COL334 Assignment-1 Report

Amaiya Singhal (2021CS50598) Om Dehlan (2021CS10076)

1. Network Analysis

Part a

- `tracert` on a windows machine was used to get the following results.
- Airtel 5G hotspot from a smartphone was used for the tests.
- **Traceroute for** `www.iitd.ac.in`

Hop No.	IP Address
1	2401:4900:5fc7:88ce::2c
2	2401:4900:20:131::1
3	2401:4900:5a60:37::1
4	2404:a800:1a00:801::225
5	2404:a800::93
6	2405:200:1607:600:49:44:220:bc
7	2405:203:89a::141e
8	2405:8a00:a:3::2
9	2405:8a00:a:a::3
10	Request timed out.
11	2001:4408:520a:10:40:128:128:1
12	2405:8a00:a:2::c5
13	2405:8a00:a:2::c6
14	2001:df4:e000:108::1
15	2001:df4:e000:26::24
16	2001:df4:e000:29::212

- **Traceroute for** `www.google.com`

Hop No.	IP Address
1	2401:4900:8097:986e::ff
2	2401:4900:20:134::1
3	2401:4900:5a60:43::1
4	2404:a800:1a00:801::229
5	2404:a800::208
6	2001:4860:1:1::d4e
7	2404:6800:8172::1
8	2001:4860:0:1::5e62
9	2001:4860:0:1a::5
10	2001:4860:0:11dd::1
11	2001:4860:0:1::5397
12	2404:6800:4002:826::2004

Part b: Observations

- On both `www.iitd.ac.in` and `www.google.com`, `tracert` defaulted to using IPv6.
- However when we tried on a linux system, both these routes defaulted to IPv4. This might be due to the difference in the types of packets sent by `tracert` and `traceroute`.
- We can force `tracert` to use IPv4 using the option: `tracert -4 <IP/Hostname>`
- **Traceroute (IPv4) for** `www.iitd.ac.in`

Hop No.	IP Address
1	192.168.218.229
2	106.208.190.1
3	Request timed out
4	182.71.124.169
5	49.44.187.164
6	Request timed out
7	Request timed out
8	136.232.148.178
9	Request timed out
10	Request timed out
11	Request timed out
12	103.27.9.24
13	103.27.9.24

- **Traceroute (IPv4) for** `www.google.com`

Hop No.	IP Address
1	192.168.218.229
2	106.208.190.1
3	117.96.31.70
4	182.71.124.173
5	116.119.112.137
6	72.14.243.0
7	74.125.244.204
8	172.253.69.58
9	108.170.248.177
10	72.14.239.247
11	142.250.183.100

- The only private IP address we noticed was that of our own device (`192.168.218.229`)
- Upon using `tracert` on `www.iitd.ac.in`, we noticed that some routers did not reply to the requests and the requests **timed out**. This happened both with IPv4 and IPv6.
- This can happen due to multiple reasons which include
  - Packets were dropped due to an issue on the network.
  - The device was not configured to reply to traffic.
  - An issue with the return path from the destination computer.

Part c: Maximum size of ping packet

- We sent multiple `ping` requests, varying the size of the packets everytime to find out the maximum packet size that was successfully sent.
- We manually varied packet size to identify the size beyond which requests start failing.
- The following table contains the packet sizes and the results, in the order we tested them for `www.iitd.ac.in`

Packet Size	Outcome
64	Successful
128	Successful
256	Successful
512	Successful
1024	Successful
2048	Request Timed Out
1536	Request Timed Out
1360	Request Timed Out
1152	Successful
1216	Successful
1280	Successful
1300	Successful
1320	Request Timed Out
1310	Successful
1315	Request Timed Out
1312	Request Timed Out
1311	Request Timed Out
1310	Successful

- The following table contains the packet sizes and the results, in the order we tested them for `www.google.com`

Packet Size	Outcome
1300	Successful
1320	Request Timed Out
1310	Successful
1315	Request Timed Out
1312	Request Timed Out
1311	Request Timed Out
1310	Successful

- **Maximum packet size successfully sent: 1310 bytes**

Observations:

- The maximum packet size was the same for both `www.iitd.ac.in` and `www.google.com`
- The maximum packet size also did not change even when we increased the timeout limit.
- There was a slight difference in the maximum packet size when we tried this on another system.
- The maximum packet size depends on the Maximum Transmission Unit (MTU) of the network. The MTU depends on a various factors such as the type of network technology, the devices and routers in the path and other configurations.

