



Indian Institute of Technology Gandhinagar

CS331 - Computer Networks

ASSIGNMENT 1

REPORT

Name - Bhavya Parmar, Romit Mohane

Roll No. - 23110059, 231100279

Date of Submission - 15 Sept 2025

Link to [GitHub Repository](#)

Table of Contents

Task 1 - DNS Resolver.....	1
Introduction & Setup.....	1
Procedure.....	1
Result.....	1
Task 2 - Traceroute Protocol Behavior.....	3
Introduction & Setup.....	3
Procedure.....	3
Questions.....	4

Task 1 - DNS Resolver

Introduction & Setup

The purpose of this task is to understand and implement packet parsing and processing logic in a programming language, specifically a custom DNS resolution support, and also give an idea of load balancing and socket programming.

Procedure

After adding the last 3 digits of our roll numbers (279 + 059), we had to select the file '**8.pcap**'.

We used the `struct` Python module to read the packed binary data of each packet and process the DNS packets from the `.pcap` file in our client.

The client, after appending the custom header to the top, sends the DNS query packets to the server's PORT using the `socket` module.

The server listens on this PORT for the incoming packets. On receiving a packet, it parses the header, and resolves the IP using the given rules. It then returns the resolved IP to the client.

The client logs these requests and responses in a `.txt` file.

Result

Running the Client and Server at 4:14 AM (morning slot), we get the following:

Custom header value (HHMMSSID)	Domain name	Resolved IP address
04144000	github.com.	192.168.1.1
04144001	bing.com.	192.168.1.2
04144102	facebook.com.	192.168.1.3
04144103	amazon.com.	192.168.1.4
04144104	linkedin.com.	192.168.1.5
04144105	stackoverflow.com.	192.168.1.1

Running the Client and Server at 2:52 PM (afternoon slot), we get the following:

Custom header value (HHMMSSID)	Domain name	Resolved IP address
14525600	github.com.	192.168.1.6
14525601	bing.com.	192.168.1.7
14525702	facebook.com.	192.168.1.8
14525703	amazon.com.	192.168.1.9
14525804	linkedin.com.	192.168.1.10
14525905	stackoverflow.com.	192.168.1.6

Running the Client and Server at 11:22 PM (night slot), we get the following:

Custom header value (HHMMSSID)	Domain name	Resolved IP address
23222000	github.com.	192.168.1.11
23222001	bing.com.	192.168.1.12
23222102	facebook.com.	192.168.1.13
23222103	amazon.com.	192.168.1.14
23222104	linkedin.com.	192.168.1.15
23222105	stackoverflow.com.	192.168.1.11

Task 2 - Traceroute Protocol Behavior

Introduction & Setup

The purpose of this task is to understand how the traceroute utility works in different operating systems by capturing the network traffic during the executions using Wireshark.

We chose to work on Windows and Linux. And we decided to trace the route to www.dev.to.

