Assignment 9

Wireshark Lab: ICMP and IP

Objective:
The purpose of this assignment is to help students understand the functionalities and behaviors of the ICMP and IP protocols through practical experiments. Students will use the Wireshark tool to capture and analyse network packets, answering questions related to packet structure, data flow, and the protocols' roles in network communication.

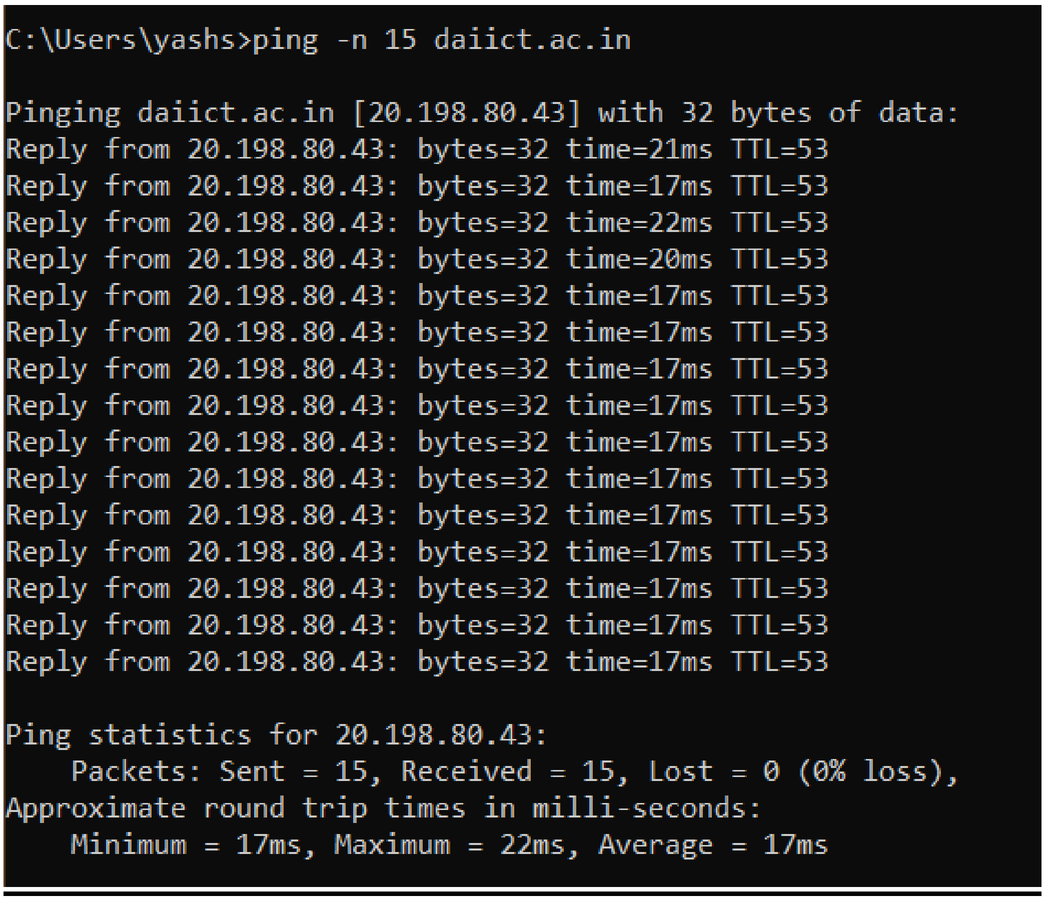
Requirements:

- Wireshark (installed on your computer)
- Access to a computer with command-line tools (Windows Command Prompt)
- Internet connection to run `ping` and `traceroute` commands

<u>Part A: Exploring ICMP with Ping and Traceroute</u>

Ping command:

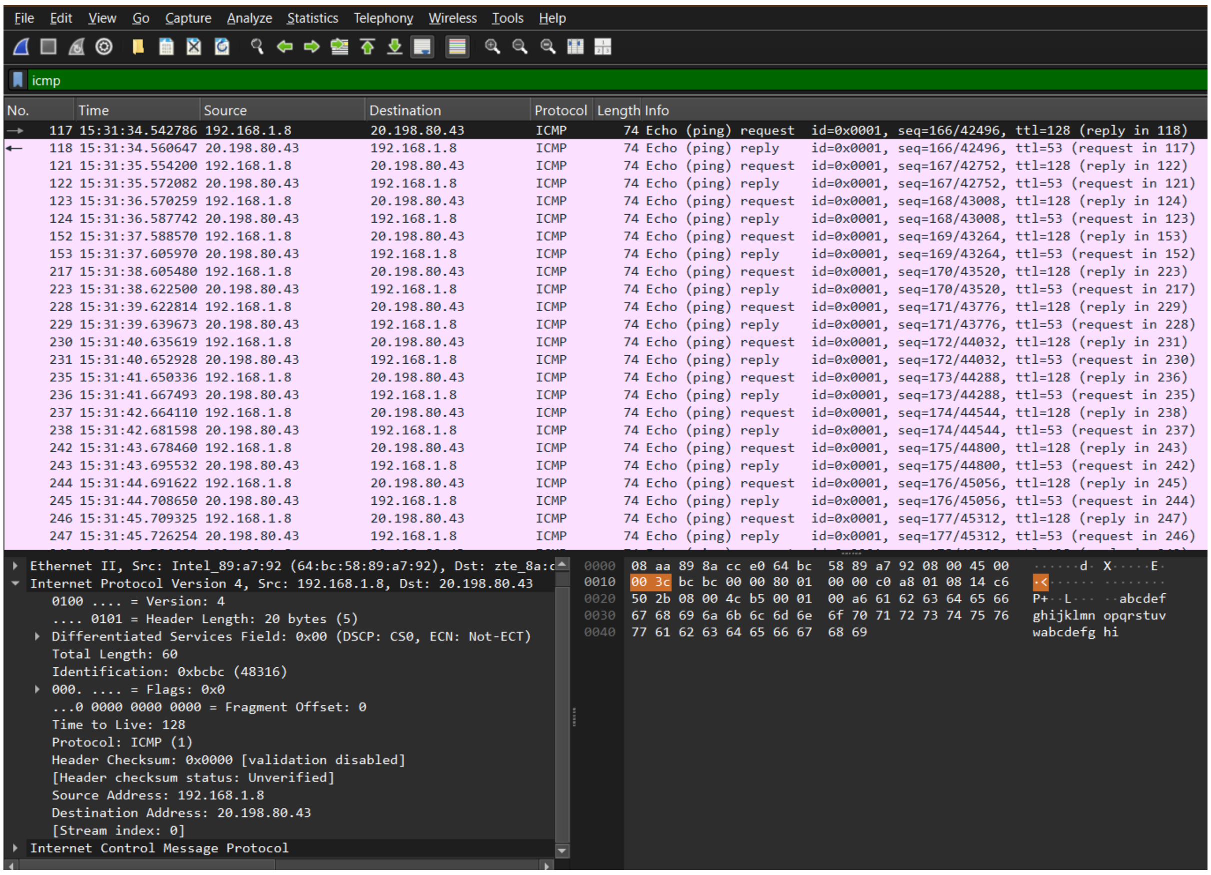
- You may recall that the **Ping program** is simple tool that allows anyone (for example, a network administrator) to verify **if a host is live or not**.
- The Ping program in the source host **sends a packet to the target IP address**; if the target is live, the Ping program in the target host responds by **sending a packet back to the source host**.
- As you might have guessed (given that this lab is about ICMP), **both Ping packets are ICMP packets**.



```
C:\Users\yashs>ping -n 15 daiict.ac.in

Pinging daiict.ac.in [20.198.80.43] with 32 bytes of data:
Reply from 20.198.80.43: bytes=32 time=21ms TTL=53
Reply from 20.198.80.43: bytes=32 time=17ms TTL=53
Reply from 20.198.80.43: bytes=32 time=22ms TTL=53
Reply from 20.198.80.43: bytes=32 time=20ms TTL=53
Reply from 20.198.80.43: bytes=32 time=17ms TTL=53
Reply from 20.198.80.43: bytes=32 time=17ms TTL=53
Reply from 20.198.80.43: bytes=32 time=17ms TTL=53
Reply from 20.198.80.43: bytes=32 time=17ms TTL=53
Reply from 20.198.80.43: bytes=32 time=17ms TTL=53
Reply from 20.198.80.43: bytes=32 time=17ms TTL=53
Reply from 20.198.80.43: bytes=32 time=17ms TTL=53
Reply from 20.198.80.43: bytes=32 time=17ms TTL=53
Reply from 20.198.80.43: bytes=32 time=17ms TTL=53
Reply from 20.198.80.43: bytes=32 time=17ms TTL=53
Reply from 20.198.80.43: bytes=32 time=17ms TTL=53

Ping statistics for 20.198.80.43:
    Packets: Sent = 15, Received = 15, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 17ms, Maximum = 22ms, Average = 17ms
```

Figure 1 Command Prompt window after entering Ping command.

