

# Computer Networks

## Assignment1

BY MANI SARTHAK & HARSHIT GUPTA

August 13, 2023

Some points to be considered:

- All the codes are managed on GitHub and can also be found in this [folder](#).
- We both have Macbook so any functionality that is only specific to Windows Command Shell or Power Shell couldn't be concluded by us.

Token Distribution:

1. Mani Sarthak : 2021CS10095 : 10
2. Harshit Gupta : 2021CS10552 : 10

## §1 Part1 : Network Analysis

### §1.1 Traceroute to servers

Procedure: Connected personal laptop (Mani's Macbook) to 4G/5G cellular network of Jio using cellular hotspot. Then traceroute to different servers were called.

#### §1.1.1 Tracerouting to [www.iitd.ac.in](http://www.iitd.ac.in)

```

Last login: Thu Aug 10 23:22:13 on ttys005
manisarthak@Manis-MacBook-Air ~ % traceroute www.iitd.ac.in
traceroute to www.iitd.ac.in (103.27.9.24), 64 hops max, 52 byte packets
 1  192.168.249.240 (192.168.249.240)  38.106 ms  7.875 ms  8.659 ms
 2  192.168.31.1 (192.168.31.1)  1240.294 ms  1532.023 ms  1510.732 ms
 3  192.168.12.73 (192.168.12.73)  32.406 ms  41.359 ms  35.202 ms
 4  192.168.13.38 (192.168.13.38)  41.656 ms
    192.168.13.58 (192.168.13.58)  28.976 ms
    192.168.13.38 (192.168.13.38)  45.161 ms
 5  dsl-tn-dynamic-125.222.22.125.airtelbroadband.in (125.22.222.125)  43.628 ms  34.377 ms  41.381 ms
 6  116.119.109.1 (116.119.109.1)  36.707 ms  45.818 ms
    116.119.109.3 (116.119.109.3)  48.083 ms
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
31  * * *
32  * * *
33  * * *
34  * * *
35  * * *
36  * * *
37  * * *
38  * * *
39  * * *
40  * * *
41  * * *
42  * * *
43  * * *
44  * * *
45  * * *
46  * * *
47  * * *

```

It seems as the ISP is blocking packets to the path to [www.iitd.ac.in](http://www.iitd.ac.in), so we now try to traceroute to [www.google.com](http://www.google.com).

### §1.1.2 Tracerouting to www.google.com

```

~ -- zsh
Last login: Thu Aug 10 23:22:19 on ttys009
manisarthak@Manis-MacBook-Air ~ % traceroute www.google.com
traceroute to www.google.com (142.250.195.4), 64 hops max, 52 byte packets
 1  192.168.249.240 (192.168.249.240)  9.219 ms  4.154 ms  4.907 ms
 2  192.168.31.1 (192.168.31.1)  43.548 ms  24.572 ms  24.612 ms
 3  192.168.12.97 (192.168.12.97)  37.709 ms  36.374 ms  59.388 ms
 4  192.168.13.48 (192.168.13.48)  28.750 ms  35.952 ms
    192.168.13.38 (192.168.13.38)  42.468 ms
 5  dsl-tn-dynamic-121.222.22.125.airtelbroadband.in (125.22.222.121)  63.242 ms
    dsl-tn-dynamic-125.222.22.125.airtelbroadband.in (125.22.222.125)  26.109 ms  26.214 ms
 6  * * 72.14.243.2 (72.14.243.2)  108.250 ms
 7  * * *
 8  142.251.54.74 (142.251.54.74)  122.568 ms
    142.251.54.62 (142.251.54.62)  70.405 ms
    142.251.52.228 (142.251.52.228)  75.906 ms
 9  * del12s09-in-f4.1e100.net (142.250.195.4)  48.616 ms
    74.125.244.196 (74.125.244.196)  40.590 ms
manisarthak@Manis-MacBook-Air ~ %

```

The traceroute to www.google.com was successful and it took 9 hops to reach the destination address.