Procedure

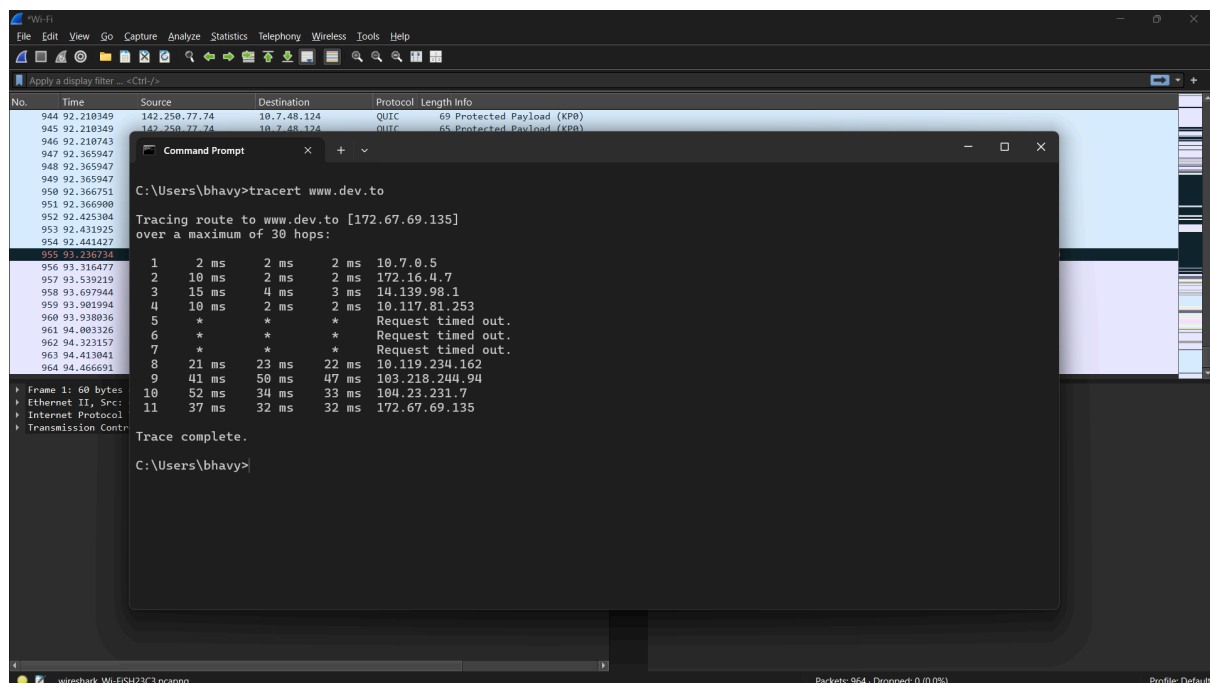
Windows

First, we opened up Wireshark and selected WiFi to capture, following which we ran the command:

```
tracert www.dev.to
```

This started to trace the route to the given destination over a maximum of 30 hops.

Upon completion, I stopped the packet capture in Wireshark and saved the file in .pcapng format.



Linux

In Linux, the procedure was similar to that of Windows... We opened Wireshark and then selected wlan0 (which is the WiFi interface) to capture. Following which we ran the command:

```
tracert www.dev.to
```

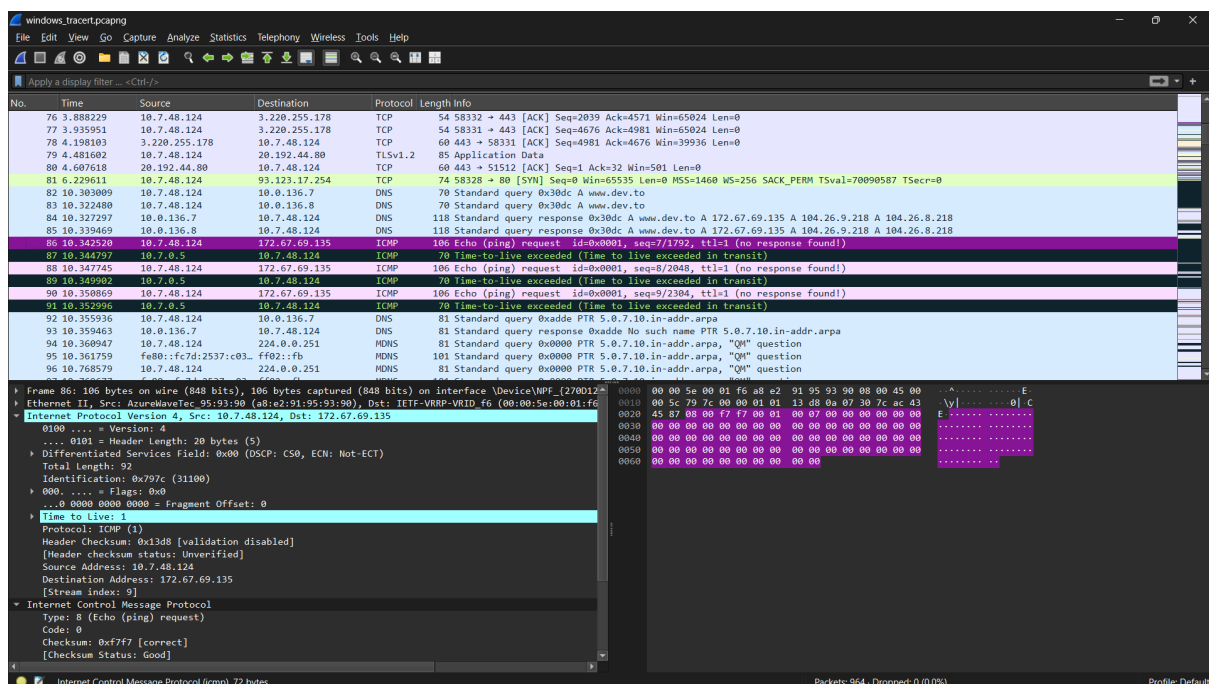
This started to trace the route to the given target over a maximum of 30 hops. And upon completion of this, I stopped the packet capture in Wireshark and saved the file in .pcapng format.

```
> traceroute www.dev.to
traceroute to www.dev.to (172.67.69.135), 30 hops max, 60 byte packets
 1  10.7.0.5 (10.7.0.5)  2.352 ms  2.180 ms  3.247 ms
 2  172.16.4.7 (172.16.4.7)  2.059 ms  2.000 ms  1.945 ms
 3  14.139.98.1 (14.139.98.1)  5.177 ms  5.046 ms  4.979 ms
 4  10.117.81.253 (10.117.81.253)  2.743 ms  2.690 ms  2.597 ms
 5  * * *
 6  * * *
 7  * * *
 8  10.119.234.162 (10.119.234.162)  24.501 ms  26.807 ms  26.757 ms
 9  103.218.244.94 (103.218.244.94)  37.680 ms  37.607 ms  38.709 ms
10  104.23.231.7 (104.23.231.7)  38.716 ms  104.23.231.5 (104.23.231.5)  34.810 ms  104.23.231.7 (104.23.231.7)  34.238 ms
11  172.67.69.135 (172.67.69.135)  36.295 ms  33.222 ms  36.856 ms
```

