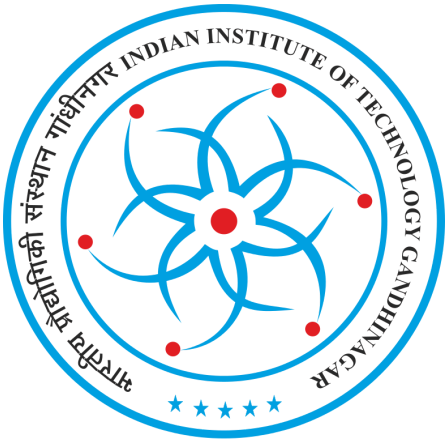
# Indian Institute of Technology, Gandhinagar




# CS - 331 Assignment - 1

# Computer Networks

Banoth Arvind Nayak 23110058

Rushitha Thipparapu 23110338

## Task-1: DNS Resolver

```
DNS Queries found in 6.pcap:
1. _apple-mobdev._tcp.local.
2. _apple-mobdev._tcp.local.
3. linkedin.com.
4. reddit.com.
5. facebook.com.
6. Brother MFC-7860DW._pdl-datastream._tcp.local.
7. Brother MFC-7860DW._pdl-datastream._tcp.local.
8. bing.com.
9. Brother MFC-7860DW._pdl-datastream._tcp.local.
10. Brother MFC-7860DW._pdl-datastream._tcp.local.
```

```
PS C:\Users\thipp\OneDrive\Desktop\CN_Task_1> python server.py
[SERVER] Listening on 127.0.0.1:9999
[RESOLVED] linkedin.com -> 192.168.1.6
[RESOLVED] reddit.com -> 192.168.1.7
[RESOLVED] facebook.com -> 192.168.1.8
[RESOLVED] bing.com -> 192.168.1.9
[RESOLVED] example.com -> 192.168.1.10
[RESOLVED] wikipedia.org -> 192.168.1.6
[RESOLVED] github.com -> 192.168.1.7
```

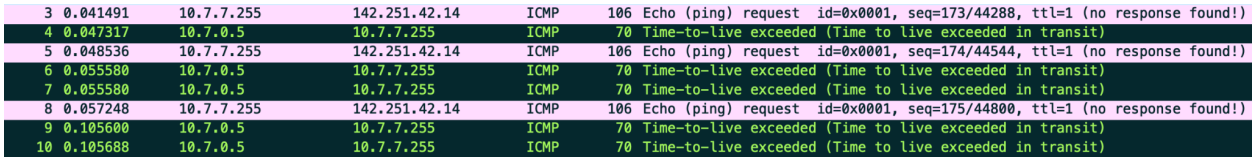
```
PS C:\Users\thipp\OneDrive\Desktop\CN_Task_1> python client.py
[CLIENT] 18041600 | linkedin.com -> 192.168.1.6
[CLIENT] 18041601 | reddit.com -> 192.168.1.7
[CLIENT] 18041602 | facebook.com -> 192.168.1.8
[CLIENT] 18041603 | bing.com -> 192.168.1.9
[CLIENT] 18041604 | example.com -> 192.168.1.10
[CLIENT] 18041605 | wikipedia.org -> 192.168.1.6
[CLIENT] 18041606 | github.com -> 192.168.1.7

[CLIENT] Report saved to report.csv (clean queries only).
```

## Task-2: Traceroute Protocol Behaviour

### Question 1: What protocol does Windows tracert use by default, and what protocol does Linux/macOS traceroute use by default?

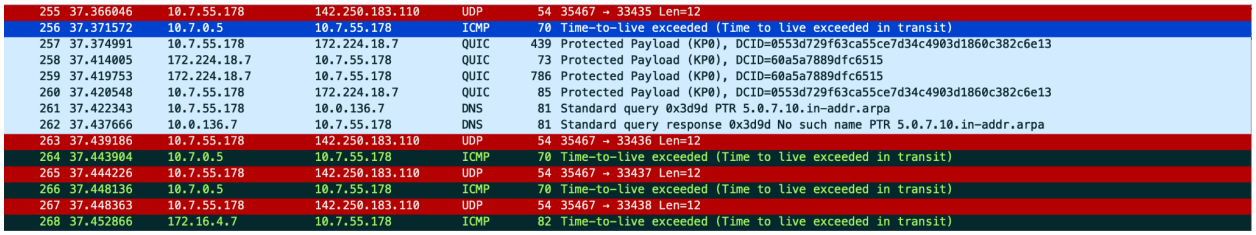
Based on network packet analysis, the default protocols are different for each operating system:

- **Windows tracert:** The Windows tracert utility sends **ICMP (Internet Control Message Protocol)** "Echo Request" packets as its probes.