### §1.2 Observations

- After calling traceroute to multiple destinations, i didn't find any case in which paths default to IPv6, so i conclude that if that occurs then it must be rare.
- By default in Mac's unix running traceroute returns IP addresses in paths in IPv4 only, however if one wants to get the IP's in IPv6 then one should use the command `traceroute6`.
- We can also observe private IP addresses in the paths like 192.168.249.240, 192.168.31.1, 198.162.13.38 and so on and these are common in the path of both. This shows that they both use the initial routers of Jio Network which is private before transferring the packets to the global networks.
- There are also routers that do not reply to traceroute requests. Example: see router number 7 in google.com traceroute call.

### §1.3 Ping

- Ping command is used to send packets to check whether an IP address is live or not.
- The `-s` flag allows us to send ping requests with different packet sizes with the default value being 56 Bytes (excluding 8 Bytes ICMP header). The real packet which is sent and received is 8 Bytes more than the data pinged, this is because of the 8 Byte ICMP header which is added to the original data.
- The maximum size of ping packets that we are able to send from MacBook M1 by using IITD WiFi for websites is mentioned below. We have used Binary Search to get to the conclusion.

Destination	Size
www.facebook.com	1472
www.nytimes.com	1472
www.indianexpress.com	8184
www.iitd.ac.in	8184

Table 1 Table showing the maximum size of ping packets without ICMP header (in Bytes) that can be sent to the destination servers.

Observations:

- As servers like www.iitd.ac.in and www.indianexpress.com have very higher sizes. Also because www.facebook.com have multiple servers and the one we are sending ping is also in proximity to us but still we are getting lesser size, can be because packet size for ping also depends on servers.
- So we conclude that the maximum size depends on the servers that we are sending and also that whether they are in our proximity or not.

## §2 Part2

### §2.1 Methodology

**Logic** : what we did here was very simple. we used linux ping command here. we observed that when we limit the TTL ( max no. of hops) such that it can't reach the destination IP, the router sends a ICMP response upon dropping the packet which contain the IP address at which packet is dropped. so by using this observation , we iterate and increased the TTL at each iteration so that we get all intermediate IP address as no. of hops is increased by 1 each time until the packet is reached to the destination and we stop there. after that we ping at each intermediate IP to get **Round Trip Time** and we report it. so traceroute functionality is replicated.

### §2.2 View of code

```
import subprocess,sys,re,socket
def ping(host,m):
    output = subprocess.run(['ping','-c','1','-m',f'{m}',host],stdout=subprocess.PIPE,stderr=subprocess.STDOUT, universal_newlines=True)
    match = re.search(r'bytes from (\d+\.\d+\.\d+\.\d+): Time', output.stdout)
    if output.returncode==2 and match!=None:
        return (False,match.group(1))
    elif output.returncode==2 and match==None:
        return (False,'a')
    return (True,'0')

def trace_route(destination):
    m=0
    a = False
    l=[]
    i=0
    while(i<64):
        m+=1
        a,b = ping(destination,m)
        if a==True:
            break
        else:
            l.append(b)
        i+=1
    for i in range(len(l)):
        if l[i]=='a':
            print(f'{i+1} '+'*')
        else:
            output = subprocess.run(['ping','-c','1',l[i]],stdout = subprocess.PIPE,stderr=subprocess.STDOUT,universal_newlines=True)
            match = re.search(r'time=(\d+\.\d+)\s*ms', output.stdout)
            if match!=None:
                print(f'{i+1} '+' '+ f'{l[i]} ({l[i]}) '+' match.group(1)+' ms')
            else:
                print(f'{i+1} '+'*')
    wip_address = socket.gethostbyname(destination)
    output = subprocess.run(['ping','-c','1',destination],stdout = subprocess.PIPE,stderr=subprocess.STDOUT,universal_newlines=True)
    match = re.search(r'time=(\d+\.\d+)\s*ms', output.stdout)
    print(f'{len(l)+1} '+' '+ f'{destination} ({wip_address}) '+' match.group(1)+' ms')

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python help.py <target_host>")
        sys.exit(1)
    target_host = sys.argv[1]
    trace_route(target_host)
```

