PROJECT REPORT

Multimedia File Sharing Application

(CSN 503 – Advanced Computer Networks)

Group 5

20535003 – Aman Juyal

20535018 – Nikhil Tirkey

20535019 – Pamanand Kumar

20535028 – Suman Narayan

20535032 - Vatsal Tiwari

20535033 – Vikash Banjare

20535034 – Vivek Suryavanshi

Member Contribution

20535003 – Aman Juyal - Receiver Connection Establishment (5)
20535018 – Nikhil Tirkey – GUI(6), Single thread file transfer (5)
20535019 – Pamanand Kumar – Sender Connection Establishment(5)
20535028 – Suman Narayan - Single thread file transfer(5)
20535032 - Vatsal Tiwari - Multithreading file transfer(6), GUI(6)
20535033 – Vikash Banjare - Single thread file transfer (5)

Contents

Project Description

Multimedia Sharing Application

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.

Working & Methodology

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 bind 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

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. 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.

Multi Thread File Transfer

Sender:
The application gets its IP from the system and assigns itself a port on which the transfer will take place. 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.
For larger files, 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.

Receiver:
The application inputs the host IP and a port on which the transfer will take place. A socket is created and the IP and port are bound to it.
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.
The receiver then start receiving the file.

GUI

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.

Source Code

Recipient:

```
import math
import os
import socket
import time
import tkinter as tk
from tkinter import messagebox, filedialog

import numpy as np

LARGE_FONT= ("Verdana", 12)

GAP = "<line_break>"
MB = int(math.pow(2, 20))
chunk = int(math.pow(2, 30))
GB = int(math.pow(2, 30))

class Page(tk.Tk):

    def __init__(self, *args, **kwargs):

        tk.Tk.__init__(self, *args, **kwargs)
        container = tk.Frame(self)

        container.pack(side="top", fill="both", expand = True)

        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}

        for F in (StartPage, PageOne):

            frame = F(container, self)

            self.frames[F] = frame

            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame(StartPage)

    def show_frame(self, cont):

        frame = self.frames[cont]
        frame.tkraise()


def wchunk(client, cno, csize, savename):
    pkt = bytearray()
    while len(pkt) < csize:
        recvfile = client.recv(csize - len(pkt))
        if not recvfile:
            raise Exception(f"chunk {cno} not fully received.")
        pkt += recvfile
    np.array(bytes(pkt)).tofile(open(savename, 'ab+'))
    return
```

```
class StartPage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)

        l_title=tk.Label(self, text="TCP File Recipient Daemon",
                         font="Arial,12")
        l_title.grid(row=0,column=0,columnspan=3,
sticky="NSEW",padx=30,pady=30)

        label_username = tk.Label(self, text="Username")
        label_password = tk.Label(self, text="Password")

        entry_username = tk.Entry(self,show="*")

        entry_password = tk.Entry(self, show="*")

        label_username.grid(row=2, column=0, sticky='NSEW',padx=10,pady=10)
        label_password.grid(row=3, column=0, sticky='NSEW',padx=10,pady=10)
        entry_username.grid(row=2, column=1,sticky='NSEW',padx=10,pady=10)
        entry_password.grid(row=3, column=1,sticky='NSEW',padx=10,pady=10)

        checkbox = tk.Checkbutton(self, text="Keep me logged in")
        checkbox.grid(row=4, column=1,sticky='NSEW',padx=10,pady=10)

        logbtn = tk.Button(self, text="Login", bg="GREEN",
fg="White",command=lambda: login_btn_clicked())
        logbtn.grid(row=5, column=1,sticky='NSEW', padx=10, pady=10)

        def login_btn_clicked():
            # print("Clicked")
            username = entry_username.get()
            password = entry_password.get()

            if len(username) and len(password) > 2:

                if username == "admin" and password == "admin":
                    controller.show_frame(PageOne)
