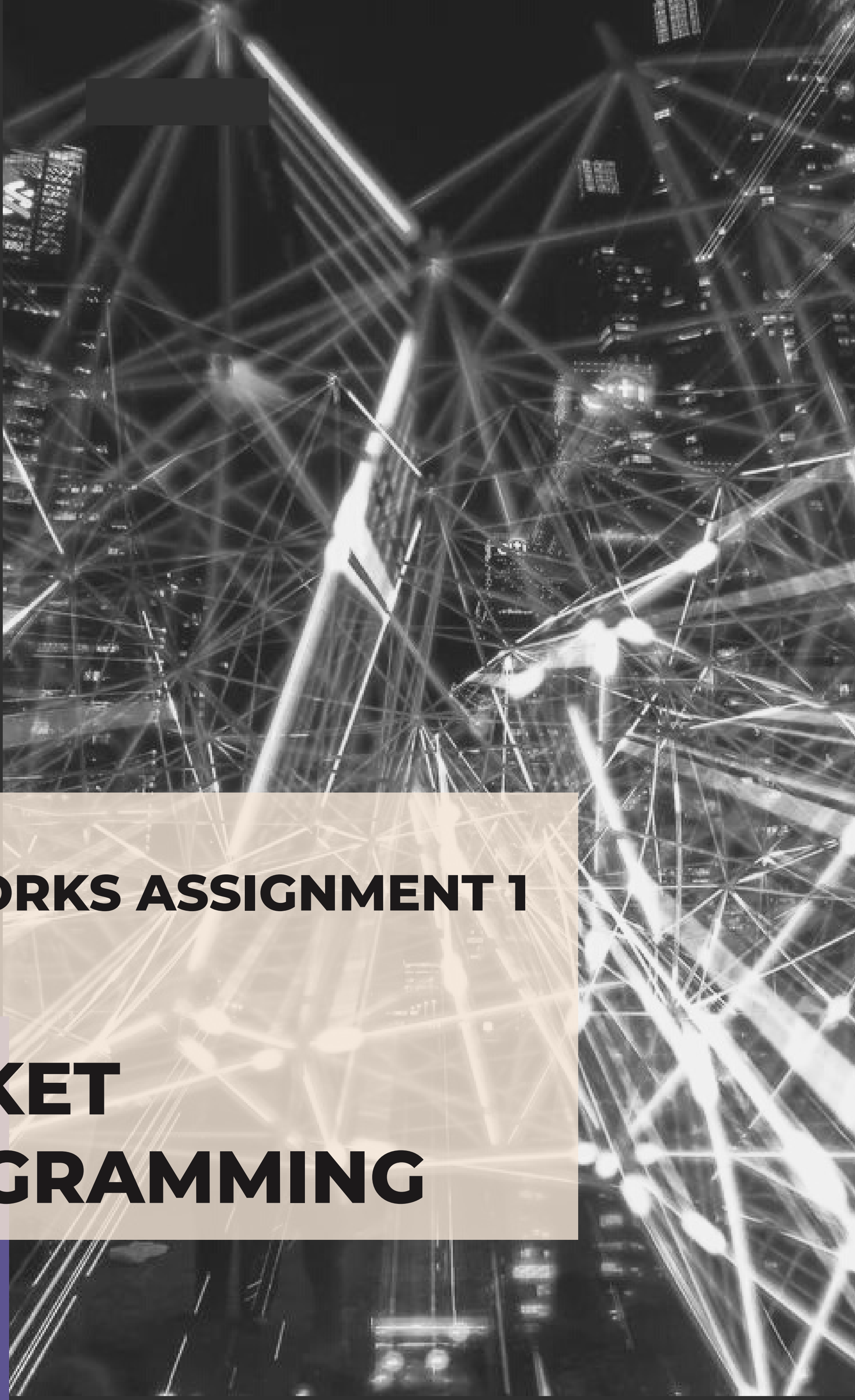
CSC3002F



NETWORKS ASSIGNMENT 1

SOCKET PROGRAMMING

BY:

SRKYUD001 - Yudhvir Sirkissoon
KTHYUS001 - Yusuf Kathrada
MTWYON001 - Yondela Matwele

SYSTEM FUNCTIONALITY

The FileHub network application utilizes a client-server model. This allows clients to upload, download to-and-from the server as well as list the available files. The application works using TCP as a protocol to transport the messages between the client and the server from the application layer. TCP was preferred as it ensures the reliability a file-sharing application requires at the expense of its heavy overheads.