### §2.3 Commands and Modules used

- ping -c 1 -m *TTL Destination\_IP* here -c is for sending only 1 packet and -m is for defining TTL ( here macos terminal is used)

- `ping -c 1 destination_ip` for getting the time for each intermediate IP
- `subprocess` module is used for running commands at console
- `re` module is used for extracting IP from text output
- `sys` module for taking input from command line
- `socket` module for getting ip address for destination website

## §2.4 Input

use the code and put input in console as:-

```
python/python3 script_trace.py destination_ip
```

## §2.5 Output for www.iitd.ac.in

```
harshits-MacBook-Air-6:computer networks col334 harshit$ python3 script_trace.py www.iitd.ac.in
1 10.184.0.13 (10.184.0.13) 6.848 ms 3.611 ms 3.459 ms
2 10.254.175.5 (10.254.175.5) 3.732 ms 3.144 ms 3.513 ms
3 10.254.236.6 (10.254.236.6) 3.330 ms 4.252 ms 4.414 ms
4 www.iitd.ac.in (10.10.211.212) 3.580 ms 4.194 ms 4.178 ms
harshits-MacBook-Air-6:computer networks col334 harshit$
```

## §2.6 design decisions

- If in a particular IP we can't get a response or we can't get a intermediate IP during hops we leave it blank with a \* mark
- we are sending 3 packet here so Round Trip Time here is for 3 packets only and we can't change no. of packets as it is by default 3 in `traceroute` as sending more than 1 packet will make many edge cases

### §3 Part3

#### §3.1 Hops analysis

For India IITD WiFi is used, for South Africa the traceroute server of Capetown city was used and for USA traceroute server of Seattle was used.

Source & Destination	utah.edu	uct.ac.za	iitd.ac.in	google.com	facebook.com
India	64+	64+	4	11	13
South Africa	19	4	20	13	19
USA	11	15	16	10	9

Table 2 Table showing the number of hops to the destination from different traceroute servers.

Observations:

- If the traceroute source and destination are close to each other geographically then they have roughly fewer hops between them. For example see (India, iitd.ac.in) , (South Africa, uct.ac.za), (USA, utah.ac.in).
- Google and Facebook have very few number of hops as compared to others. This is because they have distributed servers across the globe and the traceroute takes their closest server as destination.

#### §3.2 Latency Analysis

For India IITD WiFi is used, for South Africa the traceroute server of Capetown city was used and for USA traceroute server of Seattle was used.

Source & Destination	utah.edu	uct.ac.za	iitd.ac.in	google.com	facebook.com
India	NA	NA	5.066	8.023	31.093
South Africa	231.681	4.013	320.114	248.818	228.993
USA	22.038	294.328	270.981	74.357	86.714

Table 3 Table showing the latency between destinations and different traceroute servers (in ms).

Observations:

- Latency seems to be related to the number of hops. Latency depends on a number of factors like traffic between the routers, processing time at the routers etc. Also higher number of hops means a higher chance of getting stuck in traffic and more time consumed at routers. Hence increase in number of hops increases the latency as well.

#### §3.3 Distributed servers

We find that servers like [www.iitd.ac.in](http://www.iitd.ac.in), [www.utah.edu](http://www.utah.edu) and [www.uct.ac.za](http://www.uct.ac.za) are resolved to same IP address irrespective of from where they are getting the traceroute request, whereas servers like [www.google.com](http://www.google.com) and [www.facebook.com](http://www.facebook.com) have different destination IP's for different parts of world.

The reason for this is because big MNC's like google and Facebook need to provide their

users with as low latency as possible and also that is economically optimal for them so they create their databases across the globe each acting as a server with unique IP. However educational institutes like [utha.edu](http://utha.edu), [iitd.ac.in](http://iitd.ac.in) have a single server for them.