- At the end of the experiment, your Command Prompt Window should look something like Figure 1. From this window we see that the source ping program sent 15 query packets and received 15 responses. Note also that for each response, the source calculates the round-trip time (RTT), which for the 10 packets is on average 17 msec.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
icmp

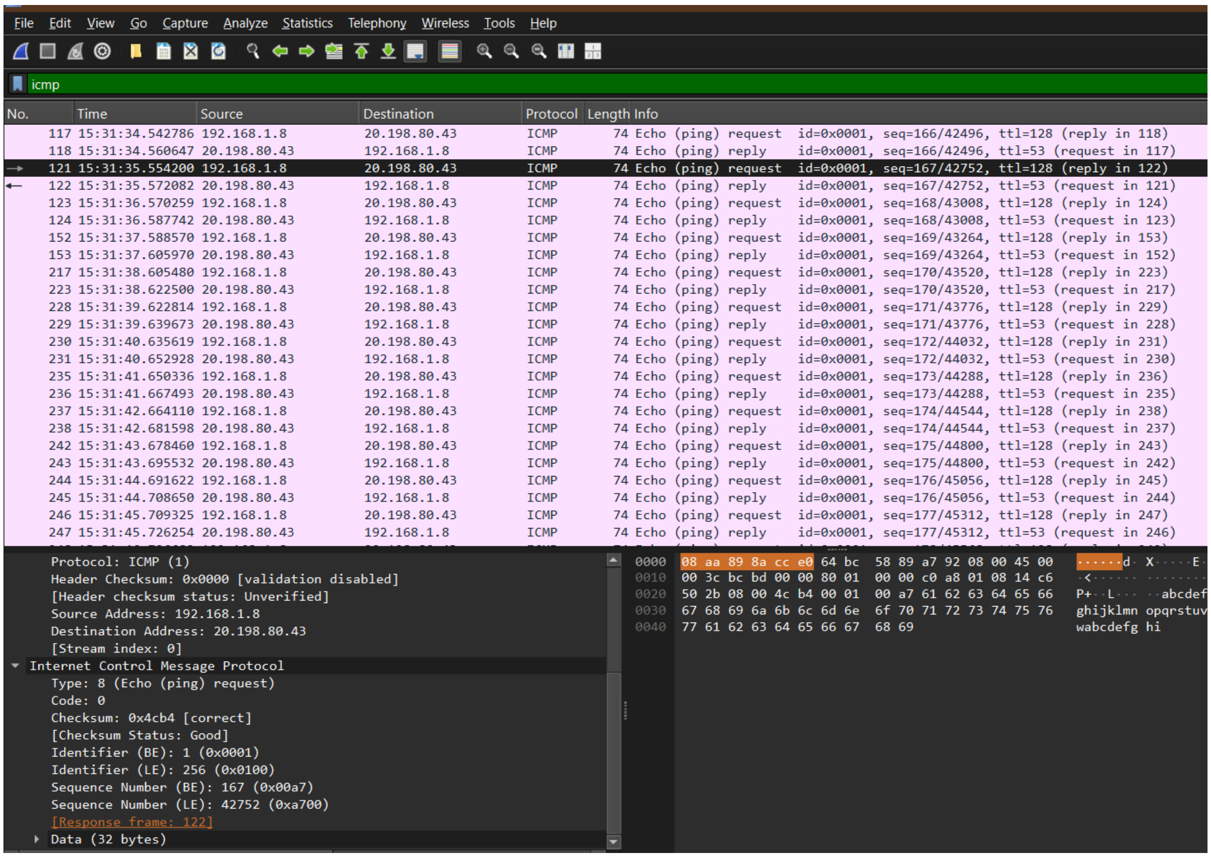
```
No.  Time            Source         Destination    Protocol Length Info
117 15:31:34.542786 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=166/42496, ttl=128 (reply in 118)
118 15:31:34.560647 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=166/42496, ttl=53 (request in 117)
121 15:31:35.554200 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=167/42752, ttl=128 (reply in 122)
122 15:31:35.572082 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=167/42752, ttl=53 (request in 121)
123 15:31:36.570259 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=168/43008, ttl=128 (reply in 124)
124 15:31:36.587742 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=168/43008, ttl=53 (request in 123)
152 15:31:37.588570 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=169/43264, ttl=128 (reply in 153)
153 15:31:37.605970 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=169/43264, ttl=53 (request in 152)
217 15:31:38.605480 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=170/43520, ttl=128 (reply in 223)
223 15:31:38.622500 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=170/43520, ttl=53 (request in 217)
228 15:31:39.622814 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=171/43776, ttl=128 (reply in 229)
229 15:31:39.639673 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=171/43776, ttl=53 (request in 228)
230 15:31:40.635619 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=172/44032, ttl=128 (reply in 231)
231 15:31:40.652928 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=172/44032, ttl=53 (request in 230)
235 15:31:41.650336 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=173/44288, ttl=128 (reply in 236)
236 15:31:41.667493 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=173/44288, ttl=53 (request in 235)
237 15:31:42.664110 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=174/44544, ttl=128 (reply in 238)
238 15:31:42.681598 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=174/44544, ttl=53 (request in 237)
242 15:31:43.678460 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=175/44800, ttl=128 (reply in 243)
243 15:31:43.695532 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=175/44800, ttl=53 (request in 242)
244 15:31:44.691622 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=176/45056, ttl=128 (reply in 245)
245 15:31:44.708650 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=176/45056, ttl=53 (request in 244)
246 15:31:45.709325 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=177/45312, ttl=128 (reply in 247)
247 15:31:45.726254 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=177/45312, ttl=53 (request in 246)
```

```
Ethernet II, Src: Intel_89:a7:92 (64:bc:58:89:a7:92), Dst: zte_8a:c
Internet Protocol Version 4, Src: 192.168.1.8, Dst: 20.198.80.43
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0xbcbc (48316)
    000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.8
    Destination Address: 20.198.80.43
    [Stream index: 0]
Internet Control Message Protocol
```

```
0000  08 aa 89 8a cc e0 64 bc  58 89 a7 92 08 00 45 00   ······d· X·····E·
0010  00 3c bc bc 00 00 80 01  00 00 c0 a8 01 08 14 c6   ·<······ ········
0020  50 2b 08 00 4c b5 00 01  00 a6 61 62 63 64 65 66   P+··L··· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

Figure 2 Wireshark output for Ping program with Internet Protocol expanded.

- Figure 2 provides a screenshot of the Wireshark output, after "**icmp**" has been entered into the filter display window. Note that the packet listing shows 30 packets: the **15 Ping queries** sent by the source and the **15 Ping responses** received by the source. Also note that the source's IP address is a private address (behind a NAT) of the form 192.168/12; the destination's IP address is that of the Web server at DAIICT.
- Now let's zoom in on the first packet (sent by the client); in the figure below, the packet contents area provides information about this packet. We see that the IP datagram within this packet has **protocol number 01**, which is **the protocol number for ICMP**. This means that the payload of the IP datagram is an ICMP packet.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
icmp

