Assignment 1

Computer Networks (CS301)

Part 1: Basic Networking Tools

Q1:

It shows information about various network interfaces on the system, along with their configuration details.

```
(base) satyamsangwan1@Satyams-MacBook-Pro ~ % ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
        nd6 options=201<PERFORMNUD,DAD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
anpi0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=400<CHANNEL_IO>
        ether 8e:9a:72:c2:77:bc
        inet6 fe80::8c9a:72ff:fec2:77bc%anpi0 prefixlen 64 scopeid 0x4
        nd6 options=201<PERFORMNUD,DAD>
        media: none
        status: inactive
anpi1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=400<CHANNEL_IO>
        ether 8e:9a:72:c2:77:bd
        inet6 fe80::8c9a:72ff:fec2:77bd%anpi1 prefixlen 64 scopeid 0x5
        nd6 options=201<PERFORMNUD,DAD>
        media: none
        status: inactive
en3: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=400<CHANNEL_IO>
        ether 8e:9a:72:c2:77:9c
        nd6 options=201<PERFORMNUD,DAD>
        media: none
        status: inactive
en4: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=400<CHANNEL_IO>
        ether 8e:9a:72:c2:77:9d
        nd6 options=201<PERFORMNUD,DAD>
        media: none
        status: inactive
en1: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 150
        options=460<TSO4,TSO6,CHANNEL_IO>
        ether 36:2d:6a:56:90:40
        media: autoselect <full-duplex>
        status: inactive
en2: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 150
        options=460<TSO4,TSO6,CHANNEL_IO>
        ether 36:2d:6a:56:90:44
        media: autoselect <full-duplex>
        status: inactive
```

When we run the ifconfig command in the terminal, it displays the information about the network interfaces and the information related to them, as shown in the above-attached image. Each interface shows the set of attributes it has, like flags, IP addresses, Mac addresses, packet count, flags, media type, subnet masks, and status. etc.

(1) Run the ifconfig command and briefly describe its output (important attribute).

- **Interface Name** (e.g., eth0, wlan0): Basically, it is the name of the interface.
- **MAC Address** (Ether): The MAC address is a unique hardware address assigned to the network interface. It consists of six pairs of hexadecimal numbers. (Note: Mac does not change.)
- **IPv4 and IPv6 Addresses**: The inet and inet6 lines show the assigned IPv4 and IPv6 addresses for the interface, along with associated subnet masks and prefix lengths.
- **MTU (Maximum Transmission Unit)**: The maximum size of the packet that can be transferred without breaking.
- **Flags**: Flags basically indicate the capabilities and the state of the interface, for example, “broadcast” - support broadcast communication.
- **Media Type**: Indicates the type of physical media the interface is connected to, such as Ethernet or wireless.
- **Status**: Indicates whether the interface is active or inactive.
- **RXCSUM** (Receive Checksum Offloading): The network card checks incoming data for errors instead of the CPU, reducing CPU workload.
- **TXCSUM** (Transmit Checksum Offloading): The network card calculates outgoing data checksums, reducing CPU usage.
- **TSO4** (TCP Segmentation Offload for IPv4): The network card breaks large outgoing data into smaller TCP segments for IPv4, improving performance
  .

- **TSO6** (TCP Segmentation Offload for IPv6): Same as TSO4 but for IPv6, enhancing efficiency for large data transfers.

(2) What options can be provided with the ifconfig command? Mention and explain at least four options.

- **up**: Activates a network interface. It makes the interface operational, allowing it to send and receive data.
- **down**: Deactivates a network interface. It takes the interface offline, preventing data transmission and reception.
- **inet <address> netmask <mask>**: Manually assigns an IPv4 address and subnet mask to the interface. For example, ifconfig eth0 inet 192.168.1.2 netmask 225.225.225.0.
- **inet6 <address> prefixlen <length>**: Manually assigns an IPv6 address and prefix length to the interface. For example, ifconfig eth0 inet6 2111:db8::1 prefixlen 64.
- -**s**: Display a short list instead of details.

Q2:

(1) What is the use of the netstat command?

- The netstat (network statistics) command is used to display a variety of network-related information and
- It provides insights into the following:
  I. Viewing Network Connections
  II. Monitoring Network Interfaces
  III. Displaying Routing Tables
  IV. Checking Network Services and Ports
  V. Viewing Network Protocol Statistics
  VI. Monitoring Network Performance
  VII. Tracking Network Traffic
  VIII. Identifying Network Usage

(2) Find all the active TCP ports on your system.