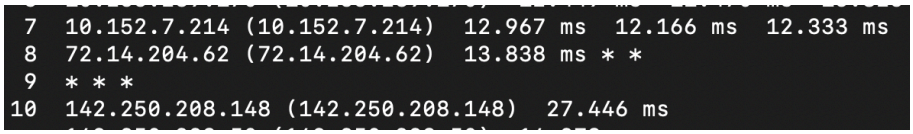
| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 0.041491 | 10.7.7.255 | 142.251.42.14 | ICMP | 106 | Echo (ping) request id=0x0001, seq=173/44288, ttl=1 (no response found!) |
| 4 | 0.047317 | 10.7.0.5 | 10.7.7.255 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 5 | 0.048536 | 10.7.7.255 | 142.251.42.14 | ICMP | 106 | Echo (ping) request id=0x0001, seq=174/44544, ttl=1 (no response found!) |
| 6 | 0.055580 | 10.7.0.5 | 10.7.7.255 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 7 | 0.055580 | 10.7.0.5 | 10.7.7.255 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 8 | 0.057248 | 10.7.7.255 | 142.251.42.14 | ICMP | 106 | Echo (ping) request id=0x0001, seq=175/44800, ttl=1 (no response found!) |
| 9 | 0.105600 | 10.7.0.5 | 10.7.7.255 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 10 | 0.105688 | 10.7.0.5 | 10.7.7.255 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |

- **macOS/Linux traceroute:** The traceroute utility on macOS and Linux sends **UDP (User Datagram Protocol)**packets to high-numbered ports as its probes.



| | | | | | | |
|---|---|---|---|---|---|---|
| 255 | 37.366046 | 10.7.55.178 | 142.250.183.110 | UDP | 54 | 35467 → 33435 Len=12 |
| 256 | 37.371572 | 10.7.0.5 | 10.7.55.178 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 257 | 37.374991 | 10.7.55.178 | 172.224.18.7 | QUIC | 439 | Protected Payload (KP0), DCID=0553d729f63ca55ce7d34c4903d1860c382c6e13 |
| 258 | 37.414005 | 172.224.18.7 | 10.7.55.178 | QUIC | 73 | Protected Payload (KP0), DCID=60a5a7889dfc6515 |
| 259 | 37.419753 | 172.224.18.7 | 10.7.55.178 | QUIC | 786 | Protected Payload (KP0), DCID=60a5a7889dfc6515 |
| 260 | 37.420548 | 10.7.55.178 | 172.224.18.7 | QUIC | 85 | Protected Payload (KP0), DCID=0553d729f63ca55ce7d34c4903d1860c382c6e13 |
| 261 | 37.422343 | 10.7.55.178 | 10.0.136.7 | DNS | 81 | Standard query 0x3d9d PTR 5.0.7.10.in-addr.arpa |
| 262 | 37.437666 | 10.0.136.7 | 10.7.55.178 | DNS | 81 | Standard query response 0x3d9d No such name PTR 5.0.7.10.in-addr.arpa |
| 263 | 37.439186 | 10.7.55.178 | 142.250.183.110 | UDP | 54 | 35467 → 33436 Len=12 |
| 264 | 37.443904 | 10.7.0.5 | 10.7.55.178 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 265 | 37.444226 | 10.7.55.178 | 142.250.183.110 | UDP | 54 | 35467 → 33437 Len=12 |
| 266 | 37.448136 | 10.7.0.5 | 10.7.55.178 | ICMP | 70 | Time-to-live exceeded (Time to live exceeded in transit) |
| 267 | 37.448363 | 10.7.55.178 | 142.250.183.110 | UDP | 54 | 35467 → 33438 Len=12 |
| 268 | 37.452866 | 172.16.4.7 | 10.7.55.178 | ICMP | 82 | Time-to-live exceeded (Time to live exceeded in transit) |

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



```
 7  10.152.7.214 (10.152.7.214)  12.967 ms  12.166 ms  12.333 ms
 8  72.14.204.62 (72.14.204.62)  13.838 ms * *
 9  * * *
10  142.250.208.148 (142.250.208.148)  27.446 ms
```