                # display a message if username and password is incorrect!
                else:
                    messagebox.showinfo(self,"Invalid username or password
! ")

            else:
                messagebox.showinfo(self,"Enter Username and Password")


class PageOne(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        clock = tk.Label(self, font=('times', 18, 'bold'),
bg='green',fg="white")
        clock.grid(row=0,column=2, sticky="NSNESWSE",padx=8,pady=8)



        def tick():
```

```
            time2=time.strftime('%H:%M:%S')
            clock.config(text=time2)
            clock.after(200,tick)
        tick()

        label = tk.Label(self, text="TCP RECIPIENT DAEMON",
font="Arial,16",bg="black",fg="White")
        label.grid(row=0, column=0, columnspan=2, padx=8, pady=8,
sticky="NSNESWSE")

        l_host=tk.Label(self,text="Enter Host NAME")
        l_host.grid(row=1, column=0, padx=8, pady=8, sticky="NSNESWSE")


        e_host=tk.Entry(self)
        e_host.grid(row=1, column=1, columnspan=2, padx=8, pady=8,
sticky="NSNESWSE")
        e_host.insert(tk.END,'127.0.0.1')


        l_port=tk.Label(self,text="Enter Port")
        l_port.grid(row=2, column=0, padx=8, pady=8, sticky="NSNESWSE")

        e_port=tk.Entry(self)
        e_port.grid(row=2, column=1, columnspan=2, padx=8, pady=8,
sticky="NSNESWSE")
        e_port.insert(tk.END,12121)

        message_label =tk.Label(self,text="Log:",font=("Arial,12"))

message_label.grid(row=3,column=0,columnspan=3,padx=10,pady=10,sticky="NSEW
")


        scrollbar_y = tk.Scrollbar(self)
        scrollbar_y.grid(row=4, column=3,rowspan=6)

        show_1=tk.Text(self,height=8, width=35,
yscrollcommand=scrollbar_y.set,
                       bg="BLACK",fg="GREEN")
        show_1.grid(row=4, column=0,rowspan=3,columnspan=3,sticky="NSEW")

        b_connect=tk.Button(self,text=" Receive",command=lambda:
my_server())
        b_connect.grid(row=14,column=0,padx=10,pady=10,sticky="nsew")
        #
        # e_data=tk.Entry(self)
        # e_data.grid(row=14,column=1,padx=10,pady=10,sticky="nsew")

        def my_server():
            # e_data_v = e_data.get()
            host = e_host.get()
            port = int(e_port.get())

            HOST, PORT = host, port
            # data  = e_data_v

                # Create a socket (SOCK_STREAM means a TCP socket)
            with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
                # Connect to server and send data
                s.connect((HOST, PORT))
```

```
                # s.sendall(bytes(data + "\n", "utf-8"))
                GAP = "<line_break>"

                data = s.recv(4096).decode()
                filename, file_size, ftype = data.split(GAP)
                filename = os.path.basename(filename)
                file_size = int(file_size)
                num = (file_size//chunk)+1
                savename = filedialog.asksaveasfilename(initialdir='/',
title=f"Save {filename} as",
                                                        filetypes=(('all
files', '*.*'), ('jpeg files','*.jpg'), ('mp4 files','*.mp4'),
                                                                   ))
                curr = 0
                if file_size > GB:
                    show_1.insert(tk.END, f'Downloading...........\n')
                    while curr < num:
                        s.send('READY'.encode())
                        cid, chunk_size = s.recv(1024).decode().split(GAP)
                        chunk_size = int(chunk_size)
                        n = int(cid)
                        wchunk(s, n, chunk_size, savename)
                        curr += 1

                else:
                    show_1.insert(tk.END, f'Downloading...........\n')
                    with open(savename, 'a+b') as f:
                        tot = 0
                        while tot < file_size:
                            recvfile = s.recv(MB)
                            if not recvfile:
                                break
                            f.write(recvfile)
                            tot += len(recvfile)
            show_1.insert(tk.END,f'{filename} received and saved as
{os.path.basename(savename)}.')
            s.close()









app = Page()
app.mainloop()
```