All the active TCP ports on the system.

```
(base) satyamsangwan1@Satyams-MacBook-Pro ~ % netstat -ap TCP
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  10.10.65.136.53192     server-3-160-185.https ESTABLISHED
tcp4       0      0  10.10.65.136.53092     13.107.42.14.https     ESTABLISHED
tcp4       0      0  10.10.65.136.52980     sa-in-f108.1e100.imaps ESTABLISHED
tcp4       0      0  10.10.65.136.52979     sa-in-f108.1e100.imaps ESTABLISHED
tcp4       0      0  10.10.65.136.52753     whatsapp-cdn-shv.https ESTABLISHED
tcp6       0      0  *.52749                *.*                    LISTEN
tcp4       0      0  *.52749                *.*                    LISTEN
tcp4       0      0  10.10.65.136.52729     sl-in-f188.1e100.https ESTABLISHED
tcp4       0      0  10.10.17.100.58649     sf-in-f108.1e100.imaps ESTABLISHED
tcp4       0      0  localhost.56647        localhost.56688        ESTABLISHED
tcp4       0      0  localhost.56688        localhost.56647        ESTABLISHED
tcp4       0      0  localhost.56649        localhost.56687        ESTABLISHED
tcp4       0      0  localhost.56687        localhost.56649        ESTABLISHED
tcp4       0      0  localhost.56645        localhost.56686        ESTABLISHED
tcp4       0      0  localhost.56686        localhost.56645        ESTABLISHED
tcp4       0      0  localhost.56646        localhost.56685        ESTABLISHED
tcp4       0      0  localhost.56685        localhost.56646        ESTABLISHED
tcp4       0      0  localhost.56646        localhost.56670        ESTABLISHED
tcp4       0      0  localhost.56670        localhost.56646        ESTABLISHED
tcp4       0      0  localhost.56649        localhost.56669        ESTABLISHED
tcp4       0      0  localhost.56669        localhost.56649        ESTABLISHED
tcp4       0      0  localhost.56648        *.*                    LISTEN
tcp4       0      0  localhost.56668        *.*                    LISTEN
tcp4       0      0  localhost.56646        *.*                    LISTEN
tcp4       0      0  localhost.56645        localhost.56667        ESTABLISHED
tcp4       0      0  localhost.56667        localhost.56645        ESTABLISHED
tcp4       0      0  localhost.56649        *.*                    LISTEN
tcp4       0      0  localhost.56647        *.*                    LISTEN
tcp4       0      0  localhost.56645        *.*                    LISTEN
tcp6       0      0  localhost.ddi-tcp-1    *.*                    LISTEN
tcp4       0      0  localhost.ddi-tcp-1    *.*                    LISTEN
tcp6       0      0  satyams-macbook-.black fe80::121d:f9cb:.1026  ESTABLISHED
tcp6       0      0  satyams-macbook-.1024  fe80::121d:f9cb:.1024  ESTABLISHED
tcp4       0      0  10.10.65.136.53187     ec2-52-206-172-3.https TIME_WAIT
tcp4       0      0  10.10.65.136.53188     ec2-52-206-172-3.https TIME_WAIT
tcp4       0      0  10.10.65.136.53190     server-3-160-185.https TIME_WAIT
tcp4       0      0  10.10.65.136.54223     17.57.145.118.5223     ESTABLISHED
```

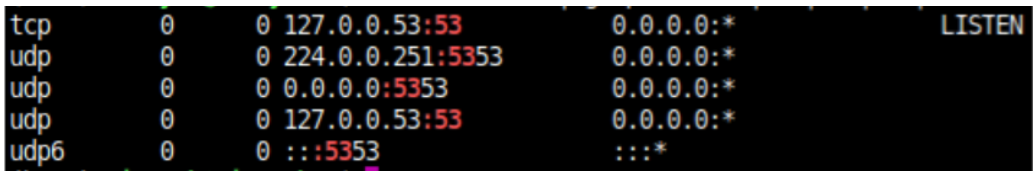
For the macOS netstat command, it is not working for the ports and the PIDs of the web browser. So, for that, I used the command “sudo lsof -iTCP -sTCP:ESTABLISHED”.

```
(base) satyamsangwan1@Satyams-MacBook-Pro ~ % sudo lsof -iTCP -sTCP:ESTABLISHED
Password:
COMMAND     PID           USER   FD   TYPE             DEVICE SIZE/OFF NODE NAME
identitys   630 satyamsangwan1   37u  IPv6 0xb9c9e426bdc11171      0t0  TCP satyams-macbo
identitys   630 satyamsangwan1   39u  IPv6 0xb9c9e426bdc10271      0t0  TCP satyams-macbo
identitys   630 satyamsangwan1   43u  IPv6 0xb9c9e426bdc10271      0t0  TCP satyams-macbo
Grammarly 26858 satyamsangwan1   15u  IPv4 0xb9c9e43057f37099      0t0  TCP 10.10.65.136:
Grammarly 26858 satyamsangwan1   35u  IPv4 0xb9c9e43057f37099      0t0  TCP 10.10.65.136:
python3.1 40063 satyamsangwan1   34u  IPv4 0xb9c9e430589a2589      0t0  TCP localhost:566
python3.1 40063 satyamsangwan1   35u  IPv4 0xb9c9e430589a3099      0t0  TCP localhost:566
python3.1 40063 satyamsangwan1   49u  IPv4 0xb9c9e43057a3da79      0t0  TCP localhost:566
python3.1 40063 satyamsangwan1   64u  IPv4 0xb9c9e43057a3f099      0t0  TCP localhost:566
python3.1 40063 satyamsangwan1   67u  IPv4 0xb9c9e43057a406b9      0t0  TCP localhost:566
python3.1 40063 satyamsangwan1   70u  IPv4 0xb9c9e43057a432f9      0t0  TCP localhost:566
Mail      40085 satyamsangwan1   33u  IPv4 0xb9c9e43057f34459      0t0  TCP 10.10.65.136:
Mail      40085 satyamsangwan1   37u  IPv4 0xb9c9e43057f34459      0t0  TCP 10.10.65.136:
Mail      40085 satyamsangwan1   40u  IPv4 0xb9c9e430589ae7e9      0t0  TCP 10.10.17.100:
Mail      40085 satyamsangwan1   61u  IPv4 0xb9c9e430589ac6b9      0t0  TCP 10.10.65.136:
Mail      40085 satyamsangwan1   63u  IPv4 0xb9c9e430589ac6b9      0t0  TCP 10.10.65.136:
python3.1 40426 satyamsangwan1   25u  IPv4 0xb9c9e43057f68f69      0t0  TCP localhost:566
python3.1 40426 satyamsangwan1   31u  IPv4 0xb9c9e430589a0459      0t0  TCP localhost:566
python3.1 40426 satyamsangwan1   64u  IPv4 0xb9c9e430589a0f69      0t0  TCP localhost:566
python3.1 40426 satyamsangwan1   66u  IPv4 0xb9c9e430589a3ba9      0t0  TCP localhost:566
python3.1 40426 satyamsangwan1   71u  IPv4 0xb9c9e43057a3e589      0t0  TCP localhost:566
python3.1 40426 satyamsangwan1   72u  IPv4 0xb9c9e43057a3fba9      0t0  TCP localhost:566
python3.1 40426 satyamsangwan1   77u  IPv4 0xb9c9e43057a411c9      0t0  TCP localhost:566
python3.1 40426 satyamsangwan1   78u  IPv4 0xb9c9e43057a41cd9      0t0  TCP localhost:566
Google    53956 satyamsangwan1   24u  IPv4 0xb9c9e430589a67e9      0t0  TCP 10.10.65.136:
Google    53956 satyamsangwan1   25u  IPv4 0xb9c9e43057f6b099      0t0  TCP 10.10.65.136:
Google    53956 satyamsangwan1   30u  IPv4 0xb9c9e430589a46b9      0t0  TCP 10.10.65.136:
Google    53956 satyamsangwan1   31u  IPv4 0xb9c9e430589ab099      0t0  TCP 10.10.65.136:
Google    53956 satyamsangwan1   40u  IPv4 0xb9c9e43057f7da79      0t0  TCP 10.10.65.136:
```

