COL334 - Computer Networks
CSE, IIT Delhi (SEM1, 2020-21)

Assignment 3
Large File Download

2018CS10388
Sharique Shamim

1. Start with writing a simple program using TCP sockets that opens a TCP connection to the server, issues a GET request, downloads the entire content, and stores it in a file, in one go.

Solution: Python script is present in folder Q1. To run the script do

`python3 Q1.py vayu.iitd.ac.in`

`python3 Q1.py norvig.com`

First receive the header, if then recieve the file byte by byte until total bytes received become equal file size.

2. Now try downloading the file in parts, by sending an HTTP GET command to download a particular range of bytes, using the Range header in HTTP.

Solution: Python script is present in folder Q2. To run the script do

`python3 Q2.py vayu.iitd.ac.in <Content-Range>`

`python3 Q2.py norvig.com <Content-Range>`

`cat get-01.txt | ncat vayu.iitd.ac.in 80`

First receive the header then receive the file byte by byte until total bytes downloaded becomes equal to specified Content Range.

3. Issue multiple GET requests on the same TCP connection, one after the other.

Solution: I have allocated the chunks to each thread dynamically i.e when a thread completes a chunk download it queries a synchronized object to pick up a new chunk. Python script is present in folder Q3.

1. To download from one server with multiple parallel tcp connections and generate plot do:

   `python3 Q3.py <Hostname> <File Location> <Chunk Size> <Number of Parallel TCP Connections> <OutputFile>`

2. To download using parallel servers Vayu and Norvig do:

   `python3 Q3Mult.py <Chunk Size> <Number of Parallel TCP Connections to Each Server>`

3. To generate the plot for Time to Download vs Number of Parallel TCP Connections do:

   `python3 plot.py`

4. To download the file using `input.csv` do:

   `python3 main.py`

```
def getRequest():
    global start
    a = start
    b = start + chunkSize
    if b > fileSize:
        b = fileSize
        start = fileSize
        return a, b
    start + = chunkSize + 1
return a, b
```

I have created a function called `getRequest` which tells the thread what content range is to be downloaded. The threads ask for the byte range from this function. A target function for the threads called `sendRequest` downloads the file. To calculate the file size I downloaded 1 byte and parsed the Header to get the file size. The threads keep running until the number of bytes downloaded becomes equal to the file size. Inside this while loop, first I receive the messages till header then I start downloading the file byte by byte in chunks of 10 kB and store it in Priority Queue with priority as the lower limit of range. Once the download is complete I write in the file and calculate MD5Sum. I have also plotted bytes downloaded vs time elapsed for each connection.

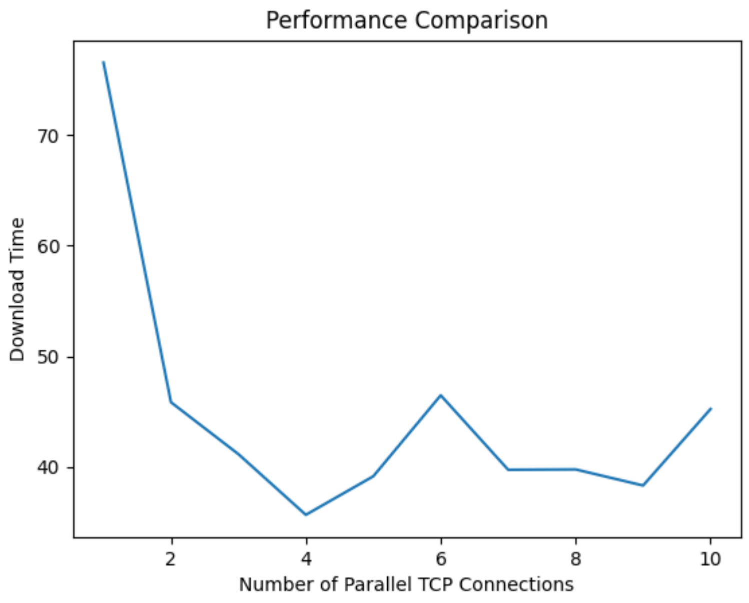
(a) Does your download time keep decreasing with more and more parallel TCP connections? Try to explain your finding.

Solution: Yes the download time decreases with increasing number of parallel TCP connections. Multiple chunks are downloaded at the same time from different connections which are then assembled to form the entire file thus reducing the overall download time. However, after a certain point it starts to increase, this is because of threading delay i.e all other threads will have to wait for a thread if it does not release a lock. The download time also depends on the internet connection. So if the internet connection is weak, the download time could increase even for more number of paralllel TCP connections.