The application can function across different operating systems, allowing users to not be restricted by their devices. The server has an always-on design enabling clients to access the application's services at their convenience. The application also provides file security as users can upload files in either an open or protected state, with only clients with the correct key being able to download and view the protected files associated with that key. The application uses SQLite as it provides a lightweight and intuitive solution for storing information about the files in the server. Finally, the server was dynamically designed in order to accommodate a variety of user inputted arguments, and where necessary, appropriate response and error messages are sent to the client.

The server sends the files as a whole to the transport layer; this is done so that the data is broken up by the TCP layer instead of the data being broken up in the application layer. This has the effect of limiting the file size that we can send according to how much RAM is available on the server. Binary files need to be 5MB minimum in size in order to be correctly transferred.

SERVER FEATURES

Sending Receiving Files

The application allows clients to upload files to the server via UPLOAD requests. The server then securely stores the client's file for later use. The client is also able to download a file from the server via the GET request and the file will be sent from the server directly to the client.

File Privacy

In the uploading of files to the server process, clients are able to specify whether they wish to upload a protected or open file. By protecting a file, the application adds a layer of security by prompting the user to attach a key. Only clients with the correct key may download a protected file whereas any client may request an open file. If a request to download a protected file is sent without the incorrect key, the client will be notified.

Querying Files

This feature allows the client to query the server to list all the available files present. The server responds with a list of files specifying the file size as well as information around the last modified date and time. After the client confirms that the desired file is present they are able to request to download the file from the server.

File Validation

The application has a file validation system which utilizes a hashing algorithm. This is to ensure that a file was not compromised during transit. The fixed-length hash values are compared prior and post transit which will verify the file has not been altered.

SQLite Database

SQlite was used to create and manage the database used to store file information. It provides only the necessary tools in storing file data and therefore provides great utility with no heavy overheads. The database creates a table at the server's initial implementation and queries are made directly in the application source code.

03 INDIVIDUAL FEATURES

MTWYON001 - Yondela Matwele

The client implementation is a command line interface that allows the user to:

- Download files from the server
    - The user can download files that are accessible to everyone and the ones that are locked using a key (provided that the user provides the correct key).
- Upload files to the server
    - If the file exists the user needs to upload a different file.
    - The uploaded file can be protected using a key.
    - Some files can be accessible to everyone which means they are open.
- List files that are in the server
    - The user can list the files that are open and ones that are protected with a key.
- The user can get help with how to use the command line interface and it guides them on which argument combinations to use.

KTHYUS001 - Yusuf Kathrada

The client implementation accepts command line arguments in which the user specifies the host/ip address, port number, request (GET/UPLOAD/LIST), privacy (OPEN/PROTECTED), key <if PROTECTED is specified>, file name <if UPLOAD or GET is specified>. If any of the fields are incorrectly entered by the user, a message notifying the user is sent outlining the expected format similar to the description above. Additionally, a help feature was added. The user can simply enter HELP into the command line and the client application with a response, assisting the user. An additional feature was added to handle file duplication and file not existing cases. This is done by traversing through the server to check if the file the client is attempting to upload matches any existing files stored on the server to prevent duplication. The file does not exist where the file the user is trying to upload does not exist, possibly due to an incorrect name or directory being entered. Finally, a checksum algorithm was added to validate that the file sent and received match and that they are correct.

