COL334 Assignment 3

Soham Gaikwad, 2018CS10394

November 2020

This assignment is done using the python programming language and invoking functions from the `socket` library. The final script is `TCPclient.py` and to check for MD5 sum, `check.py` can be used.

1. This was done by simply opening up a TCP connection using sockets and sending a single GET request to `vayu.iitd.ac.in` and then all bytes were received in one go and written to a file accordingly. The corresponding MD5 sum was matched with the original.

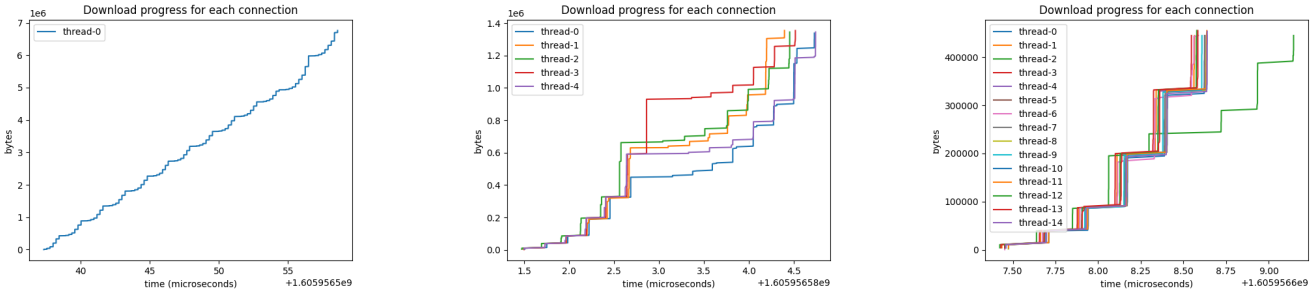
2. Parts of file can be downloaded by specifying the range in bytes in the GET header and then following the same technique as above. The response code is `HTTP 206 Partial Content` unlike the response code `HTTP 200 OK` as in the previous case. Also noticed that if invalid range is given in GET request then we get an error response.

3. To download the whole file in parts using parallel TCP connections, threading was used where chunks of file were pre-assigned to threads. An initial HEAD request is sent to find the content size of the object and then chunks are divided as uniformly as possible among threads.

   The download time seemed to be reducing in general with more number of TCP connections upto some limit. As can be seen in the following table, time reduced initially as multiple chunks were downloaded in parallel but later on it approached around 15 seconds, possibly due to the increased overhead of handling many threads on both client and server side. The following table shows the time for different number of TCP connections sampled over `vayu.iitd.ac.in`.

No. of connections	Time (seconds)
1	50 - 150
5	20 - 40
15	15 - 20
35	15 - 20
60	15 - 20

Table 1: TCP connections and time



(a) 1 TCP connection (b) 5 TCP connections (c) 15 TCP connections

Figure 1: Multiple TCP connections using multi-threading

As can be noticed in the above figures, the larger horizontal parts depict interruption in receiving due to slow internet connection or timeouts.

To spread the connections between vayu.iitd.ac.in and norvig.com, threads were assigned the servers alternatively.

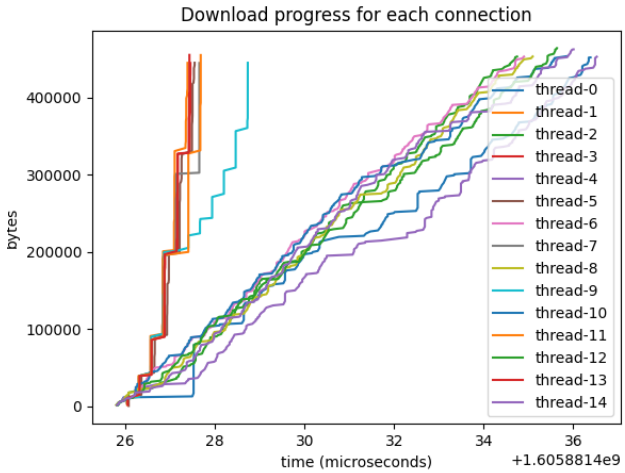


Figure 2: Spreading connections over servers

The even-numbered threads were assigned `vayu.iitd.ac.in` whereas the odd-numbered threads were assigned `norvig.com`. It is clear that the `norvig.com` server seems to be the faster one. This was faster than just downloading from `vayu.iitd.ac.in` but not much difference was observed when downloading just from `norvig.com`. As chunks are pre-assigned uniformly between threads, the program was not able to use the faster server more. But in most practical applications, we generally download from a single server except in case of torrents.

4. Whenever the connection breaks, the program catches the Exception and tries to reconnect. It also keeps track of how many chunks are downloaded till the connection was broken and then restarts downloading from where it was left by updating the corresponding parameters. A timeout of 15 or 20 seconds was set for the sockets to emulate network disconnections easily.