Multimedia Sharing Application

Group 5

20535003 - Aman Juyal

20535018 - Nikhil Tirkey

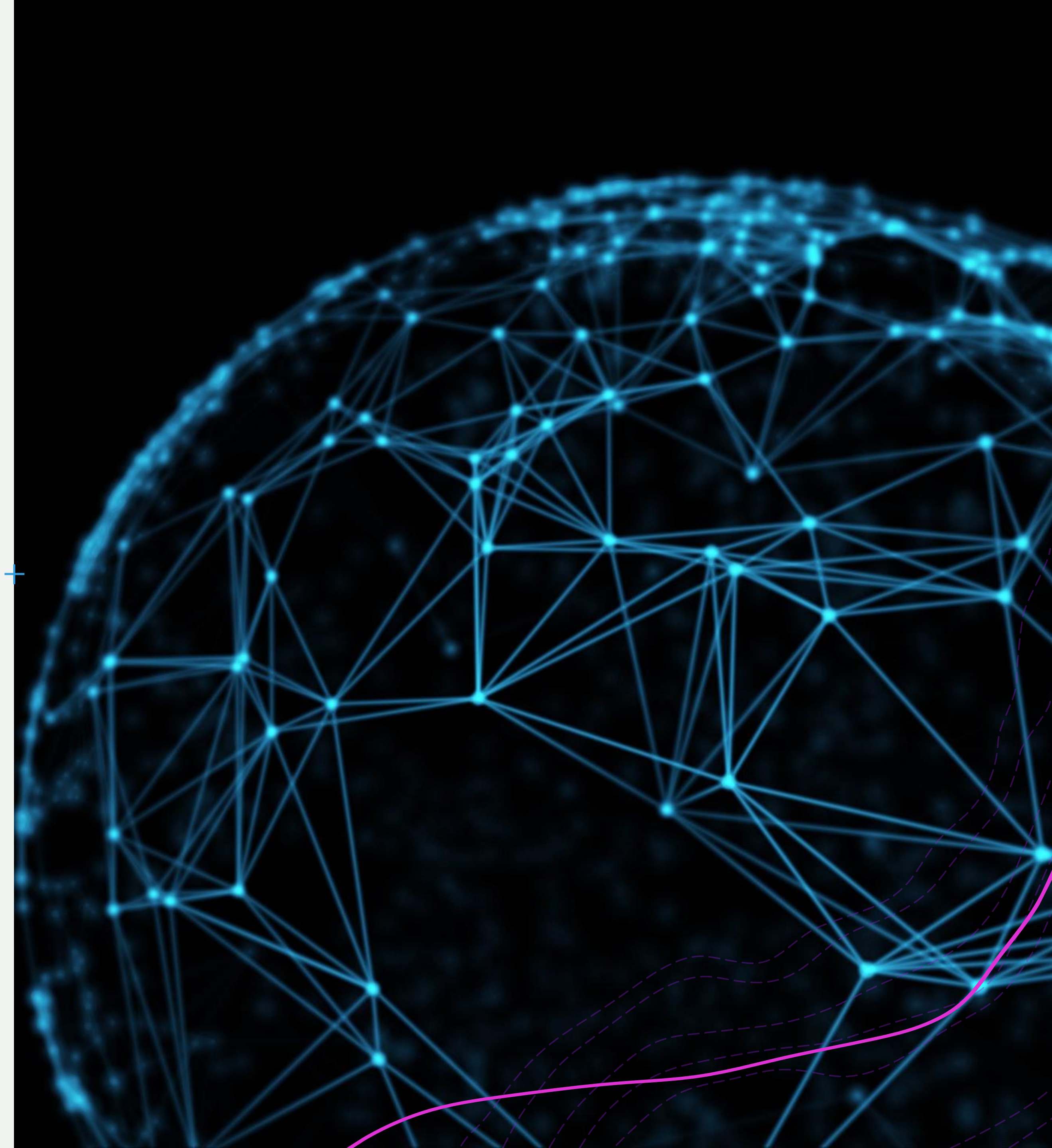
20535019 - Pamanand Kumar

20535028 - Suman Narayan

20535032 - Vatsal Tiwari

20535033 - Vikash Banjare

20535034 - Vivek Suryavanshi



Project Description

Develop an application which can share large multimedia files between two nodes o the same network using socket programming. Further optimize the application using multithreading to run faster for larger files. Show performance gain in multithreading over a single threaded program.

