

There are 4 questions.

---

1. (Network analysis)

- (a) At your home (not hostel room which is on the IITD network) or from a 4G or other cellular connection, run traceroute via your Ethernet and WiFi networks for `www.iitd.ac.in`, and note the IP addresses seen on the path. If your ISP seems to be blocking packets on the path to the `www.iitd.ac.in` network then try with different destinations like `www.google.com` or `www.facebook.com` or `www.nytimes.com` or `www.indianexpress.com`, etc..

**Solution:** traceroute to facebook.com (157.240.198.35), 64 hops max

```
1 172.18.128.1 0.348ms 0.219ms 0.241ms
2 192.168.0.1 1.163ms 0.832ms 0.805ms
3 103.99.186.10 1.625ms 1.926ms 1.905ms
4 * * *
5 103.218.244.118 19.378ms 20.204ms 3.051ms
6 74.119.78.201 4.847ms 2.472ms 2.418ms
7 173.252.67.91 2.870ms 3.534ms 2.881ms
8 157.240.198.35 2.852ms 2.306ms 2.360ms
```

- (b) Report any curious things you notice, like some paths that default to IPv6 and how you can force traceroute to use IPv4, any private IP address spaces you notice like 10.0.0.0 to 10.255.255.255, or 172.16.0.0 to 172.31.255.255, or 192.168.0.0 to 192.168.255.255, missing routers along the path that do not seem to reply to the traceroute requests, etc.

**Solution:** The curious things noticed:

Private IP address: 172.18.128.1

192.168.0.1

Missing routers : at hop 4

- (c) Ping allows you to specify the size of packets to send. What is the maximum size of ping packets that you are able to send?

**Solution:** 65500 Byte

2. (Can you replicate the traceroute functionality using ping? Ping allows you to initialize a TTL. Write a simple script in which you use ping to replicate traceroute. You can write this script in bash or perl or python or any of your favourite scripting languages.

**Solution:**

Listing 1: Traceroute Script

```
#!/bin/bash
```

```
destination=$1
```

```

max_hops=30

echo "Traceroute to $destination"

for ttl in $(seq 1 $max_hops); do
    result=$(ping -c 1 -t $ttl -W 1 $destination)
    # echo "$result"
    hostname=$(echo "$result" | grep -oE '[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+' | head -n 2 | awk 'NR==2')
    time=$(ping -c 1 $hostname | awk 'NR==2' | grep -oE "time=[0-9]+\.[0-9]* ms" )
    echo "$ttl: $hostname $time"

    echo "*****"
    if echo "$result" | grep -q "0% packet loss"; then
        break
    fi
done

```

### 3. Introduction

Consider the following web servers of educational institutions in different continents:

- University of Utah (US mid-west): [www.utah.edu](http://www.utah.edu)
- University of Cape Town (South Africa): [www.uct.ac.za](http://www.uct.ac.za)
- IIT Delhi (India): [www.iitd.ac.in](http://www.iitd.ac.in)

And consider the following web servers of large content providers:

- Google: [www.google.com](http://www.google.com)
- Facebook: [www.facebook.com](http://www.facebook.com)

The end of this document contains a list of several working traceroute servers around the world, which allow you to issue a traceroute command from there to any other hosts on the Internet. Pick some 2 traceroute servers from different continents, plus one being your own device, and do a traceroute from there to these five web servers.

Consult an AS-IP lookup service to figure out when traffic gets into the local ISP, transits to other intermediate ISPs, and finally into the destination domains. One such service is <https://hackertarget.com/as-ip-lookup/>. FYI, you can also check how different ASes are connected with one another from this collated dataset here: <https://bgp.potaroo.net/cidr/autnums.html>.

#### Traceroute Servers

Here are some traceroute servers you can use:

- Traceroute Server 1: [server1.example.com](http://server1.example.com)
- Traceroute Server 2: [server2.example.com](http://server2.example.com)
- Your Own Device: *Your IP Address*

#### Traceroute Analysis

Perform traceroutes from the selected traceroute servers to the specified web servers. Use the provided AS-IP lookup service to analyze how traffic flows through different Autonomous Systems (ASes) on its way to the destination domains.

