MINOR PROJECT-II

End Sem Report

Topic: Bandwidth sharing Download Manager

Efforts by:

Meghna Gupta	**Aniket Bhatnagar**	**Vidit Mathur**
13103731	13103534	13103615
B4	B4	B4

ABSTRACT:

This project aims to speed up download of large files using bandwidth of other nodes connected on the same local network and works on the principle of "*byte range requests*". Here the server (node that wishes to download the file) would first check for all live nodes on the network, and then send request to these nodes for help. Later all complying nodes would be sent byte ranges for downloading the file and each such node will use top-notch multi-threading to download their chunk, which shall be sent back to the server later. The server shall merge the chunk into the main file as soon as it receives the particular range file.

REQUIREMENT ANALYSIS:

Software:

Python Interpreter
Kivy framework in python for UI design

Hardware:

Router or hotspot so that multiple hosts can be connected to one network
Laptops

Functional Requirements:

1. Broadcast the request for permission to help in download as soon as the server requests a document to be downloaded.

2. Divide the document into chunks so that each client downloads that range of data and at the client side further break the chunks to download using multiple threads.

3. Client should send back the chunks downloaded as soon as the download is complete and delete the chunk from its own system to prevent any privacy issues.

Non functional Requirements:

1. Response time: A very important nonfunctional requirement in such softwares is that the software should not have any lag and should respond quickly so that the user is able to do the work efficiently.

   Our software also is having a low response time because as soon as the user who wants to download a file he/she can start the software and all the connected hosts receive the request (which is broadcasted) immediately and the response of the client also is received immediately by the user without having to wait.

2. Privacy: Privacy is a basic requirement for any software .

The privacy of the user and the fact that the data being downloaded may be private and confidential is of prime importance. So our software makes sure that any part of the document being downloaded by other hosts is not accessible to these hosts. And so this data downloaded is deleted from the client's system as soon as it is sent to the server.

3. Fault tolerance: Any good software should be tolerant to basic faults that might occur during its working to ensure effective work.

Our software is also tolerant to faults that might occur. If the server or client is no longer connected to the internet the software displays a message stating this.
Also if one or more of the clients is not able to download the fragment due to certain problem the server keeps a check on that and downloads that fragment itself to prevent any delay in work.

Design and Implementation Details

The bandwidth sharing tool was built in following steps:
Here the node which instantiates the download is referred to as server and all other as clients.

1. Server first broadcasts UDP request to all love nodes to help in download.
2. All the nodes then send their response in the form of UDP message.
3. With each affirmative response the server increments the number of positive responses. Then the file to be downloaded is divided equally with respect to the number of positive responses.
4. Also we have taken care of the case where a client after giving a positive response might not be able for the download work due to network error or any other problem.
5. In one such case if the complying node is unable to establish TCP connection then the server with the help of exception handling shall stop waiting for that client and move ahead to download the particular chunk by itself.
6. In another case the server thread dedicated for that client if found alive after a certain time is terminated and server itself downloads that part of the file to prevent any further delay in the download completion.
7. To create the UI of the downloader we have used Kivy which is an open source Python library which has a natural user interface.
8. Start and end bytes to be downloaded by each node is computed at the server.
9. Then each accepting node establishes TCP connection with the server and server sends it the start and end indices of the part to be downloaded.

10. Then each node uses top-notch multi-threading to download it's part of the file and on completion of download it is sent back to the server immediately.
11. The downloaded part is also deleted from the client side so as to prevent any misuse of data by the client.
12. At the server side all the files received by the various clients are merged in proper order.

Objectives:

The main objective of developing this download manager is to help people in being able to download a file even if they have less resources(less download speed or less time) and propose an efficient method for the same by use of resources (bandwidth) of other computers connected on the same network and speed up local range downloads with the help of multi-threading.

Eg: If a person wants to download an important 2GB file then instead of waiting for the file to get downloaded in the designated time he can take help from the fellow nodes on the network thus increasing the effective download speed and thus being able to view the file earlier.

The framework is specially useful in organisations where many computers are simultaneously connected to high speed internet connection like college hostels, hospitals or commercial offices.