To find if any of the services running in my system uses the standard ports of HTTP, DHCP, DNS, SMTP, and FTP, use this command:

Here, the standard ports of HTTP, DHCP, DNS, SMTP, and FTP are 80, 67, 53, 25 and 21.

For that, use the command sudo lsof -iTCP -sTCP:LISTEN -n -P | grep -E ':(80|53|67|68|25|21) '



Q3:

(1) The ping command is used to test the reachability and round-trip time of a network host or device over an IP network.

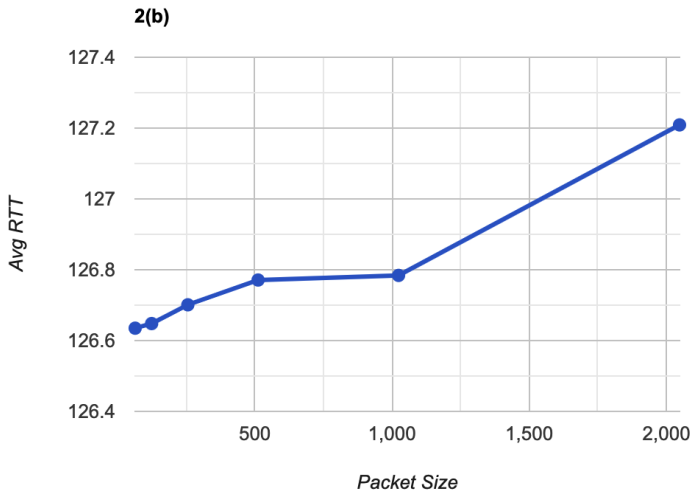
(2) a.

Host	RTT at 7 PM	RTT at 8 PM	RTT at 9 PM
www.google.com	7.195	7.912	7.092
www.instagram.com	157.598	157.171	157.950
www.youtube.com	120.451	118.396	125.396

RTT increases with the increase in geographical distance of the destination from the source.

b.

Packet Size	64	124	256	512	1024	2048
Avg RTT	126.635	126.648	126.701	126.771	126.784	127.209



c. We can clearly see from the above graph that with an increase in packet size, RTT increases.

Q4:

(1) The main use of a traceroute command is to trace the IP route from a source to a destination inside an IP network. A network traceroute shows the user the routers but also the round-trip latency from the source to each of the routers.

(2)

```
satyamsangwan1@Satyams-MacBook-Pro ~ % traceroute www.google.com
traceroute to www.google.com (142.250.182.228), 64 hops max, 52 byte packets
 1  10.10.16.1 (10.10.16.1)  4.780 ms  3.953 ms  3.969 ms
 2  static.ill.117.250.135.234.bsnl.co.in (117.250.135.234)  5.780 ms  7.215 ms  8.339 ms
 3  * * *
 4  * * *
 5  142.250.161.230 (142.250.161.230)  41.016 ms
    74.125.48.138 (74.125.48.138)  21.936 ms  23.314 ms
 6  * * *
 7  142.250.228.48 (142.250.228.48)  31.132 ms
    108.170.238.198 (108.170.238.198)  35.662 ms
    216.239.62.236 (216.239.62.236)  22.216 ms
 8  142.250.214.105 (142.250.214.105)  22.142 ms
    108.170.248.186 (108.170.248.186)  23.563 ms
    108.170.248.187 (108.170.248.187)  23.205 ms
 9  108.170.248.209 (108.170.248.209)  25.629 ms
    bom07s29-in-f4.1e100.net (142.250.182.228)  24.557 ms  22.848 ms
```

In the above images, I tried to trace the route of the host Google, but in lines 3, 4 and 6, there is * * *, which shows that the traceroute is not able to find complete paths to www.google.com.

The reasons are:

a. It may happen due to the Packet loss along the route. (Note: The packet loss may happen due to congestion in the network, hardware failures, or other issues.

b. It may also happen due to the maintenance of the network or experiencing outages, traceroute might be unable to set up complete paths to hosts in that area.

(3) Yes, it is possible to find the route to certain hosts that fail to respond with the ping experiment. Ping and traceroute are different commands. Ping uses the ICMP (Internet Control Message Protocol) to send echo request packets to the host and expects an echo reply in response. We can use the traceroute command that traces the route packets take from your computer to a destination host. It works by sending packets with increasing TTL values and noting the routers that send back ICMP "time exceeded" messages.

