MINOR- II
NETWORKING – PROJECT

"ALPHA DOWNLOADER && FILE SHARING "
(Using Internet)

BATCH : B -5

VRISHTI GAHLAUT (12103528)

DHRUV SINGH (12103533)

SUMIT BANSAL (12103530)

HAMZA KHAN (12103540)

Abstract

With the dependence of today's world on downloading data from the World Wide Web is reaching new heights, new and effective techniques must be developed to satisfy these downloading demands. This project is a step towards that direction.

We present the "Alpha downloader", a new download manager that effectively boosts the downloading speed and provides multi-client cache facilities.

The project is built solely on the python platform and uses the concept of multithreading to increase the downloading speed. Threading is a feature that enables parallel processing i.e. multiple threads execute at the same time. Thus, several packets of data can be downloaded at the same time in contrast to the orthodox downloaders which allow only a single stream of packets at a time. The "Alpha downloader" needs to be fed with the URL of the intended download page and a destination location on the server memory to begin downloading.

Along with speedy downloading, "Alpha downloader" allows multiple clients to be connected to the main server which is also the downloader using IP addresses and port numbers and handles them at the same time. Thus a page needs to be downloaded only once by the server and then any connected client can have access to it, saving the highly expensive Data usage and also saving time for multiple downloads of the same data. Also, if a particular page is already downloaded by the server, then any client requesting the same data gets it in no time as the page only needs to the sent to the client by the server.

Objective

The purpose of the project is to build a new age downloader cum server which provides high speed downloading and handles multiple clients at the same time using python as the coding platform and the concept of multithreading and server client communication.

This software is highly useful for local area networks for example a college computer lab where downloading tends to get slow due to high load on the Internet. In such a scenario, "Alpha downloader" will provide high downloading speed. Also, all students can connect to this server, therefore since most of the download requests would of the same page or data on a particular day, hence "Alpha downloader" will have to simple forward the already downloaded page to the requesting client.

Division of project work:

Dhruv - client.py, client side commumication and functioning.

Sumit - server.py, server side commumication and functioning.

Vrishti - GUI related complete designing (server and client side)

Hamza - Debuging related work , ideas , code testing.

Designing :

Language: Python 2.7 language used for create downloader application and design .

Platform : Windows 7

(Socket based application)

There are 2 python files .

- Server.py
- Client.py

To Run :

On cmd-
python server.py
python client.py

Background Study and Findings :

A **download manager** is a computer program dedicated to the task of downloading (and sometimes uploading) possibly unrelated stand-alone files from (and sometimes to) theInternet for storage. Some download managers can also be used to accelerate download speeds by downloading from multiple sources at once. Although web browsers may have download managers incorporated as a feature, they are differentiated by the fact that they do not prioritize accurate, complete and unbroken downloads of information. While some download managers are fully fledged programs dedicated to downloading any information over one or more protocols (e.g. http), many are integrated into installers or update managers and used to download parts of a specific program (or set of programs).

Download acceleration, also known, as multipart download, is a term for the method employed by software such as download managers to download a single file by splitting it in segments and using several simultaneous connections to download these segments from a single server.

The reason for doing so is to circumvent server side limitations of bandwidth per connection. Because in normal networking situations all individual connections are treated equally, rather than actual file transfers, multiple connections yields an advantage on saturated links over simple connections, both in terms of total bandwidth allocation and resilience. Many servers, however, implement a maximum number of simultaneous connections per client in order to mitigate this.

This is not to be confused with segmented downloading, which allows a client to download segments of a file simultaneously from multiple servers.

Partial Implémentation and Testing :

There are two codes . one is for server and other for clients .