(b) You can also log the download progress for each connection, and draw a graph that shows on the y-axis the number of bytes downloaded, and on the x-axis shows the time. You can check if some connections are faster than others, or some connections get stalled for some reason.

Solution: Plots can be seen below for multiple connections. From the plots we can observe that there is a lot of noise in the line. For each thread it can be seen that at times the slope of byte vs time increases and some times it decreases. Fluctuations are observed because a particular connection becomes faster whereas some other connection gets stalled for that duration.

(c) Does your download time reduce further if you download from different sources in parallel? What does this tell you about where bottlenecks lie? Is one server faster than the other? Is your program able to use this to download more from the faster server?



Solution: Vayu is faster than Norvig. For same number of total parallel connections download time is fastest for Vayu, then for Vayu and Norvig in Parallel and slowest for all connections to norvig. This is because Norvig's server takes a longer time to respond. Since Norvig is slow if all the threads for vayu complete downloading, they will have to wait for connections to norvig telling us the presence of bottleneck. In order to download faster from both servers in parallel make more connections to Vayu than Norvig.

4. Restructure your program so that in case a TCP connection breaks, the thread does not end but tries to open a new connection. Whenever a new connection succeeds, the thread resumes to download more chunks.

Solution: Python script is present in folder Q4. This part is resilient to disconnections.

1. To download from one server with multiple parallel tcp connections and generate plot do:

   ```
   python3 Q4.py <Hostname> <File Location> <Number of Parallel TCP Connections>
   <OutputFile>
   ```

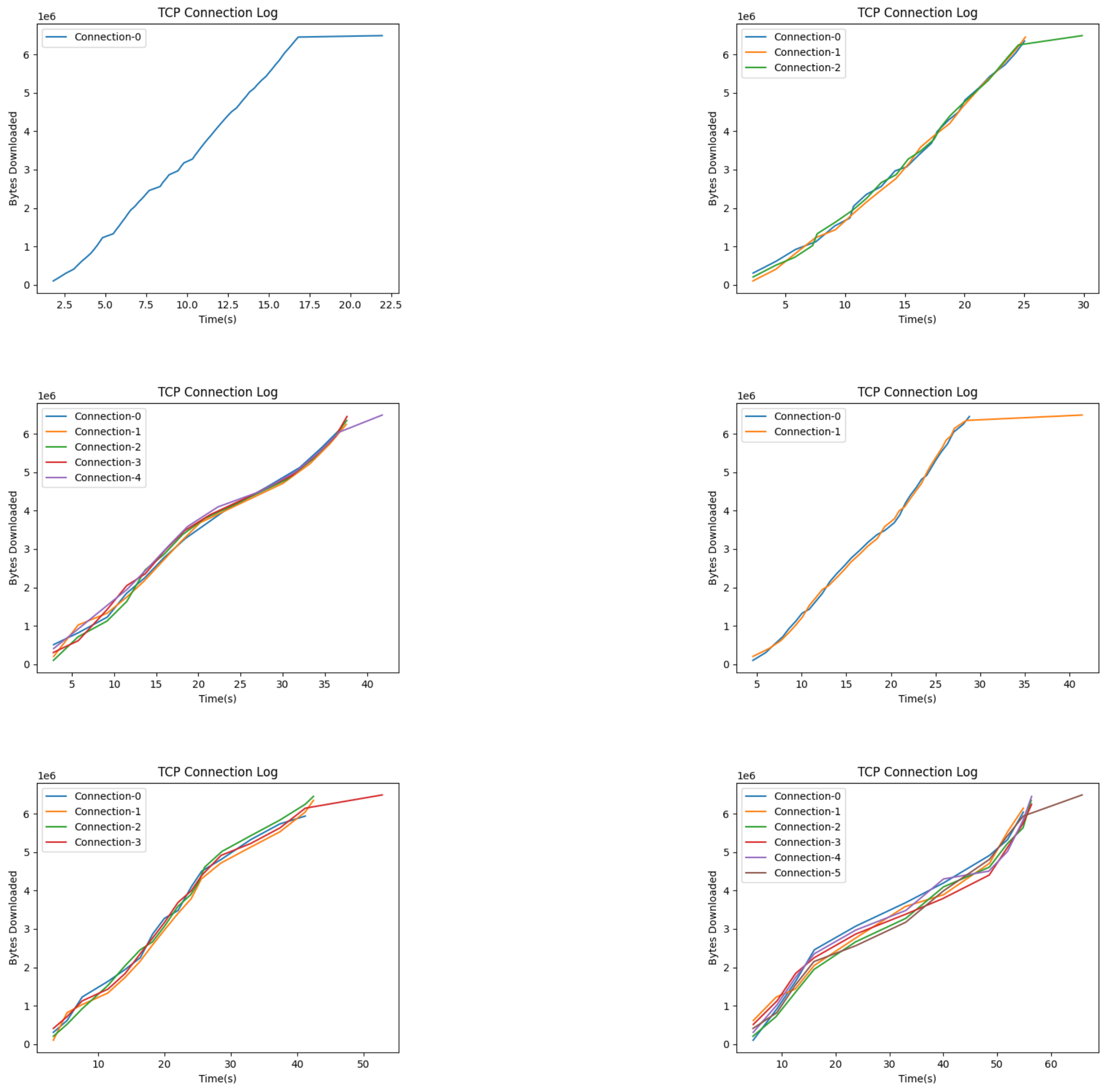
2. To download using parallel servers Vayu and Norvig do:

   ```
   python3 Q4Mult.py <Chunk Size> <Number of Parallel TCP Connections to Each
   Server>
   ```

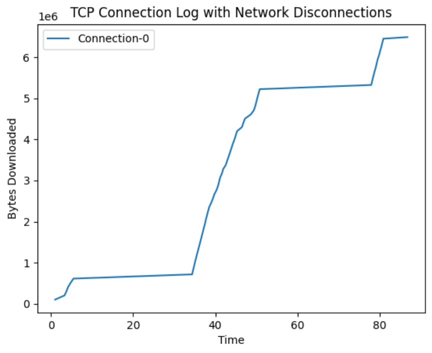
3. To download the file using `input.csv` do:

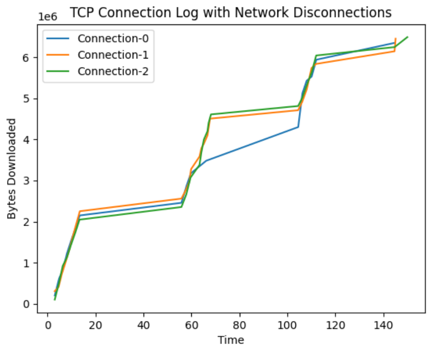
   ```
   python3 main.py
   ```

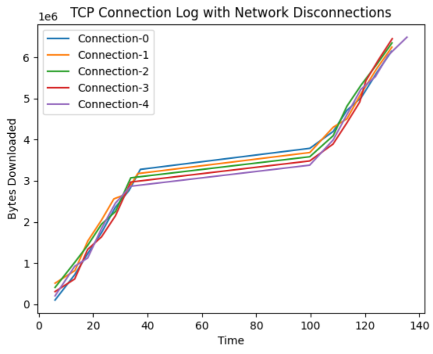
Whenever the network disrupts I handled it using try except. Whenever during the receiving the message or connecting to the server there is a network disruption close the previous socket. Create a new socket, Increase the timeout of the socket and while connection is not established with the server try connecting. Once connected, send the request and start downloading.

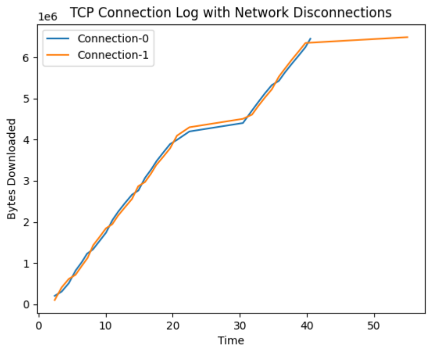


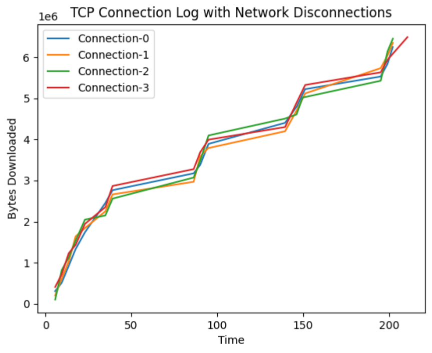
TCP Connection Log.

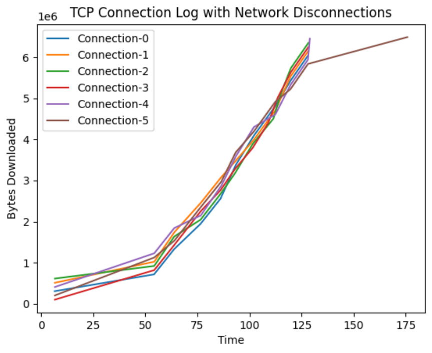












TCP Connection Log with Network Disruptions for around 20-40 seconds.