(4)

For www.youtube.com,

For www.google.com,

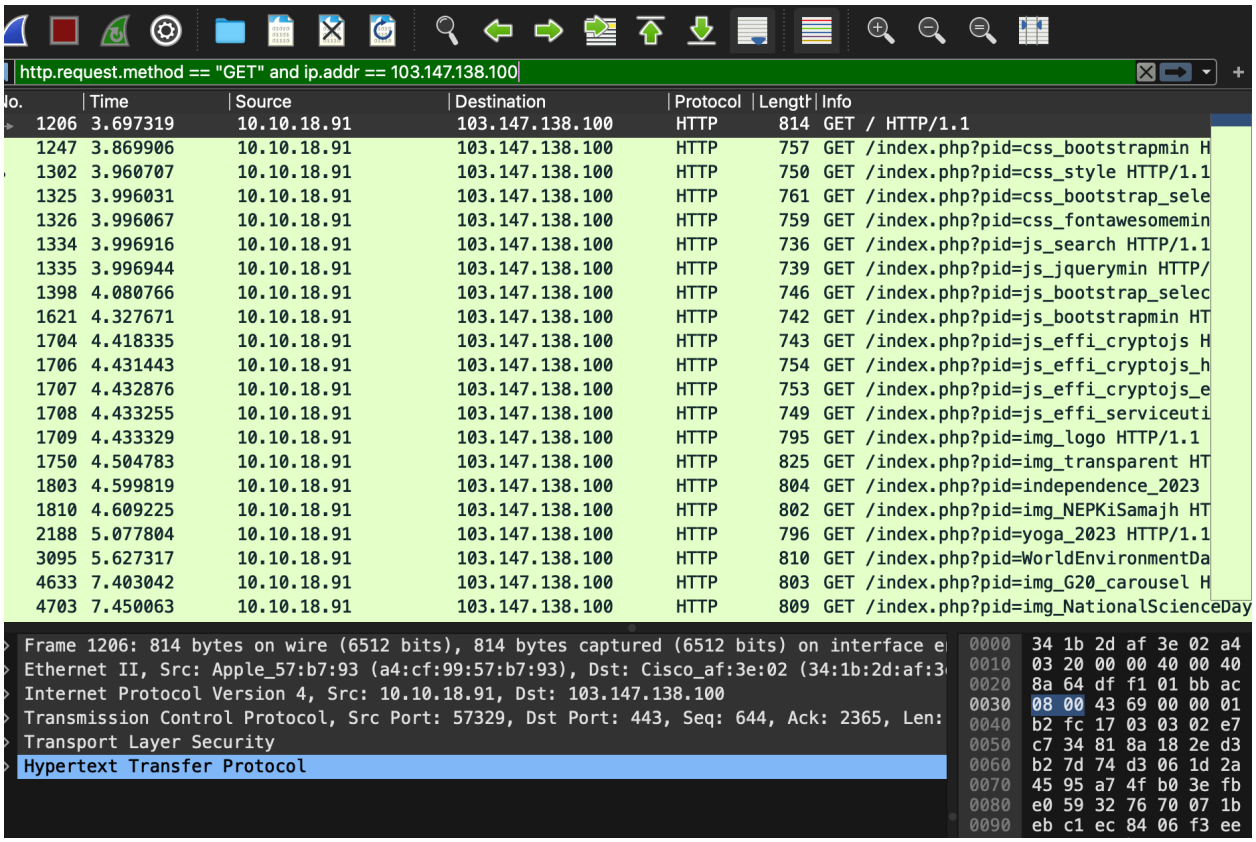
For www.instagram.com

Part 2: HTTP

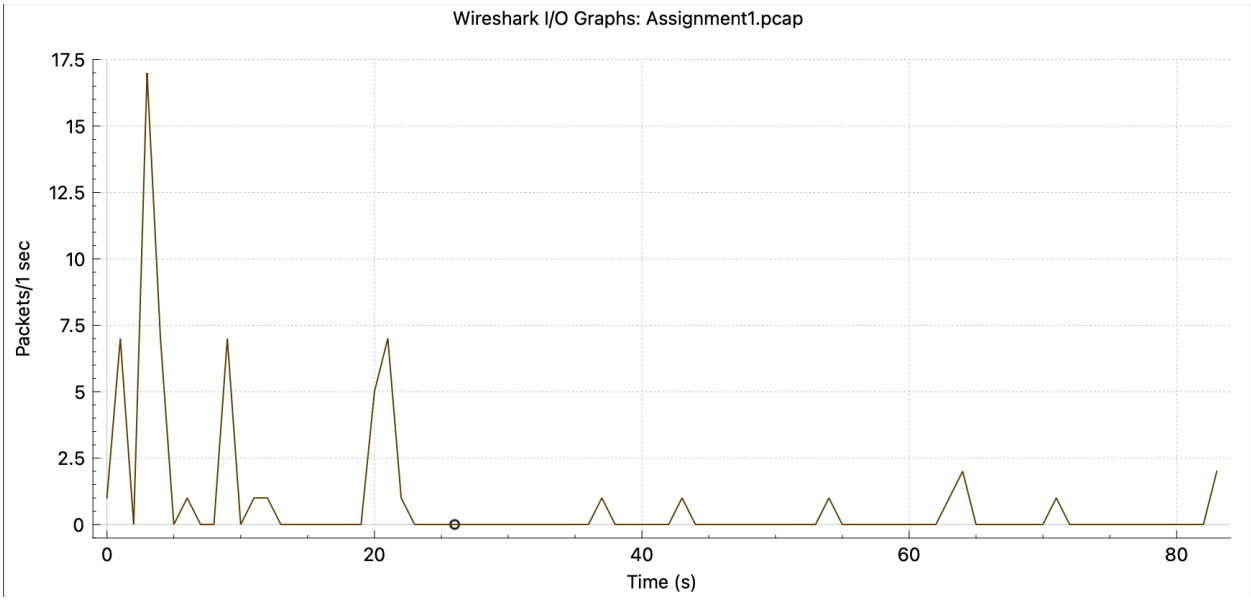
1) When you browse IIT Bhilai main page (https://www.iitbhilai.ac.in), how many GET request is sent (how many of the GET request are for embedded content, and how many GET request for the text)? Plot the IO graph for packets sent to iitbhilai.ac.in and packets received from iitbhilai.ac.in