- (a) In a neat tabular format, report the number of hops from the (3) traceroute sources to the above (5) destinations. If the pair of (traceroute source, destination) are geographically close to each other, does it roughly translate into fewer hops? Do Google and Facebook differ from the others in the number of hops required to reach them, irrespective of which traceroute source is used? Why would this be so?

Server	Google	Facebook	IIT Delhi (India)	University of Utah	University of Cape Town
Local Server	9	8	12	33	22
Canada	7	8	*	26	*
USA	13	13	*	30	*

Table 1: A 4x6 Table

**Solution:**

Yes, If traceroute source and destination are close to each other then there will be fewer hops. Yes for Google and Facebook as destination have less hops than other destination server it is because Google use CDN(Content Delivery Network), Regional Data centers to reduce number of hops.

- (b) Also report the latencies between the traceroute sources and the web-servers. Does the latency seem to be related to the number of hops, being higher when there are more hops? Why is this the case?

Latency	Google	Facebook	IIT Delhi (India)	University of Utah	University of Cape Town
	61.7	39.1	3.29	313	*
	19.199	19.199	18.195	*	46.584
	16.169	3.865	*	63.643	*

Table 2: A 4x6 Table

**Solution:**

Yes we can see that latency increases as the number of hop increases. This is because at each hop, there is a certain amount of delay to the packet's journey.

- (c) Which of the destination web-servers are resolved to the same IP address irrespective of from where you do a traceroute to them? Why do you think some web-servers are resolved to different IP addresses when queried from different parts of the world? You can also use nslookup to change the DNS server that you want to use. You can also use this dig web interface which may help speed up things for you: <https://www.digwebinterface.com/>

**Solution:**

Google and Facebook have different IPs for different traceroute servers and other three server resolved same IP address because Google and Facebook have many server all over the world. When we trace from different part of the world then it give destination IP of Google and Facebook nearby traceroute server.

- (d) If you do traceroutes from the same starting point to different IP addresses you found for the same web-server, do the paths appear different? Which ones are longer?

**Solution:**

Tracerouting done for google using its 3 Ips:

142.250.194.142 –11 hops

142.251.215.228 –18hops

142.250.80.36 – 20 hops(This is longer among these 3 IPs)

Tracerouting done for facebook using its 3 Ips:

157.240.239.35 – 16hops

157.240.3.35 –22 hops

31.13.71.36 – 19hops(This is longer among these 3 IPs)

- (e) Try tracerouting to Google and Facebook from different countries of traceroute servers around the world. Are you able to find any countries that do not seem to have their local ISPs directly peered with Google and Facebook?

**Solution:**

We trace Google from MAKATI, PHILIPPINES. There is no direct peering from Makati(IP addr:216.218.253.238, ASN: AS69639 ) to google(IP addr 142.251.46.228 ASN: AS15169).

4. (**Packet analysis** Use wireshark to grab all packets on your wired or wireless interface, while visiting an HTTP website such as <http://act4d.iitd.ac.in> from your browser. Do an `ipconfig /flushdns` before you do this activity to clear your local DNS cache. And also clear your browser cache. Report the following:
- Apply a “dns” filter on the packet trace, and see if you can find DNS queries and responses for [www.iitd.ac.in](http://www.iitd.ac.in). How long did it take for the DNS request-response to complete?
  - Apply an “http” filter on the packet trace and report the approximate number of HTTP requests that were generated. What can you tell from this observation about how webpages are structured, and how browsers render complex pages with multiple images and files?
  - Apply a filter such as “((ip.src==192.168.1.3 ip.dst==10.7.174.111) — (ip.src==10.7.174.111 ip.dst==192.168.1.3)) tcp”. As would be self-explanatory, this will filter for TCP packets moving between your browser and the web-server. Recall that the source and destination IP addresses are a part of the network layer header, which is also called the IP layer since IP (Internet Protocol) is the most common network layer protocol in use. Find the number of TCP connections that were opened between your browser and the web-server. The signature for a new TCP connection is a 3-way handshake: The client sends a SYN message to the server, the server replies with a SYNACK message, and the client then sends an ACK. You will find that several TCP connections were opened between your browser and the web-server. Is this the same as the number of HTTP requests for content objects that you found in the previous part? Do you find that some content objects are fetched over the same TCP connection? Note that TCP connections are distinguished from one another based on the source port and destination port.
  - Now try doing a trace for <http://www.indianexpress.com> and filter for “http”. What do you find, is there any HTTP traffic? Browse through the entire trace without any filters, are you able to see the contents of any HTML and Javascript files being transferred? What just happened?