* * * symbols, as seen in hop 9 of the macOS traceroute, indicate that no reply was received from the router at that hop. There are two primary reasons for this:

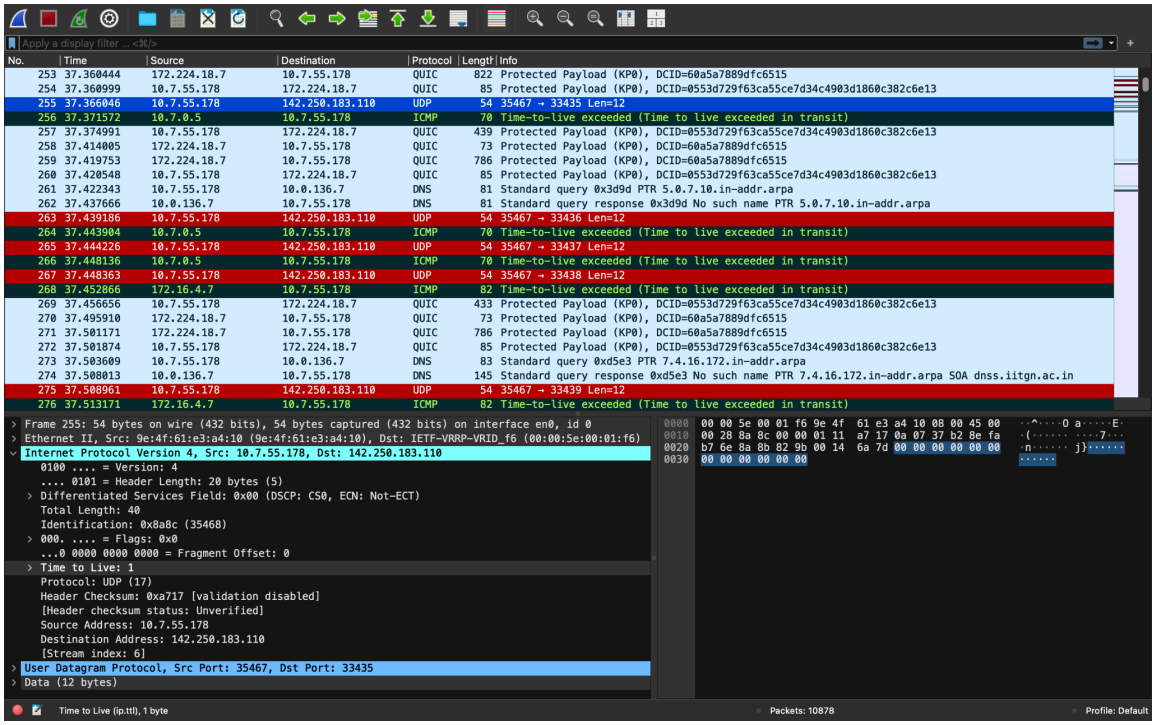
- Firewall & Security Rules: The router or a downstream firewall is often configured to deliberately drop ICMP or UDP probe packets from the public internet. This is a security measure to prevent network reconnaissance.
- ICMP Rate Limiting: To prevent being overwhelmed by requests (as in a Denial-of-Service attack), many routers are configured to only respond to a certain number of diagnostic packets per second. If our probes arrive when the router's limit has been reached, they will be ignored.

## Question 3: In Linux/MacOS traceroute, which field in the probe packets changes between successive probes sent to the destination?

