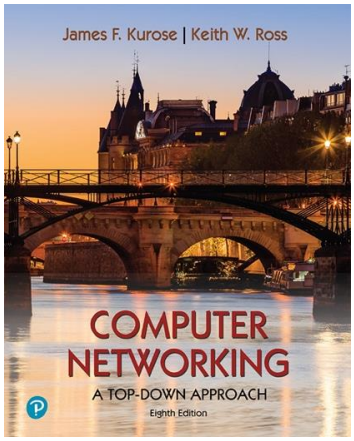
Wireshark Lab:
IP v8.1

Supplement to *Computer Networking: A Top-Down Approach, 8th ed.*, J.F. Kurose and K.W. Ross

"Tell me and I forget. Show me and I remember. Involve me and I understand." Chinese proverb



In this lab, we'll investigate the celebrated IP protocol, focusing on the IPv4 and IPv6 datagram. This lab has one part. In this part, we'll analyze packets in a trace of IPv4 datagrams sent and received by the `traceroute` program.

Before getting started, you'll probably want to review sections 1.4.3 in the text[1] and section 3.4 of RFC 2151 to update yourself on the operation of the `traceroute` program.

Capturing packets from an execution of traceroute

In order to generate a trace of IPv4 datagrams for this lab, we'll use the `traceroute` program to send datagrams of two different sizes to gaia.cs.umass.edu. Recall that `traceroute` operates by first sending one or more datagrams with the time-to-live (TTL) field in the IP header set to 1; it then sends a series of one or more datagrams towards the same destination with a TTL value of 2; it then sends a series of datagrams towards the same destination with a TTL value of 3; and so on. Recall that a router must decrement the TTL in each received datagram by 1 (actually, RFC 791 says that the router must decrement the TTL by *at least* one). If the TTL reaches 0, the router returns an ICMP message (type 11 – TTL-exceeded) to the sending host. As a result of this behavior, a datagram with a TTL of 1 (sent by the host executing `traceroute`) will cause the router one hop away from the sender to send an ICMP TTL-exceeded message back to the sender; the datagram sent with a TTL of 2 will cause the router two hops away to send an ICMP message back to the sender; the datagram sent with a TTL of 3 will cause the router three hops away to send an ICMP message back to the sender; and so on. In this manner, the host executing `traceroute` can learn the IP addresses of the routers between itself and the destination by looking at the source IP addresses in the datagrams containing the ICMP TTL-exceeded messages.

---

[1] References to figures and sections are for the 8th edition of our text, *Computer Networks, A Top-down Approach, 8h ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2020.* Our website for this book is http://gaia.cs.umass.edu/kurose_ross You'll find lots of interesting open material there.

Let's run `traceroute` and have it send datagrams of two different sizes. The larger of the two datagram lengths will require `traceroute` messages to be fragmented across multiple IPv4 datagrams.

- **Linux/MacOS.** With the Linux/MacOS `traceroute` command, the size of the UDP datagram sent towards the final destination can be explicitly set by indicating the number of bytes in the datagram; this value is entered in the `traceroute` command line immediately after the name or address of the destination. For example, to send `traceroute` datagrams of 2000 bytes towards gaia.cs.umass.edu, the command would be:

```
%traceroute gaia.cs.umass.edu 2000
```

- **Windows.** The `tracert` program provided with Windows does not allow one to change the size of the ICMP message sent by `tracert`. So it won't be possible to use a Windows machine to generate ICMP messages that are large enough to force IP fragmentation. However, you can use `tracert` to generate small, fixed length packets to perform Part 1 of this lab. At the DOS command prompt enter:

```
>tracert gaia.cs.umass.edu
```

Do the following:

- Start up Wireshark and begin packet capture. *(Capture->Start* or click on the blue shark fin button in the top left of the Wireshark window).
- Enter `traceroute` command, using gaia.cs.umass.edu as the destination, the first with a length of 56 bytes (for Windows, you should not specify length of bytes).
- Stop Wireshark tracing.