**Solution:**

(a) We have find DNS queries and responses for [www.iitd.ac.in](http://www.iitd.ac.in) ans it takes nearly 0.003641s take for the DNS request-response to complete.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	2001:df4:e000:3fc2::1	2001:df4:e000:29::1	DNS	94	Standard query 0xc5e8 AAAA www.iitd.ac.in
2	0.000694	2001:df4:e000:3fc2::1	2001:df4:e000:29::1	DNS	94	Standard query 0x32ea A www.iitd.ac.in
3	0.001189	2001:df4:e000:3fc2::1	2001:df4:e000:29::1	DNS	94	Standard query 0xe419 HTTPS www.iitd.ac.in
4	0.002947	2001:df4:e000:3fc2::1	2001:df4:e000:29::1	DNS	95	Standard query 0xb0e6 AAAA home.iitd.ac.in
5	0.003641	2001:df4:e000:29::1	2001:df4:e000:3fc2::1	DNS	122	Standard query response 0xc5e8 AAAA www.iitd.ac.in AAAA 2001:df4:e000:29::212
6	0.003753	2001:df4:e000:3fc2::1	2001:df4:e000:29::1	DNS	95	Standard query 0xf0c2 A home.iitd.ac.in
7	0.004010	2001:df4:e000:29::1	2001:df4:e000:3fc2::1	DNS	110	Standard query response 0x32ea A www.iitd.ac.in A 10.10.211.212
8	0.004442	2001:df4:e000:3fc2::1	2001:df4:e000:29::1	DNS	95	Standard query 0x71de HTTPS home.iitd.ac.in
9	0.005224	2001:df4:e000:3fc2::1	2001:df4:e000:29::1	DNS	100	Standard query 0x80f7 AAAA fonts.googleapis.com
10	0.005254	2001:df4:e000:29::1	2001:df4:e000:3fc2::1	DNS	147	Standard query response 0xe419 HTTPS www.iitd.ac.in SOA intdns.iitd.ac.in
11	0.005887	2001:df4:e000:3fc2::1	2001:df4:e000:29::1	DNS	100	Standard query 0x8309 A fonts.googleapis.com
12	0.005921	2001:df4:e000:29::1	2001:df4:e000:3fc2::1	DNS	123	Standard query response 0xb0e6 AAAA home.iitd.ac.in AAAA 2001:df4:e000:29::212
13	0.006305	2001:df4:e000:3fc2::1	2001:df4:e000:3fc2::1	DNS	111	Standard query response 0xf0c2 A home.iitd.ac.in A 10.10.211.212
14	0.006528	2001:df4:e000:3fc2::1	2001:df4:e000:29::1	DNS	100	Standard query 0xad9 HTTPS fonts.googleapis.com
15	0.008464	2001:df4:e000:29::1	2001:df4:e000:3fc2::1	DNS	148	Standard query response 0x71de HTTPS home.iitd.ac.in SOA intdns.iitd.ac.in
16	0.008530	2001:df4:e000:3fc2::1	2001:df4:e000:29::1	DNS	103	Standard query 0xd2d AAAA safebrowsing.google.com
17	0.009245	2001:df4:e000:3fc2::1	2001:df4:e000:29::1	DNS	103	Standard query 0x2692 A safebrowsing.google.com
18	0.010074	2001:df4:e000:3fc2::1	2001:df4:e000:29::1	DNS	103	Standard query 0x27b8 HTTPS safebrowsing.google.com
21	0.017437	2001:df4:e000:3fc2::1	2001:df4:e000:29::1	DNS	97	Standard query 0x77d8 AAAA fonts.gstatic.com

Figure 1: DNS filter for www.iitd.ac.in

**Solution:**

(b) There were 9 HTTP connections Wireshark captured after visiting the act4d.iitd.ac.in website and 1 HTTP connection for www.iitd.ac.in website. After observing the packets we can conclude that the webpage is structured using various resources such as HTML files, Javascript files, images etc. In addition to that there are multiple objects fetched over the same HTTP connection which is known as resource fetching.

