Assignment 3

Diwakar Prajapati

2018CS10330

1. Single TCP connection and single packet download

In this part we were required to download the entire file on only 1 TCP connection, in one go, i.e. only 1 GET request.
Since the file is very large, the response which we will get would be coming in chunks, each chunk has a header and body.
I made a parser for separating the header and the body, also extracted few field of the header like Content-Length , Content-Range.

2. Single TCP connection and multiple packet download.

In this part we are required to download the only on 1 TCP connection. But rather than sending only 1 GET request with entire byte range, we have to do it by dividing the entire byte range into multiple smaller ranges and hence making multiple GET requests.
Since we know that TCP is reliable, so whatever byte range we send we can assure than the response we will be getting in 1 TCP connection, they will be in order. This is handled by the transport layer.

3. Multiple parallel TCP connection and multiple packet download.

In this part, we are required to download the file over multiple TCP connection sending and receiving data in parallel,
We need to make the TCP connection to various hosts, example, vayu.iitd.ac.in norvig.com. We need to map a particular range of bytes to only 1 Host and only 1 TCP connection, so avoid duplicate downloads.
I experiment with multiple combinations, few important points:

- If I make only 1 TCP connection and multiple packets each of size 10000 KB, then since file size if 6488666 KB, so there will be at least 649 packets required, now this is a huge number so an overflow is occurring at the transport layer buffer and we are now able to receive all the data. Hence we require a larger sized chunk to download data only over 1 TCP connection.
- If we download the entire file only from vayu server then it takes only about 6-8 secs to download, and but if we distribute the bytes over norvig server also then it takes more time of around 15-16 secs. This is because the norvig server if farther than the vayu server, so it take more time irrespective of the number of packet and tcp connection and file size.
- In my implementation I am writing the request and listening the response parallely, So I have made 2 threads for each TCP connection, one which writes request into the socket and another which listen for the response from the socket.
- Theoretically if we increase the number of parallel connection then download time should decrease but practically it decrease and then increases after certain no of tcp connection.
- I have divided the file equally among all the tcp connection. which mean that each tcp will download approximately equal number of bytes, but since vayu server has very low round trip time, so I keep no of connection in vayu server more than the norvig server.

Figure 1: 1 TCP connection on vayu.iitd.ac.in, 1 GET Request

```
PS C:\Users\diwak\Desktop\COL\COL334\Assignments\Assignment3> java request
Get byte range:         0-6488665
Downloaded bytes:       0-6488665

FILE DOWNLOADED: 1 Packets received
TOTAL TIME = 7.9898698
MD5 sum of origianl file:       70a4b9f4707d258f559f91615297a3ec
MD5 sum of downloaded file:     70a4b9f4707d258f559f91615297a3ec
PS C:\Users\diwak\Desktop\COL\COL334\Assignments\Assignment3>
```