SENDER:

```
import tkinter as tk
import tqdm
from tkinter import messagebox, filedialog, ttk
import time
import datetime
from socket import *
import time
from time import ctime
import _thread
import numpy as np
import os
import threading
import filechunkio
import math


gb = (math.pow(2, 30))
GB = int(gb)
mb = int(math.pow(2, 20))
MB = int(mb)
chunk = GB
GAP = "<line_break>"

LARGE_FONT= ("Verdana", 12)

def my_server(show_1,HOST,PORT):


    BUFSIZE = 1024
    ADDR = (HOST, PORT)

    tcpTimeSrvrSock = socket(AF_INET,SOCK_STREAM)
    tcpTimeSrvrSock.bind(ADDR)
    tcpTimeSrvrSock.listen(5)
    currentDT = datetime.datetime.now()


    while True:
        show_1.insert(tk.END,"waiting for connection...")
        show_1.insert(tk.END,"\n")
        #print ('waiting for connection...')

        tcpTimeClientSock, addr = tcpTimeSrvrSock.accept()

        #print ('...connected from:', addr)
        show_1.insert(tk.END,"connected {}".format(addr))
        show_1.insert(tk.END,"\n")
        filename =
filedialog.askopenfilename(initialdir='C:/Users/Vatsal/PycharmProjects/unti
tled',
                                                   title="select a file",
                                                   filetypes=(('jpg files',
'*.jpg'),("mp4 files", "*.mp4"), ("all files", "*.*")))
        # filename = "Split.mp4"
        filesize = os.path.getsize(filename)
        filesize = int(filesize)

        with open(filename, 'rb') as test:
```

```
            buf = bytearray(test.read(1024))
            arr = np.frombuffer(buf)
        filetype = arr.dtype
        # with server:
        show_1.insert(tk.END,f'Sending {os.path.basename(filename)} to
{ADDR}......\n')
        st = time.time()
        sthread = threading.Thread(target=client_handler,
args=(tcpTimeClientSock, filename, filesize, filetype,))
        sthread.setDaemon(True)
        sthread.start()
        main_thread = threading.current_thread()
        for t in threading.enumerate():
            if t is main_thread:
                continue
            t.join()
        show_1.insert(tk.END,f'Done sending in {int(time.time() - st)}
seconds')
        tcpTimeClientSock.send('Thank you for connecting'.encode())
        tcpTimeClientSock.close()


    #     '''while True:
    #         data = tcpTimeClientSock.recv(BUFSIZE)
    #         if not data:
    #             break
    #         tcpTimeClientSock.send(bytes(currentDT.strftime("%I:%M:%S
%p"),'utf-8'))
    #         show_1.insert(tk.END,data.decode('utf-8'))
    #         show_1.insert(tk.END,"\n")
    #         print(data.decode('utf-8'))
    #     tcpTimeClientSock.close()
    # tcpTimeSrvrSock.close()'''

class Page(tk.Tk):

    def __init__(self, *args, **kwargs):

        tk.Tk.__init__(self, *args, **kwargs)
        container = tk.Frame(self)

        container.pack(side="top", fill="both", expand = True)

        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}

        for F in (StartPage, PageOne):

            frame = F(container, self)

            self.frames[F] = frame

            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame(StartPage)

    def show_frame(self, cont):

        frame = self.frames[cont]
```

```
        frame.tkraise()


class StartPage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)

        l_title=tk.Label(self, text="TCP File Sharing Daemon",
                         font="Tahoma,12")
        l_title.grid(row=0,column=0,columnspan=3,
sticky="NSEW",padx=30,pady=30)

        label_username = tk.Label(self, text="Username")
        label_password = tk.Label(self, text="Password")

        entry_username = tk.Entry(self,show="*")

        entry_password = tk.Entry(self, show="*")

        label_username.grid(row=2, column=0, sticky='NSEW',padx=10,pady=10)
        label_password.grid(row=3, column=0, sticky='NSEW',padx=10,pady=10)
        entry_username.grid(row=2, column=1,sticky='NSEW',padx=10,pady=10)
        entry_password.grid(row=3, column=1,sticky='NSEW',padx=10,pady=10)

        checkbox = tk.Checkbutton(self, text="Keep me logged in")
        checkbox.grid(row=4, column=1,sticky='NSEW',padx=10,pady=10)

        logbtn = tk.Button(self, text="Login", bg="GREEN",
fg="White",command=lambda: login_btn_clicked())
        logbtn.grid(row=5, column=1,sticky='NSEW', padx=10, pady=10)

        def login_btn_clicked():
            # print("Clicked")
            username = entry_username.get()
            password = entry_password.get()

            if len(username) and len(password) > 2:
                # print(username, password)

                if username == "admin" and password == "admin":
                    controller.show_frame(PageOne)
                # display a ,essage if username and password is incorrect!
                else:
                    messagebox.showinfo(self,"Invalid username or password
! ")

            else:
                messagebox.showinfo(self,"Enter Username and Password")


after_id = None


class PageOne(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        flag = True

        clock = tk.Label(self, font=('times', 18, 'bold'),
```