No.	Time	Source	Destination	Protocol	Length	Info
63	0.339216	10.194.40.185	10.10.211.212	HTTP	63	GET / HTTP/1.1
88	0.346236	10.10.211.212	10.194.40.185	HTTP	495	HTTP/1.1 302 Found (text/html)

Figure 2: HTTP request and response for www.iitd.ac.in

No.	Time	Source	Destination	Protocol	Length	Info
60	10.701181	192.168.0.109	103.27.9.24	HTTP	599	GET / HTTP/1.1
78	10.710274	103.27.9.24	192.168.0.109	HTTP	495	HTTP/1.1 302 Found (text/html)
715	44.302098	192.168.0.109	34.104.35.123	HTTP	453	HEAD /edged1/diffgen-puffin/jflookgnckckhobagIndicnbbgbonedg/1.9c7138bb9e60926af5b37df46acda922c86612393b5e8fbc5022be8a462030b...
717	44.310349	34.104.35.123	192.168.0.109	HTTP	603	HTTP/1.1 200 OK
723	44.393612	192.168.0.109	192.168.0.1	HTTP	254	GET /gateicfg5CPD.xml HTTP/1.1
728	44.397678	192.168.0.1	192.168.0.109	HTTP/X...	1309	HTTP/1.1 200 OK
735	44.424831	192.168.0.109	192.168.0.1	HTTP	337	SUBSCRIBE /upnp/control/WANCommonIFC1 HTTP/1.1
737	44.427930	192.168.0.1	192.168.0.109	HTTP	293	HTTP/1.1 200 OK
750	44.440046	192.168.0.109	192.168.0.1	HTTP/X...	363	POST /upnp/control/WANCommonIFC1 HTTP/1.1
754	44.448955	192.168.0.1	192.168.0.109	HTTP/X...	390	HTTP/1.1 200 OK
762	44.455721	192.168.0.109	192.168.0.1	HTTP/X...	367	POST /upnp/control/WANCommonIFC1 HTTP/1.1
766	44.463852	192.168.0.1	192.168.0.109	HTTP/X...	404	HTTP/1.1 200 OK
771	44.476752	192.168.0.1	192.168.0.109	HTTP/X...	218	NOTIFY /upnp/eventing/ookqzobbv HTTP/1.1 Continuation
772	44.478389	192.168.0.109	192.168.0.1	HTTP	191	HTTP/1.1 200 OK
777	44.620714	192.168.0.109	192.168.0.1	HTTP	254	GET /gateicfg5CPD.xml HTTP/1.1
783	44.631417	192.168.0.1	192.168.0.109	HTTP/X...	1309	HTTP/1.1 200 OK
790	44.657818	192.168.0.109	192.168.0.1	HTTP	337	SUBSCRIBE /upnp/control/WANCommonIFC1 HTTP/1.1
792	44.667549	192.168.0.1	192.168.0.109	HTTP	293	HTTP/1.1 200 OK
805	44.699925	192.168.0.109	192.168.0.1	HTTP/X...	363	POST /uana/control/WANCommonIFC1 HTTP/1.1

Figure 3: HTTP request and response for act4d.iitd.ac.in

**Solution:**

(c) After visiting the website act4d.iitd.ac.in, there are 6 tcp connections which are open with source in the range 61109-61116 which were the same as the number of unique HTTP connections. There

were multiple content and resources which were fetched over the same TCP connection, which efficiently uses the connection for better response and faster loading.