### §3.4 Distributed Servers of Google and Facebook

The IPv4 addresses of Google and Facebook are obtained by sending traceroute requests to them through different locations namely:

- IITD WiFi.
- Princeton traceroute server [www.net.princeton.edu](http://www.net.princeton.edu).
- HanNet Germany server.

The IPv4 addresses of the destination is summarised in the table below.

Source & Destination	www.google.com	www.facebook.com
IITD Wifi	142.250.206.142	157.240.16.35
Princeton	142.250.65.238	31.13.71.36
HanNet	172.217.18.14	157.240.210.35

Table 4 Table showing the IPv4 address of various Google and Facebook servers.

The number of hops and latency for the traceroute requests between the source and destinations can be summarised in the tables below (for latency average of the latency by three packets sent is taken).

Source & Destination	www.google.com	www.facebook.com
IITD Wifi	11	13
Princeton	20	20
HanNet	23	24

Table 5 Table showing #hops to various Google and Facebook servers from IITD WiFi.

Source & Destination	www.google.com	www.facebook.com
IITD Wifi	8.373	38.891
Princeton	334.253	314.255
HanNet	404.218	201.761

Table 6 Table showing the latency of various Google and Facebook servers from IITD WiFi.

Observation:

- Here also we see that traceroute requests to same server but with different IP address have different paths and latencies.
- Also the latencies and #hops are related to each other as increase in #hops roughly increases the latency too.
- Last but not the least, IP addresses for the server nearer to the source have very low latency as compared to other IP addresses.
- Some IP addresses are optimised for my location and hence they have very few hops and latencies but other IP's that are way too far off and hence are longer.



### §3.5 Tracerouting to Google and Facebook from different sources

Source & Destination	www.google.com	www.facebook.com
Los Angeles, USA	8	9
Berlin, Germany	14	12
London, UK	12	12
Melbourne, Australia	7	1
Johannesburg, South Africa	-	10
Mombasa, Kenya	17	15

Table 7 Table showing the number of hops to destinations from different traceroute servers.

Source & Destination	www.google.com	www.facebook.com
Los Angeles, USA	7.861	10.717
Berlin, Germany	150.234	157.013
London, UK	131.682	128.631
Melbourne, Australia	11.528	11.823
Johannesburg, South Africa	-	226.6
Mombasa, Kenya	340.216	310.067

Table 8 Table showing the average latency to destinations from different traceroute servers (in ms)

Observations:

- We notice that the #hops and latency for South African cities are significantly more than that of cities like Los Angeles, Melbourne etc.
- Assuming that the most developed cities like Los Angeles, Melbourne etc must have their local ISP's directly peered with Google and Facebook. We can say that cities like Mombasa and Johannesburg doesn't have their local ISP's directly peered with Google and Facebook.

## §4 Part4

### §4.1 Part a

In this part we started wireshark and captured the packets for 2 sites [www.iitd.ac.in](http://www.iitd.ac.in) and <http://act4d.iitd.ac.in>.

Now we apply the DNS filter and got all the dns requests and responses and after subtracting the response time with request time we get the time for dns request-response, for [www.iitd.ac.in](http://www.iitd.ac.in) it is:

Response time & type of request	<a href="http://www.iitd.ac.in">www.iitd.ac.in</a>	<a href="http://act4d.iitd.ac.in">http://act4d.iitd.ac.in</a>
AAAA	7.694 ms	3.916 ms
A	7.647 ms	3.853 ms
HTTP	7.602 ms	3.798 ms

Table 9 Table showing the response time with different request for 2 sites (in ms)