```
No.     Time            Source         Destination    Protocol Length Info
    117 15:31:34.542786 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=166/42496, ttl=128 (reply in 118)
    118 15:31:34.560647 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=166/42496, ttl=53 (request in 117)
 →  121 15:31:35.554200 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=167/42752, ttl=128 (reply in 122)
 ←  122 15:31:35.572082 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=167/42752, ttl=53 (request in 121)
    123 15:31:36.570259 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=168/43008, ttl=128 (reply in 124)
    124 15:31:36.587742 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=168/43008, ttl=53 (request in 123)
    152 15:31:37.588570 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=169/43264, ttl=128 (reply in 153)
    153 15:31:37.605970 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=169/43264, ttl=53 (request in 152)
    217 15:31:38.605480 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=170/43520, ttl=128 (reply in 223)
    223 15:31:38.622500 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=170/43520, ttl=53 (request in 217)
    228 15:31:39.622814 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=171/43776, ttl=128 (reply in 229)
    229 15:31:39.639673 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=171/43776, ttl=53 (request in 228)
    230 15:31:40.635619 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=172/44032, ttl=128 (reply in 231)
    231 15:31:40.652928 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=172/44032, ttl=53 (request in 230)
    235 15:31:41.650336 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=173/44288, ttl=128 (reply in 236)
    236 15:31:41.667493 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=173/44288, ttl=53 (request in 235)
    237 15:31:42.664110 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=174/44544, ttl=128 (reply in 238)
    238 15:31:42.681598 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=174/44544, ttl=53 (request in 237)
    242 15:31:43.678460 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=175/44800, ttl=128 (reply in 243)
    243 15:31:43.695532 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=175/44800, ttl=53 (request in 242)
    244 15:31:44.691622 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=176/45056, ttl=128 (reply in 245)
    245 15:31:44.708650 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=176/45056, ttl=53 (request in 244)
    246 15:31:45.709325 192.168.1.8    20.198.80.43   ICMP     74 Echo (ping) request  id=0x0001, seq=177/45312, ttl=128 (reply in 247)
    247 15:31:45.726254 20.198.80.43   192.168.1.8    ICMP     74 Echo (ping) reply    id=0x0001, seq=177/45312, ttl=53 (request in 246)
```

```
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.8
    Destination Address: 20.198.80.43
    [Stream index: 0]
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x4cb4 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence Number (BE): 167 (0x00a7)
    Sequence Number (LE): 42752 (0xa700)
    [Response frame: 122]
  ▸ Data (32 bytes)
```

```
0000  08 aa 89 8a cc e0 64 bc  58 89 a7 92 08 00 45 00   ······d· X·····E·
0010  00 3c bc bd 00 00 80 01  00 00 c0 a8 01 08 14 c6   ·<······ ········
0020  50 2b 08 00 4c b4 00 01  00 a7 61 62 63 64 65 66   P+··L··· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