Questions

1. What protocol does Windows tracert use by default, and what protocol does Linux traceroute use by default?

Ans. Windows tracert uses the ICMP (Internet Control Message Protocol) by default, and sends echo request packets to the destination IP, probe-wise (3 messages in one probe), with the TTL (Time To Live) increasing at each probe. The intermediate routers respond with ICMP “Time Exceeded” messages.



The selected 86th packet is the first packet sent by the tracert operation via the ICMP protocol in the above image.

Whereas Linux traceroute uses UDP (User Datagram Protocol) by default, whilst also sending packets probe-wise (3 messages in one probe) to high-numbered ports (starting at 33434). Intermediate routers reply with ICMP "Time-to-live exceeded," and the destination replies with ICMP "Port Unreachable" (since no service listens on those high ports). The image showing the pcap file for Linux for the same is below:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	104.18.12.149	10.1.144.86	TLSv1.2	105	Application Data
2	0.254443304	10.7.13.177	172.64.148.235	TLSv1.2	105	Application Data
3	0.293563505	172.64.148.235	10.7.13.177	TLSv1.2	105	Application Data
4	0.293640788	10.7.13.177	172.64.148.235	TCP	66	59000 → 443 [ACK] Seq=40 Ack=62 Min=62 Len=0 TSval=2045687729 TSecr=400036093
5	3.384008371	172.64.148.235	10.7.13.177	TLSv1.2	90	Application Data
6	3.384367682	10.7.13.177	172.64.148.235	TLSv1.2	90	Application Data
7	3.422163818	172.64.148.235	10.7.13.177	TCP	66	443 → 59000 [ACK] Seq=25 Ack=29 Min=19 Len=0 TSval=593932160 TSecr=2045690820
8	3.967651260	10.7.13.177	10.0.136.7	DNS	70	Standard query 0x4216 A www.dev.to
9	3.967683586	10.7.13.177	10.0.136.7	DNS	70	Standard query 0x2a00 AAAA www.dev.to
10	3.989127159	10.0.136.7	10.7.13.177	DNS	154	Standard query response 0x2a00 AAAA www.dev.to AAAA 2606:4700:20::681a:9da AAAA 2606:4700:20::ac43:4587 AAAA 2606:4700:20::681a:8
11	3.990934269	10.0.136.7	10.7.13.177	DNS	110	Standard query response 0x4216 A www.dev.to A 172.67.69.135 A 104.26.9.218 A 104.26.8.218
12	3.991357734	10.7.13.177	172.67.69.135	UDP	74	51032 → 33434 Len=32
13	3.991466122	10.7.13.177	172.67.69.135	UDP	74	58779 → 33435 Len=32
14	3.991526666	10.7.13.177	172.67.69.135	UDP	74	55572 → 33436 Len=32
15	3.991638074	10.7.13.177	172.67.69.135	UDP	74	59489 → 33437 Len=32
16	3.991642405	10.7.13.177	172.67.69.135	UDP	74	35214 → 33438 Len=32
17	3.991697453	10.7.13.177	172.67.69.135	UDP	74	53750 → 33439 Len=32
18	3.991753675	10.7.13.177	172.67.69.135	UDP	74	53930 → 33440 Len=32
19	3.991807932	10.7.13.177	172.67.69.135	UDP	74	47316 → 33441 Len=32
20	3.991951635	10.7.13.177	172.67.69.135	UDP	74	40630 → 33442 Len=32
21	3.992008043	10.7.13.177	172.67.69.135	UDP	74	48546 → 33443 Len=32
22	3.992096464	10.7.13.177	172.67.69.135	UDP	74	41233 → 33444 Len=32
23	3.992153260	10.7.13.177	172.67.69.135	UDP	74	41594 → 33445 Len=32
24	3.992202668	10.7.13.177	172.67.69.135	UDP	74	32889 → 33446 Len=32
25	3.992292063	10.7.13.177	172.67.69.135	UDP	74	36489 → 33447 Len=32
26	3.992367006	10.7.13.177	172.67.69.135	UDP	74	49787 → 33448 Len=32
27	3.992428887	10.7.13.177	172.67.69.135	UDP	74	41003 → 33449 Len=32
28	3.993222802	10.7.0.5	10.7.13.177	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
29	3.993622849	10.7.0.5	10.7.13.177	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
30	3.993622970	172.16.4.7	10.7.13.177	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
31	3.993623076	172.16.4.7	10.7.13.177	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
32	3.993623182	172.16.4.7	10.7.13.177	ICMP	102	Time-to-live exceeded (Time to live exceeded in transit)
33	3.994517001	10.7.13.177	10.0.136.7	DNS	81	Standard query 0xeeef PTR 5.0.7.10.in-addr.arpa
34	3.994732506	10.117.81.253	10.7.13.177	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
35	3.994732980	10.117.81.253	10.7.13.177	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
36	3.994733089	10.117.81.253	10.7.13.177	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
37	3.994733372	10.7.0.5	10.7.13.177	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
38	3.996911221	14.139.98.1	10.7.13.177	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
39	3.996911512	14.139.98.1	10.7.13.177	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
40	3.996911641	14.139.98.1	10.7.13.177	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)