2. Replicating `traceroute` using `ping`

- We used `bash` for writing the script.
- The user has to input the maximum number of hops and the destination address.
- The script is shown below:

```
#!/bin/usr/env bash
reach=false
for (( i=1 ; i<=$1 ; i++ ));
do
    if [ "$reach" = false ]; then
        echo
        echo -n "$i. "
        for x in {1..3}
        do
            c=$( ping -m $i -c 1 $2 -W 0)
            d=$( echo $c | awk '{print $10}' )
            d=$( echo $d | sed 's/://' )
            # d=${d%?}
            reach=false
            if [[ "$c" == *"0 packets received"* ]]; then
                if [[ "$d" == "statistics" ]]; then
                    echo "Request Timed Out"
                else
                    e=$( ping $d -c 1 -W 1 | tail -n 1 | awk '{print $4}' )
                    e=$( echo $e | sed 's/\//\ /' | awk '{print $1}' )
                    if [[ "$e" != "0" ]]; then
                        echo "$d : $e ms"
                    else
                        echo "$d : X ms"
                    fi
                    # echo "$e"
                fi
            else
                d=$( echo $c | awk '{print $3}' )
                e=$( ping $2 -c 1 -W 1 | tail -n 1 | awk '{print $4}' )
                e=$( echo $e | sed 's/\//\ /' | awk '{print $1}' )
                echo "Destination $2 $d Reached : $e ms"
                reach=true
                # break
            fi
        done
    else
        break
    fi
done

if [ "$reach" = false ]; then
    echo "MAXIMUM HOPS EXHAUSTED"
fi
```

Working of the code

- We use a for loop which runs as many times as the number of maximum hops, setting the `ttl` (Time to Live) equal to the iteration number (starting from 1) and send 1 packet at a time.
- This is done thrice for each value of `ttl`, to get times for 3 packets, similar to `traceroute`.
- We get the IP address where the ttl gets exceeded for each iteration and it is printed to represent that particular hop.
- For getting the time, we ping the IP address found for each hop. This is also done thrice for each hop. If the ping is unsuccessful then X ms is printed.
- From the output of the ping commands, we extract the required information using `tail, awk` and `sed` commands.

Examples

- `www.iitd.ac.in`

```
admin@Amaiyas-MacBook-Air ass1 % bash 2.sh 20 www.iitd.ac.in

1. 10.184.32.13 : 15.513 ms
10.184.32.13 : 19.231 ms
10.184.32.13 : 11.313 ms

2. 10.254.175.1 : 35.712 ms
10.254.175.1 : 9.769 ms
10.254.175.1 : 32.350 ms

3. 10.254.236.2 : 5.762 ms
10.254.236.2 : 8.302 ms
10.254.236.2 : 20.435 ms

4. Destination www.iitd.ac.in (10.10.211.212): Reached : 3.695 ms
Destination www.iitd.ac.in (10.10.211.212): Reached : 10.539 ms
Destination www.iitd.ac.in (10.10.211.212): Reached : 7.241 ms
```

(a) Our script

```
admin@Amaiyas-MacBook-Air ass1 % traceroute www.iitd.ac.in
traceroute to www.iitd.ac.in (10.10.211.212), 64 hops max, 52 byte packets
 1  10.184.32.13 (10.184.32.13)  25.135 ms  7.484 ms  3.430 ms
 2  10.254.175.5 (10.254.175.5)  4.469 ms
    10.254.175.1 (10.254.175.1)  4.149 ms  4.102 ms
 3  10.254.236.26 (10.254.236.26)  3.655 ms  5.035 ms
    10.254.236.2 (10.254.236.2)  3.625 ms
 4  www.iitd.ac.in (10.10.211.212)  3.532 ms  5.277 ms  4.609 ms
```