   When I browse IIT Bhilai main page (https//www.iitbhilai.ac.in),

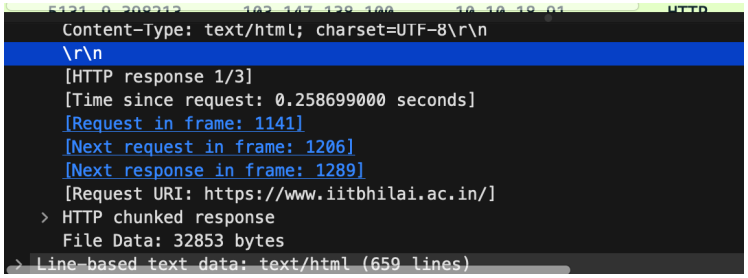
   a. It has 38 get requests sent.
   b. Which has 25 for embedded content (images)
   c. 13 are the script/styles (text)



The Plot of the IO graph is below:



2)For each HTTP GET request, as you see above, find out (i) the total amount of data being received in the corresponding HTTP response message



- GET / HTTP/1.1 - **32853 bytes**
- GET /index.php?pid=css_bootstrapmin HTTP/1.1 - **121033 bytes**
- GET /index.php?pid=css_style HTTP/1.1 - **19232 bytes**
- GET /index.php?pid=css_bootstrap_select HTTP/1.1 - **6065 bytes**
- GET /index.php?pid=css_fontawesomemin HTTP/1.1 - **31004 bytes**
- GET /index.php?pid=js_jquerymin HTTP/1.1 - **116840 bytes**

- GET /index.php?pid=js_search HTTP/1.1 - **379 bytes**
- GET /index.php?pid=js_bootstrap_select HTTP/1.1 - **31697 bytes**
- GET /index.php?pid=js_bootstrapmin HTTP/1.1 - **37045 bytes**
- GET /index.php?pid=js_effi_cryptojs HTTP/1.1 - **47944 bytes**
- GET /index.php?pid=js_effi_cryptojs_hmacsha256 HTTP/1.1 - **302 bytes**
- GET /index.php?pid=js_effi_cryptojs_encbase64 HTTP/1.1 - **1100 bytes**
- GET /index.php?pid=WorldEnvironmentDay2023 HTTP/1.1 - **No response**
- GET /index.php?pid=news1 HTTP/1.1 - **no response**
- GET /index.php?pid=img_NationalConference HTTP/1.1 - **no response**
- GET /index.php?pid=yoga_2023 HTTP/1.1 - **no response**
- GET /index.php?pid=img_G20_carousel HTTP/1.1 - **no response**
- GET /index.php?pid=img_NationalScienceDay HTTP/1.1 - **no response**

Similarly, this can be done for the other GET requests.

3)For the response to your HTTP GET request, get the image reconstructed by the hex editor.

4)

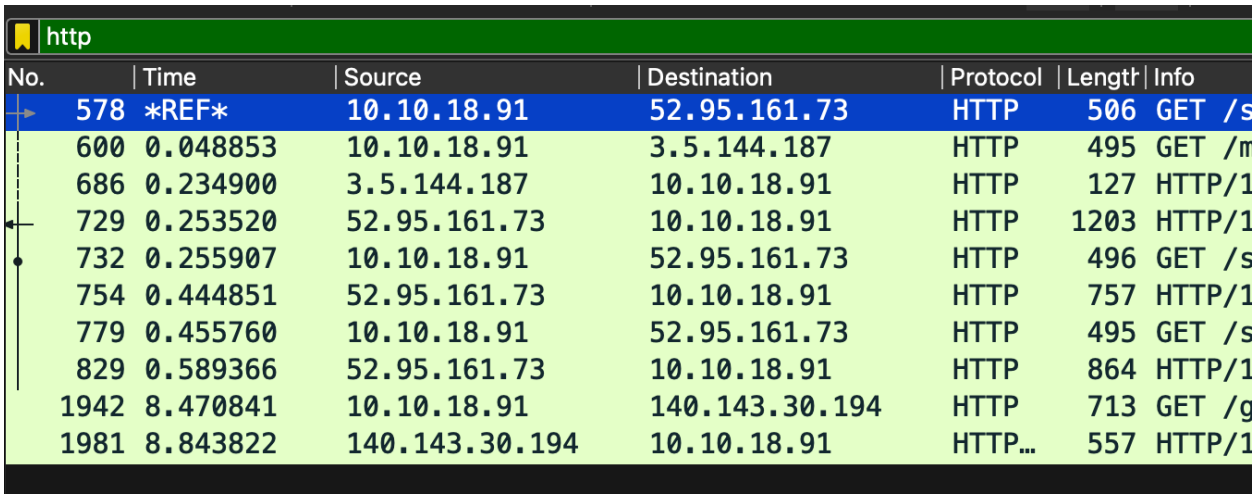
a. In this part, we have to inspect the content of the first HTTP GET request from the browser to the server and check if it has the “IF-MODIFIED-SINCE” line :
No. I don’t see any “IF-MODIFIED-SINCE” line in the first HTTP GET request.

b. In this part of the question, we have to inspect the contents of the server response:
Yes, the server explicitly returned the contents of the file, and this is because it shows the Line-based text data in the server response.