No.	Time	Source	Destination	Protocol	Length	Info
54	1.272413	10.194.49.234	10.237.26.108	TCP	66	61109 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
55	1.273643	10.194.49.234	10.237.26.108	TCP	66	61110 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
61	1.331688	10.237.26.108	10.194.49.234	TCP	66	80 → 61109 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=536 SACK_PERM WS=64
62	1.331688	10.237.26.108	10.194.49.234	TCP	66	80 → 61110 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=536 SACK_PERM WS=64
63	1.331931	10.194.49.234	10.237.26.108	TCP	54	61109 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
64	1.332019	10.194.49.234	10.237.26.108	TCP	54	61110 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
65	1.332530	10.194.49.234	10.237.26.108	TCP	590	61109 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=536 [TCP segment of a reassembled PDU]
66	1.332530	10.194.49.234	10.237.26.108	HTTP	65	GET / HTTP/1.1
71	1.400205	10.194.49.234	10.237.26.108	TCP	590	[TCP Retransmission] 61109 → 80 [PSH, ACK] Seq=12 Ack=1 Win=131072 Len=536
78	1.771769	10.194.49.234	10.237.26.108	TCP	590	[TCP Retransmission] 61109 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=536
82	2.136861	10.237.26.108	10.194.49.234	TCP	54	80 → 61109 [ACK] Seq=1 Ack=537 Win=6912 Len=0
83	2.136998	10.194.49.234	10.237.26.108	TCP	65	[TCP Retransmission] 61109 → 80 [PSH, ACK] Seq=537 Ack=1 Win=131072 Len=11
84	2.153914	10.237.26.108	10.194.49.234	TCP	54	80 → 61109 [ACK] Seq=1 Ack=548 Win=6912 Len=0
108	2.224869	10.237.26.108	10.194.49.234	TCP	65	[TCP Dup ACK 84#1] 80 → 61109 [ACK] Seq=548 Ack=548 Win=6912 Len=0 SLE=12 SRE=548
109	2.224869	10.237.26.108	10.194.49.234	TCP	590	80 → 61109 [ACK] Seq=1 Ack=548 Win=6912 Len=536 [TCP segment of a reassembled PDU]
110	2.224869	10.237.26.108	10.194.49.234	TCP	590	80 → 61109 [ACK] Seq=537 Ack=548 Win=6912 Len=536 [TCP segment of a reassembled PDU]
111	2.224869	10.237.26.108	10.194.49.234	TCP	590	80 → 61109 [ACK] Seq=1073 Ack=548 Win=6912 Len=536 [TCP segment of a reassembled PDU]
112	2.225551	10.194.49.234	10.237.26.108	TCP	54	61109 → 80 [ACK] Seq=548 Ack=2145 Win=131072 Len=0
122	2.228395	10.237.26.108	10.194.49.234	TCP	65	[TCP Dup ACK 84#1] 80 → 61109 [ACK] Seq=2145 Ack=548 Win=6912 Len=0 SLE=1 SRE=537
140	2.282181	10.237.26.108	10.194.49.234	TCP	66	[TCP Dup ACK 84#1] 80 → 61109 [ACK] Seq=2145 Ack=548 Win=6912 Len=0 SLE=537 SRE=548
148	2.282181	10.237.26.108	10.194.49.234	TCP	590	80 → 61109 [ACK] Seq=2145 Ack=548 Win=6912 Len=536 [TCP segment of a reassembled PDU]
149	2.282181	10.237.26.108	10.194.49.234	TCP	590	80 → 61109 [ACK] Seq=2681 Ack=548 Win=6912 Len=536 [TCP segment of a reassembled PDU]
150	2.282376	10.194.49.234	10.237.26.108	TCP	54	61109 → 80 [ACK] Seq=548 Ack=3217 Win=131072 Len=0
151	2.282755	10.237.26.108	10.194.49.234	TCP	590	80 → 61109 [ACK] Seq=3217 Ack=548 Win=6912 Len=536 [TCP segment of a reassembled PDU]
152	2.282755	10.237.26.108	10.194.49.234	HTTP/XML	574	HTTP/1.1 200 OK
157	2.282888	10.194.49.234	10.237.26.108	HTTP	54	61109 → 80 [ACK] Seq=548 Ack=4273 Win=131072 Len=0
228	2.538920	10.194.49.234	10.237.26.108	HTTP	587	GET /act4d/templates/beez/css/template.css HTTP/1.1
229	2.540470	10.194.49.234	10.237.26.108	HTTP	587	GET /act4d/templates/beez/css/position.css HTTP/1.1
230	2.541059	10.194.49.234	10.237.26.108	TCP	66	61113 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
231	2.542069	10.194.49.234	10.237.26.108	TCP	66	61114 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
232	2.543447	10.194.49.234	10.237.26.108	TCP	66	61115 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
233	2.547290	10.194.49.234	10.237.26.108	TCP	66	61116 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
266	2.687767	10.237.26.108	10.194.49.234	TCP	54	80 → 61109 [ACK] Seq=4273 Ack=1081 Win=8000 Len=0
267	2.687767	10.237.26.108	10.194.49.234	TCP	66	80 → 61113 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=536 SACK_PERM WS=64
268	2.687767	10.237.26.108	10.194.49.234	TCP	66	80 → 61115 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=536 SACK_PERM WS=64
269	2.687767	10.237.26.108	10.194.49.234	TCP	54	80 → 61110 [ACK] Seq=1 Ack=534 Win=6912 Len=0
270	2.687767	10.237.26.108	10.194.49.234	TCP	66	80 → 61115 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=536 SACK_PERM WS=64
271	2.687767	10.237.26.108	10.194.49.234	TCP	66	80 → 61114 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=536 SACK_PERM WS=64
272	2.687767	10.237.26.108	10.194.49.234	TCP	590	80 → 61109 [ACK] Seq=4273 Ack=1081 Win=8000 Len=536 [TCP segment of a reassembled PDU]
273	2.687767	10.237.26.108	10.194.49.234	TCP	590	80 → 61109 [ACK] Seq=4809 Ack=1081 Win=8000 Len=536 [TCP segment of a reassembled PDU]
274	2.687767	10.237.26.108	10.194.49.234	HTTP	98	HTTP/1.1 200 OK (text/css)