Server.py

```
no_of_connections=0
no_of_acceptance=0
start_flag = False;
connect_flag = False;


def download(i):
   if i==0:
      start=0
   else:
      start=i*packet_size_each_thread[i-1]
   start1=str(start)
   end=start+packet_size_each_thread[i]-1
   end1=str(end)
   req = urllib2.Request(url, headers={'Range':'bytes='+start1+'-'+end1})
   u = urllib2.urlopen(req)
   block_sz = 8192
   file_size_dl = start
```

```
    temp='\0'
    while True:
        data=u.read(block_sz)
        if not data:
            break
        print('%d Fetched %s from %s' % (i,len(data), url))
        temp=temp+data

    temp=temp[1:]
    print "\n\n"
    print len(temp),"\n\n"
    f.seek(start)
    f.write(temp)
```

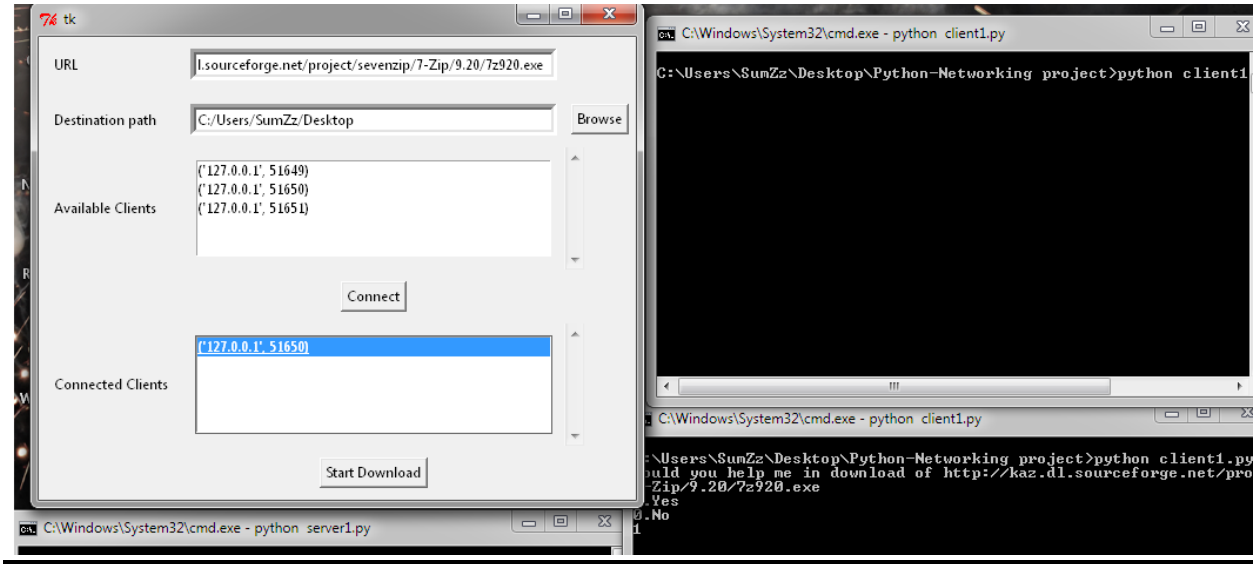
```
def start_parallel():
    #result = Queue.Queue()
    threads = [threading.Thread(target=download, args = (i,)) for i in range(no_of_threads)]
    for t in threads:
        t.start()
    for t in threads:
        t.join()
```