SRKYUD001 - Yudhvir Sirkissoon

The srkyud001_client implements a user-friendly interface allowing its users to download files from the server using the GET command, upload files to the server using the UPLOAD command, and list files on the server USING the LIST command. All command formats can be seen by using the '--help' flag after the desired command. e.g. UPLOAD --help
The general user input follows this format:

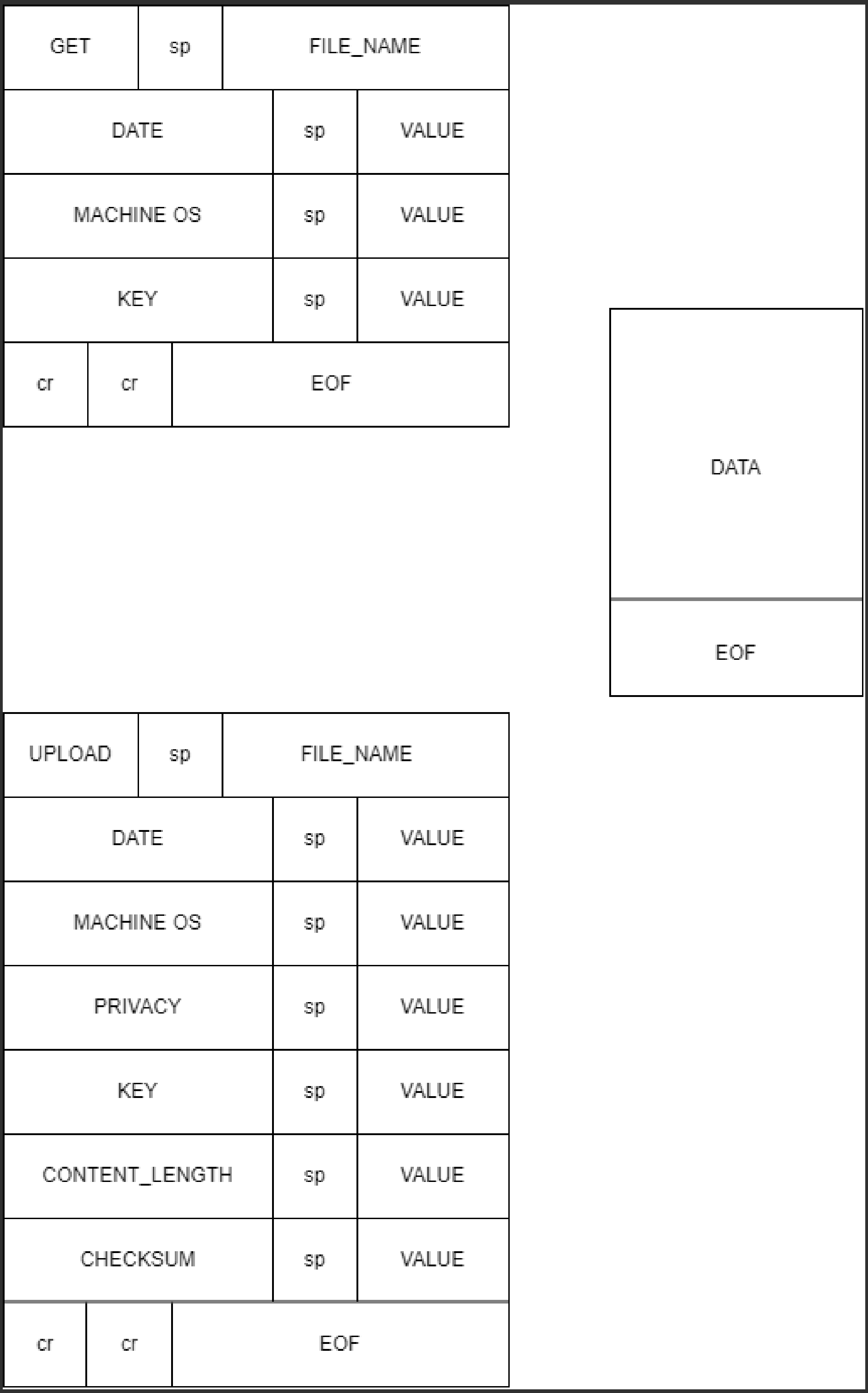
[server-ip] [port] [command] [file_name] <options>

where <options> are the file privacy and/or the file key
The server implements a sha256 algorithm to ensure that the contents of the file are sent correctly. The sqlite database ensures a reliable representation of the data stored on the server side.