Socket programming

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

Sender Connection Establishment

+ First of all we import socket which is necessary.
+ Then we made a socket object and reserved a port on our pc.
+ After that we binded our server to the specified port. Passing an empty string means that the server can listen to incoming connections from other computers as well.
+ After that we put the server into listen mode.
+ At last we make a while loop and start to accept all incoming connections and close those connections after a thank you message to all connected sockets.

Receiver Connection Establishment

- First of all we make a socket object.
- Then we connect to sender on the port on which our server runs and lastly we receive data from the server and close the connection.

Single Thread File Transfer: Sender

+ A socket is created and the IP and port are bound to it.
+ The sender then enters to listening mode and waits for the receiver to establish connection.
+ Once the connection is established the sender sends the file name and file size to the receiver and then starts transmitting the data.

Single Thread File Transfer: Receiver

- . A socket is created and is connected to the IP and port of the host.
- The receiver then establishes the connection using the sockets.
- Once the connection is established the receiver receives the file name and file size from the sender.

Multithread File Transfer-Sender

- Once the connection is established the sender sends the file name and file size, along with the file type to the receiver and then starts transmitting the data.
- The file is divided into chunks of fixed size and then the program uses multiple threads to send these chunks of the file to the receiver .
- As soon as a thread receives 'READY' from receiver, it sends the chunk assigned to it along with its information.

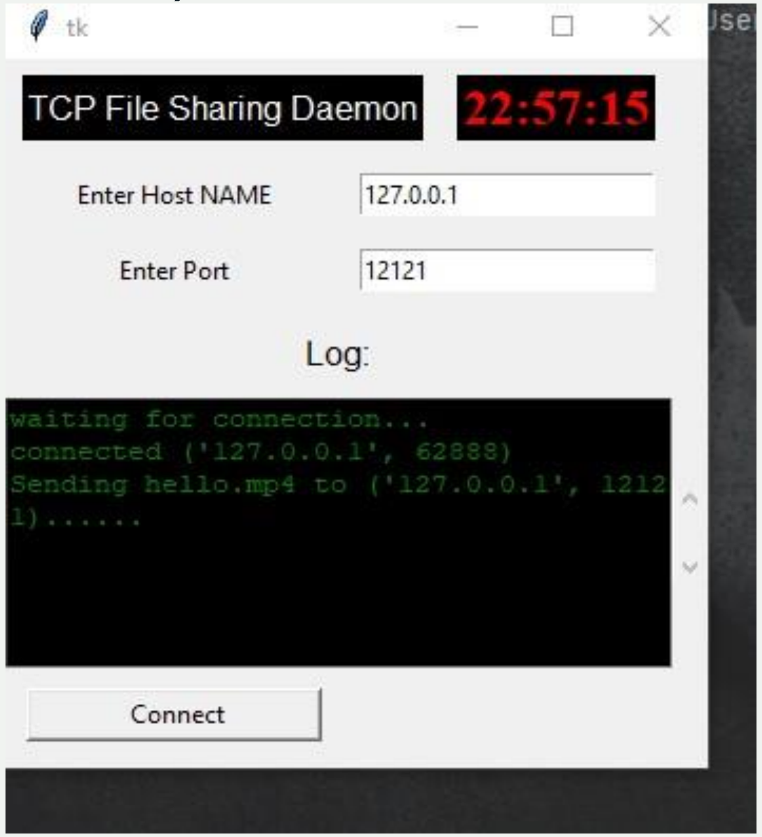
Multithread File Transfer-Recipient

- The sender sends a "READY" message to notify the sender that it is ready to receive bytes.
- It receives the chunk info first, and then goes on receive and write the data into file accordingly.
- Chunk info consists of <chunk id, chunk size> separated by a delimiter.