(b) traceroute

- `www.google.com`

```
admin@Amaiyas-MacBook-Air ass1 % bash 2.sh 20 www.google.com

1. 10.184.32.13 : 13.068 ms
10.184.32.13 : 4.417 ms
10.184.32.13 : 22.841 ms

2. 10.254.175.5 : 11.266 ms
10.254.175.5 : 6.033 ms
10.254.175.5 : 13.669 ms

3. 10.255.1.34 : X ms
10.255.1.34 : X ms
10.255.1.34 : X ms

4. 10.119.233.65 : X ms
10.119.233.65 : X ms
10.119.233.65 : X ms

5. Request Timed Out
Request Timed Out
Request Timed Out

6. Request Timed Out
Request Timed Out
Request Timed Out

7. 10.119.234.162 : X ms
10.119.234.162 : X ms
10.119.234.162 : X ms

8. 72.14.195.56 : 14.434 ms
72.14.195.56 : 13.737 ms
72.14.195.56 : 7.944 ms

9. 108.170.251.97 : 7.190 ms
108.170.251.97 : 8.924 ms
108.170.251.97 : 23.216 ms

10. 142.251.76.169 : X ms
142.251.76.169 : X ms
142.251.76.169 : X ms

11. Destination www.google.com (142.250.207.196): Reached : 43.437 ms
Destination www.google.com (142.250.207.196): Reached : 6.478 ms
Destination www.google.com (142.250.207.196): Reached : 6.493 ms
admin@Amaiyas-MacBook-Air ass1 % traceroute www.google.com
```

(a) Our script

```
admin@Amaiyas-MacBook-Air ass1 % traceroute www.google.com
traceroute to www.google.com (142.250.207.196), 64 hops max, 52 byte packets
 1  10.184.32.13 (10.184.32.13)  15.718 ms  3.253 ms  3.842 ms
 2  10.254.175.1 (10.254.175.1)  3.997 ms
    10.254.175.5 (10.254.175.5)  3.532 ms  3.771 ms
 3  10.255.1.34 (10.255.1.34)  3.917 ms  6.920 ms  4.343 ms
 4  10.119.233.65 (10.119.233.65)  4.051 ms  3.950 ms  5.466 ms
 5  * * *
 6  * * *
 7  10.119.234.162 (10.119.234.162)  16.299 ms  6.487 ms  6.572 ms
 8  72.14.195.56 (72.14.195.56)  9.247 ms
    72.14.194.160 (72.14.194.160)  6.866 ms
    72.14.195.56 (72.14.195.56)  24.727 ms
 9  108.170.251.97 (108.170.251.97)  11.741 ms  12.865 ms  12.068 ms
10  142.251.76.169 (142.251.76.169)  43.937 ms  12.916 ms
    142.251.76.171 (142.251.76.171)  9.217 ms
11  del12s10-in-f4.1e100.net (142.250.207.196)  7.760 ms  12.806 ms  46.199 ms
```

(b) traceroute

3. Internet Architecture

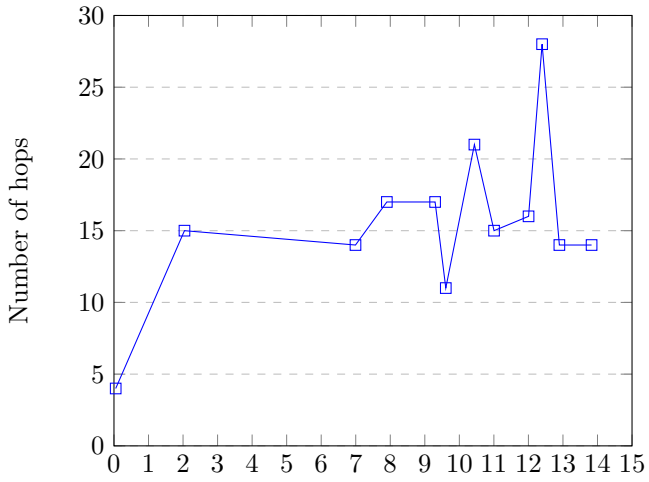
- We chose servers in **Sydney**, **Chicago** and **London** and also ran `traceroute` from a device in Delhi.
- This provides a good set of different geographical locations all around the world.

Part a: Variation in number of hops

- The following table shows the number of hops from 3 different serves to 5 different destinations.