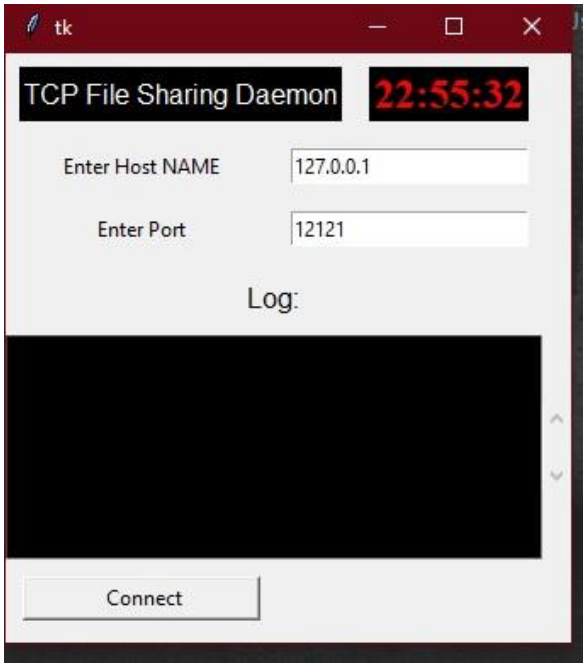
```
bg='black',fg="red")
        clock.grid(row=0,column=2, sticky="NSNESWSE",padx=8,pady=8)

        def tick():
            time2=time.strftime('%H:%M:%S')
            clock.config(text=time2)
            clock.after(200,tick)
        tick()

        label = tk.Label(self, text="TCP File Sharing Daemon",
font="Arial,16",bg="black",fg="White")
        label.grid(row=0, column=0, columnspan=2, padx=8, pady=8,
sticky="NSNESWSE")

        l_host=tk.Label(self,text="Enter Host NAME")
        l_host.grid(row=1, column=0, padx=8, pady=8, sticky="NSNESWSE")

        e_host=tk.Entry(self)
        e_host.grid(row=1, column=1, columnspan=2, padx=8, pady=8,
sticky="NSNESWSE")
        e_host.insert(tk.END,'127.0.0.1')


        l_port=tk.Label(self,text="Enter Port")
        l_port.grid(row=2, column=0, padx=8, pady=8, sticky="NSNESWSE")

        e_port=tk.Entry(self)
        e_port.grid(row=2, column=1, columnspan=2, padx=8, pady=8,
sticky="NSNESWSE")
        e_port.insert(tk.END,12121)

        message_label=tk.Label(self,text="Log:",font=("Arial,12"))

message_label.grid(row=3,column=0,columnspan=3,padx=10,pady=10,sticky="NSEW
")


        scrollbar_y = tk.Scrollbar(self)
        scrollbar_y.grid(row=4, column=3,rowspan=6)

        show_1=tk.Text(self,height=8, width=35,
yscrollcommand=scrollbar_y.set,
                       bg="Black",fg="Green")
        show_1.grid(row=4, column=0,rowspan=3,columnspan=3,sticky="NSEW")



        b_connect=tk.Button(self,text=" Connect",command=lambda: connect())
        b_connect.grid(row=14,column=0,padx=10,pady=10,sticky="nsew")

        # b_disconnect=tk.Button(self,text=" disconnect",command=lambda:
disconnec())
        # b_disconnect.grid(row=14,column=1,padx=10,pady=10,sticky="nsew")


        def runner():
            global after_id
            global secs
            secs += 1
            if secs % 2 == 0:  # every other second
                e_host_v=e_host.get()
```

```
                e_port_v=int(e_port.get())

            after_id = self.after(1000, runner)  # check again in 1 second

        def connect():
            # CONNECT COM PORT
            e_host_v=e_host.get()
            e_port_v=int(e_port.get())
            _thread.start_new_thread(my_server,(show_1,e_host_v,e_port_v))
            #start_new_thread(my_server,(show_1,e_host_v,e_port_v))
            global secs
            secs = 0
            #runner()  # start repeated checking


        # def disconnec():
        #     global after_id
        #     if after_id:
        #         self.after_cancel(after_id)
        #         after_id = None

def client_handler(soc, file_name, file_size, file_type):

    with soc, open(file_name, 'rb') as f:
        soc.send(f"{file_name}{GAP}{file_size}{GAP}{file_type}".encode())

        if file_size <= GB:
           while True:
                fsend = f.read(MB)
                if not fsend:
                    break
                soc.sendall(fsend)


        else:

            num = 0
            threads = []

            while True:

                offset = chunk * num
                if offset >= file_size:
                    break
                buff = min(chunk, file_size - offset)
                buff = int(buff)
                size = offset + buff

                fp = filechunkio.FileChunkIO(file_name, 'rb',
offset=int(offset), bytes=buff)
                chunk_id = num
                num += 1
                t = threading.Thread(target=sendfile, args=(soc, fp, buff,
chunk_id,))
                threads.append(t)
                t.setDaemon(True)
                t.start()


            mthread = threading.current_thread()
            for thread in threads:
```

