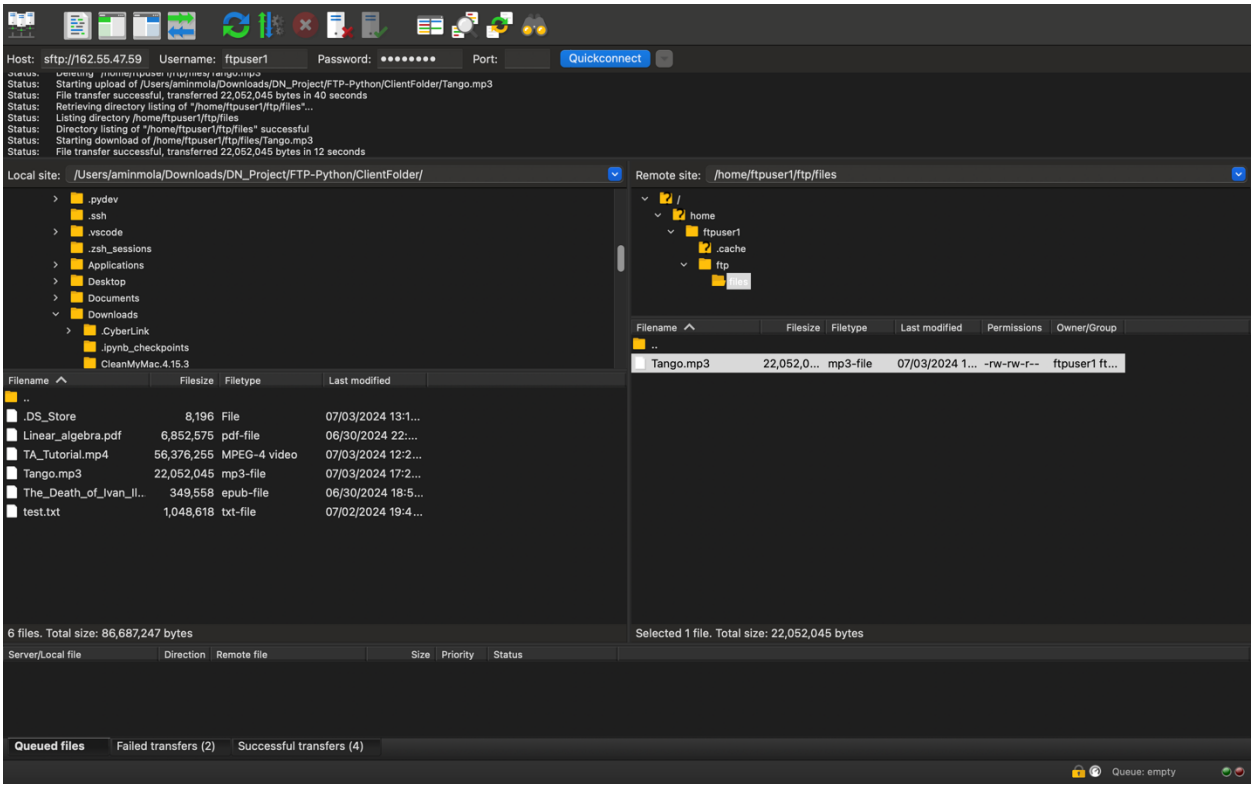




Transferring file from the Server to Client (Get Command):