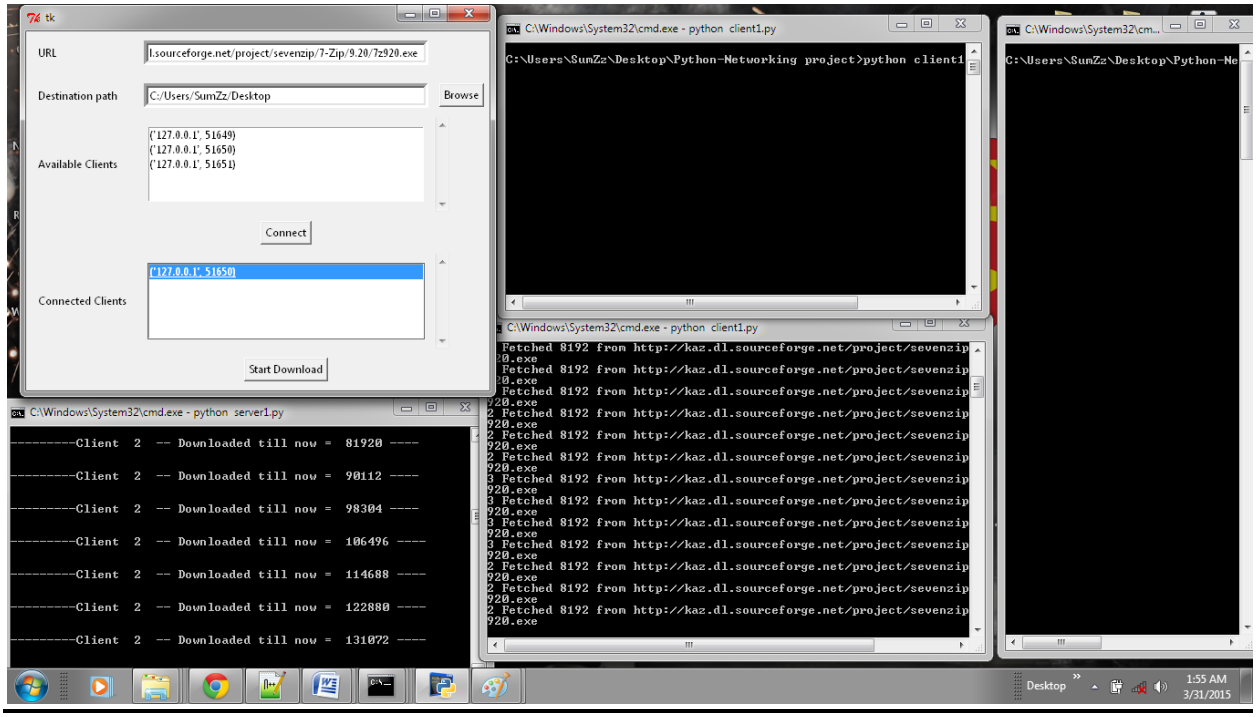
Client.py

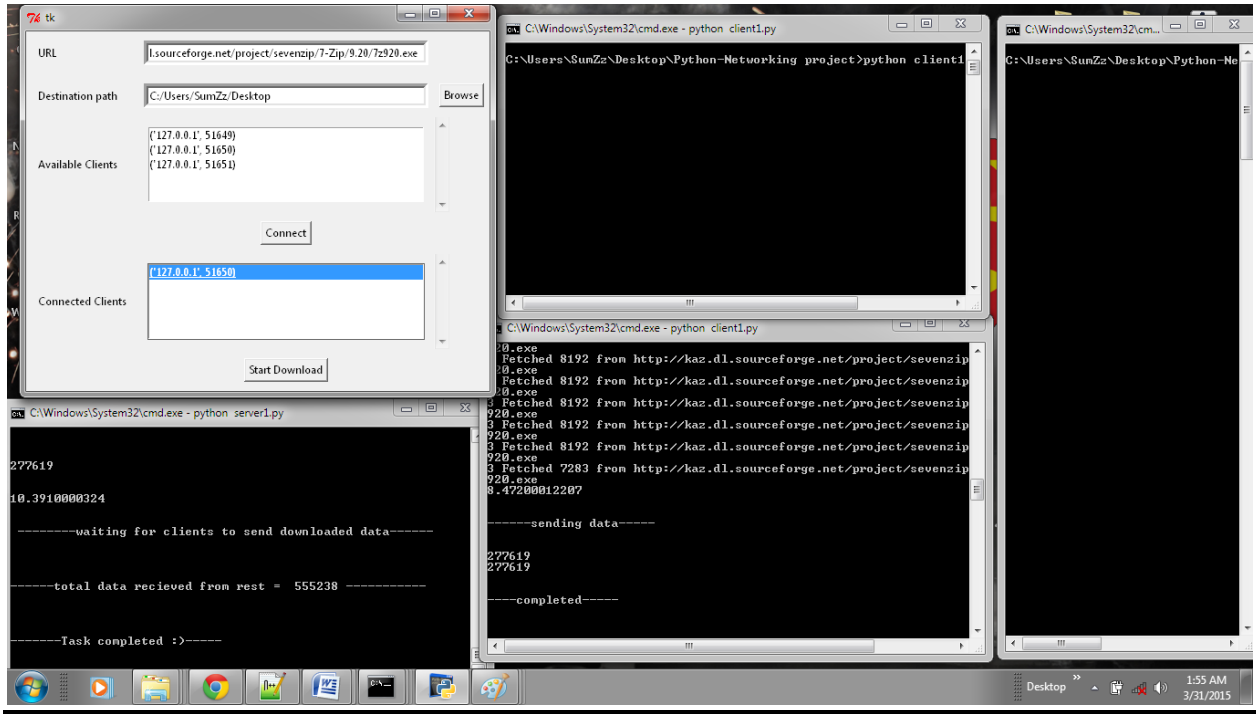
```
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

client_socket.connect(('localhost', 8080))


request_to_download=client_socket.recv(8812)

print request_to_download

print "1.Yes\n0.No"


choice=raw_input()


if choice=='1':

   response="acceptance=1"

else:

   response="acceptance=0"


client_socket.send(response)


if choice=='1':

   client_no=int(client_socket.recv(8812))

no_of_threads=2

#client_no=1
```

```
if choice=='1':


    url = client_socket.recv(8812)

    start_from=int(client_socket.recv(8812))

    packet_size_each_client=int(client_socket.recv(8812))


    print "\n\n------amount to download= ",packet_size_each_client,"--------\n\n"


    packet_size_each_thread=[]

    i=0

    while i<no_of_threads:

        if i != no_of_threads-1:

            packet_size_each_thread.append(packet_size_each_client/no_of_threads)

        else:

            packet_size_each_thread.append((packet_size_each_client/no_of_threads) +
(packet_size_each_client%no_of_threads))

        i+=1


    #print packet_size_each

    u = urllib2.urlopen(url)


    main_data=[]

    main_data.append([])

    main_data.append([])
```