2. Some hops in your traceroute output may show *. Provide at least two reasons why a router might not reply.**

Ans. In my tracert to www.dev.to, hops 5, 6, and 7 displayed ***, indicating that no response was received from those routers. This does not mean the route was broken, since subsequent hops (8 onwards) continued to respond, and the tracert was still able to reach the destination successfully. So, mostly the reasons could be the following:

- ➔ The ICMP responses for diagnostics using tracert may be considered of lower priority than the actual traffic data on that router, and hence may be dropped.
- ➔ A router may forward the packet on, but the ICMP response it generates may take a different return path that doesn't make it back to us.
- ➔ Some routers may be configured to ignore TTL-expired packets entirely to avoid exposing their internal topology.

tracert and captured packets in Wireshark for Windows:

```
Tracing route to www.dev.to [172.67.69.135]
over a maximum of 30 hops:
```

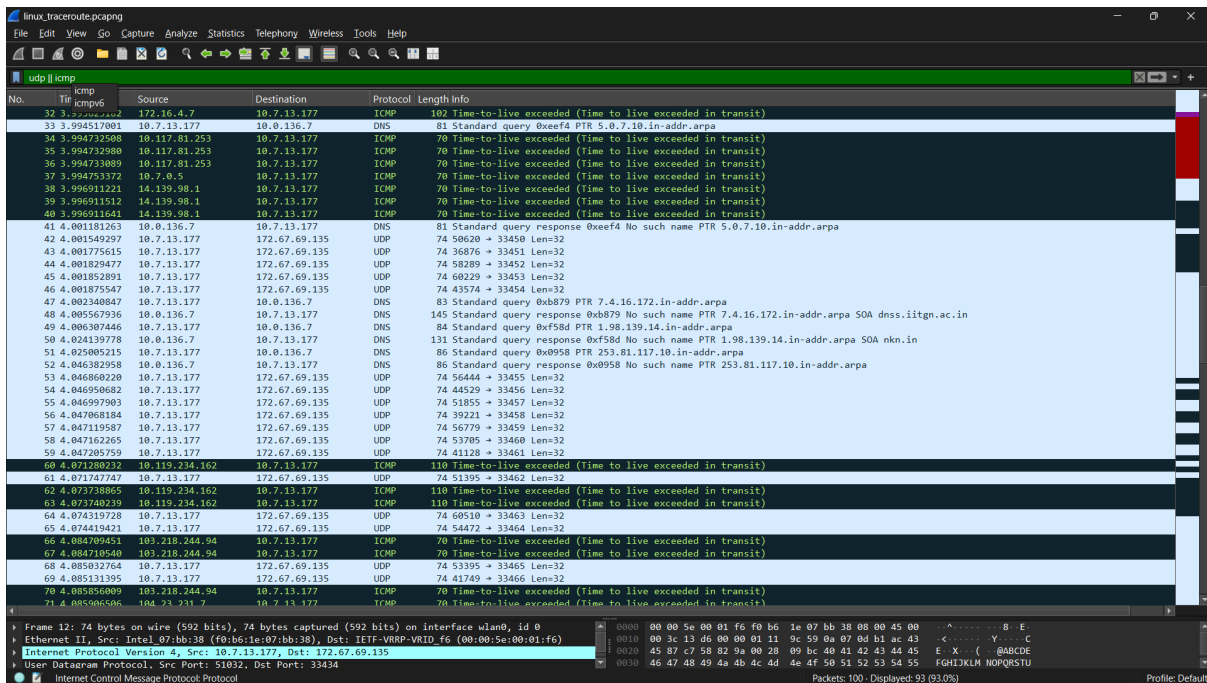
```
  0  0 ms  0 ms  0 ms  10.7.0.5
  1  10 ms  2 ms  2 ms  172.16.4.7
  2  15 ms  4 ms  3 ms  14.139.98.1
  3  10 ms  2 ms  2 ms  10.117.81.253
  4  *      *      *      Request timed out.
  5  *      *      *      Request timed out.
  6  *      *      *      Request timed out.
  7  21 ms  23 ms  22 ms  10.119.234.162
  8  41 ms  50 ms  47 ms  103.218.244.94
  9  52 ms  34 ms  33 ms  104.23.231.7
 10  37 ms  32 ms  32 ms  172.67.69.135
```

