Anjali Singh (2023JCS2565) & Sharath Kumar Reddy Gandham(2023JCS2542) & Prince Gupta (2023JCS2561)

1. Give the Client Code.

Solution:

```
import socket
import time
import math
import hashlib
import numpy as np
import matplotlib.pyplot as plt
vayu_server=("vayu.iitd.ac.in",9802)
#vayu_server=("10.194.32.174",9802)
my_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
my_ip=("10.194.62.203",12457)
my_socket.bind(my_ip)

def check(a,b):
    es=[]
    if a==b:
        return es
    for g in b:
        if g not in a:
            es.append(g)
    return es



def main():
    ma=11000
    d=0
    q=0
    window_list=[0]*ma
    while(d<ma):
        window_list[d]=q
        d+=1
        q+=1448
    data="SendSize\nReset\n\n"
    while True:
        my_socket.sendto(data.encode('utf-8'),vayu_server)
        try:
            received_data,ip= my_socket.recvfrom(2048)
            if received_data:
                break
        except Exception as e:
            print(str(e))
            continue

    off=received_data.decode('utf-8').split("\n")
    offset=off[0][6:]
    offset=(int)(offset)
    w_size=1
    res={}
```

```
max_size=math.ceil(offset/1448)
w=window_list[:max_size]
last_request=w[max_size-1]
count=0
c=0
x={}
y={}
start_time = time.time()
while(len(w)>0):
    #print(len(w))
    con=False
    gf=w[0:w_size]
    ind=0
    for k in w:
        c=0
        if(ind>=w_size):
            break
        max_bytes=1448
        if(k==last_request):
            max_bytes=offset-k
        if max_bytes==0:
            break
        data=f"Offset: {k}\nNumBytes: {max_bytes}\n\n"
        my_socket.sendto(data.encode('utf-8'),vayu_server)
        x[k]=(time.time() - start_time)/1000
        time.sleep(0.0073*(w_size-c)/w_size)
        ind+=1
    current_pkt=[]
    my_socket.setblocking(0)
    ra_data=""
    while(True):
        try:
            received_data,ip=my_socket.recvfrom(2048)
            ra_data+=received_data.decode('utf-8')
        except BlockingIOError:
            break

    i_data=ra_data.split('Offset: ')
    #print(len(i_data))
    for s in i_data[1:]:
        ss=s.split('\n')
        last_r_byte=(int)(ss[0])
        if(last_r_byte not in current_pkt):
            current_pkt.append(last_r_byte)
        if(last_r_byte not in res):
            res[last_r_byte]=s.split('\n\n')[1]
            y[last_r_byte]=(time.time() -start_time)/1000
        if ss[2]=="Squished":
            count+=1
            con=True
            time.sleep(0.001)

    l=check(current_pkt,gf)
    w=l+w[w_size:]
    c=len(l)
```

```
        if(c!=0 or con==True):
            if(w_size>=2):
                w_size=(int)(w_size*(w_size-c)/w_size)
                if w_size==0:
                    w_size=2
            else:
                w_size=2
        else:
            w_size+=1

    li=list(res.keys())
    li.sort()
    s_res={k:res[k] for k in li}
    stri=''
    for key in s_res:
        stri+=s_res[key]

    byte_data=stri.encode('utf-8')
    md5_hash=hashlib.md5(byte_data).hexdigest()
    print(md5_hash)

    print(count)
    data=f"Submit: 2023JCS2542@sap\nMD5: {md5_hash}\n\n"
    while(True):
        my_socket.sendto(data.encode('utf-8'),vayu_server)
        try:
            time.sleep(0.1)
            result,ip=my_socket.recvfrom(2048)
            print(result.decode('utf-8'))
            break
        except Exception as e:
            print(str(e))

    my_socket.close()

    l1=list(x.keys())
    l1.sort()
    l2=list(y.keys())
    l2.sort()
    x1 = {k:x[k] for k in l1}
    y1 = {k:y[k] for k in l2}
    send=[x1[k] for k in x1]
    rec=[y1[k] for k in y1]
    z=[rec[i]-send[i] for i in range(0,len(send))]

    plt.plot(z,l1,label="Round Trip Time",color='green',linestyle='-')
    plt.plot(send,l1,label="Sending Time",color='blue',linestyle='-')
    plt.plot(rec,l1,label="Recieving Time",color='red',linestyle='-')
    plt.xlabel("Time Taken(seconds)")
    plt.ylabel("Offset Value")

    plt.title("Offset vs Time Taken")
    plt.legend()

    plt.grid(True)
    plt.show()
```

```
if __name__=="__main__":
    main()
```