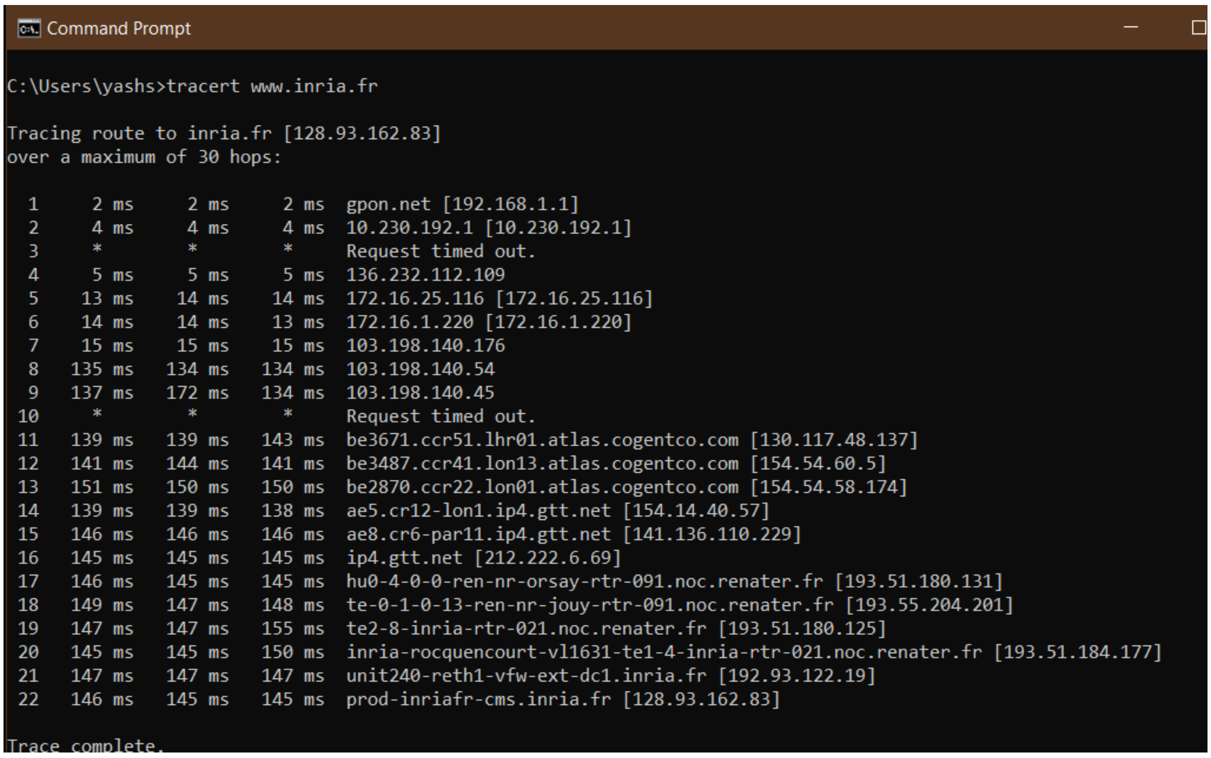
Figure 3 Wireshark capture of ping packet with ICMP packet expanded.

- Figure 3 focuses on the same ICMP but has expanded **the ICMP protocol information** in the packet contents window.
- Observe that this ICMP packet is of Type 8 and Code 0 - a so-called ICMP "**echo request**" packet.
- Also note that this ICMP packet contains a **checksum**, an **identifier**, and a **sequence number**.

Traceroute command:

- **Traceroute** program can be used to figure out the **path a packet takes from source to destination**.
- **Traceroute** is implemented in different ways in Unix/Linux/MacOS and in Windows.
- In **Unix/Linux**, the source sends a **series of UDP packets** to the target destination using an unlikely destination port number.
- In **Windows**, the source sends a **series of ICMP packets** to the target destination.
- For both operating systems, the program sends the first packet with TTL=1, the second packet with TTL=2, and so on.
- Recall that a **router** will **decrement a packet's TTL value** as the packet passes through the router. When a packet arrives at a router with **TTL=1**, the router **sends an ICMP error packet** back to the source. In the following, we'll use the **native Windows *tracert* program.**