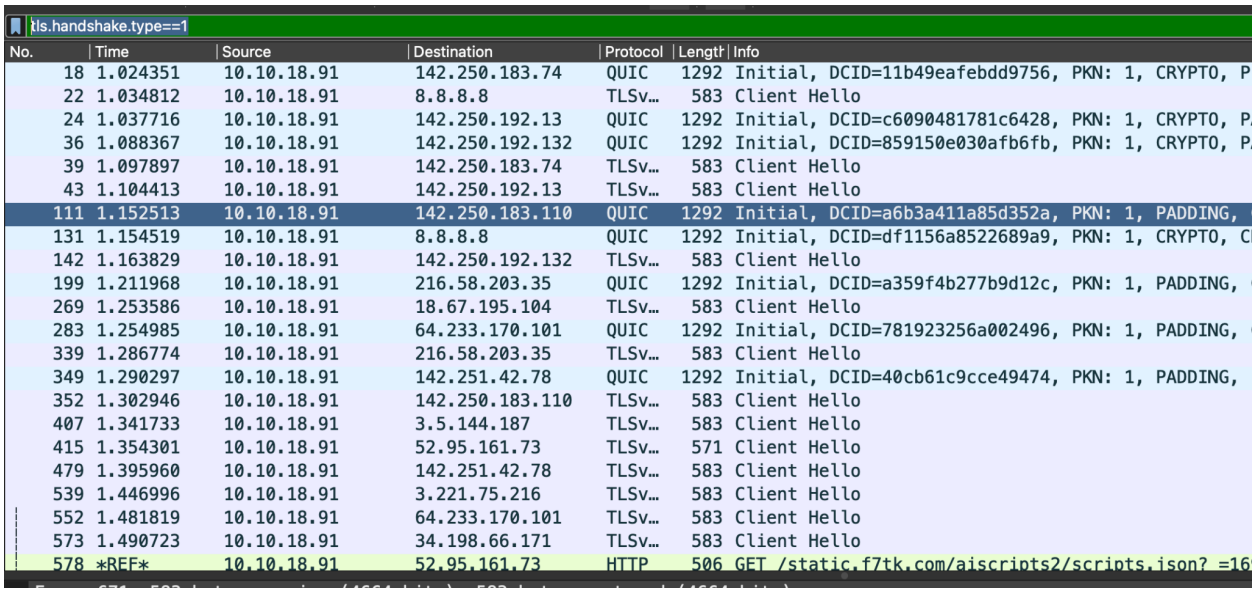
c. In this part, we have to write about the information of “IF-MODIFIED-SINCE”:
to ask a web server for a resource only if it has been modified since a specific date and time. This header helps optimise network usage by reducing unnecessary data transfer, as the server can respond with the resource only if it has been updated since the specified time; otherwise, it returns a "Not Modified" status, indicating that the client can use its cached copy.

d. In this part, we have to write about the status code and phrase returned from the server in the response to the second HTTP GET:
The HTTP GET is 200 and OK. No, the server did not return the contents of the file because Line-based text data is not present in the response.

5) the solution to this part is as follows:



- It took 8.84382 ms to load the facebook.com page, as mentioned in the about images. This is the difference between the time intervals when the first request is sent from client to server, and the last response is received from server to client.



- In this, **32** connections are used to download this page because **32** times the handshake protocol was followed.
- These connections are persistent because for every time a connection is made, multiple packets are transferred between the server and the client. 5 objects have been transferred between these connections, which can be calculated by applying filter “http and ip.dst==client_ip_address” in Wireshark.

Part 3:

For www.iitbhilai.ac.in

1. Firstly, I run the command **dig . NS** then outputs the information of the root servers. (for www.iitbhilai.ac.in)

- NS: This is the type of DNS record you are querying for. NS stands for "Name Server," and it's used to identify the authoritative name servers responsible for a domain.
- .: The dot represents the root domain of the DNS hierarchy. In DNS, the root domain is the top-level domain from which all other domains branch out.
- dig: This is the command itself that invokes the dig tool.

```
satyamsangwan1@Satyams-MacBook-Pro ~ % dig . NS

; <<>> DiG 9.10.6 <<>> . NS
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35618
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       77622   IN      NS      h.root-servers.net.
.                       77622   IN      NS      i.root-servers.net.
.                       77622   IN      NS      j.root-servers.net.
.                       77622   IN      NS      k.root-servers.net.
.                       77622   IN      NS      l.root-servers.net.
.                       77622   IN      NS      m.root-servers.net.
.                       77622   IN      NS      a.root-servers.net.
.                       77622   IN      NS      b.root-servers.net.
.                       77622   IN      NS      c.root-servers.net.
.                       77622   IN      NS      d.root-servers.net.
.                       77622   IN      NS      e.root-servers.net.
.                       77622   IN      NS      f.root-servers.net.
.                       77622   IN      NS      g.root-servers.net.
```