Part 1: Basic IPv4

In your trace, you should be able to see the series of UDP segments (in the case of MacOS/Linux) or ICMP Echo Request messages (Windows) sent by `traceroute` on your computer, and the ICMP TTL-exceeded messages returned to your computer by the intermediate routers. In the questions below, we'll assume you're using a MacOS/Linux computer; the corresponding questions for the case of a Windows machine should be clear. Your screen should look similar to the screenshot in Figure 2, where we have used the display filter "udp||icmp" (see the light-green-filled display-filter field in Figure 2) so that only UDP and/or ICMP protocol packets are displayed.

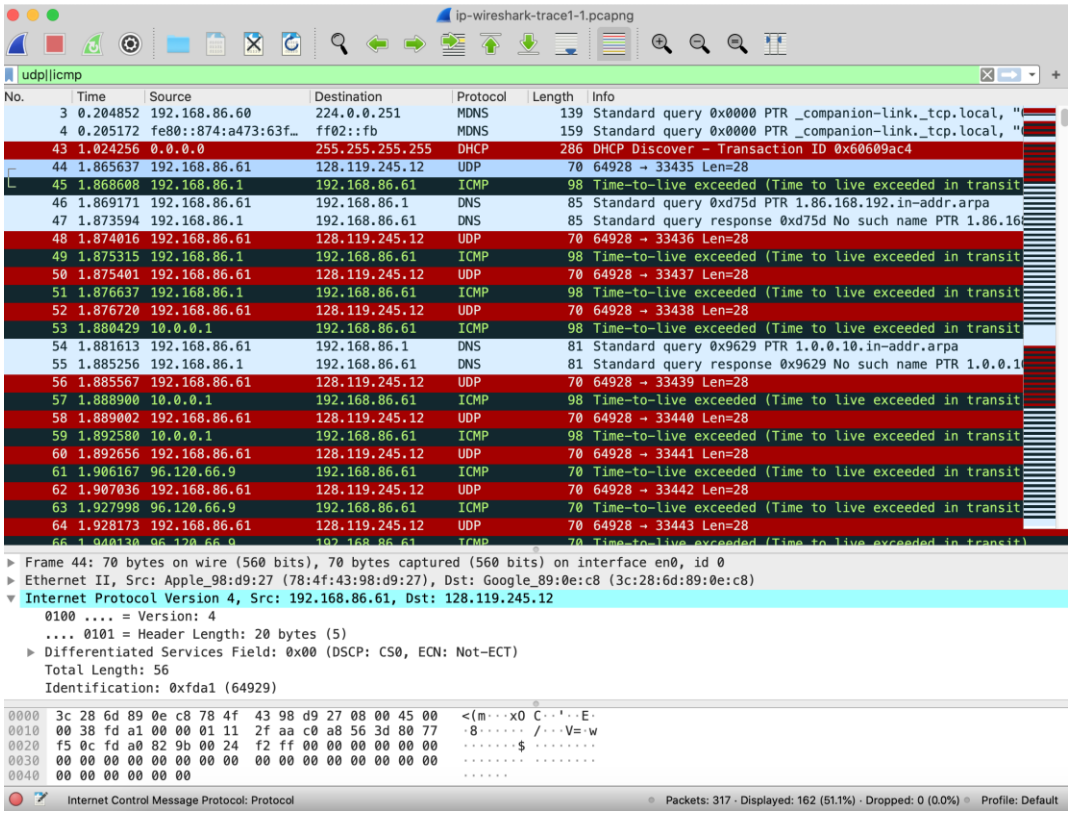


Figure 2: Wireshark screenshot, showing UDP and ICMP packets in the tracefile *ip-wireshark-trace1-1.pcapng*

Answer the following questions. If you're doing this lab as part of class, your teacher will provide details about how to hand in assignments, whether written or in an LMS.

- Select the first UDP(Windows ICMP) segment sent by your computer via the `traceroute` command to gaia.cs.umass.edu. (Hint: this is 44th packet in the trace file in the *ip-wireshark-trace1-1.pcapng* file in footnote 2). Expand the Internet Protocol part of the packet in the packet details window.
    1. 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?
    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. Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.