Hostname	Sydney	Chicago	London	Delhi
`www.utah.edu`	14 (12896 km)	15 (2041 km)	17 (7901 km)	28 (12396 km)
`www.uct.ac.za`	15* (11005 km)	14* (13829 km)	11* (9607.7 km)	17 (9298 km)
`www.iitd.ac.in`	21 (10436 km)	16 (12004 km)	14 (6993 km)	16 (50 km)
`www.google.com`	11	11	10	10
`www.facebook.com`	14	11	12	9

Variation of number of hops with geographical distance



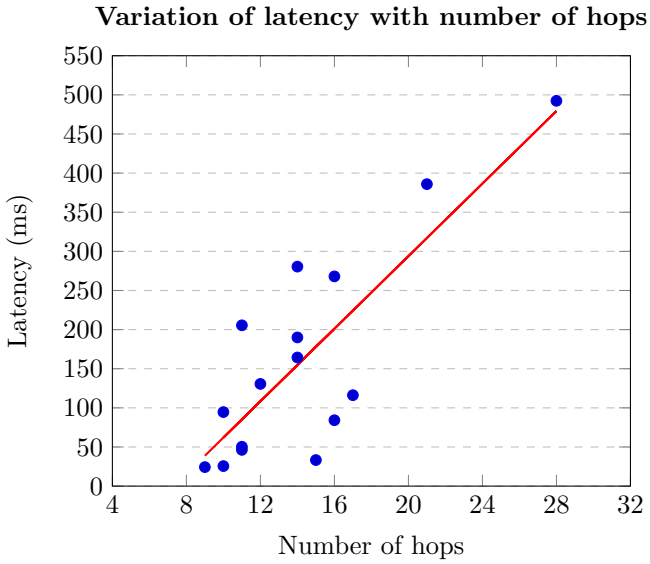
Observations:

- We were not able to reach `www.uct.ac.za` from any of the servers. The last IP reached by all of them was the same so the analysis has been done till that point.
- As the graph shows, there is **no relation** between the number of hops with the geographical distance.
- This is because a hop does not account for the distance between the source and the target. It could take me 10 hops to reach a device in the same room if I connect 10 routers, whereas it could take me just 1 hop to reach another continent.
- While sitting in the campus, it took me 16 hops to `traceroute` to `www.iitd.ac.in`, which is the same as that from Chicago which is over 12000 kms away
- Irrespective of the source, **Google** and **Facebook** take roughly a constant number of hops (10-11) which is less than the number of hops required for the other servers.
- This is because Google and Facebook use a **Content Delivery Network** and have their servers all over the world. The DNS directs the request to the geographically closest server which is optimised for that location. So the number of hops is relatively less.
- This is done so as to reduce latency and improve the delivery of content to the users.

Part b: Variation in latency

- The following table shows the variation in latencies with different source and target locations

Hostname	Sydney	Chicago	London	Delhi
`www.utah.edu`	164.4	33.3	116.2	492.4
`www.uct.ac.za`	380.2*	245*	171.4*	527.6*
`www.iitd.ac.in`	385.9	268	190	84.2
`www.google.com`	205.5	50.3	94.6	25.6
`www.facebook.com`	280.5	46.33	130.6	24.3



Observations:

- As we can observe from the graph, latency **increases** as the number of hops increases.
- Each hop represents a router or a switch that the packet passes through. Each hop adds a delay due to various reasons including
  - **Processing Time:** Each hop requires some time to process the incoming packet and forward it on to the next hop on the path.
  - **Queuing Delay:** The device might be busy processing other packets or there can be congestion at the hop adding to the delays.
  - **Propogation Delay:** Some time is taken to physically traverse the medium (like wireless channels or fibre optic cables) between 2 hops.
- Although there are many other factors affecting the latency other than the number of hops, an increase in the number of hops does imply a higher latency in general.

Part c: Differences in IP of the same host

- The following table shows the IP addresses for different addresses from 4 different sources.