262	9.224278	2001:df4:e00...	2001:df4:e00...	DNS	94	Standard query 0x4db9 AAAA <a href="http://www.iitd.ac.in">www.iitd.ac.in</a>
263	9.224325	2001:df4:e00...	2001:df4:e00...	DNS	94	Standard query 0x294b A <a href="http://www.iitd.ac.in">www.iitd.ac.in</a>
264	9.224369	2001:df4:e00...	2001:df4:e00...	DNS	94	Standard query 0xc2ac HTTPS <a href="http://www.iitd.ac.in">www.iitd.ac.in</a>
265	9.231971	2001:df4:e00...	2001:df4:e00...	DNS	147	Standard query response 0xc2ac HTTPS <a href="http://www.iitd.ac.in">www.iitd.ac.in</a> SOA intdns.iitd.ac.in
266	9.231972	2001:df4:e00...	2001:df4:e00...	DNS	110	Standard query response 0x294b A <a href="http://www.iitd.ac.in">www.iitd.ac.in</a> A 10.10.211.212
267	9.231972	2001:df4:e00...	2001:df4:e00...	DNS	122	Standard query response 0x4db9 AAAA <a href="http://www.iitd.ac.in">www.iitd.ac.in</a> AAAA 2001:df4:e000:29::212

pic for [www.iitd.ac.in](http://www.iitd.ac.in)

51	5.500456	2001:df4:e00...	2001:df4:e00...	DNS	96	Standard query 0x1691 AAAA <a href="http://act4d.iitd.ac.in">act4d.iitd.ac.in</a>
52	5.500517	2001:df4:e00...	2001:df4:e00...	DNS	96	Standard query 0x22ba A <a href="http://act4d.iitd.ac.in">act4d.iitd.ac.in</a>
53	5.500574	2001:df4:e00...	2001:df4:e00...	DNS	96	Standard query 0x1872 HTTPS <a href="http://act4d.iitd.ac.in">act4d.iitd.ac.in</a>
54	5.504370	2001:df4:e00...	2001:df4:e00...	DNS	112	Standard query response 0x22ba A <a href="http://act4d.iitd.ac.in">act4d.iitd.ac.in</a> A 10.237.26.108
55	5.504372	2001:df4:e00...	2001:df4:e00...	DNS	149	Standard query response 0x1872 HTTPS <a href="http://act4d.iitd.ac.in">act4d.iitd.ac.in</a> SOA intdns.iitd.ac.in
56	5.504372	2001:df4:e00...	2001:df4:e00...	DNS	149	Standard query response 0x1691 AAAA <a href="http://act4d.iitd.ac.in">act4d.iitd.ac.in</a> SOA intdns.iitd.ac.in