2. Explain the algorithm you have implemented. Initialization:

Solution:

The client sets up a UDP socket to communicate with a server at IP address "**10.17.6.5**" on port **9802**. It binds the socket to the IP address "**10.194.32.174**" on port 12457.
The client sends an initial message "**SendSize**" to the server and receives a response that helps us to determine the offset value (variable offset).
After this, the Algorithm that we follow is explained below:

Algorithm:

- Make a list of possible sending offset requests in a list.
- Send the requests of window_size from the list.
- After sending every offset request we wait a certain amount of time to avoid getting Squished from the server.
- After sending collect all the responses from the server.
- Find the offsets which are received and which are not received.
- Add the received offsets to a dictionary key as an offset and value as data corresponding to that offset.
- Which offsets we received from the server, we delete that offset from the list.
- Which offsets have not received we still place it in that list only.
- Based on how many requested offsets we received in complete window size we make changes to the window size for net iteration.
- The Algorithm runs until the request offset list gets empty. (It represents that we sent all the required offset requests and successfully received also).
- Now we retrieve the data from the dictionary and arrange in sorted order of key values which represent the offset.
- Find hash of the complete data.
- Submit to the server after finding hash.
- Close the connection.
- Plot the graphs for Round Trip time and request sending time, data receiving time with offset values.

Some Mechanisms used here:

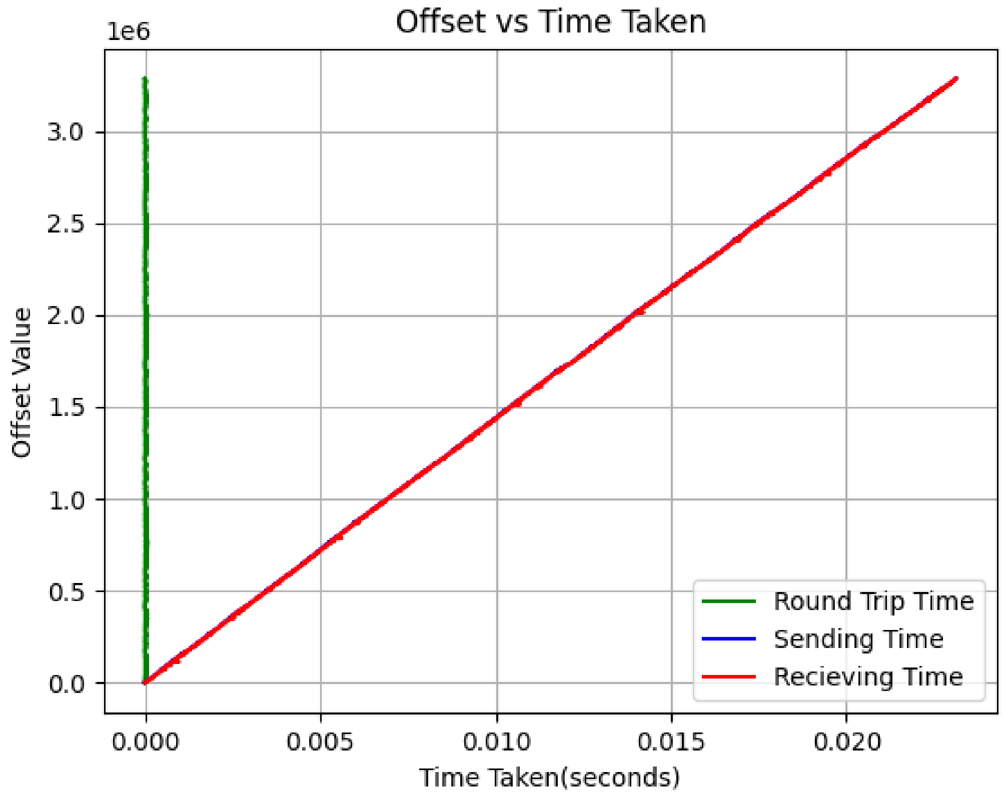
- **Error Handling:** The client checks for missing packets and retransmits them if necessary. It also handles exceptions that might occur during socket operations.