Trace complete.

```
366 33.017140 10.7.48.124 172.67.69.135 ICMP 106 Echo (ping) request id=0x0001, seq=19/4864, ttl=5 (no response found!)
378 36.720374 10.7.48.124 172.67.69.135 ICMP 106 Echo (ping) request id=0x0001, seq=20/5120, ttl=5 (no response found!)
384 40.719946 10.7.48.124 172.67.69.135 ICMP 106 Echo (ping) request id=0x0001, seq=21/5376, ttl=5 (no response found!)
391 44.726594 10.7.48.124 172.67.69.135 ICMP 106 Echo (ping) request id=0x0001, seq=22/5632, ttl=6 (no response found!)
460 48.719964 10.7.48.124 172.67.69.135 ICMP 106 Echo (ping) request id=0x0001, seq=23/5888, ttl=6 (no response found!)
508 52.712599 10.7.48.124 172.67.69.135 ICMP 106 Echo (ping) request id=0x0001, seq=24/6144, ttl=6 (no response found!)
594 56.723420 10.7.48.124 172.67.69.135 ICMP 106 Echo (ping) request id=0x0001, seq=25/6400, ttl=7 (no response found!)
612 60.710026 10.7.48.124 172.67.69.135 ICMP 106 Echo (ping) request id=0x0001, seq=26/6656, ttl=7 (no response found!)
625 64.723878 10.7.48.124 172.67.69.135 ICMP 106 Echo (ping) request id=0x0001, seq=27/6912, ttl=7 (no response found!)
631 68.712645 10.7.48.124 172.67.69.135 ICMP 106 Echo (ping) request id=0x0001, seq=28/7168, ttl=8 (no response found!)
632 68.734196 10.119.234.162 10.7.48.124 ICMP 110 Time-to-live exceeded (Time to live exceeded in transit)
```

traceroute and captured packets in Wireshark for Linux:

```
> traceroute www.dev.to
traceroute to www.dev.to (172.67.69.135), 30 hops max, 60 byte packets
 0 10.7.0.5 (10.7.0.5) 2.352 ms 2.180 ms 3.247 ms
 1 172.16.4.7 (172.16.4.7) 2.059 ms 2.000 ms 1.945 ms
 2 14.139.98.1 (14.139.98.1) 5.177 ms 5.046 ms 4.979 ms
 3 10.117.81.253 (10.117.81.253) 2.743 ms 2.690 ms 2.597 ms
 4 * * *
 5 * * *
 6 * * *
 7 * * *
 8 10.119.234.162 (10.119.234.162) 24.501 ms 26.807 ms 26.757 ms
 9 103.218.244.94 (103.218.244.94) 37.680 ms 37.607 ms 38.709 ms
10 104.23.231.7 (104.23.231.7) 38.716 ms 104.23.231.5 (104.23.231.5) 34.810 ms 104.23.231.7 (104.23.231.7) 34.238 ms
11 172.67.69.135 (172.67.69.135) 36.295 ms 33.222 ms 36.856 ms
```