SNAPSHOTS:


tk
URL
l.sourceforge.net/project/sevenzip/7-Zip/9.20/7z920.exe
Destination path
C:/Users/SumZz/Desktop
Browse
Available Clients
('127.0.0.1', 51649)
('127.0.0.1', 51650)
('127.0.0.1', 51651)
Connect
Connected Clients
('127.0.0.1', 51650)
Start Download
C:\Windows\System32\cmd.exe - python server1.py
C:\Windows\System32\cmd.exe - python client1.py
C:\Users\SumZz\Desktop\Python-Networking project>python client1
C:\Windows\System32\cmd.exe - python client1.py
Yes
No


tk
URL
l.sourceforge.net/project/sevenzip/7-Zip/9.20/7z920.exe
Destination path
C:/Users/SumZz/Desktop
Browse
Available Clients
('127.0.0.1', 51649)
('127.0.0.1', 51650)
('127.0.0.1', 51651)
Connect
Connected Clients
('127.0.0.1', 51650)
Start Download
C:\Windows\System32\cmd.exe - python server1.py
---------Client 2 -- Downloaded till now = 81920 ----
---------Client 2 -- Downloaded till now = 90112 ----
---------Client 2 -- Downloaded till now = 98304 ----
---------Client 2 -- Downloaded till now = 106496 ----
---------Client 2 -- Downloaded till now = 114688 ----
---------Client 2 -- Downloaded till now = 122880 ----
---------Client 2 -- Downloaded till now = 131072 ----
C:\Windows\System32\cmd.exe - python client1.py
C:\Users\SumZz\Desktop\Python-Networking project>python client1
C:\Windows\System32\cmd.exe - python client1.py
Fetched 8192 from http://kaz.dl.sourceforge.net/project/sevenzip
C:\Windows\System32\cm...
C:\Users\SumZz\Desktop\Python-Ne
Desktop
1:55 AM
3/31/2015



References:

- Wikipedia .
- Research paper related to downloader .
- Python documentation file and tutorials .
- Stack over flow problem discussions.
- Downloader working .