- **Congestion_Window:** The algorithm dynamically adjusts the window size based on successful transmissions and missing packets.
- **Delay Management:** Delays between packet transmissions are managed to optimize data flow.

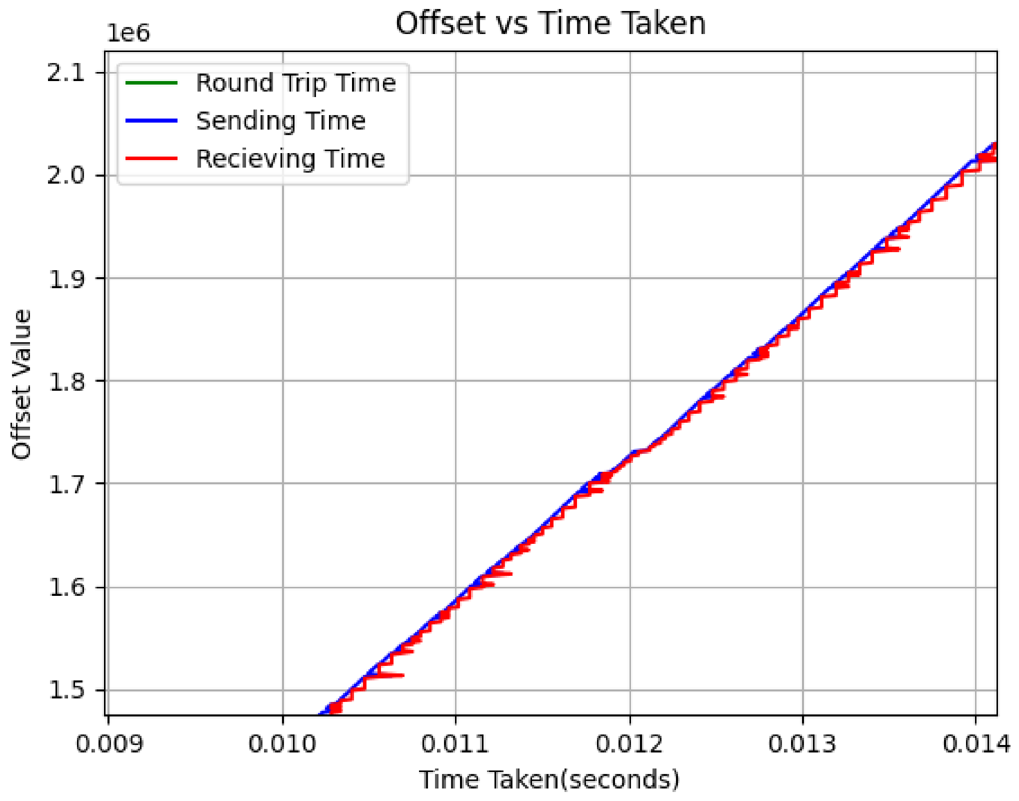
3. Show graphs of downloading the file with offset number vs the Request sent time, Data received time, and Round Trip Time.

Solution:

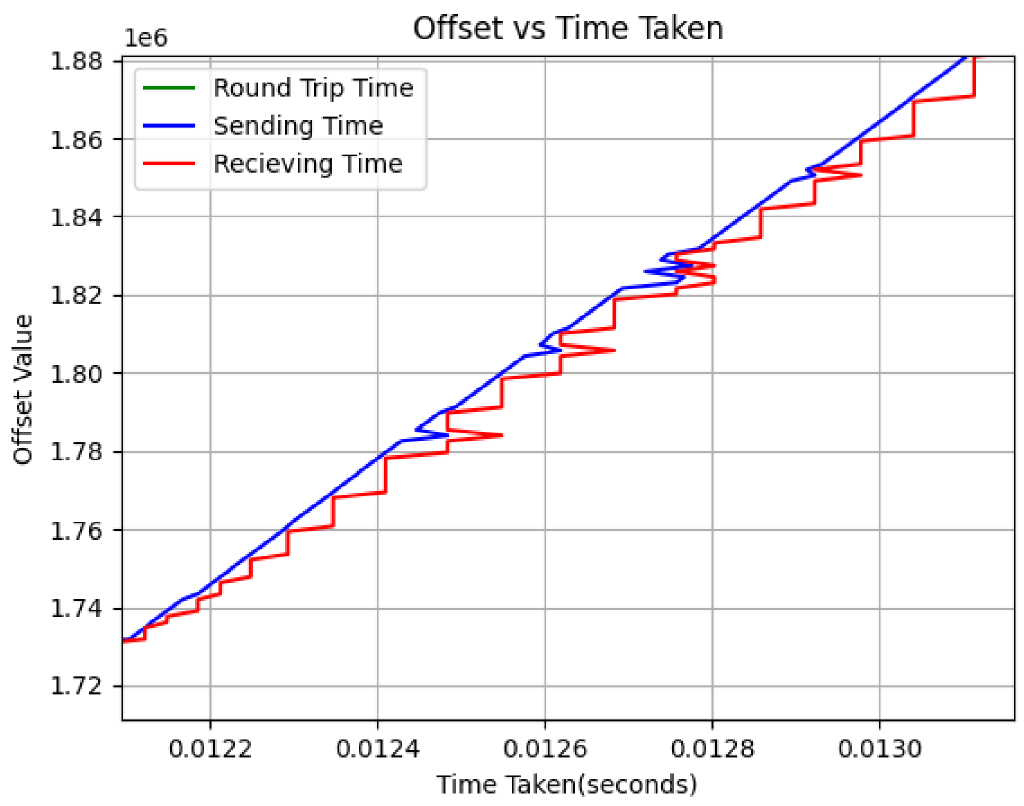
In the graphs the horizontal axis represents the Time in seconds and the vertical axis represents the Offset.



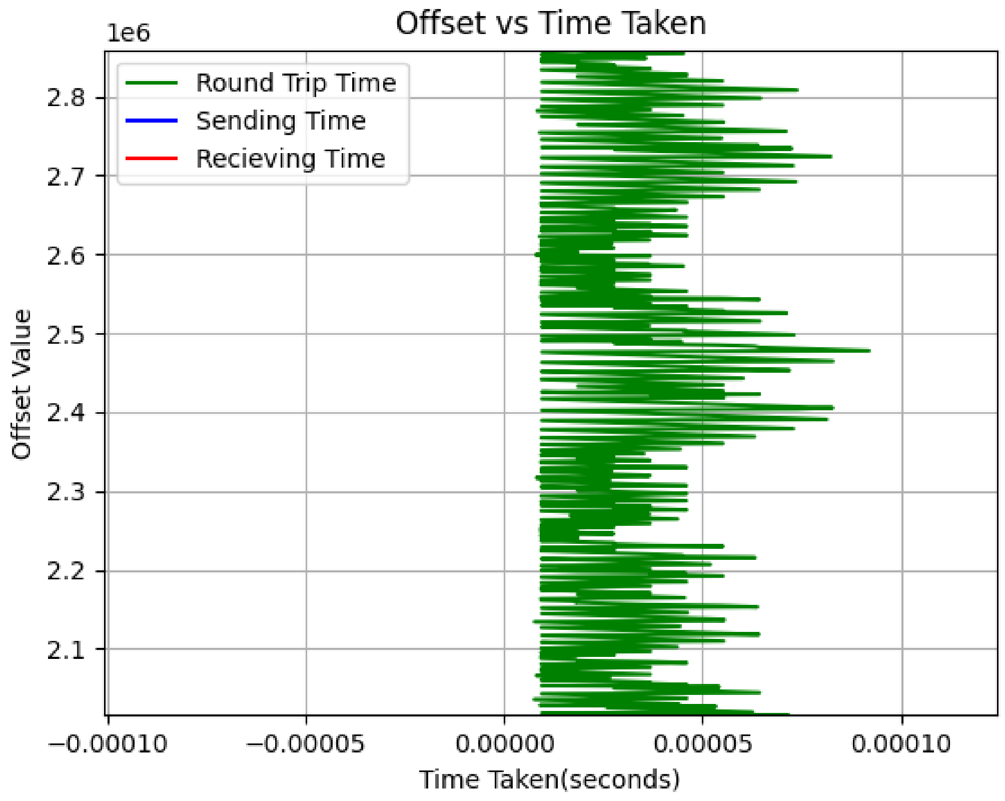
This is the graph obtained by running the code for receiving data from the Vayu server and shows components such as round trip time, request sending time, and data receiving time.