DIVISION OF WORK:

Meghna Gupta

- Developed the file transfer system.
- Built the UDP request broadcast and response management system.
- Implemented the try-except block in order to be able to recognise any error of client not being able to establish TCP connection.
- Designed the server side UI using kivy.

Aniket Bhatnagar

- Developed the multi threaded download function.
- Developed the TCP part of server-client communication for communicating url and byte ranges to clients.
- Implemented the try-except block in order to be able to recognise any error as to client getting disconnected from the network after TCP connection.
- Designed the client side UI using kivy.

Vidit Mathur

- Developed the system to divide the file to be downloaded into chunks so that each chunk can be downloaded by each live node.
- Developed the UDP request and accept functionality for server-client communication.
- Developed the merge functionality at the server side and client side.
- Developed synchronization strategy and established timeout conditions to handle multiple clients simultaneously.

PARTIAL IMPLEMENTATION:

CODE:

Multi-threading Download function

```
def fun(start,end,rurl):          #Downloader function
    global ext
    req = urllib2.Request(rurl)
    req.headers['Range'] = 'bytes=%s-%s' % (start, end)
    site = urllib2.urlopen(req)
    f = open(str(start)+ext, "wb")
    f.write(site.read())
    f.close()
    site.close()
    f = open(str(start)+ext, "rb")
    print "File on disk after download:",len(f.read())
    f.close()
def mgfiles(start):
    global ext
    f=open("final"+ext,"ab")
    f1=open(str(start)+ext,"rb")
    f.write(f1.read())
    f.close()

def download(rurl,start,end,val_ext):
    global ext
    ext=val_ext
    #site = urllib2.urlopen(rurl)
    #meta= site.info()
    size=end-start+1
    print "Total File size to be downloaded: "+str(size)
    N=5
    d=(size)/N
    start1=0
```

```
    end1=d-1
    arr=[[0]*2 for i in range(N)]
    arr[0]=[start1,end1]
    for i in range(1,N):
        if i!=N-1:
            start1=end1+1
            end1=start1+d-1
            arr[i]=[start1,end1]
        else:
            start1=end1+1
            end1=size-1
            arr[i]=[start1,end1]
    th=[]
    for i in range (N):
        t = Thread(target=fun, args=(arr[i][0],arr[i][1],rurl,))
        th.append(t)
    for i in range(N):
        th[i].start()
    for i in range(N):
        th[i].join()
    for i in range(N):
        mgfiles(arr[i][0])
    for i in range(N):
        os.remove(str(arr[i][0])+ext)
```

Server side multi-client TCP regulator function

```
def handleclient(connsocket,start,end,i):
    msg=rurl+' '+str(start)+' '+str(end)+' '+str(i)
    connsocket.send(ext)
    connsocket.send(msg)
    f=open(str(i)+ext,'wb')
    while True:
        l=connsocket.recv(1024)
        if not l:
            break
        f.write(l)
    f.close()
    print "Recvd succesfully"
    connsocket.close()
def acc_tcp():
    #Divide file into ranges
    global yescount
```

```
    N=yescount
    d=(size)/N
    start1=0
    end1=d-1
    arr=[[0]*2 for i in range(N)]
    arr[0]=[start1,end1]

    for i in range(1,N):
            if i!=N-1:
                    start1=end1+1
                    end1=start1+d-1
                    arr[i]=[start1,end1]
            else:
                    start1=end1+1
                    end1=size-1
                    arr[i]=[start1,end1]
    #Set server host,port and socket
    host = myip
    port = 50005
    s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    # Bind the socket to the port
    s.bind((host, port))
    s.listen(5)
    print 'Server binded and ready to use for TCP'
    i=0
    while i<yescount:
            try:
                    s.settimeout(250*yescount)
                    connsocket,addr = s.accept()
                    t = threading.Thread(target=handleclient,
args=(connsocket,arr[i][0],arr[i][1],i,))
                    th.append(t)
                    i=i+1
            except socket.timeout:
                    #In case the client under consideration fails to make TCP connection
                    print "Problem1: Server itself downloads chunk "+str(arr[i][0])+"-"
+str(arr[i][1])
                    download_dhaga.download(rurl,arr[i][0],arr[i][1],ext)
                    os.rename('final'+ext,str(i)+ext)
                    mgfiles(i)
                    os.remove(str(i)+ext)
                    i=i+1

    for i in range(len(th)):
                    th[i].start()
```

```
    for i in range(len(th)):
                th[i].join(60.0)
                if th[i].isAlive():
                        print "Problem2: Server itself downloads chunk "+str(arr[i][0])+"-"
+str(arr[i][1])
                        download_dhaga.download(rurl,arr[i][0],arr[i][1],ext)
                        os.rename('final'+ext,str(i)+ext)

    for i in range(len(th)):
                mgfiles(i)
    for i in range(len(th)):
        os.remove(str(i)+ext)
    s.close()
```