pic for <http://act4d.iitd.ac.in>

### §4.2 Part b

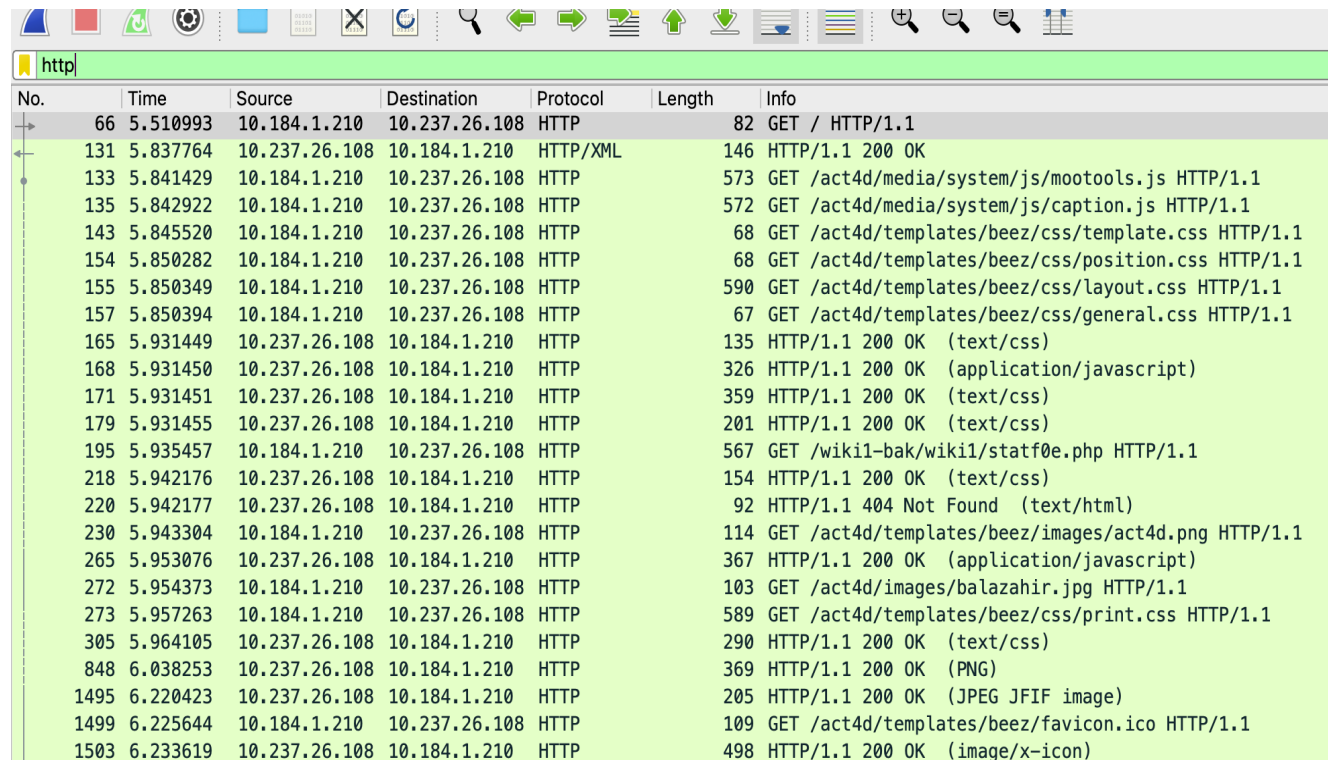
Now applying the http filter and get those packets

- no. of http request generated for [www.iitd.ac.in](http://www.iitd.ac.in) is :- **1** beacuse this site has secured http so 1 request is visible not css ,js etc.
- no. of http request generated for <http://act4d.iitd.ac.in> is :- **12** beacuse this website is not secured so we can see how many request these pages have

with seeing the http request for act4d website we can see how webpages are structured and how browser render complex images and files. we can see. so we can see that first application javascript is requested and then templates pages such as html , css , text etc. are requested and then rendered. after they are completed media pages such as images, video , favicon etc. are requested and then rendered.

No.	Time	Source	Destination	Protocol	Length	Info
274	9.238055	2001:df4:e00...	2001:df4:e00...	HTTP	624	GET / HTTP/1.1
276	9.245608	2001:df4:e00...	2001:df4:e00...	HTTP	527	HTTP/1.1 302 Found (text/html)

pic for [www.iitd.ac.in](http://www.iitd.ac.in)



