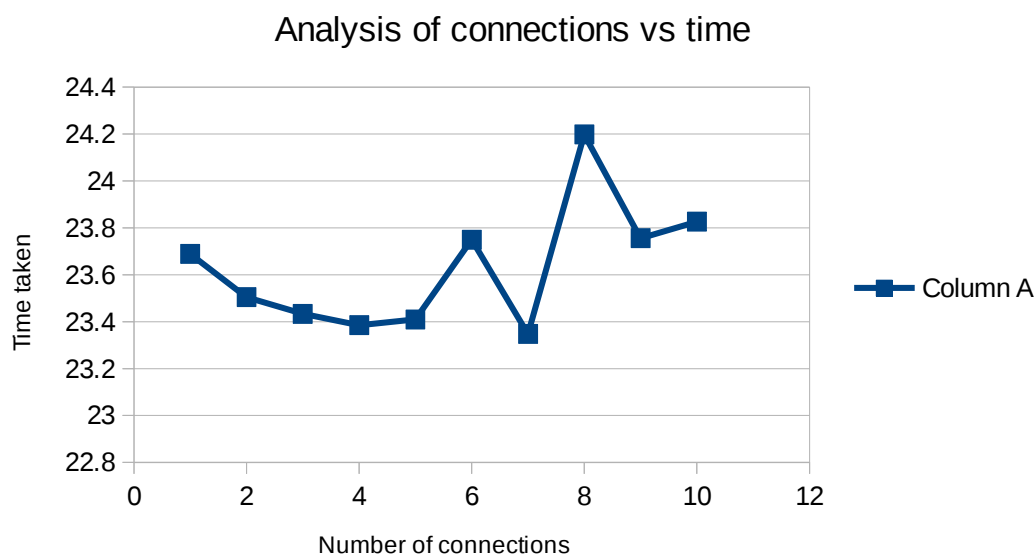


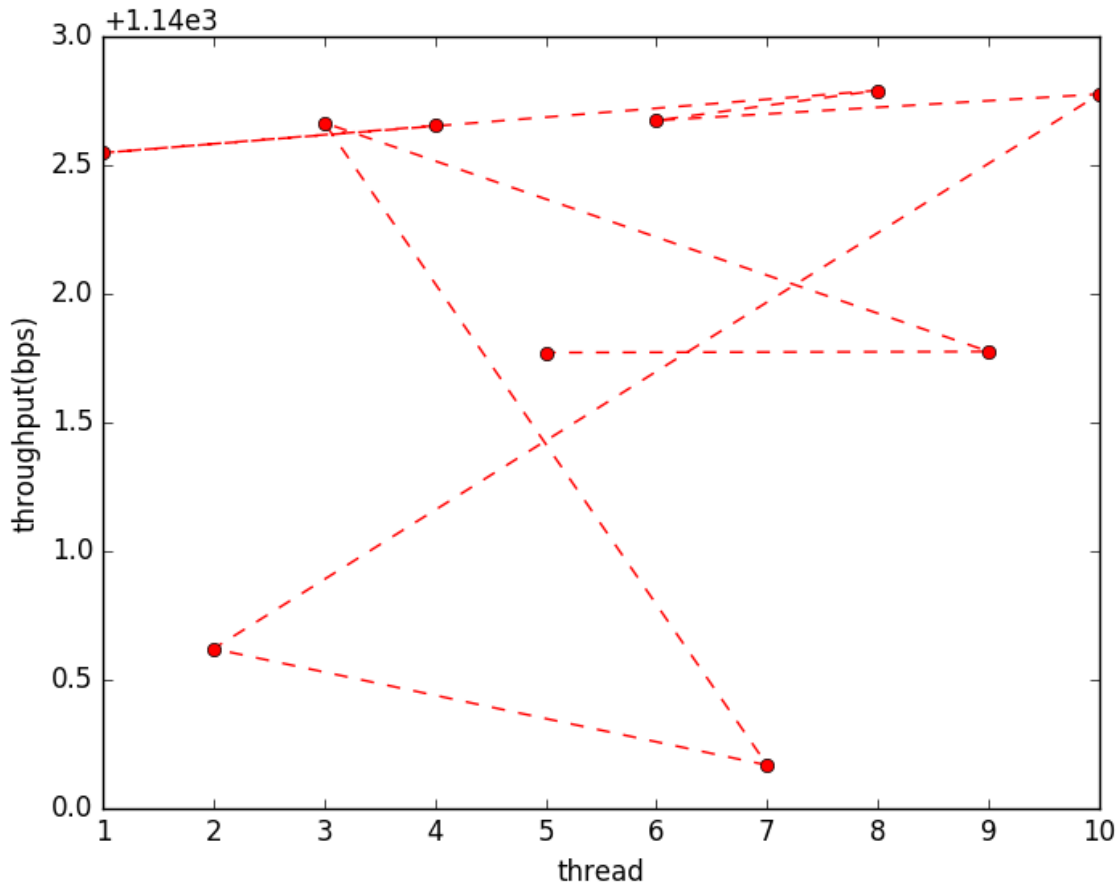
COMPUTER NETWORKS 2
Sreekar B
CS14BTECH11033
Assignment II
File download using range request

2) a) Given below is the graph for the time taken using different number of connections to download a file, we can see that till 5 connections, the time taken to download the file decreasing, since this is a small file downloaded and the the number of connections is significantly adding some profit by decreasing the time taken to download whereas by making it 6-8 connections it took comparatively more time which is adding some profit in download speed. It also depends on the current traffic.