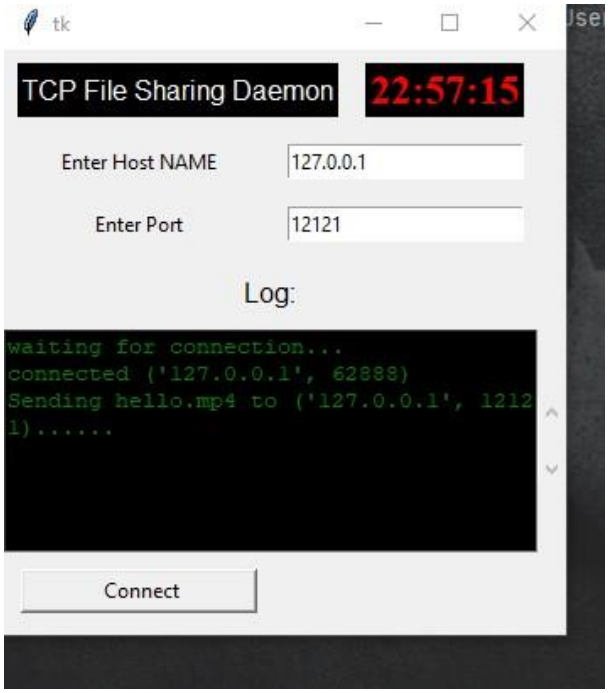
```
                if thread is mthread:
                    continue
                thread.join()

def sendfile(c, fp, size, cid):
    with fp:
        while True:
            response = c.recv(1024).decode()
            if response == 'READY':
                break

        try:
            c.send(f"{cid}{GAP}{size}".encode())
            while True:
                fsend = fp.readall()
                if not fsend:
                    break
                c.sendall(fsend)


                if len(fsend) < MB:
                    break
        except Exception as e:
            print(f'Exception!!     {e}')
            return

    return




app = Page()
app.mainloop()
```