No.	Time	Source	Destination	Protocol	Length	Info
66	5.510993	10.184.1.210	10.237.26.108	HTTP	82	GET / HTTP/1.1
131	5.837764	10.237.26.108	10.184.1.210	HTTP/XML	146	HTTP/1.1 200 OK
133	5.841429	10.184.1.210	10.237.26.108	HTTP	573	GET /act4d/media/system/js/mootools.js HTTP/1.1
135	5.842922	10.184.1.210	10.237.26.108	HTTP	572	GET /act4d/media/system/js/caption.js HTTP/1.1
143	5.845520	10.184.1.210	10.237.26.108	HTTP	68	GET /act4d/templates/beeze/css/template.css HTTP/1.1
154	5.850282	10.184.1.210	10.237.26.108	HTTP	68	GET /act4d/templates/beeze/css/position.css HTTP/1.1
155	5.850349	10.184.1.210	10.237.26.108	HTTP	590	GET /act4d/templates/beeze/css/layout.css HTTP/1.1
157	5.850394	10.184.1.210	10.237.26.108	HTTP	67	GET /act4d/templates/beeze/css/general.css HTTP/1.1
165	5.931449	10.237.26.108	10.184.1.210	HTTP	135	HTTP/1.1 200 OK (text/css)
168	5.931450	10.237.26.108	10.184.1.210	HTTP	326	HTTP/1.1 200 OK (application/javascript)
171	5.931451	10.237.26.108	10.184.1.210	HTTP	359	HTTP/1.1 200 OK (text/css)
179	5.931455	10.237.26.108	10.184.1.210	HTTP	201	HTTP/1.1 200 OK (text/css)
195	5.935457	10.184.1.210	10.237.26.108	HTTP	567	GET /wiki1-bak/wiki1/statf0e.php HTTP/1.1
218	5.942176	10.237.26.108	10.184.1.210	HTTP	154	HTTP/1.1 200 OK (text/css)
220	5.942177	10.237.26.108	10.184.1.210	HTTP	92	HTTP/1.1 404 Not Found (text/html)
230	5.943304	10.184.1.210	10.237.26.108	HTTP	114	GET /act4d/templates/beeze/images/act4d.png HTTP/1.1
265	5.953076	10.237.26.108	10.184.1.210	HTTP	367	HTTP/1.1 200 OK (application/javascript)
272	5.954373	10.184.1.210	10.237.26.108	HTTP	103	GET /act4d/images/balazahir.jpg HTTP/1.1
273	5.957263	10.184.1.210	10.237.26.108	HTTP	589	GET /act4d/templates/beeze/css/print.css HTTP/1.1
305	5.964105	10.237.26.108	10.184.1.210	HTTP	290	HTTP/1.1 200 OK (text/css)
848	6.038253	10.237.26.108	10.184.1.210	HTTP	369	HTTP/1.1 200 OK (PNG)
1495	6.220423	10.237.26.108	10.184.1.210	HTTP	205	HTTP/1.1 200 OK (JPEG JFIF image)
1499	6.225644	10.184.1.210	10.237.26.108	HTTP	109	GET /act4d/templates/beeze/favicon.ico HTTP/1.1
1503	6.233619	10.237.26.108	10.184.1.210	HTTP	498	HTTP/1.1 200 OK (image/x-icon)

pic for <http://act4d.iitd.ac.in>

### §4.3 Part c

Now we apply the `((ip.src==192.168.1.3 && ip.dst==10.7.174.111) or (ip.src==10.7.174.111 && ip.dst==192.168.1.3)) && TCP` filter and answer of 3 questions are given below:

- now we need to find the no. of tcp connections opened between my browser and web-server. as stated a tcp connection is 3-way handshake, our browser,client, sends SYN message to server , server replies with SYN-ACK message and then client sends an ACK. now to find no. of connections opened we just need to count no. of SYN request and we get that count as **6**
- now we find that this is not same and less than number of HTTP requests for content objects in previous part due to the fact that multiple http objects get transferred over same tcp connection due to HTTP/1.1 protocol being alive to reduce latency
- Yes, as stated above multiple HTTP content being fetched over same TCP connection due to the fact the HTTP/1.1 protocol is present and its ability to reuse existing connections for multiple requests to reduce latency.

### §4.4 Part d

Now applying trace for <http://www.indianexpress.com> and we apply the http filter. surprisingly we found that there is no HTTP traffic visible over the network as we only found 1 http request for indianexpress.com visible of HTTP/1.1 protocol visible

now browsing through the entire trace without any filters but we cannot see any contents of html and javascript files being transferred. so the reason for that is that this website

and many other uses **HTTP secure** which encrypts the traffic, making it difficult to directly see the contents of the transferred files in plaintext.

therefore we are not able to view the HTTP traffic like the act4d.iitd.ac.in