GUI Implementation

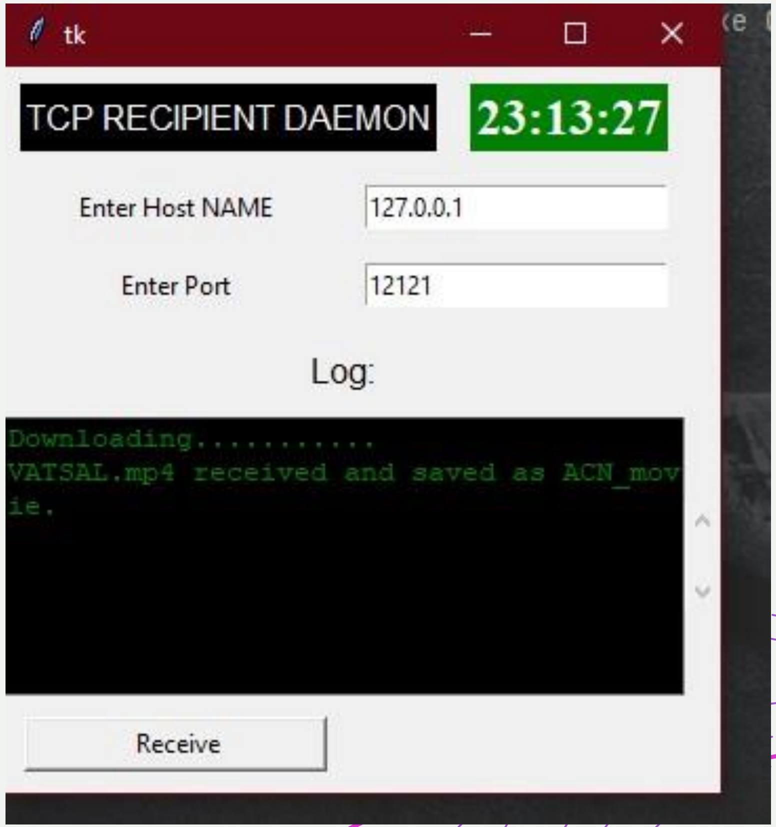
- Tkinter is the Python interface used to create GUI for both the sender and receiver.
- The GUI displays information like connected and sending file at senders window and downloading file and finished status at receivers window.

- Sender Window(sending status)



```
waiting for connection...
connected ('127.0.0.1', 62888)
Sending hello.mp4 to ('127.0.0.1', 1212
1)......
```

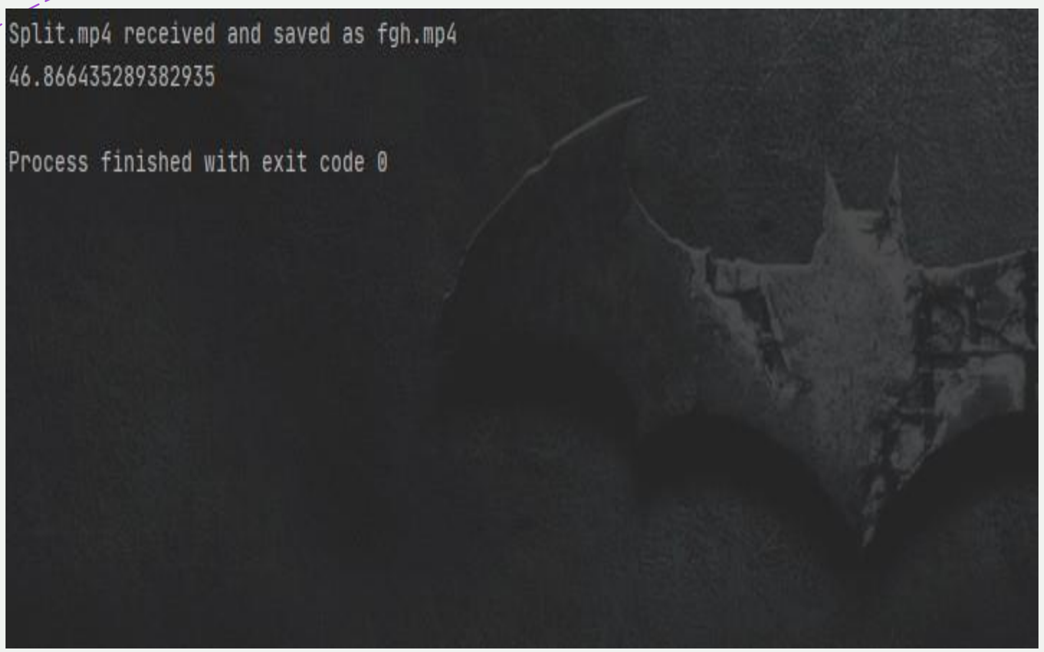
- Receiver Window(finished status)



```
Downloading...........
VATSAL.mp4 received and saved as ACN_mov
ie.
```