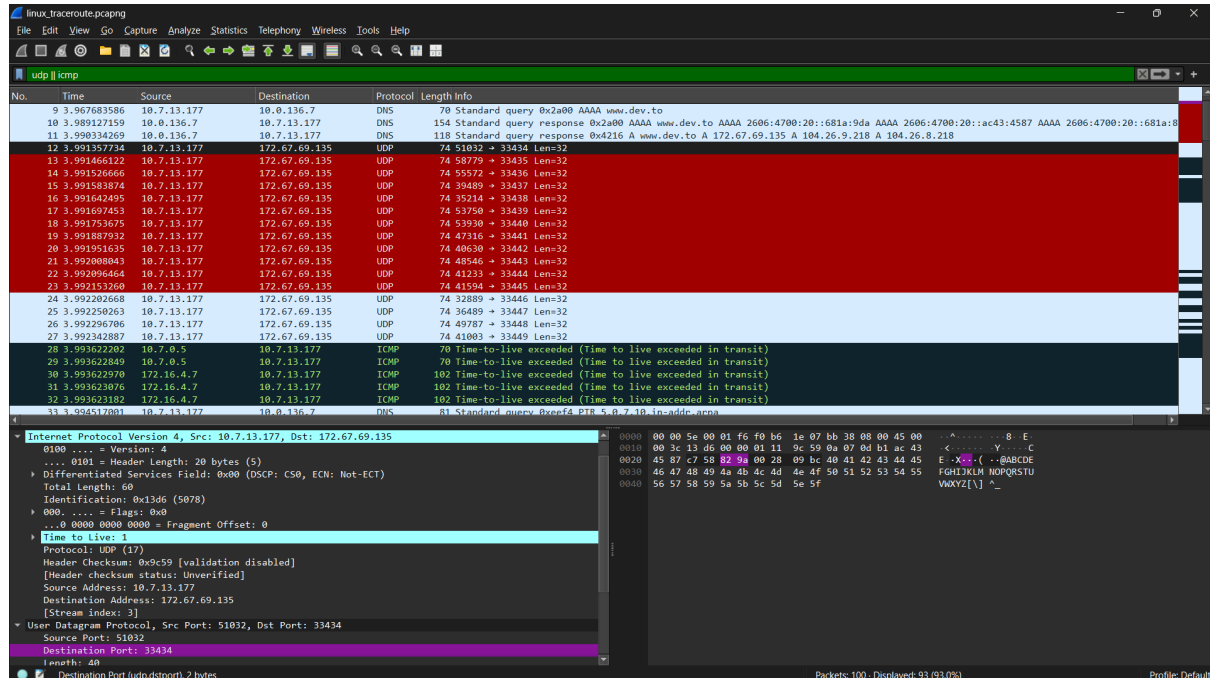


Here we can see that the packets 34, 35, 36 are received from IP 10.117.81.253, which is at hop 4, giving the ICMP TTL exceeded message, and then there is a large gap before it receives the same from the IP of hop 8.

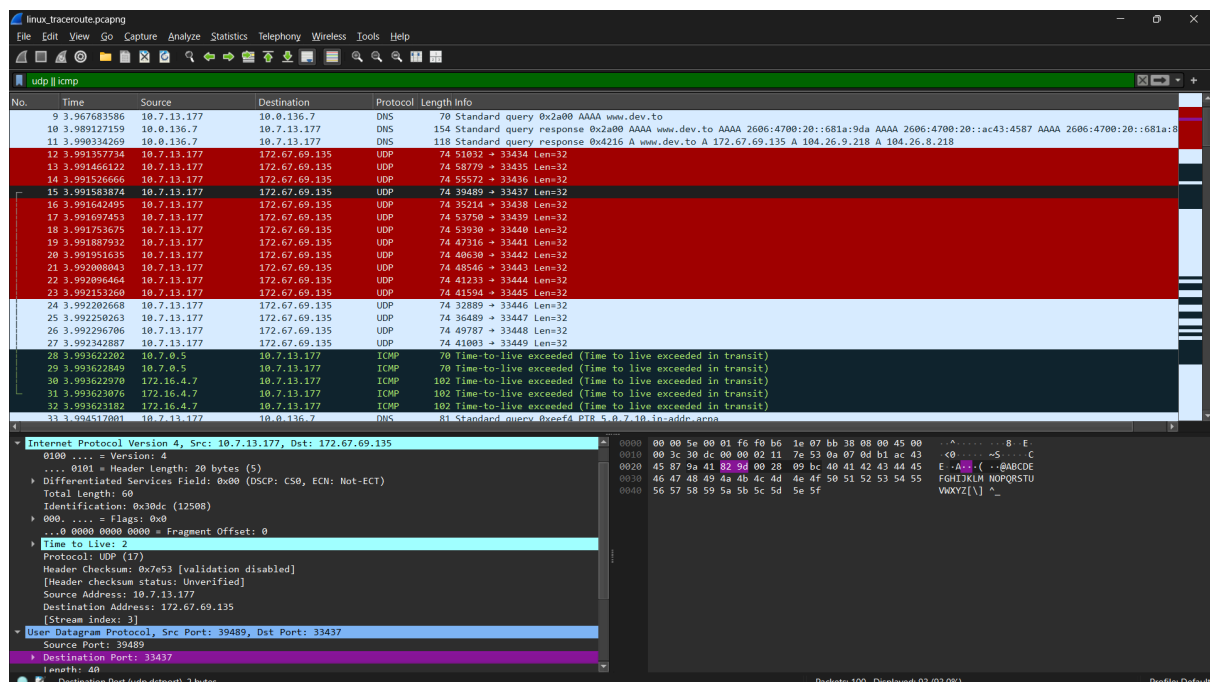
3. In Linux traceroute, which field in the probe packets changes between successive probes sent to the destination?