City	www.utah.edu
Sydney	155.98.186.21
Chicago	155.98.186.21
London	155.98.186.21
Delhi	155.98.186.21

City	www.iitd.ac.in
Sydney	2001:df4:e000:29::212
Chicago	2001:df4:e000:29::212
London	2001:df4:e000:29::212
Delhi	2001:df4:e000:29::212

City	www.google.com
Sydney	2607:f8b0:4008:80a::2004
Chicago	2607:f8b0:4005:813::2004
London	2607:f8b0:4008:80a::2004
Delhi	2404:6800:4002:82e::2004

City	www.facebook.com
Sydney	2a03:2880:f12c:183:face:b00c:0:25de
Chicago	2a03:2880:f131:83:face:b00c:0:25de
London	2a03:2880:f131:83:face:b00c:0:25de
Delhi	2a03:2880:f12f:83:face:b00c:0:25de

Observations:

- `www.utah.edu` is resolved to the same IP from **all the locations**.
- `www.iitd.ac.in` is resolved to the same IP from **all the locations**.
- `www.google.com` is resolved to the same IP from Sydney and London.
- `www.facebook.com` is resolved to the same IP from Chicago and London.
- The difference in the IP addresses of the same web-server is because of a technique used by big firms called **Content Deliver Network (CDN)**.
- A large number of servers are setup in several geographical locations. The request is served from the server which is geographically closer to the user to reduce the latency and improve the delivery of content to the user. The DNS directs the user to the CDN server that is optimised for that location.

Part d: Tracerouting to different IP addresses of the same web-server

- The following table contains the number of hops for the different IP addresses of `www.google.com` and `www.facebook.com` with the source location as Delhi.

Hostname	IP Address 1 (Sydney)	IP Address 2 (Chicago)	IP Address 3 (Delhi)
www.google.com	19	16	10
www.facebook.com	27	25	12

Observations:

- The paths are different for the different IP addresses of the same web-server.
- The lengths of the paths are **different** as well as the IP addresses encountered on the way.
- As Sydney and Chicago are thousands of kilometres away, the paths to those IP addresses are much longer than the path taken by the Delhi IP Address.
- This is a clear example of the working of the Content Delivery Network.
- The DNS serves the IP address of the CDN server optimised for our location hence the number of hops required is less when using the Delhi IP address.
- The IP address served by the DNS in Chicago is optimised for that geographical region. Whereas it takes more than twice the number of hops when tracerouting to that address from Delhi. This explains the reason why a CDN is used.

Using AS Lookup

- We tried out performing AS lookup and identifying the local ISP, intermediate ISP and the destination ISP for the following routes and noted these observations.

Route	Local ISP	Intermediate ISP	Destination ISP
`Chicago - IITD`	ASN-TELSTRA-GLOBAL	RELIANCEJIO-IN NKN-CORE-NW NKN NICNET-VSNL-BOARDER-AP	IITDEL-AS-IN
`Sydney - Google`	HURRICANE	-	GOOGLE
`London - Facebook`	HURRICANE	NTT-LTD-2914	FACEBOOK

Part e: Countries with local ISP not directly peered with Google or Facebook

Google

- As observed in the previous parts, irrespective of the source location, it took around 10-11 hops to reach `www.google.com`.
- So if the number of hops is significantly higher than this then it could indicate that the local ISP is not directly peering with Google.
- We also checked if we saw the Google services IP immediately after our local ISP, if this was seen then it indicates that our ISP is peering directly with Google.
- The following table shows our observations.

Country	Hops	Latency(ms)	Direct Peering
`London (UK)`	10	94.6	YES
`Singapore (Singapore)`	9	174.6	YES
`Nairobi (Kenya)`	15	325	NO
`Delhi (India)`	10	25.6	YES
`Bangkok (Thailand)`	15	249.5	NO
`Cape Town (S. Africa)`	16	289.9	NO
`Chicago (USA)`	11	50.3	YES

- Kenya, Thailand and South Africa showed a greater number of hops and significantly higher latency.
- Also, there were more intermediate ISPs involved between the local ISP and Google.

Facebook

- Similarly, as observed in the previous parts it took around 12-14 hops to reach `www.facebook.com`.
- We repeated the same procedure as done for Google.
- The following table shows our observations.