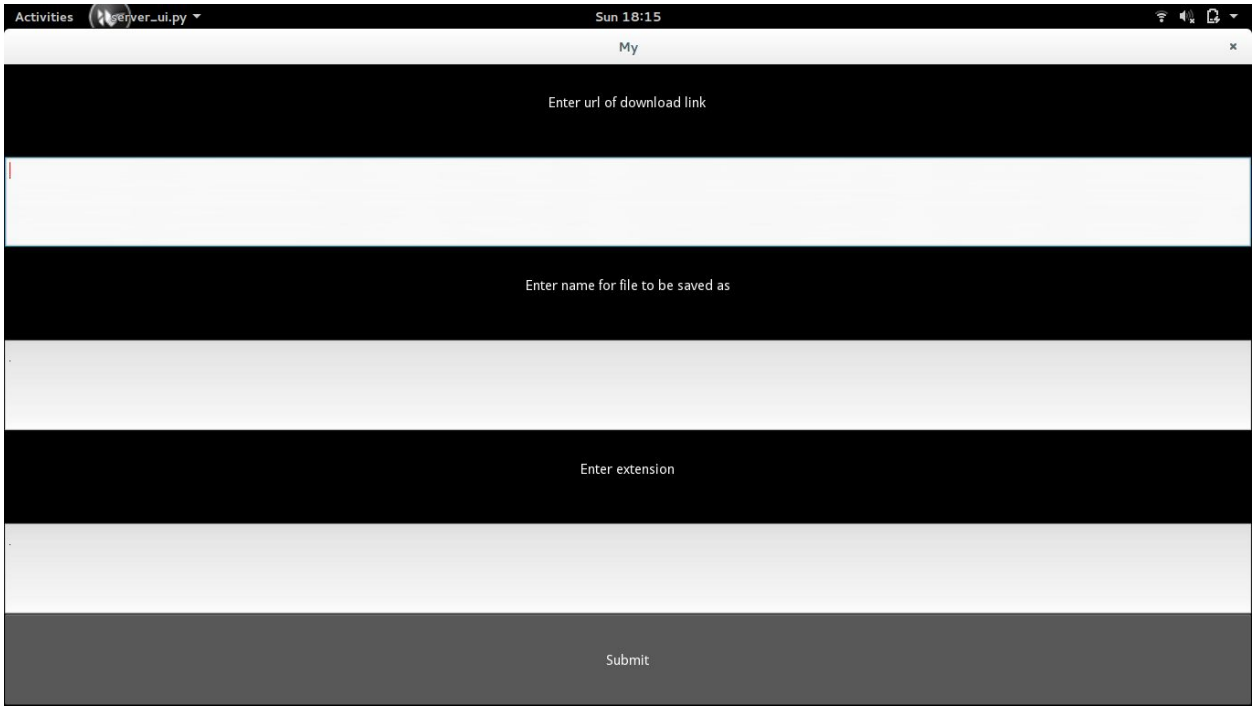
Server side multi-client UDP regulator function