2. In the second step, run **dig @e.root-servers.net. www.iitbhilai.ac.in . NS +norec**, which gives the information about the registry, for example, ns1.registry.in.

```
satyamsangwan1@Satyams-MacBook-Pro ~ % dig @e.root-servers.net. www.iitbhilai.ac.in . NS +norec

; <<>> DiG 9.10.6 <<>> @e.root-servers.net. www.iitbhilai.ac.in . NS +norec
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43652
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 13
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.iitbhilai.ac.in.           IN      A

;; AUTHORITY SECTION:
in.                     172800  IN      NS      ns1.registry.in.
in.                     172800  IN      NS      ns2.registry.in.
in.                     172800  IN      NS      ns3.registry.in.
in.                     172800  IN      NS      ns4.registry.in.
in.                     172800  IN      NS      ns5.registry.in.
in.                     172800  IN      NS      ns6.registry.in.

;; ADDITIONAL SECTION:
ns1.registry.in.        172800  IN      A       37.209.192.12
ns2.registry.in.        172800  IN      A       37.209.194.12
ns3.registry.in.        172800  IN      A       37.209.196.12
ns4.registry.in.        172800  IN      A       37.209.198.12
ns5.registry.in.        172800  IN      A       156.154.100.20
ns6.registry.in.        172800  IN      A       156.154.101.20
ns1.registry.in.        172800  IN      AAAA    2001:dcd:1::12
ns2.registry.in.        172800  IN      AAAA    2001:dcd:2::12
ns3.registry.in.        172800  IN      AAAA    2001:dcd:3::12
ns4.registry.in.        172800  IN      AAAA    2001:dcd:4::12
ns5.registry.in.        172800  IN      AAAA    2001:502:2eda::20
ns6.registry.in.        172800  IN      AAAA    2001:502:ad09::20
```

3. In the third step, I run the command **dig @ns1.registry.in. www.iitbhilai.ac.in . NS +norec,** which gives the information about the DNS.

```
satyamsangwan1@Satyams-MacBook-Pro ~ % dig @ns1.registry.in. www.iitbhilai.ac.in . NS +norec,
Invalid option: +norec,
Usage:  dig [@global-server] [domain] [q-type] [q-class] {q-opt}
            {global-d-opt} host [@local-server] {local-d-opt}
            [ host [@local-server] {local-d-opt} [...]]

Use "dig -h" (or "dig -h | more") for complete list of options
satyamsangwan1@Satyams-MacBook-Pro ~ % dig @ns1.registry.in. www.iitbhilai.ac.in . NS +norec

; <<>> DiG 9.10.6 <<>> @ns1.registry.in. www.iitbhilai.ac.in . NS +norec
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61816
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 3
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;www.iitbhilai.ac.in.           IN      A

;; AUTHORITY SECTION:
iitbhilai.ac.in.        3600    IN      NS      dns2.iitbhilai.ac.in.
iitbhilai.ac.in.        3600    IN      NS      dns1.iitbhilai.ac.in.

;; ADDITIONAL SECTION:
dns2.iitbhilai.ac.in.   3600    IN      A       103.147.138.111
dns1.iitbhilai.ac.in.   3600    IN      A       103.147.138.110
```

4. In the fourth step, I run the command **dig @dns1.iitbhilai.ac.in. www.iitbhilai.ac.in . NS +norec,** after running this command, I finally got the IP address of www.iitbhilai.ac.in, which is 103.147.138.100.

```
satyamsangwan1@Satyams-MacBook-Pro ~ %  dig @dns1.iitbhilai.ac.in. www.iitbhilai.ac.in . NS +norec

; <<>> DiG 9.10.6 <<>> @dns1.iitbhilai.ac.in. www.iitbhilai.ac.in . NS +norec
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23694
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.iitbhilai.ac.in.           IN      A

;; ANSWER SECTION:
www.iitbhilai.ac.in.    10800   IN      A       103.147.138.100

;; AUTHORITY SECTION:
iitbhilai.ac.in.        10800   IN      NS      dns1.iitbhilai.ac.in.

;; ADDITIONAL SECTION:
dns1.iitbhilai.ac.in.   10800   IN      A       103.147.138.110
```

For www.facebook.com

1. Firstly, I run the command **dig . NS** then outputs the information of the root servers. (for www.facebook.com)

- NS: This is the type of DNS record you are querying for. NS stands for "Name Server," and it's used to identify the authoritative name servers responsible for a domain.
- .: The dot represents the root domain of the DNS hierarchy. In DNS, the root domain is the top-level domain from which all other domains branch out.
- dig: This is the command itself that invokes the dig tool.

```
satyamsangwan1@Satyams-MacBook-Pro ~ % dig . NS

; <<>> DiG 9.10.6 <<>> . NS
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38619
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       56439   IN      NS      g.root-servers.net.
.                       56439   IN      NS      j.root-servers.net.
.                       56439   IN      NS      e.root-servers.net.
.                       56439   IN      NS      l.root-servers.net.
.                       56439   IN      NS      d.root-servers.net.
.                       56439   IN      NS      a.root-servers.net.
.                       56439   IN      NS      b.root-servers.net.
.                       56439   IN      NS      i.root-servers.net.
.                       56439   IN      NS      m.root-servers.net.
.                       56439   IN      NS      h.root-servers.net.
.                       56439   IN      NS      c.root-servers.net.
.                       56439   IN      NS      k.root-servers.net.
.                       56439   IN      NS      f.root-servers.net.
```

2. In the second step, run **dig @e.root-servers.net. www.facebook.com . NS +norec**, which gives the information about the next part.

```
satyamsangwan1@Satyams-MacBook-Pro ~ % dig @e.root-servers.net. www.facebook.com . NS +norec

; <<>> DiG 9.10.6 <<>> @e.root-servers.net. www.facebook.com . NS +norec
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35389
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.facebook.com.              IN      A

;; AUTHORITY SECTION:
com.                    172800  IN      NS      a.gtld-servers.net.
com.                    172800  IN      NS      b.gtld-servers.net.
com.                    172800  IN      NS      c.gtld-servers.net.
com.                    172800  IN      NS      d.gtld-servers.net.
com.                    172800  IN      NS      e.gtld-servers.net.
com.                    172800  IN      NS      f.gtld-servers.net.
com.                    172800  IN      NS      g.gtld-servers.net.
com.                    172800  IN      NS      h.gtld-servers.net.
com.                    172800  IN      NS      i.gtld-servers.net.
com.                    172800  IN      NS      j.gtld-servers.net.
com.                    172800  IN      NS      k.gtld-servers.net.
com.                    172800  IN      NS      l.gtld-servers.net.
com.                    172800  IN      NS      m.gtld-servers.net.
```