Figure 2: 1 TCP connection on vayu.iitd.ac.in, 2 GET request

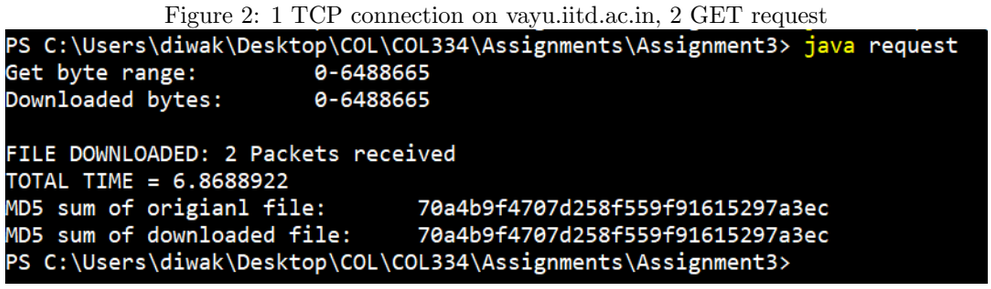


Figure 3: 2 TCP connection on vayu.iitd.ac.in, 1 GET request per TCP connection

Figure 4: 2 at vayu.iitd.ac.in, 2 GET request per TCP connection

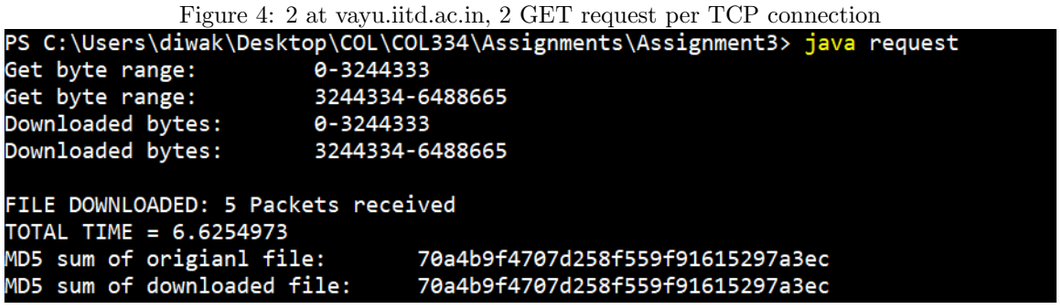


Figure 5: 1 TCP at vayu.iitd.ac.in and 1 TCP at norvig.com, 2 GET request per TCP connection

```
PS C:\Users\diwak\Desktop\COL\COL334\Assignments\Assignment3> java request
Get:    0-3244333       vayu.iitd.ac.in
Get:    3244334-6488665         norvig.com
Received:       0-3244333       vayu.iitd.ac.in
Received:       3244334-6488665         norvig.com

FILE DOWNLOADED: 34 Packets received
TOTAL TIME = 17.8519648
MD5 sum of origianl file:       70a4b9f4707d258f559f91615297a3ec
MD5 sum of downloaded file:     70a4b9f4707d258f559f91615297a3ec
```

Figure 6: 3 TCP at vayu and 3 TCP at norvig

```
PS C:\Users\diwak\Desktop\COL\COL334\Assignments\Assignment3> java request
Get:    0-1081444       vayu.iitd.ac.in
Get:    1081445-2162889         vayu.iitd.ac.in
Get:    2162890-3244334         vayu.iitd.ac.in
Get:    3244335-4325779         norvig.com
Get:    4325780-5407224         norvig.com
Get:    5407225-6488665         norvig.com
Received:       0-1081444       vayu.iitd.ac.in
Received:       2162890-3244334         vayu.iitd.ac.in
Received:       1081445-2162889         vayu.iitd.ac.in
Received:       3244335-4325779         norvig.com
Received:       4325780-5407224         norvig.com
Received:       5407225-6488665         norvig.com

FILE DOWNLOADED: 330 Packets received
TOTAL TIME = 14.2758639
MD5 sum of origianl file:       70a4b9f4707d258f559f91615297a3ec
MD5 sum of downloaded file:     70a4b9f4707d258f559f91615297a3ec
```