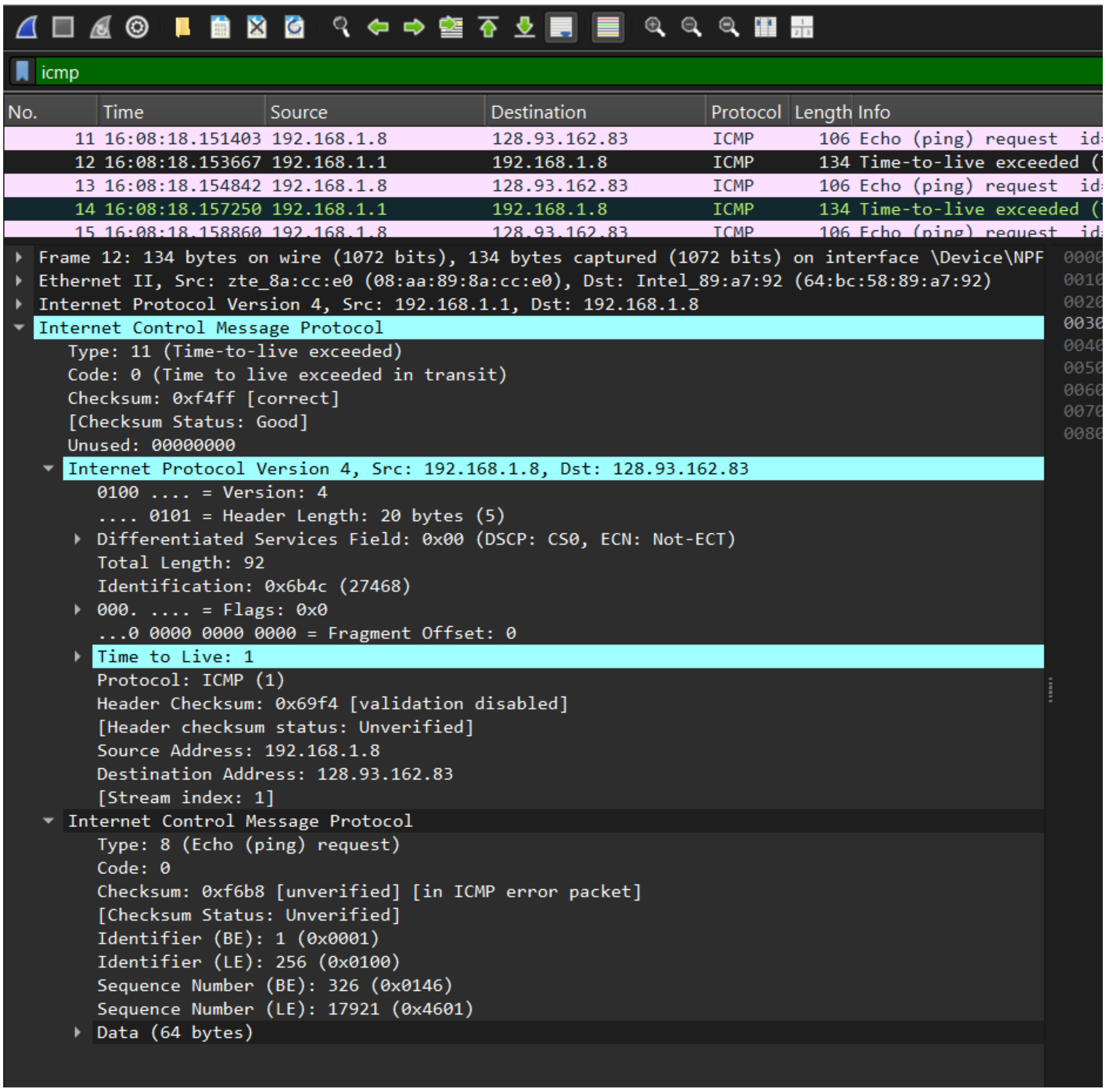
(Note that on a Windows machine, the command is "*tracert*" and not "*traceroute*".)



```
C:\Users\yashs>tracert www.inria.fr

Tracing route to inria.fr [128.93.162.83]
over a maximum of 30 hops:

  1     2 ms     2 ms     2 ms  gpon.net [192.168.1.1]
  2     4 ms     4 ms     4 ms  10.230.192.1 [10.230.192.1]
  3     *        *        *     Request timed out.
  4     5 ms     5 ms     5 ms  136.232.112.109
  5    13 ms    14 ms    14 ms  172.16.25.116 [172.16.25.116]
  6    14 ms    14 ms    13 ms  172.16.1.220 [172.16.1.220]
  7    15 ms    15 ms    15 ms  103.198.140.176
  8   135 ms   134 ms   134 ms  103.198.140.54
  9   137 ms   172 ms   134 ms  103.198.140.45
 10     *        *        *     Request timed out.
 11   139 ms   139 ms   143 ms  be3671.ccr51.lhr01.atlas.cogentco.com [130.117.48.137]
 12   141 ms   144 ms   141 ms  be3487.ccr41.lon13.atlas.cogentco.com [154.54.60.5]
 13   151 ms   150 ms   150 ms  be2870.ccr22.lon01.atlas.cogentco.com [154.54.58.174]
 14   139 ms   139 ms   138 ms  ae5.cr12-lon1.ip4.gtt.net [154.14.40.57]
 15   146 ms   146 ms   146 ms  ae8.cr6-par11.ip4.gtt.net [141.136.110.229]
 16   145 ms   145 ms   145 ms  ip4.gtt.net [212.222.6.69]
 17   146 ms   145 ms   145 ms  hu0-4-0-0-ren-nr-orsay-rtr-091.noc.renater.fr [193.51.180.131]
 18   149 ms   147 ms   148 ms  te-0-1-0-13-ren-nr-jouy-rtr-091.noc.renater.fr [193.55.204.201]
 19   147 ms   147 ms   155 ms  te2-8-inria-rtr-021.noc.renater.fr [193.51.180.125]
 20   145 ms   145 ms   150 ms  inria-rocquencourt-vl1631-te1-4-inria-rtr-021.noc.renater.fr [193.51.184.177]
 21   147 ms   147 ms   147 ms  unit240-reth1-vfw-ext-dc1.inria.fr [192.93.122.19]
 22   146 ms   145 ms   145 ms  prod-inriafr-cms.inria.fr [128.93.162.83]

Trace complete.
```

Figure 4 Command Prompt window displays the results of the Traceroute program.