Instantaneous throughput

instantaneous throughput of each thread downloading a file
given a plot obtained to download a file with 10 tcp connections



we can see that the throughput became random

IMPLEMENTATION

This program takes url to download and the number parallel connections initially it obtains a connection from the server, since this is a http server the standard port number is 80 and the url of the main server which is given as the netloc by using urlparse function and use it as HOST address to obtain connection, It then resolves the ip and connects to the server the client sends a request to get the meta data of the file we want to download, this meta data contains the information of the content size.

The header is created as "GET "+url.path+ " HTTP/1.1\nHOST: "+HOST + "\r\n\r\n"

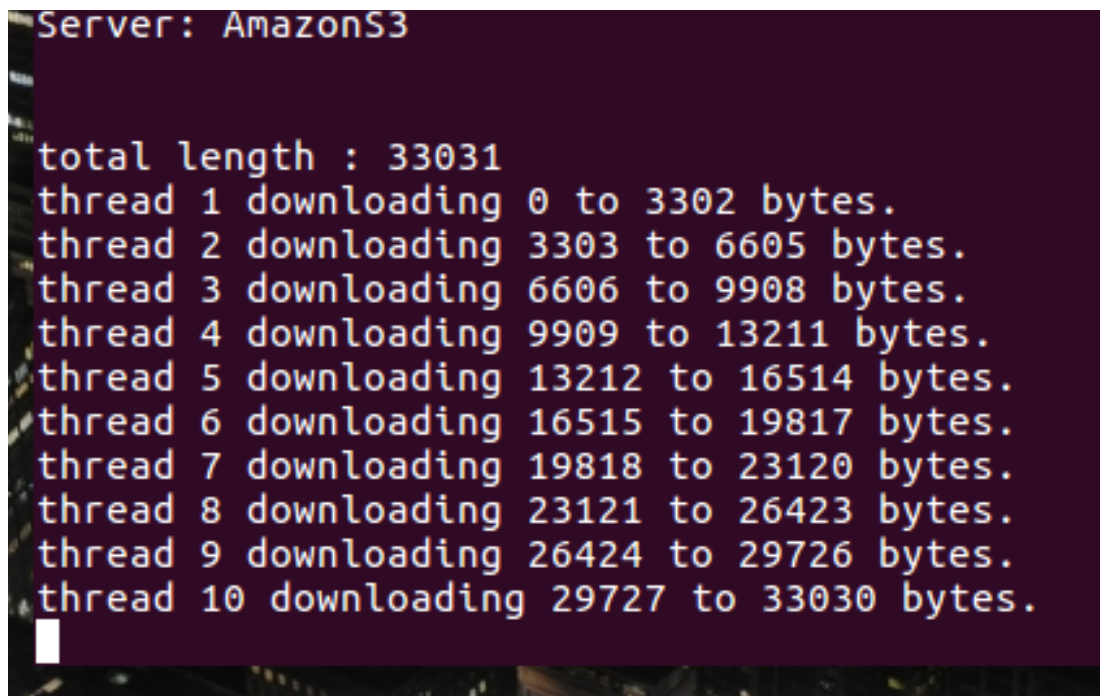
this is the standard format and url.path is the file and HTTP1.1 is the type of http and gives the host information and get the content file

the response from the server is as follows

```
HTTP/1.1 200 OK
x-amz-id-2:
or/iQjtNPPGH2eIeD2Gow9ly/RkXus/m7ikG6yEhCgEJvWpplvLyuRgxjm8biJ
uOI6+cjMZ6s8A=
x-amz-request-id: E5B5F2CE14E8D7D9
Date: Thu, 09 Mar 2017 18:15:22 GMT
Last-Modified: Tue, 07 Mar 2017 11:12:59 GMT
ETag: "3297badc39b7564ea5432e40a7b32a58"
Content-Type: image/jpeg
Content-Length: 33031
Server: AmazonS3
```

The content-Length is the size of the file

we are given a number of parallel streams so we share the data among the threads equally



```
Server: AmazonS3

total length : 33031
thread 1 downloading 0 to 3302 bytes.
thread 2 downloading 3303 to 6605 bytes.
thread 3 downloading 6606 to 9908 bytes.
thread 4 downloading 9909 to 13211 bytes.
thread 5 downloading 13212 to 16514 bytes.
thread 6 downloading 16515 to 19817 bytes.
thread 7 downloading 19818 to 23120 bytes.
thread 8 downloading 23121 to 26423 bytes.
thread 9 downloading 26424 to 29726 bytes.
thread 10 downloading 29727 to 33030 bytes.
```

we can see that the total length 33031 is divided most equally among 10 threads each thread includes a range request in the header

```
"GET "+url.path+ " HTTP/1.1\nHOST: "+HOST+"\nRange: bytes="+str(st)+"-"+str(fi)+"\r\n\r\n"
```

we here str(st) and str(fi) are the starting and ending bytes respectively for partial download

In each thread we write the contents each file named part0,part1..... partN-1 these files are then appended to form the main file after the threads finishes its execution.

Finally these chunk files are removed from the memory. Using os.remove()

Each thread obtains a seperate connection. After obtaining the connection it sends range requests and receives data.

Parallel connections seems to be profitable by observing the above graphs.

Commands

```
python download.py url_to_download Number_of_connections
```