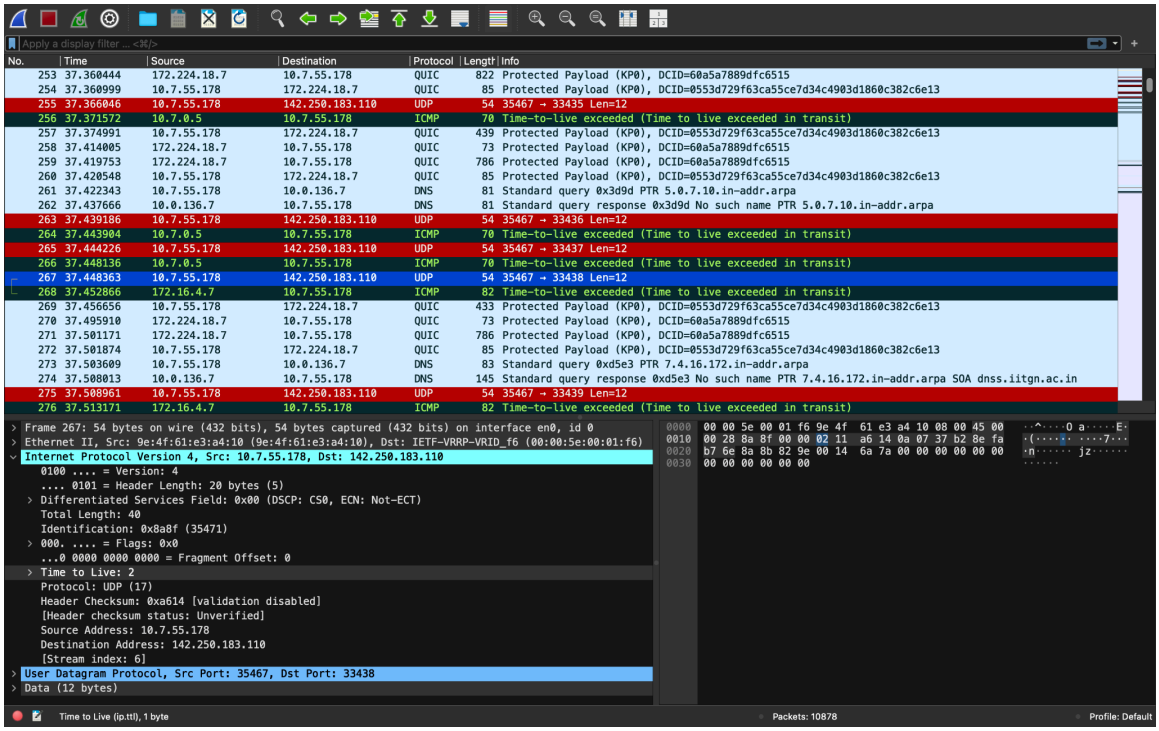
In macOS and Linux traceroute, the field that changes in the IP header of each successive outgoing probe is the Time to Live (TTL).

The first set of probes is sent with a TTL of 1. The second set is sent with a TTL of 2, and so on. When a router receives a packet, it decrements the TTL. If the TTL becomes 0, the router discards the packet and sends an ICMP "Time-to-live exceeded" message back to the source. This is the core mechanism that allows traceroute to discover each hop.

**Screenshot:The first outgoing UDP probe with Internet Protocol Version 4 section and Time to Live: 1" field.**




**Screenshot:The first outgoing UDP probe with Internet Protocol Version 4 section and Time to Live: 2" field.**




## Question 4: At the final hop, how is the response different compared to the intermediate hop?

```
banotharvindnayak@BANOTHs-MacBook-Pro ~ % traceroute www.youtube.com
traceroute: Warning: www.youtube.com has multiple addresses; using 142.250.183.110
traceroute to youtube-ui.l.google.com (142.250.183.110), 64 hops max, 40 byte packets
 1  10.7.0.5 (10.7.0.5)  7.512 ms  4.996 ms  4.138 ms
 2  172.16.4.7 (172.16.4.7)  4.679 ms  4.427 ms  3.179 ms
 3  14.139.98.1 (14.139.98.1)  6.214 ms  7.290 ms  6.514 ms
 4  10.117.81.253 (10.117.81.253)  4.133 ms  4.464 ms  4.447 ms
 5  10.154.8.137 (10.154.8.137)  13.299 ms  12.325 ms  12.962 ms
 6  10.255.239.170 (10.255.239.170)  12.449 ms  11.470 ms  15.825 ms
 7  10.152.7.214 (10.152.7.214)  12.967 ms  12.166 ms  12.333 ms
 8  72.14.204.62 (72.14.204.62)  13.838 ms * *
 9  * * *
10  142.250.208.148 (142.250.208.148)  27.446 ms
    142.250.228.50 (142.250.228.50)  14.278 ms
    142.251.77.100 (142.251.77.100)  15.699 ms
11  142.250.226.134 (142.250.226.134)  14.126 ms  14.646 ms  58.232 ms
12  bom12s13-in-f14.1e100.net (142.250.183.110)  14.275 ms  13.468 ms  14.017 ms
```