- At the end of the experiment, your Command Prompt Window should look something like Figure 4.
- In this figure, the **client Traceroute program is in Ahmedabad,gujarat** and the **target destination is in France**.
- From this figure we see that for **each TTL value**, the source program **sends three probe packets**.
- Traceroute displays the RTTs for each of the probe packets, as well as the IP address (and possibly the name) of the router that returned the ICMP TTL-exceeded message.



```
icmp

No.   Time             Source        Destination    Protocol Length Info
   11 16:08:18.151403 192.168.1.8   128.93.162.83  ICMP     106 Echo (ping) request  id
   12 16:08:18.153667 192.168.1.1   192.168.1.8    ICMP     134 Time-to-live exceeded (
   13 16:08:18.154842 192.168.1.8   128.93.162.83  ICMP     106 Echo (ping) request  id
   14 16:08:18.157250 192.168.1.1   192.168.1.8    ICMP     134 Time-to-live exceeded (
   15 16:08:18.158860 192.168.1.8   128.93.162.83  ICMP     106 Echo (ping) request  id

> Frame 12: 134 bytes on wire (1072 bits), 134 bytes captured (1072 bits) on interface \Device\NPF
> Ethernet II, Src: zte_8a:cc:e0 (08:aa:89:8a:cc:e0), Dst: Intel_89:a7:92 (64:bc:58:89:a7:92)
> Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.8
v Internet Control Message Protocol
     Type: 11 (Time-to-live exceeded)
     Code: 0 (Time to live exceeded in transit)
     Checksum: 0xf4ff [correct]
     [Checksum Status: Good]
     Unused: 00000000
   v Internet Protocol Version 4, Src: 192.168.1.8, Dst: 128.93.162.83
        0100 .... = Version: 4
        .... 0101 = Header Length: 20 bytes (5)
      > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        Total Length: 92
        Identification: 0x6b4c (27468)
      > 000. .... = Flags: 0x0
        ...0 0000 0000 0000 = Fragment Offset: 0
      > Time to Live: 1
        Protocol: ICMP (1)
        Header Checksum: 0x69f4 [validation disabled]
        [Header checksum status: Unverified]
        Source Address: 192.168.1.8
        Destination Address: 128.93.162.83
        [Stream index: 1]
   v Internet Control Message Protocol
        Type: 8 (Echo (ping) request)
        Code: 0
        Checksum: 0xf6b8 [unverified] [in ICMP error packet]
        [Checksum Status: Unverified]
        Identifier (BE): 1 (0x0001)
        Identifier (LE): 256 (0x0100)
        Sequence Number (BE): 326 (0x0146)
        Sequence Number (LE): 17921 (0x4601)
      > Data (64 bytes)
```

Figure 5 Wireshark window of ICMP fields expanded for one ICMP error packet.

- Figure 5 displays the Wireshark window for an ICMP packet returned by a router.
- **Note that this ICMP error packet contains many more fields than the Ping ICMP messages**

Instructions to be followed:

1. Open a command-line interface on your computer.
2. Start Wireshark and begin capturing packets on your network interface.
3. Run the `ping` command to send ICMP echo requests to a remote server:
   `ping -n 10 gaia.cs.umass.edu`
4. Observe and capture the ICMP packets generated by the `ping` command. After the capture, stop Wireshark and *save the capture file*.
5. Use the `tracert` (on Windows) command to see the path packets take to a remote server:
   `tracert gaia.cs.umass.edu`
6. Save the output of the `traceroute` command and stop Wireshark capture.

Questions:

(Q1-Q4 => Ping-ICMP)

(from Q5 => tracert-ICMP)

1. What is the IP address of your local host and the destination host?
2. Why do ICMP packets do not contain source and destination port numbers?
3. Examine one of the captured ICMP echo request packets. What are the `Type` and `Code` values, and what do they signify?
4. Examine the corresponding ping reply packet. What are the ICMP type and code numbers? What other fields does this ICMP packet have? How many bytes are the checksum, sequence number and identifier fields?
5. Examine the ICMP echo packet in your screenshot. Is this different from the ICMP ping query packets in the first half of this lab? If yes, how so?
6. Compare the captured ICMP echo request packets to the responses. Identify any differences in the packet details (such as identifiers and sequence numbers).
7. Why might some `traceroute` hops not return any response?
8. Examine the ICMP error packet in your screenshot. It has more fields than the ICMP echo packet. What is included in those fields?

9. Examine the last three ICMP packets received by the source host. How are these packets different from the ICMP error packets? Why are they different?
10. Within the tracert measurements, is there a link whose delay is significantly longer than others? Refer to the screenshot in Figure 4, is there a link whose delay is significantly longer than others? On the basis of the router names, can you guess the location of the two routers on the end of this link?