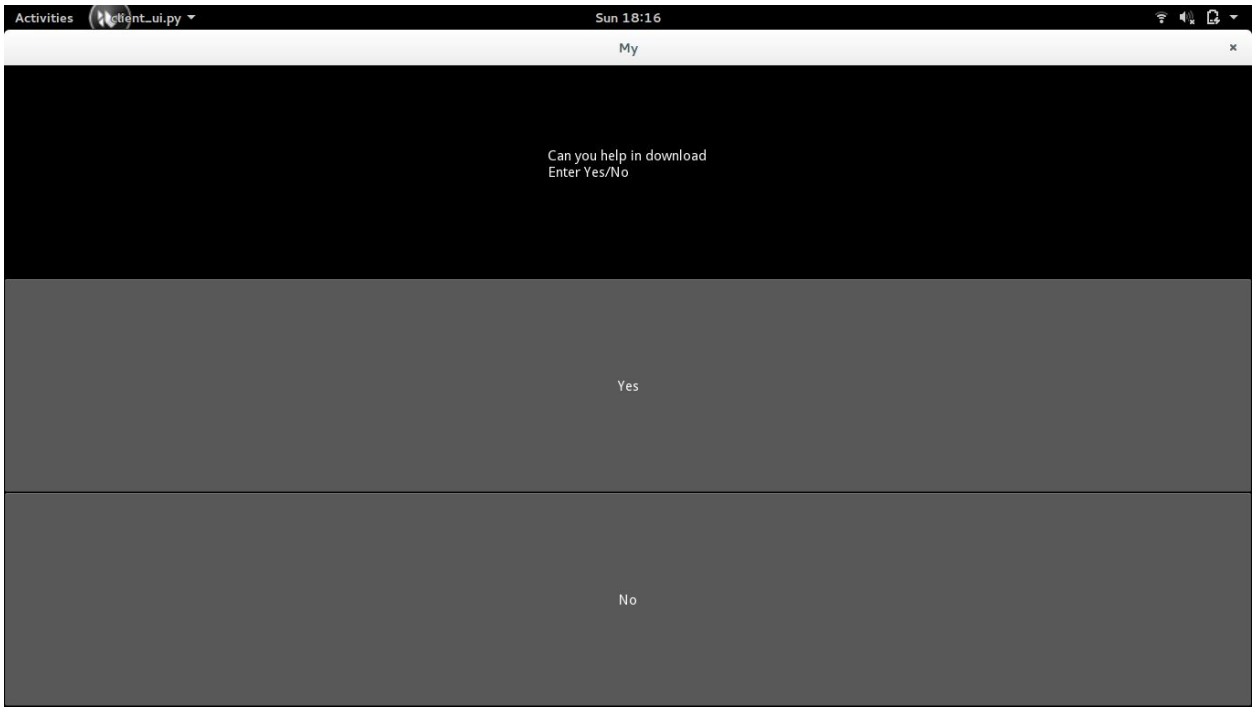
```
def broad_udp():
    global yescount
    message="Can you help in download"
    #Set server host,port and socket
    host = myip
    port = 50005
    totno=len(live_ips)
    message=message+'#'+str(totno)
    for x in live_ips:
        try:
                s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
                s.bind((host,port))
                s.settimeout(6)
                s.sendto(message,(x,50008))
                print x
                reply=""
                reply,caddr=s.recvfrom(2048)
                if reply=='Yes':
                        yescount=yescount+1
                s.close()
        except socket.timeout:
                print "caught timeout"
    print yescount
```