Host: sftp://162.55.47.59 Username: ftpuser1 Password: •••••••• Port: Quickconnect
Status: Deleting "/home/ftpuser1/ftp/files/Tango.mp3"
Status: Starting upload of /Users/aminmola/Downloads/DN_Project/FTP-Python/ClientFolder/Tango.mp3
Status: File transfer successful, transferred 22,052,045 bytes in 40 seconds
Status: Retrieving directory listing of "/home/ftpuser1/ftp/files"...
Status: Listing directory /home/ftpuser1/ftp/files
Status: Directory listing of "/home/ftpuser1/ftp/files" successful
Status: Starting download of /home/ftpuser1/ftp/files/Tango.mp3
Local site: /Users/aminmola/Downloads/DN_Project/FTP-Python/ClientFolder/
Remote site: /home/ftpuser1/ftp/files
.pydev
.ssh
.vscode
.zsh_sessions
Applications
Desktop
Documents
Downloads
.CyberLink
.ipynb_checkpoints
CleanMyMac.4.15.3
/
home
ftpuser1
.cache
ftp
files
Filename Filesize Filetype Last modified
.. .DS_Store 8,196 File 07/03/2024 13:1...
Linear_algebra.pdf 6,852,575 pdf-file 06/30/2024 22:...
TA_Tutorial.mp4 56,376,255 MPEG-4 video 07/03/2024 12:2...
The_Death_of_Ivan_Il.. 349,558 epub-file 06/30/2024 18:5...
test.txt 1,048,618 txt-file 07/02/2024 19:4...
Filename Filesize Filetype Last modified Permissions Owner/Group
..
Tango.mp3 22,052,0... mp3-file 07/03/2024 1... -rw-rw-r-- ftpuser1 ft...
5 files. Total size: 64,635,202 bytes
Selected 1 file. Total size: 22,052,045 bytes
Server/Local file Direction Remote file Size Priority Status
sftp://ftpuser1@162.55...
/Users/aminmola/Downl <<-- /home/ftpuser1/ftp/files/Ta... 22,052,045 Normal Transferring
00:00:06 elapsed 00:00:21 left 19.0% 4,194,304 bytes (875.3 KiB/s)
Queued files (1) Failed transfers (2) Successful transfers (3)
Queue: 21.1 MiB


Host: sftp://162.55.47.59 Username: ftpuser1 Password: •••••••• Port: Quickconnect
Status: Starting upload of /Users/aminmola/Downloads/DN_Project/FTP-Python/ClientFolder/Tango.mp3
Status: File transfer successful, transferred 22,052,045 bytes in 40 seconds
Status: Retrieving directory listing of "/home/ftpuser1/ftp/files"...
Status: Listing directory /home/ftpuser1/ftp/files
Status: Directory listing of "/home/ftpuser1/ftp/files" successful
Status: Starting download of /home/ftpuser1/ftp/files/Tango.mp3
Status: File transfer successful, transferred 22,052,045 bytes in 12 seconds
Local site: /Users/aminmola/Downloads/DN_Project/FTP-Python/ClientFolder/
Remote site: /home/ftpuser1/ftp/files
.pydev
.ssh
.vscode
.zsh_sessions
Applications
Desktop
Documents
Downloads
.CyberLink
.ipynb_checkpoints
CleanMyMac.4.15.3
/
home
ftpuser1
.cache
ftp
files
Filename Filesize Filetype Last modified
.. .DS_Store 8,196 File 07/03/2024 13:1...
Linear_algebra.pdf 6,852,575 pdf-file 06/30/2024 22:...
TA_Tutorial.mp4 56,376,255 MPEG-4 video 07/03/2024 12:2...
Tango.mp3 22,052,045 mp3-file 07/03/2024 17:2...
The_Death_of_Ivan_Il.. 349,558 epub-file 06/30/2024 18:5...
test.txt 1,048,618 txt-file 07/02/2024 19:4...
Filename Filesize Filetype Last modified Permissions Owner/Group
..
Tango.mp3 22,052,0... mp3-file 07/03/2024 1... -rw-rw-r-- ftpuser1 ft...
6 files. Total size: 86,687,247 bytes
Selected 1 file. Total size: 22,052,045 bytes
Server/Local file Direction Remote file Size Priority Status
Queued files Failed transfers (2) Successful transfers (4)
Queue: empty

Part B – Firewall Configuration on FTP Server

We the following command I have set up firewall configuration and enable it on my server.

```
root@ubuntu-4gb-fsn1-11:~# ls
root@ubuntu-4gb-fsn1-11:~# sudo ufw status
Status: inactive
root@ubuntu-4gb-fsn1-11:~# sudo ufw allow 20/tcp
sudo ufw allow 21/tcp
sudo ufw allow 10000:10100/tcp
Rules updated
Rules updated (v6)
Rules updated
Rules updated (v6)
Rules updated
Rules updated (v6)
root@ubuntu-4gb-fsn1-11:~# sudo ufw status
Status: inactive
root@ubuntu-4gb-fsn1-11:~# sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
root@ubuntu-4gb-fsn1-11:~# sudo ufw reload
Firewall reloaded
root@ubuntu-4gb-fsn1-11:~#
```

Why is it Important to Configure Firewall Rules for FTP Traffic?