Figure 4: TCP response for act4d.iitd.ac.in

**Solution:**

(d) Wireshark didn't capture any packets of HTTP and Javascript after visiting Indianexpress.com. This is because Indian Express uses HTTPS SSL(socket secure layer) or TLS(Transport layer security) to encrypt their traffic. Wireshark can capture the initial handshake of the SSL/TLS connection, but it cannot decrypt the actual content of the encrypted packets without the proper SSL/TLS keys.

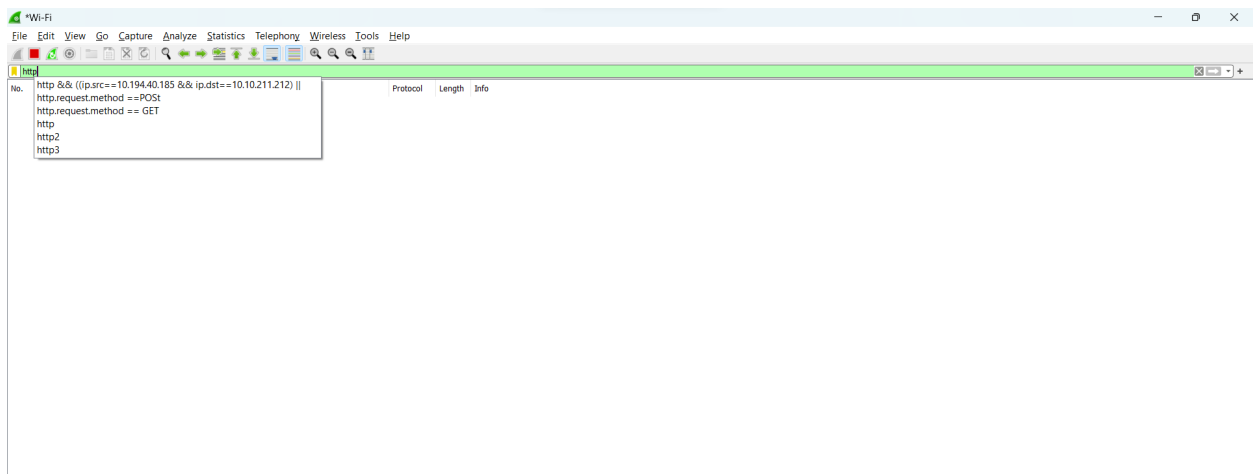


Figure 5: HTTP request and response for www.indianexpress.com