Ans. In Linux traceroute, two fields change:

- The TTL (Time-To-Live) field → increments for each hop (1, 2, 3, ...).
- The UDP destination port number → changes for each probe, starting at 33434 and incrementing (33435, 33436, ...).



In the above image, we can see that the TTL is 1 and the destination port is 33434 (this is for packet 12, which is the first UDP packet sent of hop 1). And in the below image, we can see that the TTL is 2 and the destination port is 33435 (which is for packet 15, corresponding to the first packet sent of hop 2).



4. At the final hop, how is the response different compared to the intermediate hop?

Ans. In Windows, at the final hop, each ICMP echo request of the final hop receives an echo reply from the destination IP.

869	85.607921	10.7.48.124	172.67.69.135	ICMP	106 Echo (ping) request	id=0x0001, seq=37/9472, ttl=11 (reply in 870)
870	85.645357	172.67.69.135	10.7.48.124	ICMP	106 Echo (ping) reply	id=0x0001, seq=37/9472, ttl=53 (request in 869)
871	85.646247	10.7.48.124	172.67.69.135	ICMP	106 Echo (ping) request	id=0x0001, seq=38/9728, ttl=11 (reply in 872)
872	85.678630	172.67.69.135	10.7.48.124	ICMP	106 Echo (ping) reply	id=0x0001, seq=38/9728, ttl=53 (request in 871)
873	85.679834	10.7.48.124	172.67.69.135	ICMP	106 Echo (ping) request	id=0x0001, seq=39/9984, ttl=11 (reply in 874)
874	85.711903	172.67.69.135	10.7.48.124	ICMP	106 Echo (ping) reply	id=0x0001, seq=39/9984, ttl=53 (request in 873)

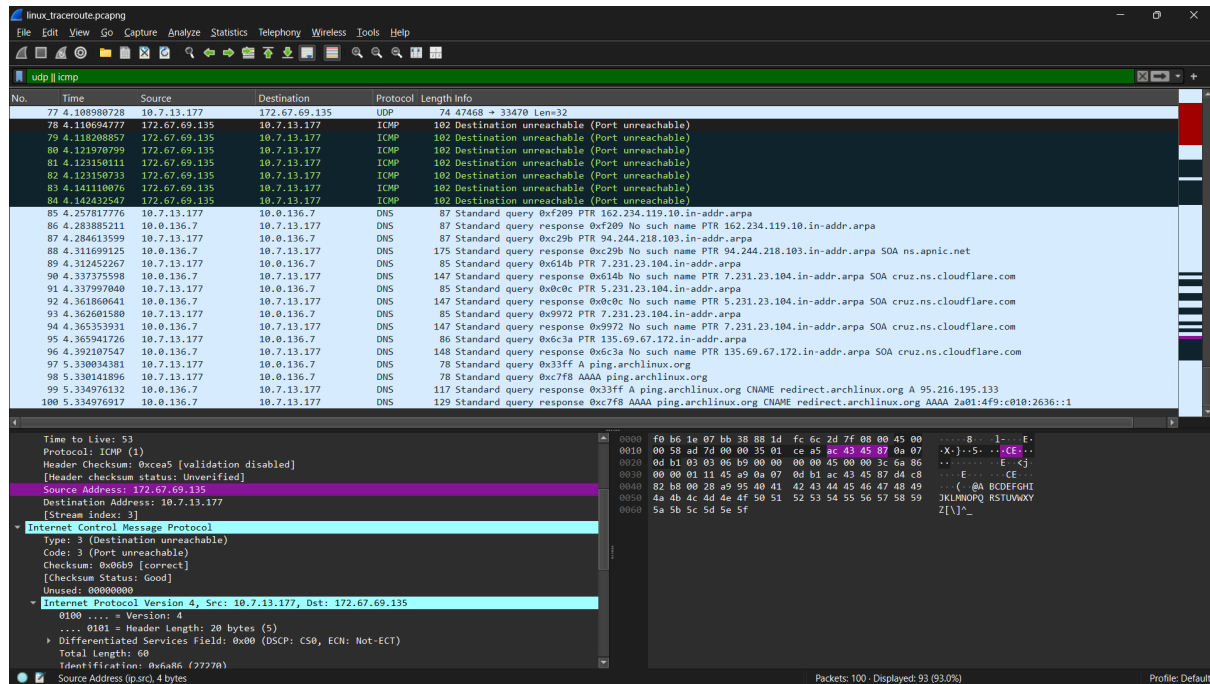
Whereas at the intermediate hops, each ICMP echo request was followed by an ICMP error message saying “Time-to-live exceeded”, as the packets didn’t reach the destination within the fixed TTL.