Configuring firewall rules for FTP traffic is crucial for several reasons:

1. **Security:** Firewalls help protect your system from unauthorized access and malicious attacks. By explicitly allowing only the necessary ports, you reduce the attack surface and prevent unauthorized access to other services on your server.
2. **Control:** Proper firewall configuration ensures that only legitimate FTP traffic is allowed. This control helps in monitoring and managing the traffic to and from your FTP server.
3. **Compliance:** Many organizations have security policies and compliance requirements that mandate the use of firewalls to protect sensitive data and systems. Configuring firewall rules ensures adherence to these policies.

What are the Specific Ports Used for FTP, and How Do You Open Them in the Firewall?

FTP uses the following specific ports:

- **Port 21:** Used for the control connection (commands and responses).
- **Port 20:** Used for the data connection in active mode (not commonly used with modern passive mode setups).
- **Passive Mode Ports:** A range of ports (e.g., 10000-10100) used for data connections in passive mode. This range is configurable in the FTP server settings.

Part C- Backing Up vsftpd Configuration Files

I have use the following command to back up the configuration file as below:

```
root@ubuntu-4gb-fsn1-11:~# sudo cp /etc/vsftpd.conf /etc/vsftpd.conf.orig
root@ubuntu-4gb-fsn1-11:~# ls
root@ubuntu-4gb-fsn1-11:~#
```

Why is it important to back up configuration files before editing them?

Backing up configuration files before making changes is crucial for several reasons:

- **Recovery from Errors**: If a mistake is made while editing the configuration file, having a backup allows you to revert to the original working state.
- **System Stability**: Incorrect configurations can lead to system instability or service failures. A backup ensures that the system can be quickly restored to a stable state.
- **Audit Trail**: Backups serve as a record of changes, which can be useful for troubleshooting and understanding how configurations have evolved over time.
- **Learning from Mistakes**: Comparing the backup with the edited file helps in identifying what went wrong and learning from the errors.

Modifying the vsftpd Configuration to Change the Default Directory

Why is it beneficial to change the default directory for FTP users?

Changing the default directory for FTP users can offer several benefits:

- **Security**: Limiting users to a specific directory reduces the risk of unauthorized access to sensitive areas of the filesystem.
- **Organization**: Segregating FTP data from other system files can help keep the server organized and easier to manage.
- **Customization**: Allows setting up different directories for different users or groups based on their needs, improving the user experience.

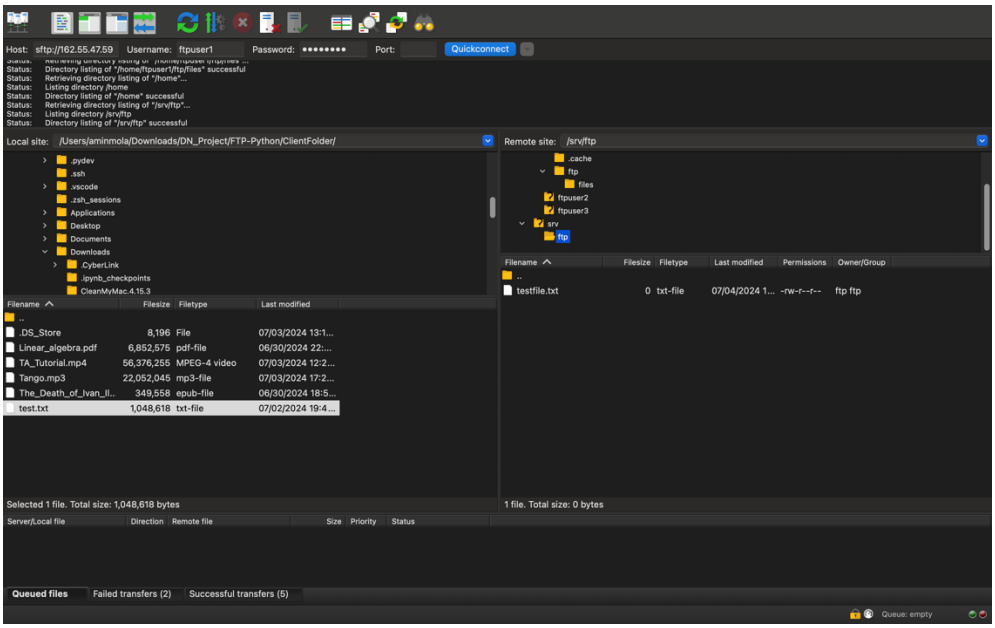
Change the Default Directory for FTP Users
To change the default directory for FTP users first I have create a directory as below and edit the configuration file by adding the local_root=/srv/ftp line and restart the vsftpd service.