Result & Analysis

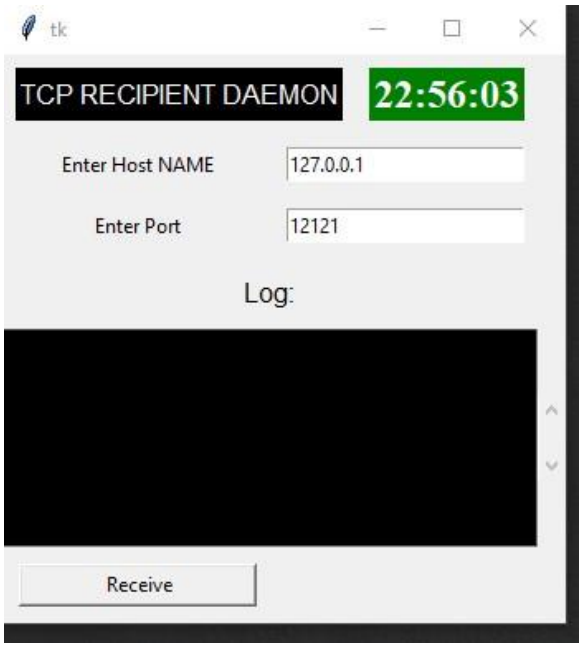
Sender GUI:



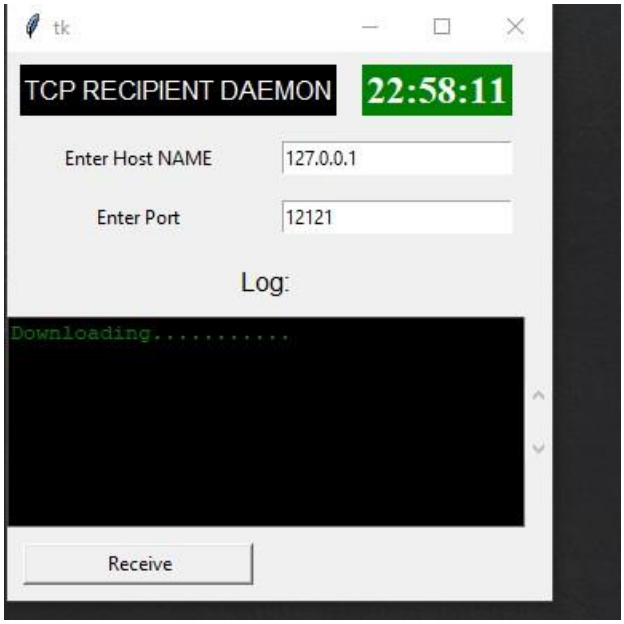
Sender sending:



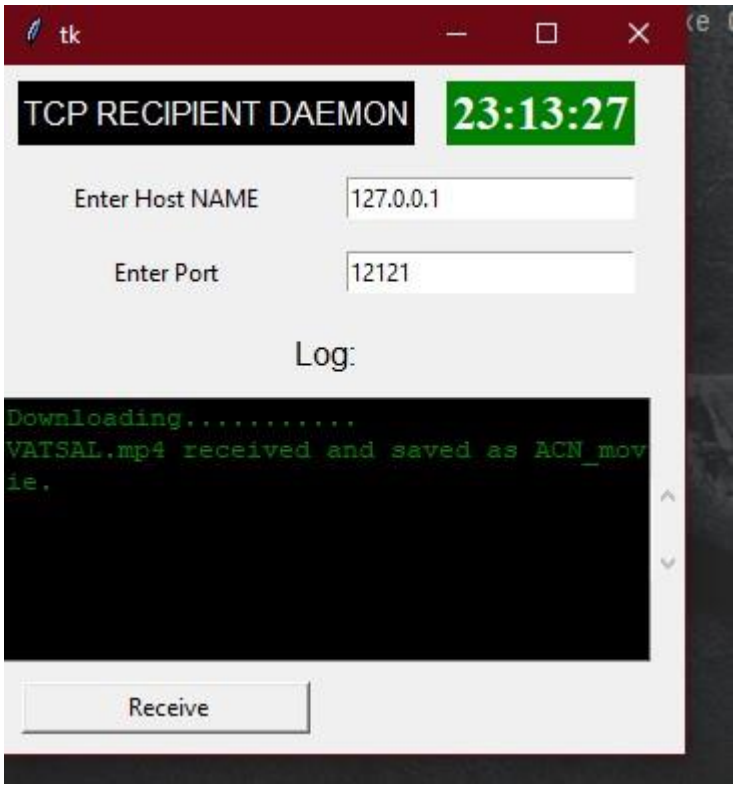
Receiver GUI:



Receiver Downloading:

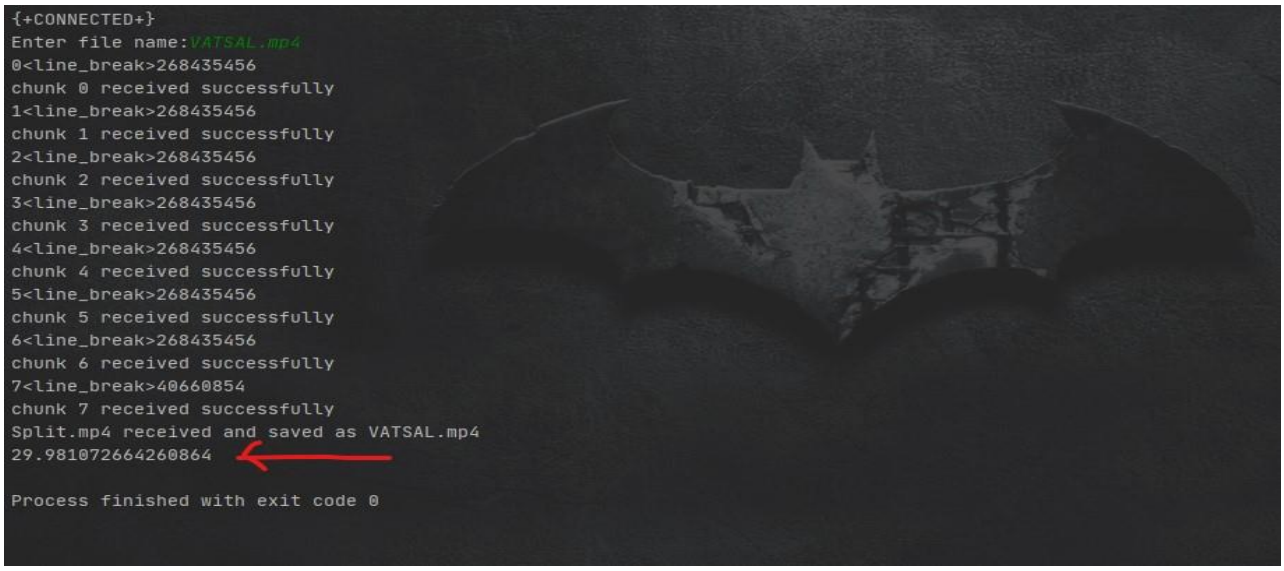


Receiver Finished:



Time using single thread:

Time using multiple thread:



REFERENCES

1. https://docs.python.org/3/library/tkinter.html
2. StackOverflow
3. https://pypi.org/project/filechunkio/
4. GitHub
5. YouTube