126	16.303820	10.7.48.124	172.67.69.135	ICMP	106 Echo (ping) request	id=0x0001, seq=10/2560, ttl=2 (no response found!)
127	16.314366	172.16.4.7	10.7.48.124	ICMP	134 Time-to-live exceeded	(Time to live exceeded in transit)
128	16.316690	10.7.48.124	172.67.69.135	ICMP	106 Echo (ping) request	id=0x0001, seq=11/2816, ttl=2 (no response found!)
129	16.318724	172.16.4.7	10.7.48.124	ICMP	134 Time-to-live exceeded	(Time to live exceeded in transit)
130	16.319976	10.7.48.124	172.67.69.135	ICMP	106 Echo (ping) request	id=0x0001, seq=12/3072, ttl=2 (no response found!)
131	16.321826	172.16.4.7	10.7.48.124	ICMP	134 Time-to-live exceeded	(Time to live exceeded in transit)
145	21.889113	10.7.48.124	172.67.69.135	ICMP	106 Echo (ping) request	id=0x0001, seq=13/3328, ttl=3 (no response found!)
146	21.904199	14.139.98.1	10.7.48.124	ICMP	70 Time-to-live exceeded	(Time to live exceeded in transit)
147	21.906945	10.7.48.124	172.67.69.135	ICMP	106 Echo (ping) request	id=0x0001, seq=14/3584, ttl=3 (no response found!)
148	21.911332	14.139.98.1	10.7.48.124	ICMP	70 Time-to-live exceeded	(Time to live exceeded in transit)
149	21.912971	10.7.48.124	172.67.69.135	ICMP	106 Echo (ping) request	id=0x0001, seq=15/3840, ttl=3 (no response found!)
150	21.916788	14.139.98.1	10.7.48.124	ICMP	70 Time-to-live exceeded	(Time to live exceeded in transit)

Whereas in Linux, at intermediate hops, routers send back ICMP TTL exceeded messages when the TTL reaches zero. We can see that in the image below that packet 28 is the ICMP TTL exceeded message received for our first UDP packet sent on hop 1, which had a TTL of 1 and destination port 33434.

The screenshot shows the output of the `linux.traceroute` command. The main table displays the sequence of hops, including source and destination IP addresses, protocols, and lengths. The path starts from 10.7.13.177 and ends at 10.7.13.177. The final hop (74) shows a successful UDP packet reaching the destination port 33434, followed by an ICMP Port Unreachable error message (Type 3, Code 3). The detailed view of the final hop shows the packet details, including the source and destination addresses, the destination port, and the ICMP error message.

And at the final hop (destination), since the UDP probe reaches the host but no service is listening on the high port, the host replies with an ICMP Port Unreachable error message (Type 3, Code 3). We can see that in the image below, the packet 74 comes straight from our target IP.



5. Suppose a firewall blocks UDP traffic but allows ICMP — how would this affect the results of Linux traceroute vs. Windows tracert?

Ans.

- Linux traceroute (UDP-based): The probes would never reach the destination. No ICMP Port Unreachable responses would come back, so traceroute would fail or hang.
- Windows tracert (ICMP-based): It would still work correctly, since ICMP Echo Requests and Replies are allowed through the firewall.

Thus, Windows tracert is more resilient in this scenario.