Country	Hops	Latency(ms)	Direct Peering
`London (UK)`	12	130.8	YES
`Singapore (Singapore)`	8	171.3	YES
`Nairobi (Kenya)`	14	309.3	NO
`Delhi (India)`	9	24.3	YES
`Bangkok (Thailand)`	15	264.1	NO
`Cape Town (S. Africa)`	19	226.6	NO
`Chicago (USA)`	11	46.3	YES

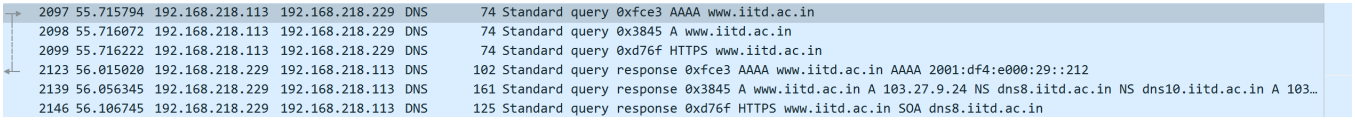
- In this case as well, Kenya, Thailand and South Africa do not seem to have their local ISPs directly peering with Facebook.

4. Wireshark

Link of the Wireshark files: Wireshark Files

Part a: DNS

- The following table shows the DNS queries and responses found for `www.iitd.ac.in`.



- The different types of queries are:
  - **DNS (Domain Name System)**: translates human readable domain names (for example, www.amazon.com) to machine readable IP addresses (for example, 192.0.2.44).
  - **A**: stands for **address** and this is the most fundamental type of DNS record: it indicates the IPv4 address of a given domain.
  - **AAAA**: these records match a domain name to an IPv6 address.
- The HTTPS DNS record provides more detailed information than other record types (like A or AAAA) about the services available for a specific domain. This record type plays a crucial role in establishing secure network connections like `https` by providing essential details.
- For example, HTTPS DNS can communicate information about supported protocols and ports, and can even specify alternate servers to which clients can connect.
- The following shows the request-response time taken.

S.No. of Request	S.No. of Response	Type	Time(s)
2097	2123	`AAAA`	$56.015020 - 55.715794 = 0.299226$
2098	2139	`A`	$56.056345 - 55.716072 = 0.340273$
2099	2146	`HTTPS`	$56.106745 - 55.716222 = 0.390523$

Part b: HTTP

- **HTTP (Hypertext Transfer Protocol)**: the set of rules for transferring files – such as text, images, sound, video and other multimedia files – over the web.
- HTTP functions as a request–response protocol in the client–server model. A web browser, for example, may be the client whereas a process, named web server, running on a computer hosting one or more websites may be the server.
- The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content or performs other functions on behalf of the client, returns a response message to the client. The response contains completion status information about the request and may also contain requested content in its message body.

- The following table shows the HTTP requests and responses for opening `http://act4d.iitd.ac.in`

307	43.310374	192.168.218.113	103.27.9.5	HTTP	579	GET / HTTP/1.1
550	44.583474	103.27.9.5	192.168.218.113	HTTP/XML	372	HTTP/1.1 200 OK
774	46.208767	192.168.218.113	103.27.9.5	HTTP	565	GET /act4d/templates/beez/css/template.css HTTP/1.1
834	47.265488	192.168.218.113	103.27.9.5	HTTP	565	GET /act4d/templates/beez/css/position.css HTTP/1.1
843	47.302105	192.168.218.113	103.27.9.5	HTTP	563	GET /act4d/templates/beez/css/layout.css HTTP/1.1
844	47.302203	192.168.218.113	103.27.9.5	HTTP	564	GET /act4d/templates/beez/css/general.css HTTP/1.1
845	47.302251	192.168.218.113	103.27.9.5	HTTP	546	GET /act4d/media/system/js/mootools.js HTTP/1.1
859	47.325174	192.168.218.113	103.27.9.5	HTTP	545	GET /act4d/media/system/js/caption.js HTTP/1.1
870	47.506902	103.27.9.5	192.168.218.113	HTTP	1171	HTTP/1.1 200 OK (text/css)
878	47.509979	192.168.218.113	103.27.9.5	HTTP	540	GET /wiki1-bak/wiki1/statf0e.php HTTP/1.1
903	47.632041	103.27.9.5	192.168.218.113	HTTP	443	HTTP/1.1 200 OK (text/css)
911	47.673968	103.27.9.5	192.168.218.113	HTTP	77	HTTP/1.1 200 OK (text/css)
930	47.733787	103.27.9.5	192.168.218.113	HTTP	110	HTTP/1.1 200 OK (text/css)
961	47.947916	103.27.9.5	192.168.218.113	HTTP	1362	HTTP/1.1 200 OK (application/javascript)
990	48.061418	103.27.9.5	192.168.218.113	HTTP	604	HTTP/1.1 404 Not Found (text/html)
991	48.065999	192.168.218.113	103.27.9.5	HTTP	611	GET /act4d/templates/beez/images/act4d.png HTTP/1.1
1002	48.248933	103.27.9.5	192.168.218.113	HTTP	1022	HTTP/1.1 200 OK (application/javascript)
1005	48.252784	192.168.218.113	103.27.9.5	HTTP	600	GET /act4d/images/balazahir.jpg HTTP/1.1
1006	48.255712	192.168.218.113	103.27.9.5	HTTP	562	GET /act4d/templates/beez/css/print.css HTTP/1.1
1111	48.590764	103.27.9.5	192.168.218.113	HTTP	1326	HTTP/1.1 200 OK (text/css)
1507	49.428648	103.27.9.5	192.168.218.113	HTTP	661	HTTP/1.1 200 OK (PNG)
1788	50.524196	103.27.9.5	192.168.218.113	HTTP	941	HTTP/1.1 200 OK (JPEG JFIF image)
1790	50.531372	192.168.218.113	103.27.9.5	HTTP	606	GET /act4d/templates/beez/favicon.ico HTTP/1.1