Figure 7: 3 TCP at vayu and 3 TCP at norvig

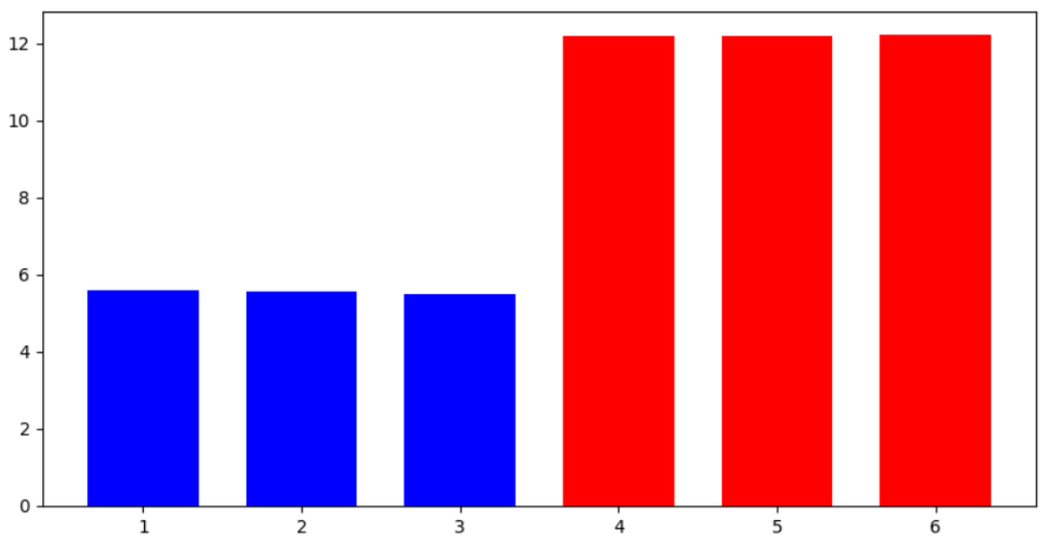
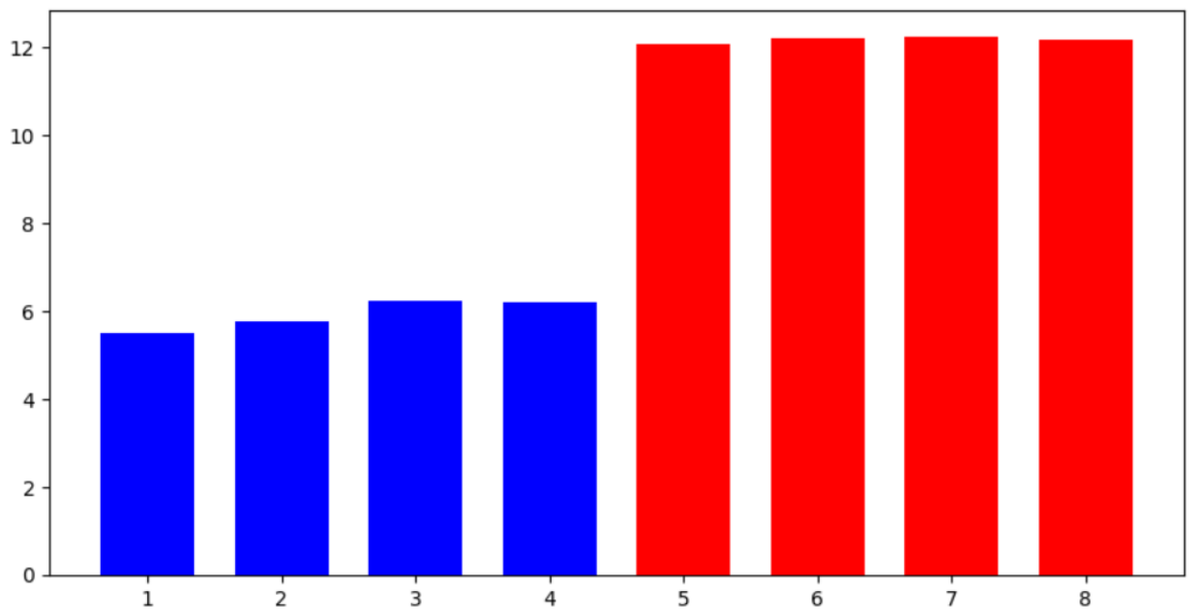


Figure 8: 4 TCP at vayu and 4 TCP at norvig

```
PS C:\Users\diwak\Desktop\COL\COL334\Assignments\Assignment3> java request
Get:    0-811083        vayu.iitd.ac.in
Get:    811084-1622167  vayu.iitd.ac.in
Get:    1622168-2433251         vayu.iitd.ac.in
Get:    2433252-3244335         vayu.iitd.ac.in
Get:    3244336-4055419         norvig.com
Get:    4055420-4866503         norvig.com
Get:    4866504-5677587         norvig.com
Get:    5677588-6488665         norvig.com
Received:       0-811083        vayu.iitd.ac.in
Received:       811084-1622167  vayu.iitd.ac.in
Received:       2433252-3244335         vayu.iitd.ac.in
Received:       1622168-2433251         vayu.iitd.ac.in
Received:       3244336-4055419         norvig.com
Received:       4055420-4866503         norvig.com
Received:       4866504-5677587         norvig.com
Received:       5677588-6488665         norvig.com

FILE DOWNLOADED: 656 Packets received
TOTAL TIME = 13.7419829
MD5 sum of origianl file:       70a4b9f4707d258f559f91615297a3ec
MD5 sum of downloaded file:     70a4b9f4707d258f559f91615297a3ec
```