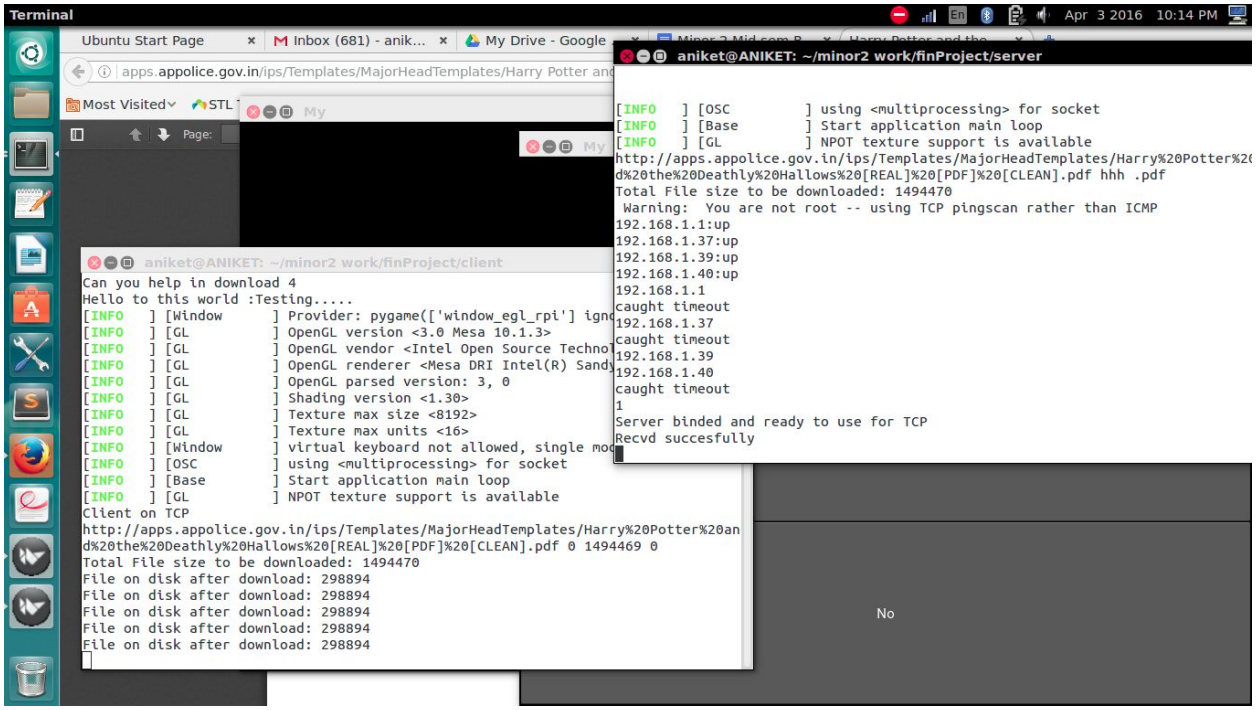
Client UI in kivy

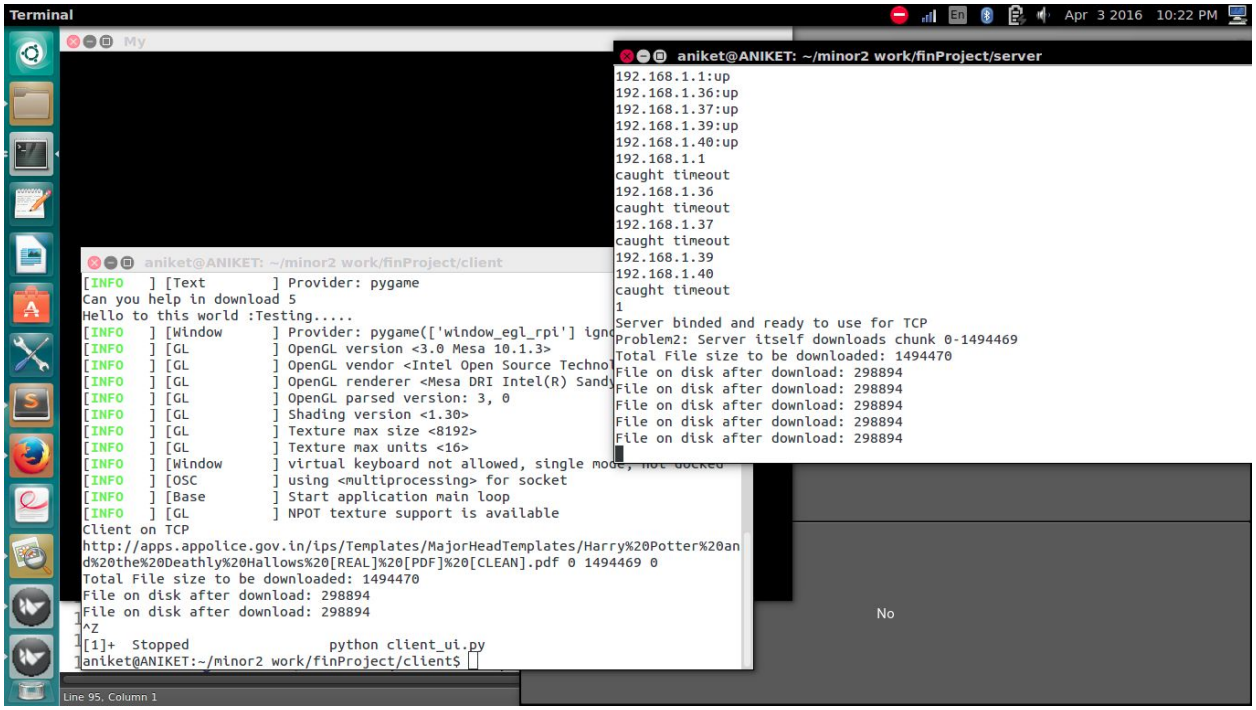
```
class MyApp(App):

def build(self):
        #UDP part
        global clientSocket
        global serverip
        global serveraddr
        global totno
        clientSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        clientSocket.bind((myip,50008))
        msg,serveraddr=clientSocket.recvfrom(2048)
        serverip=serveraddr[0]
        msg,totno=msg.split('#')          #totno represents total no of nodes on
network
        print msg
        totno=int(totno)
        box = BoxLayout(orientation='vertical')
        label1 = Label(text=msg+'\nEnter Yes/No\n')
        btn1 = ToggleButton(text='Yes', state='normal')
        btn1.bind(on_press=ifyes)
        btn2 = ToggleButton(text='No', state='normal')
        btn2.bind(on_press=ifno)
        box.add_widget(label1)
        box.add_widget(btn1)
        box.add_widget(btn2)
        return box

        time.sleep(10)

MyApp().run()
```