PROTOCOL SPECIFICATION

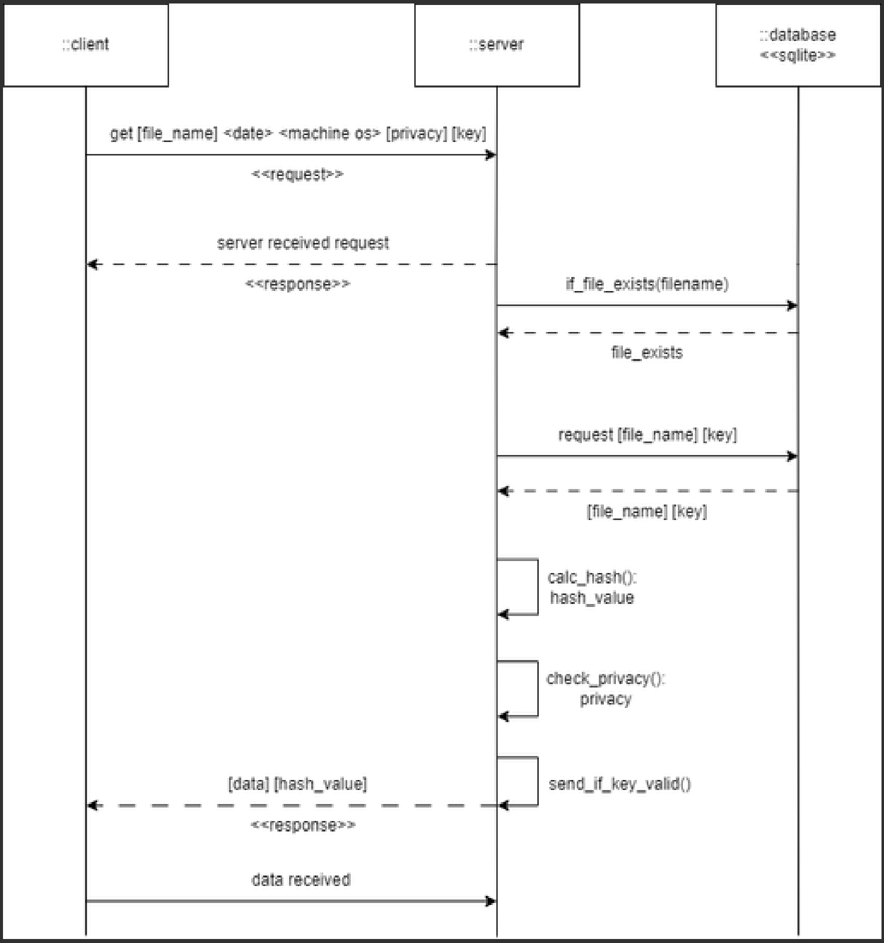
04

MESSAGE FORMATS/STRUCTURE

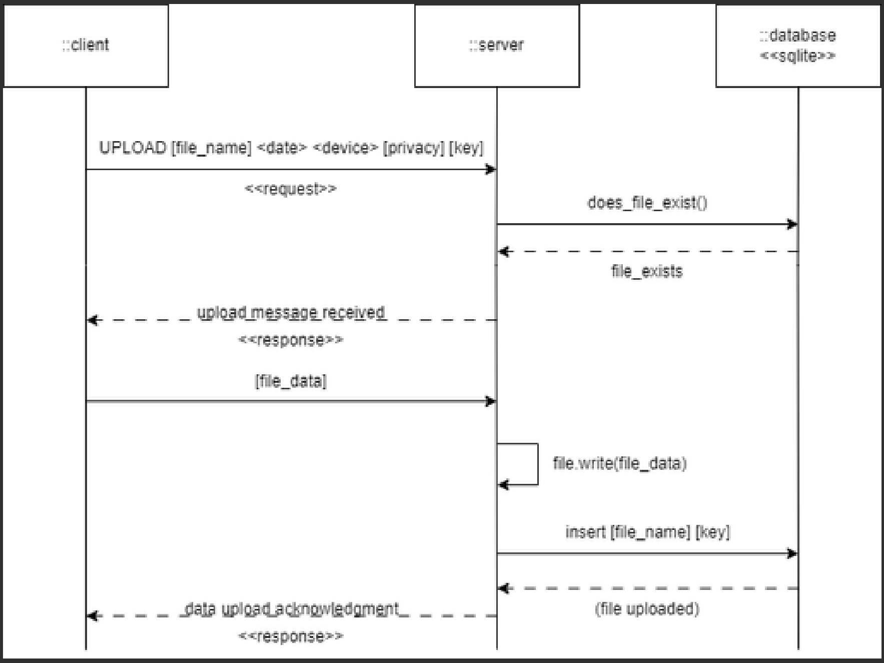


SEQUENCE DIAGRAMS

05



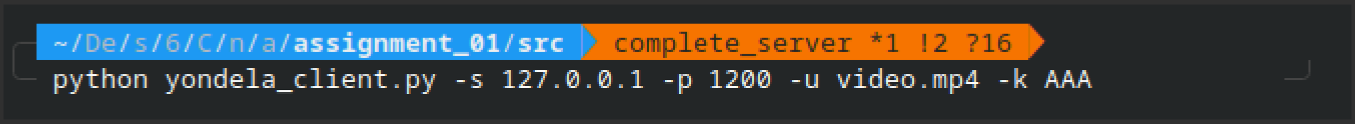
GET REQUEST



UPLOAD REQUEST