Intermediate Hop: As seen in our  screenshots, an intermediate router ( hop 8, 72.14.204.62) replies with an ICMP "Time-to-live exceeded" message. This happens because the router decremented the packet's TTL to zero.

```
PS C:\Users\saila> tracert www.youtube.com

Tracing route to youtube-ui.l.google.com [142.251.42.14]
over a maximum of 30 hops:

  1     6 ms     7 ms    48 ms  10.7.0.5
  2    44 ms     7 ms    10 ms  172.16.4.7
  3    11 ms    27 ms     7 ms  14.139.98.1
  4    40 ms    85 ms    18 ms  10.117.81.253
  5    10 ms    12 ms    12 ms  10.154.8.137
  6    11 ms    13 ms    11 ms  10.255.239.170
  7    66 ms    13 ms    46 ms  10.152.7.214
  8    11 ms    11 ms    14 ms  72.14.204.62
  9    11 ms    12 ms    12 ms  142.251.49.177
 10    12 ms    10 ms    12 ms  209.85.248.61
 11    11 ms    12 ms    12 ms  bom12s19-in-f14.1e100.net [142.251.42.14]

Trace complete.
```

Final Hop: The final destination (hop 11 in Windows trace, bom12s19-in-f14.1e100.net) receives the probe packet with a TTL greater than zero.

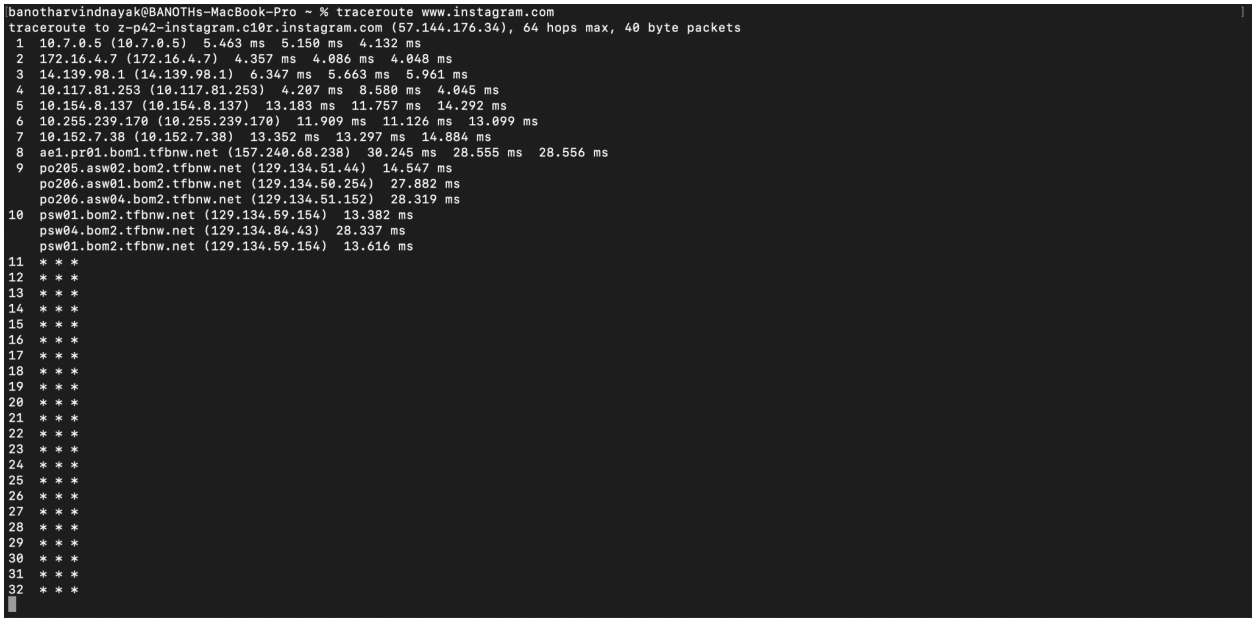
- For Windows tracert, the destination server receives an ICMP Echo Request and sends back a standard ICMP "Echo reply".
- For macOS traceroute, the destination server receives the UDP packet on a high-numbered, unused port. Its operating system responds with an ICMP "Destination unreachable (Port unreachable)" message. This specific "unreachable" message signals to traceroute that it has successfully reached the end.

## Question 5: Suppose a firewall blocks UDP traffic but allows ICMP. How would this affect the results of Linux traceroute vs. Windows tracert?

A firewall that blocks UDP traffic but allows ICMP would have opposite effects on the two utilities.