Submission:

- Submit **screenshots** of the **Command Prompt window**.

- When answering questions below, you should hand in a **printout of the packet(s) within the trace** that you used to answer the question asked. Use the printout to explain your answer.

(*To print a packet, use File->Print, choose Selected packet only, choose Packet summary line, and select the minimum amount of packet detail that you need to answer the question.*)

- Answer all questions in a separate single document, clearly labelling each part.
- Provide detailed explanations with references to packet structures wherever necessary.

Part B: Analyzing IP Protocol Behavior

Instructions to be followed:

Capturing packets from an execution of traceroute:

1. Start-up Wireshark and begin packet capture.
    a. *(Capture->Start* or click on the blue shark fin button in the top left of the Wireshark window).
2. Run the `ping` command again (for windows, in windows command prompt.)
    a. **"ping gaia.cs.umass.edu -l 3000"(-l is to determine size of packet sent)**
3. Save this capture for answering related questions.
4. Now, try to capture some IPv6 packets by visiting websites that support IPv6, such as `youtube.com` with help "**ping www.youtube.com**"

Questions:

(Q1-Q10 => IPv4)

(from Q11-Q17 =>IPv4 Fragmentation)

(from Q18-end =>IPv6)

1. Select the first UDP segment sent by your computer via the traceroute command to gaia.cs.umass.edu. Expand the Internet Protocol part of the packet in the packet details window. What is the IP address of your computer?
2. What is the value in the time-to-live (TTL) field in this IPv4 datagram's header? (search in 1st ICMP packet in trace).
3. What is the value in the upper layer protocol field in this IPv4 datagram's header? [Note: the answers for Linux/MacOS differ from Windows here].
4. How many bytes are in the IP header?
5. How many bytes are in the payload of the IP datagram? Explain how you determined the number of payload bytes
6. Are the values of the TTL fields similar, across all of ICMP packets from all of the routers?

7. Has this IP datagram been fragmented? Explain how you determined whether the datagram has been fragmented.
8. Describe the pattern you see in the values in the Identification field of the IP datagrams being sent by your computer.
9. Which fields in the IP datagram always change from one datagram to the next within this series of UDP segments sent by your computer destined to 128.119.245.12, via traceroute? Why?
10. Which fields in this sequence of IP datagrams (containing UDP segments) stay constant? Why?

(**Note**: if you find your packet has not been fragmented, you should download the zip file “http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip” and extract the trace file ip-wireshark-trace1-1.pcapng . If your computer has an Ethernet or WiFi interface, a packet size of 3000 should cause fragmentation)

11. Find the first IP datagram containing the first part of the segment sent to 128.119.245.12 sent by your computer via the traceroute command to gaia.cs.umass.edu, after you specified that the traceroute packet length should be 3000. (Hint: This is packet 179 in the ip-wireshark-trace1-1.pcapng trace file in footnote 2. Packets 179, 180, and 181 are three IP datagrams created by fragmenting the first single 3000-byte UDP segment sent to 128.119.145.12). Has that segment been fragmented across more than one IP datagram? (Hint: the answer is yes !)
12.
13. What information in the IP header indicates that this datagram been fragmented?
14. What information in the IP header for this packet indicates whether this is the first fragment versus a latter fragment?
15. How many bytes are there in is this IP datagram (header plus payload)?
16. What fields change in the IP header between the first and second fragment?
17. Now find the IP datagram containing the third fragment of the original UDP segment. What information in the IP header indicates that this is the last fragment of that segment?

(The DNS AAAA request type is used to resolve names to IPv6 IP addresses.)

18. What is the IPv6 address of the computer making the DNS AAAA request? Give the IPv6 source address for this datagram in the exact same form as displayed in the Wireshark window

(Recall that an IPv6 address is shown as 8 sets of 4 hexadecimal digits, with each set separated by colons, and with leading zeros omitted. If an IPv6 address has two colons in a row (‘::’), this is shorthand meaning that all of the intervening bytes between the two colons are zero. Thus, for example,

fe80::1085:6434:583:e79 is shorthand for fe80:0000:0000:0000:1085:6434:0583:0e79. Make sure you understand this example)

19. What is the IPv6 destination address for this datagram? Give this IPv6 address in the exact same form as displayed in the Wireshark window.
20. For IPv6 packets, describe any notable differences in structure compared to IPv4. How are headers formatted differently?
21. How much payload data is carried in the 2nd IPv6 datagram?
22. What is the upper layer protocol to which 2nd datagram’s payload will be delivered at the destination?

(Lastly, find the IPv6 DNS response to the IPv6 DNS AAAA request made in the this trace. This DNS response contains IPv6 addresses for youtube.com)

23. How many IPv6 addresses are returned in the response to this AAAA request?
24. What is the first of the IPv6 addresses returned by the DNS for youtube? Give this IPv6 address in the exact same shorthand form as displayed in the Wireshark window.