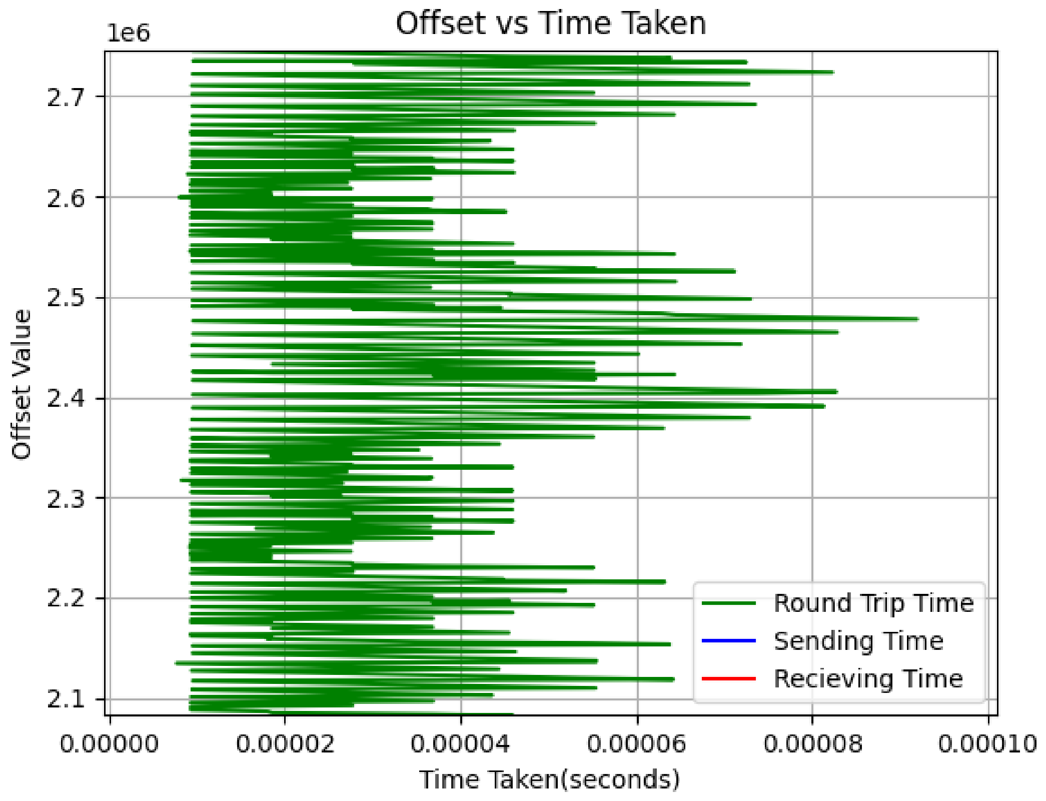
This is the zoomed view of the graph obtained by running the code for receiving data from the Vayu server which shows components such as request sending time and data receiving time.



This is the more zoomed view of the graph obtained by running the code for receiving data from the Vayu server which shows components such as request sending time and data receiving time.



This is the zoomed view of the graph obtained by running the code for receiving data from the Vayu server which shows components such as round trip time.



This is the more zoomed view of the graph obtained by running the code for receiving data from the Vayu server which shows components such as round trip time.

The total time on average that our client took to receive the entire data is **23.146 seconds** with an average of **5 penalties**.