- Linux/macOS traceroute: This utility would fail. Its UDP probe packets would be blocked by the firewall. The command would time out and show * * * for all hops beyond the firewall because it would never receive any replies.
- Windows tracert: This utility would succeed. Since its probes are ICMP packets and the firewall allows ICMP, the trace would pass through the firewall without issue and likely complete successfully.

If traceroute encountered such a firewall, the output would look similar to the trace to www.instagram.com, where all hops after a certain point are blocked and result in timeouts.



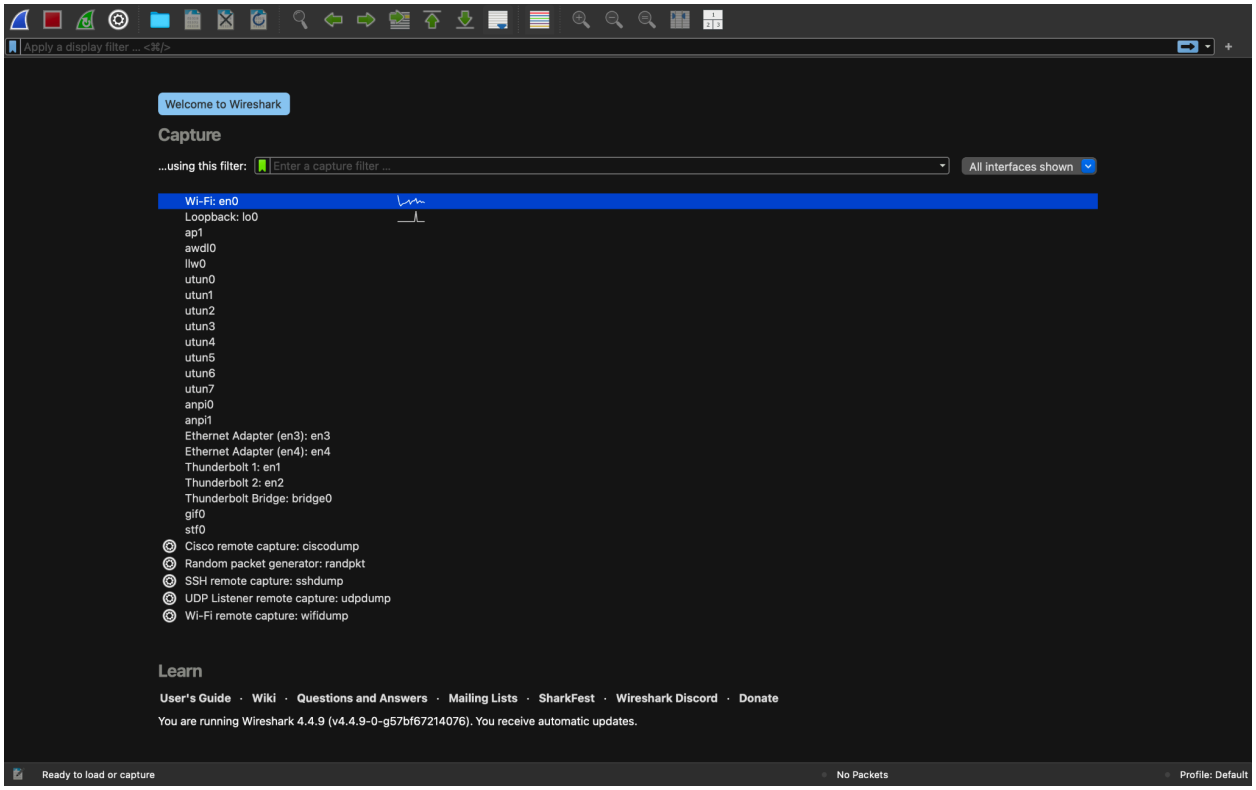

While this screenshot doesn't prove the firewall's specific rule is "block UDP, allow ICMP," it perfectly demonstrates the *result* of what would happen to traceroute in that scenario.

# Instructions

## ## Part 1: How to Perform the Experiment (macOS)

These are the steps we used to capture the traffic for the traceroute command on our Mac.

1. Open Applications: Start both Wireshark and the Terminal app.
2. Select Network Interface in Wireshark: On the Wireshark welcome screen, find your active network connection. This is typically en0 (Wi-Fi) and will have a moving line graph next to it. Click on it once to select it.




