Name: Khushei Meghana Meda

SRN: PES1201800416

Week number: 2

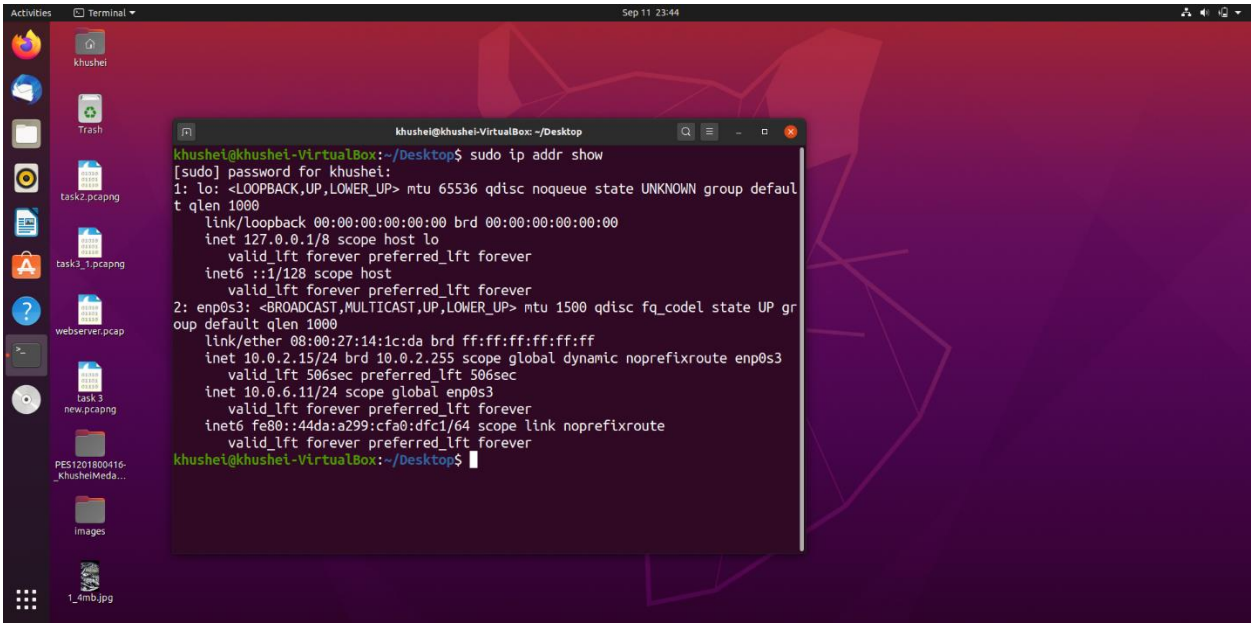
Name of experiment: HTTP Persistent and Non-Persistent Connections

Date: 14-09-2020

Objectives of the experiment: To understand persistent and non-persistent HTTP connections and corresponding performance impact.