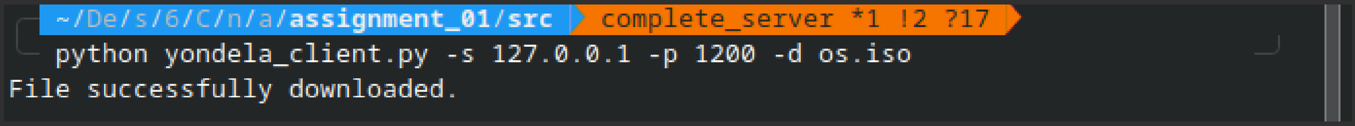
DEMO IMAGES

UPLOAD REQUEST



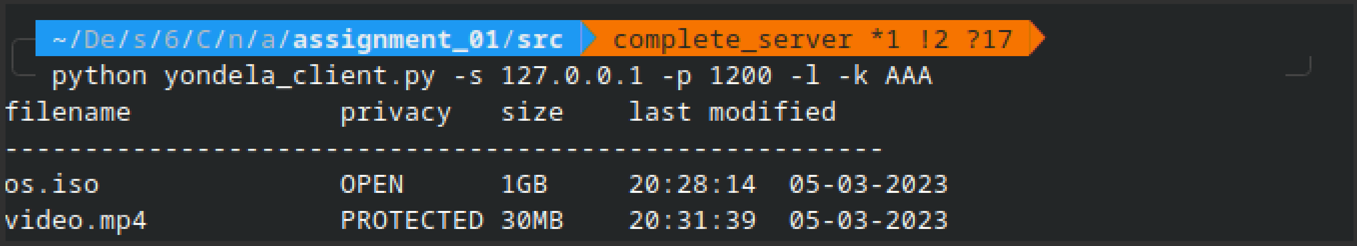
```
~/De/s/6/C/n/a/assignment_01/src  complete_server *1 !2 ?16
python yondela_client.py -s 127.0.0.1 -p 1200 -u video.mp4 -k AAA
```