3. Start Capturing: Click the blue icon  in the top-left corner to begin capturing all network traffic.


4. Run the Command: Immediately switch to your Terminal window, type the following command, and press Enter:
   traceroute www.youtube.com


5. Wait for Completion: Watch the Terminal until the command finishes running and you see the normal command prompt again.

```
banotharvindnayak@BANOTHs-MacBook-Pro ~ % traceroute www.youtube.com
traceroute: Warning: www.youtube.com has multiple addresses; using 142.250.183.110
traceroute to youtube-ui.l.google.com (142.250.183.110), 64 hops max, 40 byte packets
 1  10.7.0.5 (10.7.0.5)  7.512 ms  4.996 ms  4.138 ms
 2  172.16.4.7 (172.16.4.7)  4.679 ms  4.427 ms  3.179 ms
 3  14.139.98.1 (14.139.98.1)  6.214 ms  7.290 ms  6.514 ms
 4  10.117.81.253 (10.117.81.253)  4.133 ms  4.464 ms  4.447 ms
 5  10.154.8.137 (10.154.8.137)  13.299 ms  12.325 ms  12.962 ms
 6  10.255.239.170 (10.255.239.170)  12.449 ms  11.470 ms  15.825 ms
 7  10.152.7.214 (10.152.7.214)  12.967 ms  12.166 ms  12.333 ms
 8  72.14.204.62 (72.14.204.62)  13.838 ms * *
 9  * * *
10  142.250.208.148 (142.250.208.148)  27.446 ms
    142.250.228.50 (142.250.228.50)  14.278 ms
    142.251.77.100 (142.251.77.100)  15.699 ms
11  142.250.226.134 (142.250.226.134)  14.126 ms  14.646 ms  58.232 ms
12  bom12s13-in-f14.1e100.net (142.250.183.110)  14.275 ms  13.468 ms  14.017 ms
```

6. Stop Capturing: Switch back to Wireshark and click the red square icon 🟥 to stop the capture


**7.** Filter the Results: In the display filter bar at the top of Wireshark, type (udp.port >= 33434) or icmp and press Enter. This will hide all irrelevant traffic and show you only the traceroute packets.

8. We then saved the file and named it **tracert_youtube_mac.pcapng** and we've also provided the file in our github repository.

## ## Part 2: How to Perform the Experiment (Windows)

The steps followed on Windows are almost the same as mac but there are only a few differences. Such as:

### 1. The Command-Line Tool

The application We use to run the command is different.

- **macOS:** We used the **Terminal** app.
- **Windows:** We used the **Command Prompt** (`cmd`).

### 2. The Command Itself

The name of the command and its output format vary slightly.

```
PS C:\Users\saila> tracert www.youtube.com

Tracing route to youtube-ui.l.google.com [142.251.42.14]
over a maximum of 30 hops:

  1     6 ms     7 ms    48 ms  10.7.0.5
  2    44 ms     7 ms    10 ms  172.16.4.7
  3    11 ms    27 ms     7 ms  14.139.98.1
  4    40 ms    85 ms    18 ms  10.117.81.253
  5    10 ms    12 ms    12 ms  10.154.8.137
  6    11 ms    13 ms    11 ms  10.255.239.170
  7    66 ms    13 ms    46 ms  10.152.7.214
  8    11 ms    11 ms    14 ms  72.14.204.62
  9    11 ms    12 ms    12 ms  142.251.49.177
 10    12 ms    10 ms    12 ms  209.85.248.61
 11    11 ms    12 ms    12 ms  bom12s19-in-f14.1e100.net [142.251.42.14]

Trace complete.
```

- **macOS:** The command is `traceroute www.youtube.com`. It provides detailed timing for each of the three probes sent.
- **Windows:** The command is `tracert www.youtube.com`. It presents the timing for the three probes first, followed by the IP address or name of the hop.

**3. The Wireshark Filter**

Because the underlying protocols are different, you must use a different filter in Wireshark to isolate the relevant traffic.

- **macOS:** The best filter is `(udp.port >= 33434) or icmp`. This is because `traceroute` sends **UDP** packets and receives ICMP replies.
- **Windows:** The correct filter is simply `icmp`. This is because `tracert` uses only the **ICMP** protocol for both its outgoing probes and the incoming replies.

**4.** We then saved the file and named it **tracert_youtube_windows.pcapng** and we've also provided the file in our github repository.