3. In the third step, I run the command **dig @a.gtld-servers.net. www.facebook.com. NS +norec,** which gives the information about the next part.

```
satyamsangwan1@Satyams-MacBook-Pro ~ %  dig @a.gtld-servers.net. www.facebook.com. NS +norec

; <<>> DiG 9.10.6 <<>> @a.gtld-servers.net. www.facebook.com. NS +norec
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43039
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.facebook.com.              IN      NS

;; AUTHORITY SECTION:
facebook.com.           172800  IN      NS      a.ns.facebook.com.
facebook.com.           172800  IN      NS      b.ns.facebook.com.
facebook.com.           172800  IN      NS      c.ns.facebook.com.
facebook.com.           172800  IN      NS      d.ns.facebook.com.

;; ADDITIONAL SECTION:
a.ns.facebook.com.      172800  IN      A       129.134.30.12
a.ns.facebook.com.      172800  IN      AAAA    2a03:2880:f0fc:c:face:b00c:0:35
b.ns.facebook.com.      172800  IN      A       129.134.31.12
b.ns.facebook.com.      172800  IN      AAAA    2a03:2880:f0fd:c:face:b00c:0:35
c.ns.facebook.com.      172800  IN      A       185.89.218.12
c.ns.facebook.com.      172800  IN      AAAA    2a03:2880:f1fc:c:face:b00c:0:35
d.ns.facebook.com.      172800  IN      A       185.89.219.12
d.ns.facebook.com.      172800  IN      AAAA    2a03:2880:f1fd:c:face:b00c:0:35
```

4. In the fourth step, I run the command **dig @a.ns.facebook.com. www.facebook.com . NS +norec,** after running this command, I finally refused the IP address.

For www.stanford.edu

1. Firstly, I run the command **dig . NS** then outputs the information of the root servers. (for www.stanford.edu)

```
satyamsangwan1@Satyams-MacBook-Pro ~ % dig . NS

; <<>> DiG 9.10.6 <<>> . NS
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63778
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       40319   IN      NS      c.root-servers.net.
.                       40319   IN      NS      d.root-servers.net.
.                       40319   IN      NS      e.root-servers.net.
.                       40319   IN      NS      f.root-servers.net.
.                       40319   IN      NS      g.root-servers.net.
.                       40319   IN      NS      h.root-servers.net.
.                       40319   IN      NS      i.root-servers.net.
.                       40319   IN      NS      j.root-servers.net.
.                       40319   IN      NS      k.root-servers.net.
.                       40319   IN      NS      l.root-servers.net.
.                       40319   IN      NS      m.root-servers.net.
.                       40319   IN      NS      a.root-servers.net.
.                       40319   IN      NS      b.root-servers.net.
```

2. In the second step, run **dig @a.root-servers.net. www.stanford.edu . NS +norec**, which gives the information about the next part.

```
satyamsangwan1@Satyams-MacBook-Pro ~ %  dig @a.root-servers.net. www.stanford.edu . NS +norec

; <<>> DiG 9.10.6 <<>> @a.root-servers.net. www.stanford.edu . NS +norec
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18710
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1472
;; QUESTION SECTION:
;www.stanford.edu.              IN      A

;; AUTHORITY SECTION:
edu.                    172800  IN      NS      a.edu-servers.net.
edu.                    172800  IN      NS      b.edu-servers.net.
edu.                    172800  IN      NS      c.edu-servers.net.
edu.                    172800  IN      NS      d.edu-servers.net.
edu.                    172800  IN      NS      e.edu-servers.net.
edu.                    172800  IN      NS      f.edu-servers.net.
edu.                    172800  IN      NS      g.edu-servers.net.
edu.                    172800  IN      NS      h.edu-servers.net.
edu.                    172800  IN      NS      i.edu-servers.net.
edu.                    172800  IN      NS      j.edu-servers.net.
edu.                    172800  IN      NS      k.edu-servers.net.
edu.                    172800  IN      NS      l.edu-servers.net.
edu.                    172800  IN      NS      m.edu-servers.net.
```

3. In the third step, I run the command **dig @a.edu-servers.net. www.stanford.edu . NS +norec,** which gives the information about the next part.

```
satyamsangwan1@Satyams-MacBook-Pro ~ % dig @a.edu-servers.net. www.stanford.edu . NS +norec

; <<>> DiG 9.10.6 <<>> @a.edu-servers.net. www.stanford.edu . NS +norec
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52066
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 7
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.stanford.edu.              IN      A

;; AUTHORITY SECTION:
stanford.edu.           172800  IN      NS      avallone.stanford.edu.
stanford.edu.           172800  IN      NS      atalante.stanford.edu.
stanford.edu.           172800  IN      NS      argus.stanford.edu.
stanford.edu.           172800  IN      NS      ns5.dnsmadeeasy.com.
stanford.edu.           172800  IN      NS      ns6.dnsmadeeasy.com.
stanford.edu.           172800  IN      NS      ns7.dnsmadeeasy.com.

;; ADDITIONAL SECTION:
avallone.stanford.edu.  172800  IN      A       204.63.224.53
avallone.stanford.edu.  172800  IN      AAAA    2620:6c:40c0:0:204:63:224:53
atalante.stanford.edu.  172800  IN      A       171.64.7.61
atalante.stanford.edu.  172800  IN      AAAA    2607:f6d0:0:d32::ab40:73d
argus.stanford.edu.     172800  IN      A       171.64.7.115
argus.stanford.edu.     172800  IN      AAAA    2607:f6d0:0:9113::ab40:773
```

4. In the fourth step, I run the command **dig @ avallone.stanford.edu. www.stanford.edu . NS +norec**