

CN project



**By:**

M Nouman Javaid

**CN Project**

**Task to Do/Requirements:**

1. Single Connection
2. Multiple Connection
3. Connections Protocols (TCP/UDP)
4. Local Server and Online Download
5. Resume Download
6. Speed and download bytes

**Note:** No HTTP libraries to be used

**Task Done:**

1. Single Connection
2. Multiple Connection
3. Connections Protocols (TCP only)

**Reason:**

HTTP uses TCP to provide reliable data transfer while UDP is not a reliable data transfer protocol.

1. Local Server Download and Online Download

**Local Server:**

Local Server used is Chrome web server. (An extension of chrome)

1. Resume Download
2. Speed and download bytes

**Note:** No HTTP libraries has been used

**Description:**

My file downloading process takes place an in number of steps:

**Step 1:** Get the size of the file to be download (1st HTTP request)

**Step 2:** Divide the file size on the number of chunks /connections provided by the user to get the chunk size

**Step 3:** Next critical step was to create an HTTP fetch request here I used a mathematical approach.Based on chunk size retrieved and no of chunks, I calculated the initial and final length, which will be provided as range of my http request (2nd HTTP request)

**Step 4:** After then, I start receiving data

In multi-connection, it was difficult to keep track of each thread downloaded data. So I start to record data corresponding to each thread file name. And also append these threaded file names in an array so later I can open them one after the other and store the whole data as a single downloaded file and then delete the rest chunks

**Step 5:** While receiving data I note down the time of receive and data downloaded, then using this information I calculated the download progress and speed of download

**Step 6:** If step 4 is blocked them a resume features has also been introduced to start downloading the same file from same index bit. The problem here was to find the data resume point. Again using mathematical approach I add downloaded bytes length to a range initial length. So when it starts again, it should start from the point where it left till the rest defined as final length. I-e

Range = (Initial + downlength) – final

**Step 7:** Clean up the left over or other minimal jobs