Observations:

- There were **12** HTTP GET requests generated for rendering the website.
  - 1 for /
  - 5 for css files
  - 2 for js files
  - 1 for .php file
  - 1 for .png file
  - 1 for .jpg file
  - 1 for .ico file
- From this we can see that for loading a complex webpage, a number of HTTP requests are generated, seperately for each component of the webpage needed for rendering, such css file, js files or images.
- A fetch request is of the format `GET [path of object to fetch] HTTP/1.1` where 1.1 is the version of HTTP used.
- A response to a request is of the form `HTTP/1.1 [response status] (file type recieved)` for example, 200 OK is the response for a correctly completed request.

Part c: TCP

- **Transmission Control Protocol (TCP)**: a communications standard that enables application programs and computing devices to exchange messages over a network. It is designed to send packets across the internet and ensure the successful delivery of data and messages over networks.
- To make sure that each message reaches its target location intact, the TCP/IP model breaks down the data into small bundles to improve efficiency and afterward reassembles the bundles on the opposite end.
- We used the filter
  `((ip.src==192.168.218.113 && ip.dst==103.27.9.5) ||`
  `(ip.src==103.27.9.5 && ip.dst==192.168.218.113)) && tcp`
  for monitoring the connection and then matched the port used for each HTTP request to the TCP connection.
- The following table shows the Ports and the corresponding HTTP requests handled by each of them.

No.	Port 1	Port 2	S. Nos. of HTTP requests handled
1	50068	80	307
2	50069	80	-
3	50082	80	834
4	50083	80	843
5	50084	80	774, 878, 991
6	50085	80	859, 1006
7	50086	80	845, 1005, 1790
8	50087	80	844

Observations:

- We found that there were 9 TCP connections which is not equal to the HTTP requests (12).
- On checking the port used for HTTP requests we see that multiple content objects may be fetched over the same TCP connection.
- This has multiple benefits. Using the same connection for multiple requests reduces the latency involved in establishment of each connection and improves the efficiency of the network.

Part d: Tracing Indian Express

Observations:

- No HTTP traffic was found when running trace for `http://www.indianexpress.com`.
- We were redirected to `https://indianexpress.com`
- A lot of communication between our browser and webserver was found to be through the TLS protocol, i.e., **Transport Layer Security**
- TLS encrypts the communication and thus we do not see any of the HTML and JS files in the trace as they are encrypted.
- TLS is a widely adopted security protocol designed to facilitate privacy and data security for communications over the Internet.
- A primary use case of TLS is encrypting the communication between web applications and servers, such as web browsers loading a website. TLS can also be used to encrypt other communications such as email, messaging, and voice over IP (VoIP).