SNAPSHOTS:


Activities
server_ui.py
Sun 18:15
My
Enter url of download link
Enter name for file to be saved as
Enter extension
Submit


Activities
client_ui.py
Sun 18:16
My
Can you help in download
Enter Yes/No
Yes
No


Terminal
Apr 3 2016 10:14 PM
aniket@ANIKET: ~/minor2 work/finProject/server
[INFO] [OSC] using <multiprocessing> for socket
[INFO] [Base] Start application main loop
[INFO] [GL] NPOT texture support is available
http://apps.appolice.gov.in/ips/Templates/MajorHeadTemplates/Harry%20Potter%20
d%20the%20Deathly%20Hallows%20[REAL]%20[PDF]%20[CLEAN].pdf hhh .pdf
Total File size to be downloaded: 1494470
Warning: You are not root -- using TCP pingscan rather than ICMP
192.168.1.1:up
192.168.1.37:up
192.168.1.39:up
192.168.1.40:up
192.168.1.1
caught timeout
192.168.1.37
caught timeout
192.168.1.39
192.168.1.40
caught timeout
1
Server binded and ready to use for TCP
Recvd succesfully
aniket@ANIKET: ~/minor2 work/finProject/client
Can you help in download 4
Hello to this world :Testing.....
[INFO] [Window] Provider: pygame([
[INFO] [GL] OpenGL version <3.0 Mesa 10.1.3>
[INFO] [GL] OpenGL vendor <Intel Open Source Techno
[INFO] [GL] OpenGL renderer <Mesa DRI Intel(R) Sandy
[INFO] [GL] OpenGL parsed version: 3, 0
[INFO] [GL] Shading version <1.30>
[INFO] [GL] Texture max size <8192>
[INFO] [GL] Texture max units <16>
[INFO] [Window] virtual keyboard not allowed, single mo
[INFO] [OSC] using <multiprocessing> for socket
[INFO] [Base] Start application main loop
[INFO] [GL] NPOT texture support is available
Client on TCP
http://apps.appolice.gov.in/ips/Templates/MajorHeadTemplates/Harry%20Potter%20an
d%20the%20Deathly%20Hallows%20[REAL]%20[PDF]%20[CLEAN].pdf 0 1494469 0
Total File size to be downloaded: 1494470
File on disk after download: 298894
File on disk after download: 298894
File on disk after download: 298894
File on disk after download: 298894
File on disk after download: 298894
Ubuntu Start Page
Inbox (681) - anik...
My Drive - Google
apps.appolice.gov.in/ips/Templates/MajorHeadTemplates/Harry Potter ant
Most Visited
Page:
No