DOWNLOAD REQUEST

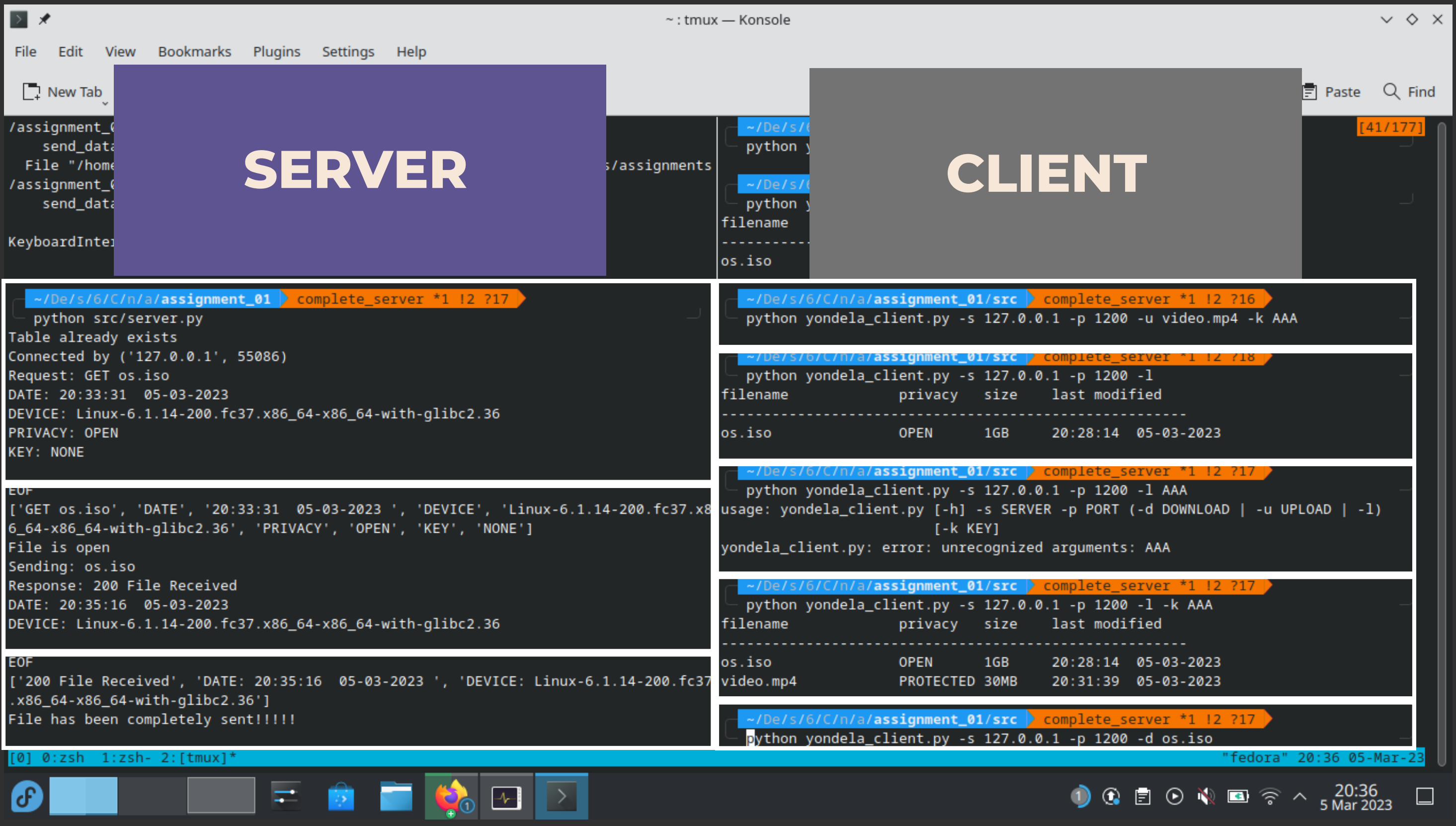


```
~/De/s/6/C/n/a/assignment_01/src  complete_server *1 !2 ?17
python yondela_client.py -s 127.0.0.1 -p 1200 -d os.iso
File successfully downloaded.
```