Figure 9: 4 TCP at vayu and 4 TCP at norvig



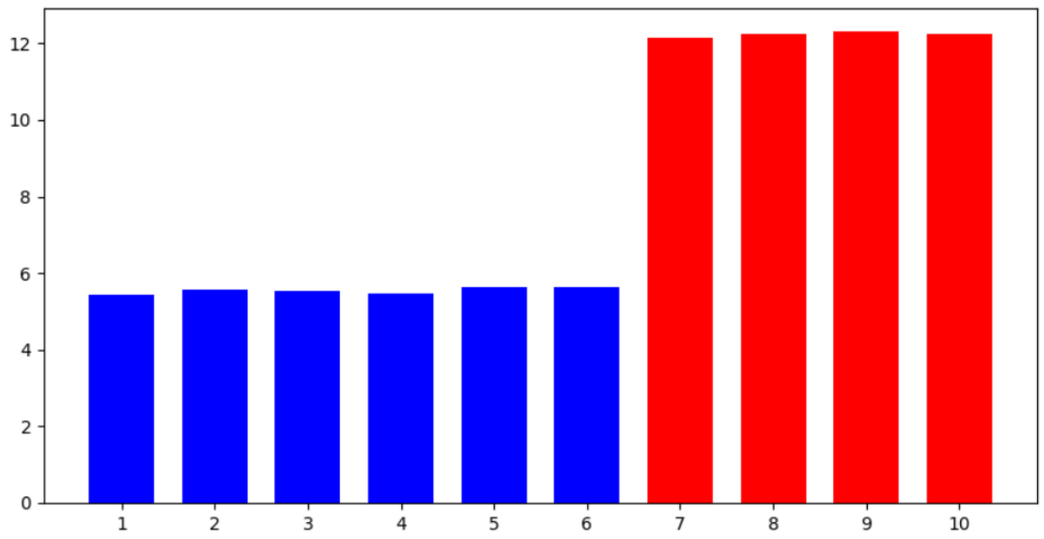
.

Figure 10: 6 TCP at vayu and 4 TCP at norvig

```
PS C:\Users\diwak\Desktop\COL\COL334\Assignments\Assignment3> java request
Get:    0-648866        vayu.iitd.ac.in
Get:    648867-1297733  vayu.iitd.ac.in
Get:    1297734-1946600         vayu.iitd.ac.in
Get:    1946601-2595467         vayu.iitd.ac.in
Get:    2595468-3244334         vayu.iitd.ac.in
Get:    3244335-3893201         vayu.iitd.ac.in
Get:    3893202-4542068         norvig.com
Get:    4542069-5190935         norvig.com
Get:    5190936-5839802         norvig.com
Get:    5839803-6488665         norvig.com
Received:       0-648866        vayu.iitd.ac.in
Received:       648867-1297733  vayu.iitd.ac.in
Received:       1946601-2595467         vayu.iitd.ac.in
Received:       1297734-1946600         vayu.iitd.ac.in
Received:       2595468-3244334         vayu.iitd.ac.in
Received:       3244335-3893201         vayu.iitd.ac.in
Received:       3893202-4542068         norvig.com
Received:       4542069-5190935         norvig.com
Received:       5190936-5839802         norvig.com
Received:       5839803-6488665         norvig.com

FILE DOWNLOADED: 650 Packets received
TOTAL TIME = 14.3479393
MD5 sum of origianl file:       70a4b9f4707d258f559f91615297a3ec
MD5 sum of downloaded file:     70a4b9f4707d258f559f91615297a3ec
```

Figure 11: 6 TCP at vayu and 4 TCP at norvig



We can see from above graphs that, time in downloading from norvig is much larger than downloading it from vayu, this means that bottleneck lies toward norvig.
In each case we can see the MD5 sum of downloaded file matches the original MD5 sir provided by sir. This means that our file have been correctly download.

4. Resistance to Internet disconnections.

By default, each TCP has a timeout of 15 mins, after which the tcp connection breaks if it does not get any response from the server, so we have to handle in in such a manner that, even if we the internet get disconnected of the server goes down.
I have done is by using try catch, when ever there is time out it raises and exception, I am catching this using try catch block. I try executes then we know that there was no time out, so I have made a boolean variable done, which is true when it has received all the data from current TCP connection. But if there had been time out then done will remain false. Now after joining all the threads, if check again if it is false or not. If it is false then I again send request to download the file.
My implementation is such that it is also resistant to multiple disconnections.