Terminal
Apr 3 2016 10:22 PM
aniket@ANIKET: ~/minor2 work/finProject/server
192.168.1.1:up
192.168.1.36:up
192.168.1.37:up
192.168.1.39:up
192.168.1.40:up
192.168.1.1
caught timeout
192.168.1.36
caught timeout
192.168.1.37
caught timeout
192.168.1.39
192.168.1.40
caught timeout
1
Server binded and ready to use for TCP
Problem2: Server itself downloads chunk 0-1494469
Total File size to be downloaded: 1494470
File on disk after download: 298894
File on disk after download: 298894
File on disk after download: 298894
File on disk after download: 298894
File on disk after download: 298894
aniket@ANIKET: ~/minor2 work/finProject/client
[INFO] [Text] Provider: pygame
Can you help in download 5
Hello to this world :Testing.....
[INFO] [Window] Provider: pygame([
[INFO] [GL] OpenGL version <3.0 Mesa 10.1.3>
[INFO] [GL] OpenGL vendor <Intel Open Source Techno
[INFO] [GL] OpenGL renderer <Mesa DRI Intel(R) Sandy
[INFO] [GL] OpenGL parsed version: 3, 0
[INFO] [GL] Shading version <1.30>
[INFO] [GL] Texture max size <8192>
[INFO] [GL] Texture max units <16>
[INFO] [Window] virtual keyboard not allowed, single mo
[INFO] [OSC] using <multiprocessing> for socket
[INFO] [Base] Start application main loop
[INFO] [GL] NPOT texture support is available
Client on TCP
http://apps.appolice.gov.in/ips/Templates/MajorHeadTemplates/Harry%20Potter%20an
d%20the%20Deathly%20Hallows%20[REAL]%20[PDF]%20[CLEAN].pdf 0 1494469 0
Total File size to be downloaded: 1494470
File on disk after download: 298894
File on disk after download: 298894
^Z
[1]+ Stopped python client_ui.py
aniket@ANIKET:~/minor2 work/finProject/client$
Line 95, Column 1
No

Testing:

We have used try and except pairs wherever there was possibility of an error or an exception so that the user can directly get the message as to what problem occurred in the software. At any point if the user encounters a problem with the working of the downloader an appropriate message will be printed.

Also we have prevented some errors using this method by setting the timeout duration according to the present number of live nodes on the network so that the server does wait for the appropriate amount of time for the response of all the clients and does not terminate until all live nodes running the client scripts have given their nod or rejected the request for the download work.

This also helped us while developing the downloader as whenever we came across any problem in the code or the functionality of the downloader we were able to make out the exact part of the code where the problem was persisting without having to slog hours in trying to make out the function where the error was.

Gantt Chart

- Basic server design ---> a
- Basic client design ---> b
- Basic file sharer ---> c
- Fetch connected (live) nodes ---> d
- Multi-threaded download manager ---> e
- Developing file merge system ---> f
- Thread-up and supply ranges ---> g
- Receive ranges and accordingly start download ---> h
- Merging of byte divided file ranges ---> i
- Receiver for data from client ---> j
- Send final file and delete rest ---> k
- Generate UI ---> l
- Set up broadcast ---> m
- Design broadcast receivers ---> n
- Generate timeouts ---> o
- Notifications system to notify server of failure ---> p
- Management of failure ---> q

REFERENCES:

1)http://security.stackexchange.com/questions/36198/how-to-find-live-hosts-on-my-network
2)http://serverfault.com/questions/153776/nmap-find-all-alive-host names-and-ips-in-lan
3) http://www.tutorialspoint.com/python/python_networking.htm
4) https://docs.python.org/2/howto/sockets.html
5)http://stackoverflow.com/questions/27241804/sending-a-file-over-tcp-sockets-in-python
6)http://www.bogotobogo.com/python/python_network_programming_server_client_ file_transfer.php
7) https://kivy.org/

8) https://kivy.org/docs/guide/widgets.html
9) https://kivy.org/docs/guide/basic.html
10) https://docs.python.org/2/tutorial/errors.html
11)http://stackoverflow.com/questions/15696461/import-python-script-into-another
12)http://www.binarytides.com/python-socket-programming-tutorial/