LIST REQUEST



```
~/De/s/6/C/n/a/assignment_01/src  complete_server *1 !2 ?17
python yondela_client.py -s 127.0.0.1 -p 1200 -l -k AAA
filename                privacy    size     last modified
-------------------------------------------------------
os.iso                  OPEN       1GB      20:28:14  05-03-2023
video.mp4               PROTECTED  30MB     20:31:39  05-03-2023
```



```
/assignment_(
    send_data
  File "/home                                           s/assignments
/assignment_(
    send_data

KeyboardInte
```

```
~/De/s/6/C/n/a/assignment_01  complete_server *1 !2 ?17
python src/server.py
Table already exists
Connected by ('127.0.0.1', 55086)
Request: GET os.iso
DATE: 20:33:31  05-03-2023
DEVICE: Linux-6.1.14-200.fc37.x86_64-x86_64-with-glibc2.36
PRIVACY: OPEN
KEY: NONE
```

```
EOF
['GET os.iso', 'DATE', '20:33:31  05-03-2023 ', 'DEVICE', 'Linux-6.1.14-200.fc37.x8
6_64-x86_64-with-glibc2.36', 'PRIVACY', 'OPEN', 'KEY', 'NONE']
File is open
Sending: os.iso
Response: 200 File Received
DATE: 20:35:16  05-03-2023
DEVICE: Linux-6.1.14-200.fc37.x86_64-x86_64-with-glibc2.36
```

```
EOF
['200 File Received', 'DATE: 20:35:16  05-03-2023 ', 'DEVICE: Linux-6.1.14-200.fc37
.x86_64-x86_64-with-glibc2.36']
File has been completely sent!!!!!
```

```
~/De/s/(
python y

~/De/s/(
python y
filename
----------
os.iso
                                                          [41/177]
```

```
~/De/s/6/C/n/a/assignment_01/src  complete_server *1 !2 ?16
python yondela_client.py -s 127.0.0.1 -p 1200 -u video.mp4 -k AAA
```

```
~/De/s/6/C/n/a/assignment_01/src  complete_server *1 !2 ?18
python yondela_client.py -s 127.0.0.1 -p 1200 -l
filename                privacy    size     last modified
-------------------------------------------------------
os.iso                  OPEN       1GB      20:28:14  05-03-2023
```

```
~/De/s/6/C/n/a/assignment_01/src  complete_server *1 !2 ?17
python yondela_client.py -s 127.0.0.1 -p 1200 -l AAA
usage: yondela_client.py [-h] -s SERVER -p PORT (-d DOWNLOAD | -u UPLOAD | -l)
                         [-k KEY]
yondela_client.py: error: unrecognized arguments: AAA
```

```
~/De/s/6/C/n/a/assignment_01/src  complete_server *1 !2 ?17
python yondela_client.py -s 127.0.0.1 -p 1200 -l -k AAA
filename                privacy    size     last modified
-------------------------------------------------------
os.iso                  OPEN       1GB      20:28:14  05-03-2023
video.mp4               PROTECTED  30MB     20:31:39  05-03-2023
```

```
~/De/s/6/C/n/a/assignment_01/src  complete_server *1 !2 ?17
python yondela_client.py -s 127.0.0.1 -p 1200 -d os.iso
```

```
[0] 0:zsh  1:zsh- 2:[tmux]*                                  "fedora" 20:36 05-Mar-23
```

20:36
5 Mar 2023

POTENTIAL IMPROVEMENTS AND EXTENSIONS

- **ADD TIMEOUT FUNCTION TO PREVENT SERVER FROM 'HANGING'**
- **USER DOWNLOAD REQUEST HISTORY**
- **DELETING FILES ON THE SERVER THAT CLIENTS HAVE UPLOAD**
- **UPDATING OF KEYS FOR PROTECTED FILES**
- **ADDING A PROGRESS BAR FOR UPLOADING/DOWNLOADING**