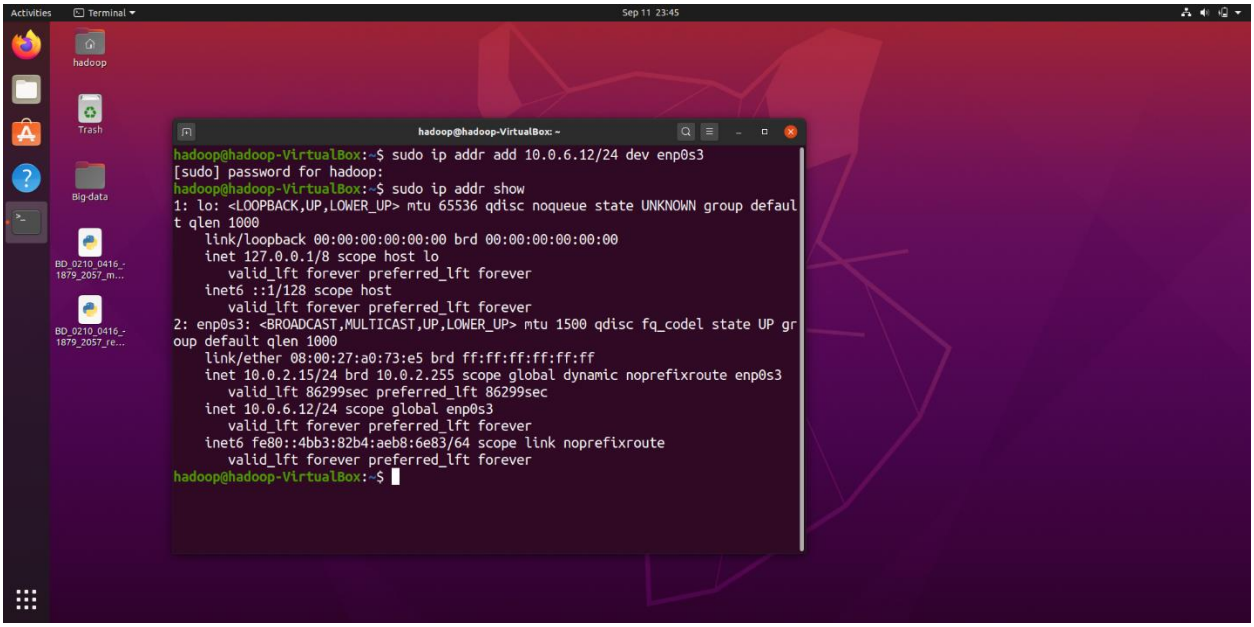
Server IP address set

The ip address 10.0.6.11 has been added to the server machine.



Client IP address set

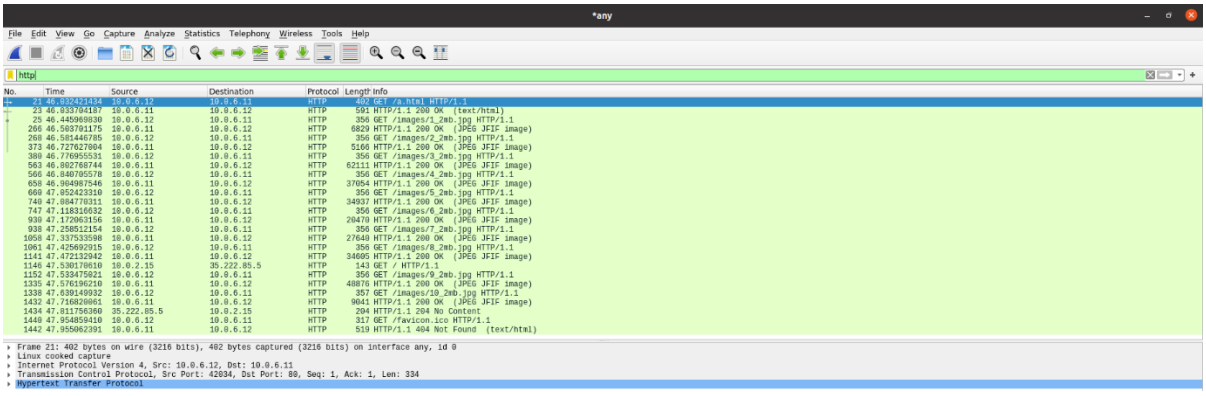
The ip address 10.0.6.12 has been added to the client machine.



For the below observations, I have loaded 10 images, each of which is approximately 2mb in size. The path to these images is given in the file named a.html.

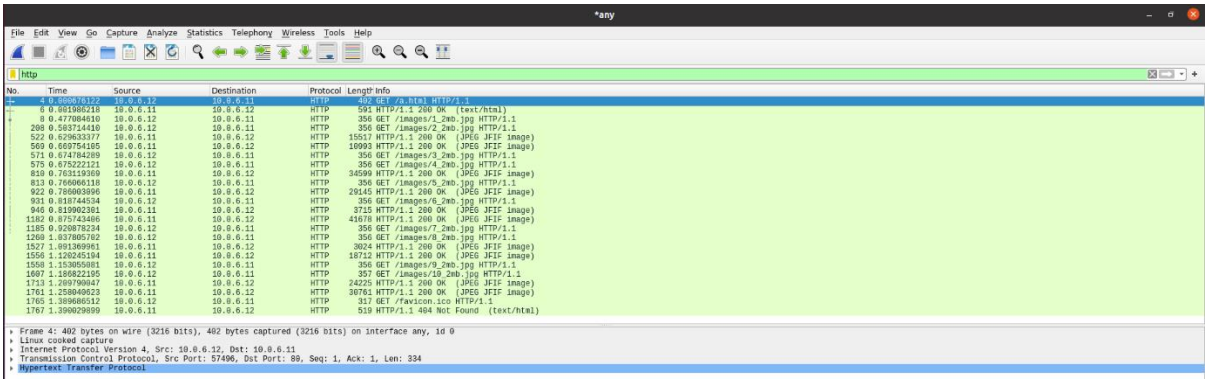
Non-persistent connection:

47.716820061 - 46.032421434 = 1.684398627



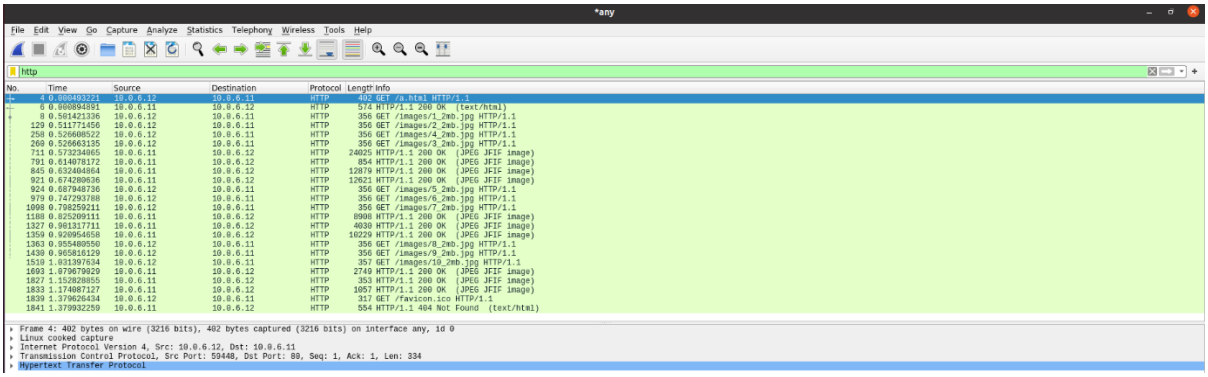
2 persistent connections:

1.258040623 - 0.000676122 = 1.257364501



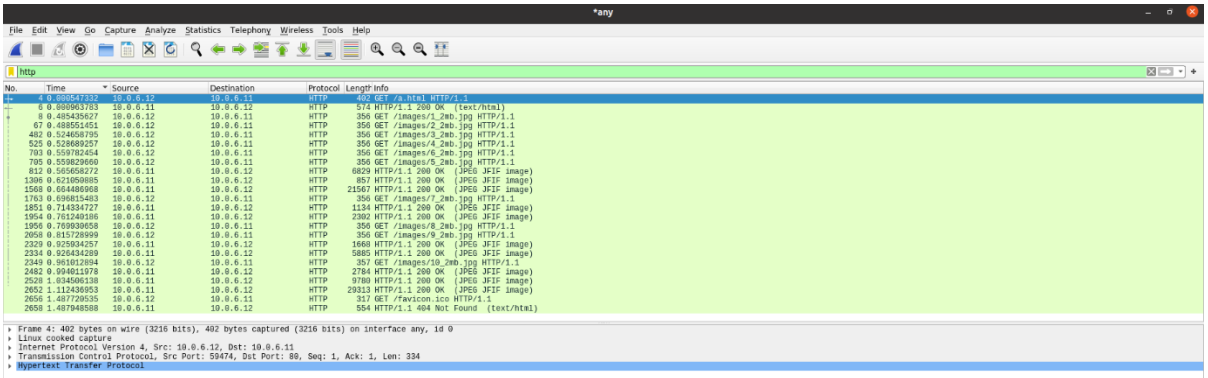
4 persistent connections:

1.174087127 - 0.000493221 = 1.173593906



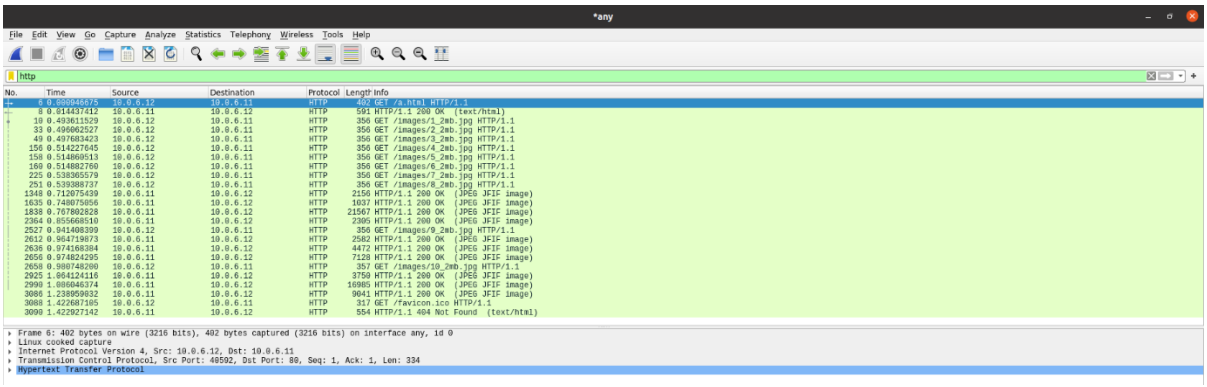
6 persistent connections:

1.112436953 - 0.000547332 = 1.111889621



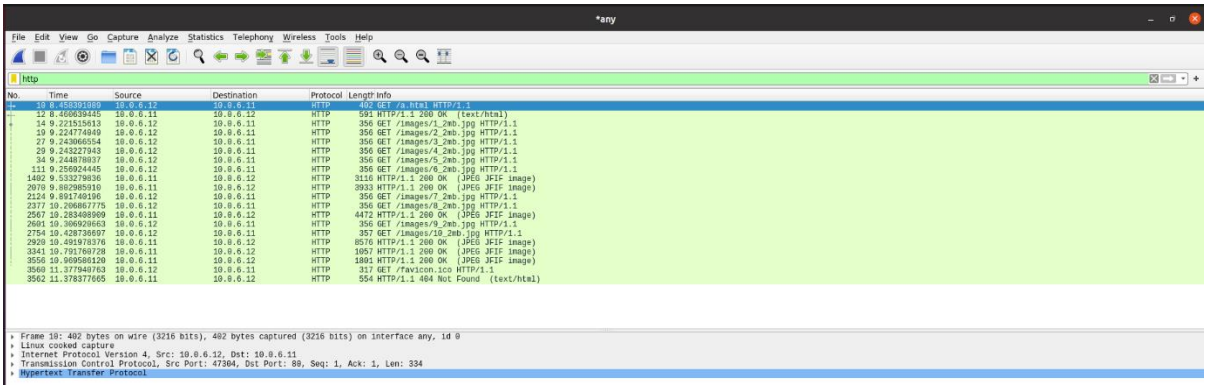
8 persistent connections:

1.238959032-0.000946675 = 1.237065682



10 persistent connections:

10.969586120-8.458391089 = 2.511195031



Observations:

No. of persistent connections	Time taken to display entire page
0	1.684398627
2	1.257364501
4	1.173593906
6	1.111889621
8	1.237065682
10	2.511195031

So, we notice that the time taken decreases up to 6 persistent connections and increases beyond that. 6 is the optimal number of persistent connections.

In its simplest implementation, a HTTP client creates a new TCP connection to the destination server, writes the request, and receives the response. The server then closes the TCP connection to release resources. Creating a new TCP connection requires a 'three-way handshake', and tearing it down also involves a two-way shutdown procedure. Repeatedly creating and closing TCP connections, one for each message, is inefficient. Using keepalive connections to hold open the TCP connection between the client and the server after a HTTP transaction has completed so that if the client needs to conduct another HTTP transaction, reduces the latency, as we notice up to 6 persistent connections.

However, after 6 persistent connections, the time increases. This could be because the size of each connection reduces and thus the throughput as well reduces. The number of optimal persistent connections is based on a collection of trade-offs: the higher the limit, the higher the client and server overhead, but at the additional benefit of higher request parallelism. Six connections per host is simply a safe middle ground.