sudo mkdir /srv/ftp
sudo chown ftp:ftp /srv/ftp

Creating a New File in the New Directory and Checking Accessibility

I have created a new file using the following command and as you can see it in screen shot is not available for the ftpuser1.

```
*** System restart required ***
Pending kernel upgrade!
Running kernel version:
  6.8.0-31-generic
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.8.0-36-generic.
Last login: Wed Jul  3 15:15:19 2024 from 162.55.47.59
root@ubuntu-4gb-fsn1-11:~# ls
root@ubuntu-4gb-fsn1-11:~# cd /srv/ftp
root@ubuntu-4gb-fsn1-11:/srv/ftp# sudo touch testfile.txt
sudo chown ftp:ftp testfile.txt
root@ubuntu-4gb-fsn1-11:/srv/ftp#
```



Part D- Secure FTP

1. Encrypt FTP Traffic: Configure the FTP Server to Use SSL/TLS Encryption
Like previous part I have generate SSL key as below and change configuration file as following and restart the vsftpd service.

Generate SSL Certificate and Key**:**

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/vsftpd.pem -out /etc/ssl/private/vsftpd.pem

```
ssl_enable=YES
rsa_cert_file=/etc/ssl/private/vsftpd.pem
rsa_private_key_file=/etc/ssl/private/vsftpd.pem
ssl_ciphers=HIGH
force_local_data_ssl=YES
force_local_logins_ssl=YES
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
require_ssl_reuse=NO
ssl_tlsv1_1=YES
ssl_tlsv1_2=YES
```

Why is it important to encrypt FTP traffic?

Encrypting FTP traffic is important because it ensures that the data transferred between the FTP client and server is secure and cannot be intercepted or tampered with by malicious actors. Without encryption, sensitive information such as usernames, passwords, and data files can be easily captured and exploited.

What are the benefits of using SSL/TLS encryption for FTP?

The benefits of using SSL/TLS encryption for FTP include:

Data Protection: Encrypts data during transmission, protecting it from eavesdropping and interception.

Authentication: Ensures that the server the client is connecting to is legitimate, preventing man-in-the-middle attacks.

Integrity: Protects data from being altered during transmission, ensuring that the received data is the same as the sent data.

Compliance: Helps meet regulatory requirements for data protection and privacy.

2. Limit User Access: Implement Measures to Limit User Access to Specific Directories and Functionalities
With the following lines in the configuration file we can limit access:

```
chroot_local_user=YES
allow_writeable_chroot=YES
```

Why do we need to secure FTP servers?

Securing FTP servers is essential for several reasons:

- **Prevent Unauthorized Access**: Ensures that only authorized users can access the server and its data, preventing unauthorized access to sensitive information.
- **Protect Data Integrity**: Prevents unauthorized users from modifying or deleting data on the server.
- **Ensure Availability**: Protects the server from attacks that could disrupt its availability, such as denial-of-service attacks.
- **Compliance**: Helps organizations comply with data protection regulations and standards, protecting both the organization and its users.

There are several methods to limit user access on an FTP server:

1. **Chroot Jail**: Restrict users to their home directories, preventing them from accessing the broader filesystem.

   ```
   chroot_local_user=YES
   allow_writeable_chroot=YES
   ```

2. **User-Specific Configuration**: Create user-specific configuration files to define access controls for individual users.

   ```
   sudo nano /etc/vsftpd/user_list
   ```

3. **Restrict Write Permissions**: Limit write permissions to only those users who need it, preventing unauthorized modification of files.

   ```
   write_enable=NO
   ```

4. **Limit Command Set**: Restrict the set of FTP commands available to users, limiting their ability to perform certain actions.

   ```
   cmds_allowed=ABOR,CWD,DELE,LIST,MDTM,MLSD,NLST,RETR,RNFR,RNTO,STOR
   ```

5. **User Authentication**: Use strong authentication mechanisms, such as SSH keys or multi-factor authentication, to verify user identity.
6. **Access Control Lists (ACLs)**: Use ACLs to specify which users can access specific directories and files.