Result & Analysis

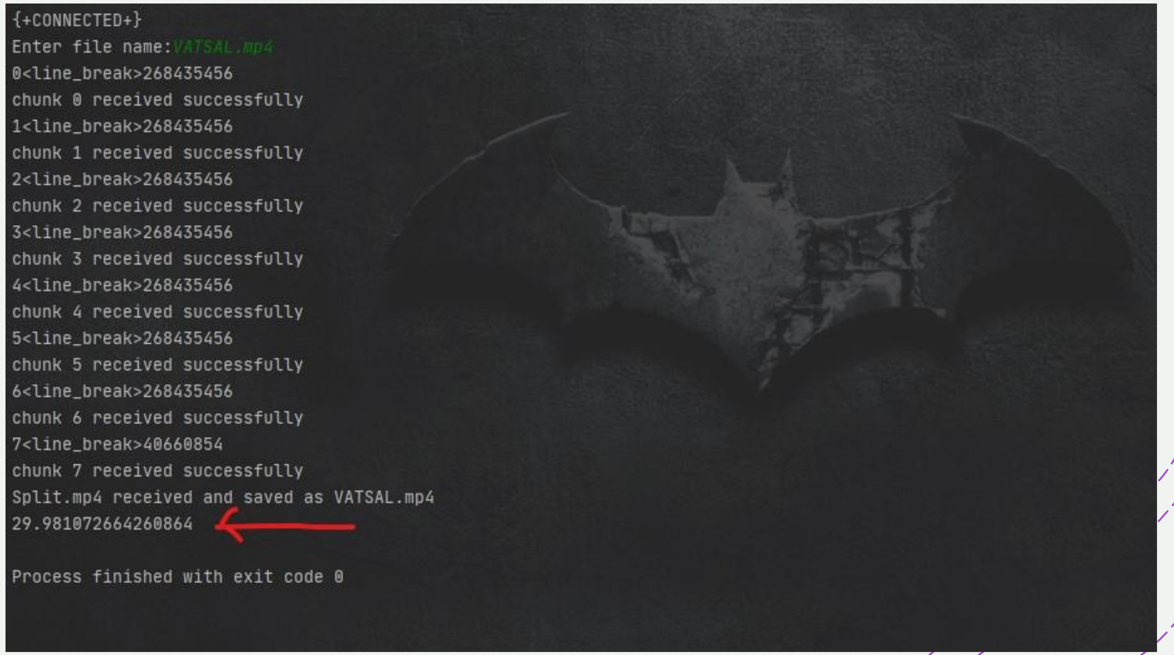
+ Single thread(time elapsed)



```
Split.mp4 received and saved as fgh.mp4
46.866435289382935

Process finished with exit code 0
```

+ Multi thread(time elapsed)



```
{+CONNECTED+}
Enter file name:VATSAL.mp4
0<line_break>268435456
chunk 0 received successfully
1<line_break>268435456
chunk 1 received successfully
2<line_break>268435456
chunk 2 received successfully
3<line_break>268435456
chunk 3 received successfully
4<line_break>268435456
chunk 4 received successfully
5<line_break>268435456
chunk 5 received successfully
6<line_break>268435456
chunk 6 received successfully
7<line_break>40660854
chunk 7 received successfully
Split.mp4 received and saved as VATSAL.mp4
29.981072664260864

Process finished with exit code 0
```