Part 4: Bandwidth and Transfer Rate control

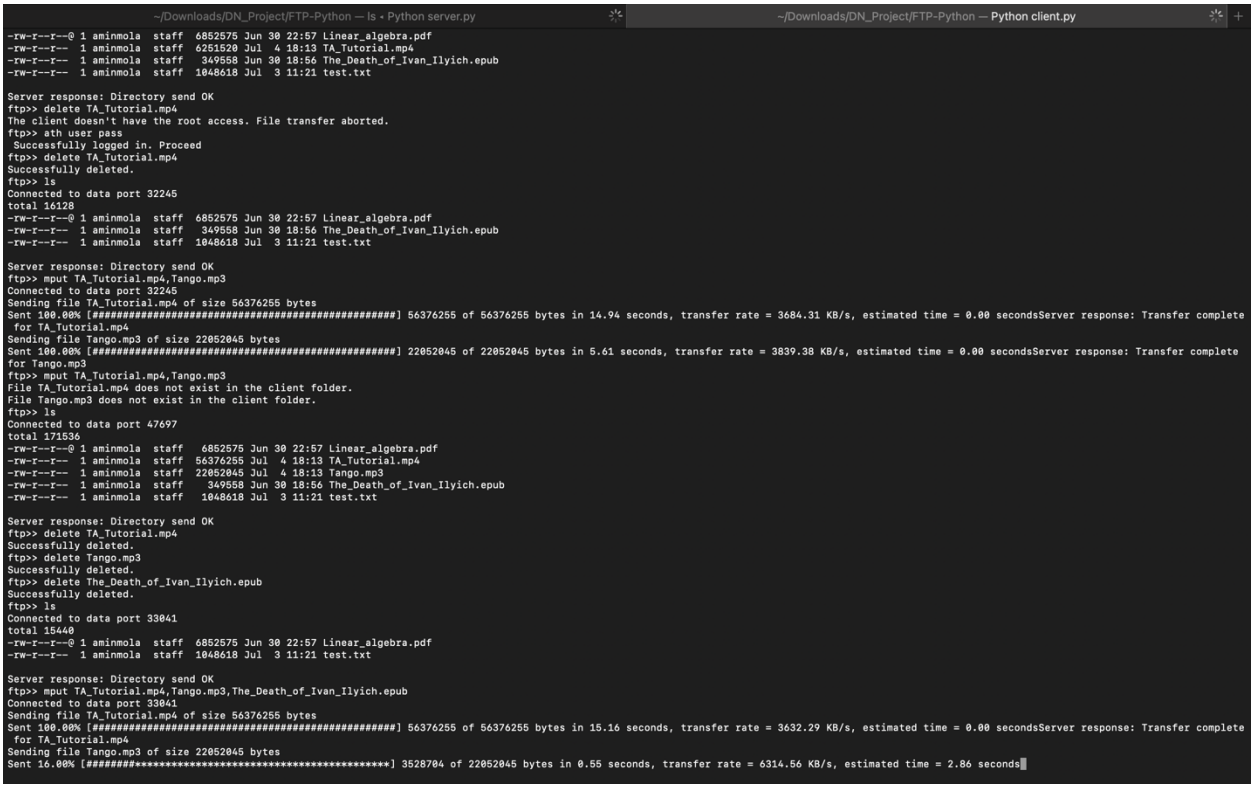
Measurement of Transfer Rate:

For Transfer Rate measurement I have record the time for each command that includes a file transferring (put, mput, get) and also calculate the transferred size so I can have the transfer rate.

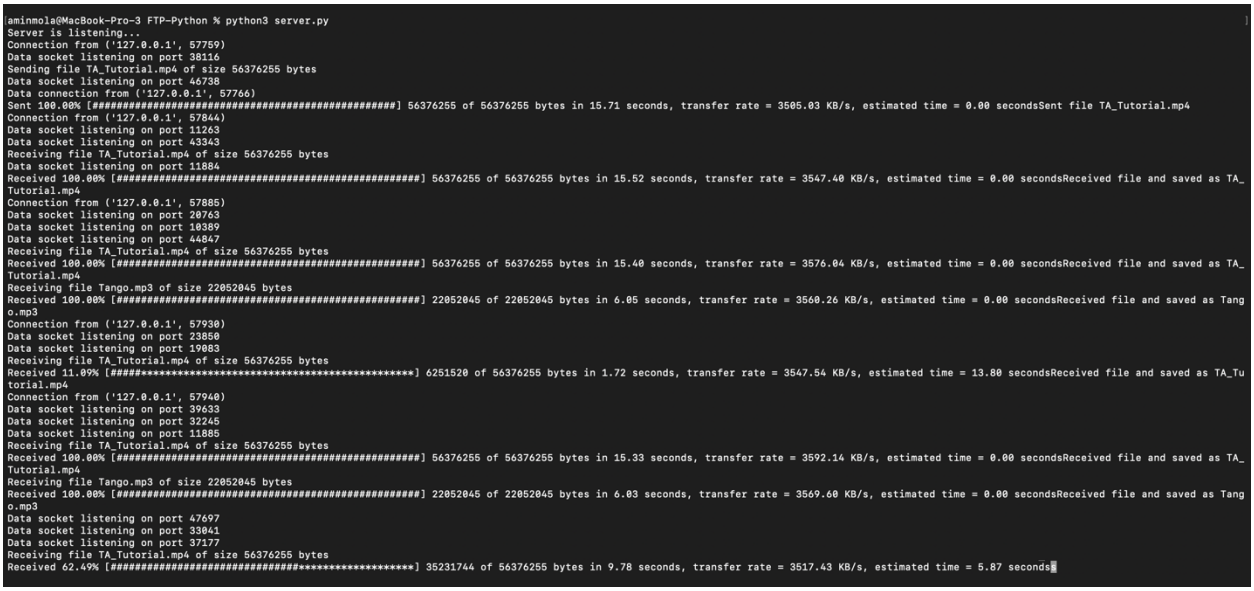
Limiting Bandwidth:

Based on the maximum Bandwidth I have set a sleep time so we can control the transfer rate and congestion if there is such a phenomenon. I also added the estimated time based on the the file size and current transfer rate

Client side:



Server side:


aminmola@MacBook-Pro-3 FTP-Python % python3 server.py
Server is listening...
Connection from ('127.0.0.1', 57759)
Data socket listening on port 38116
Sending file TA_Tutorial.mp4 of size 56376255 bytes
Data socket listening on port 46738
Data connection from ('127.0.0.1', 57766)
Sent 100.00% [##] 56376255 of 56376255 bytes in 15.71 seconds, transfer rate = 3505.03 KB/s, estimated time = 0.00 secondsSent file TA_Tutorial.mp4
Connection from ('127.0.0.1', 57844)
Data socket listening on port 11263
Data socket listening on port 43343
Receiving file TA_Tutorial.mp4 of size 56376255 bytes
Data socket listening on port 11884
Received 100.00% [##] 56376255 of 56376255 bytes in 15.52 seconds, transfer rate = 3547.40 KB/s, estimated time = 0.00 secondsReceived file and saved as TA_
Tutorial.mp4
Connection from ('127.0.0.1', 57885)
Data socket listening on port 20763
Data socket listening on port 10389
Data socket listening on port 44847
Receiving file TA_Tutorial.mp4 of size 56376255 bytes
Received 100.00% [##] 56376255 of 56376255 bytes in 15.40 seconds, transfer rate = 3576.04 KB/s, estimated time = 0.00 secondsReceived file and saved as TA_
Tutorial.mp4
Receiving file Tango.mp3 of size 22052045 bytes
Received 100.00% [##] 22052045 of 22052045 bytes in 6.05 seconds, transfer rate = 3560.26 KB/s, estimated time = 0.00 secondsReceived file and saved as Tang
o.mp3
Connection from ('127.0.0.1', 57930)
Data socket listening on port 23850
Data socket listening on port 19083
Receiving file TA_Tutorial.mp4 of size 56376255 bytes
Received 11.09% [#####**] 6251520 of 56376255 bytes in 1.72 seconds, transfer rate = 3547.54 KB/s, estimated time = 13.80 secondsReceived file and saved as TA_Tu
torial.mp4
Connection from ('127.0.0.1', 57940)
Data socket listening on port 39633
Data socket listening on port 32245
Data socket listening on port 11885
Receiving file TA_Tutorial.mp4 of size 56376255 bytes
Received 100.00% [##] 56376255 of 56376255 bytes in 15.33 seconds, transfer rate = 3592.14 KB/s, estimated time = 0.00 secondsReceived file and saved as TA_
Tutorial.mp4
Receiving file Tango.mp3 of size 22052045 bytes
Received 100.00% [##] 22052045 of 22052045 bytes in 6.03 seconds, transfer rate = 3569.60 KB/s, estimated time = 0.00 secondsReceived file and saved as Tang
o.mp3
Data socket listening on port 47697
Data socket listening on port 33041
Data socket listening on port 37177
Receiving file TA_Tutorial.mp4 of size 56376255 bytes
Received 62.49% [###**********] 35231744 of 56376255 bytes in 9.78 seconds, transfer rate = 3517.43 KB/s, estimated time = 5.87 seconds