Next, let's look at the *sequence* of UDP (Windows ICMP) segments being sent from your computer via `traceroute`, destined to 128.119.245.12. The display filter that you can enter to do this is "ip.src==<Your IP> and ip.dst==128.119.245.12 and udp and !icmp" (for Windows "ip.src==<Your IP> and ip.dst==128.119.245.12 and !udp and icmp").

This will allow you to easily move sequentially through just the datagrams containing just these segments. Your screen should look similar to Figure 3.

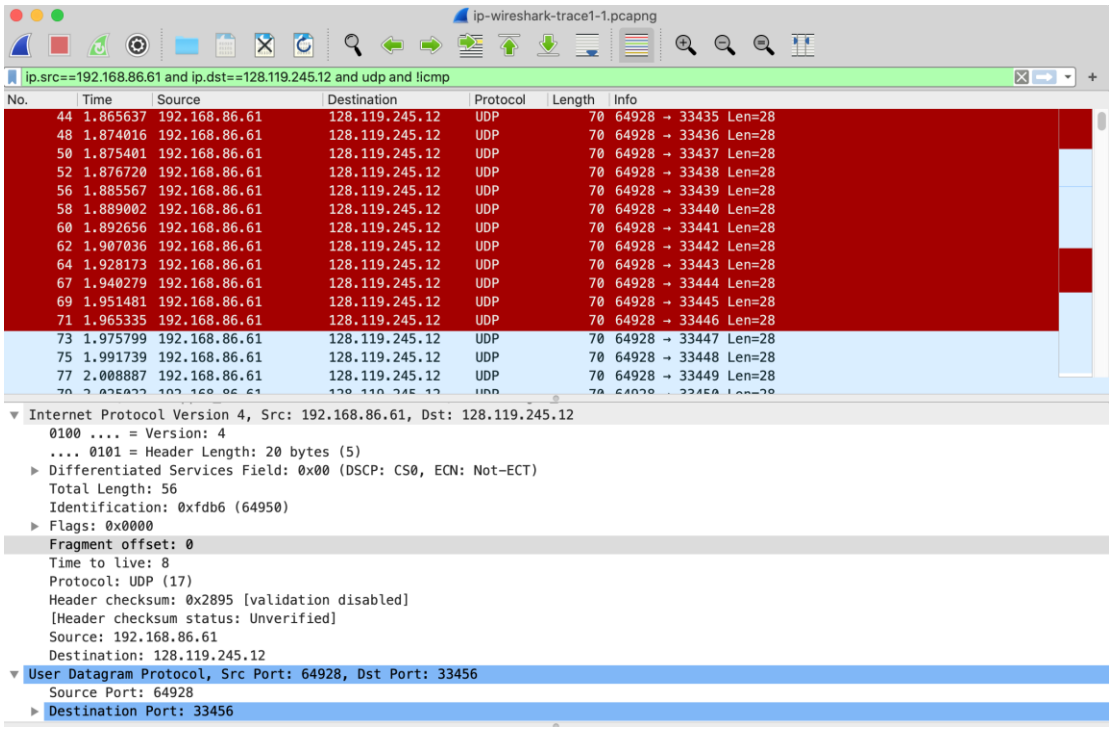


Figure 3: Wireshark screenshot, showing up segments in the tracefile using the display filter `ip.src==192.168.86.61 and ip.dst==128.119.245.12 and udp and !icmp`

7. 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?
8. Which fields in this sequence of IP datagrams (containing UDP segments) stay constant? Why?
9. Describe the pattern you see in the values in the Identification field of the IP datagrams being sent by your computer.

Now let's take a look at the ICMP packets being returned to your computer by the intervening routers where the TTL value was decremented to zero (and hence caused the ICMP error message to be returned to your computer). The display filter that you can use to show just these packets is "ip.dst==<Your IP> and icmp".

10. What is the upper layer protocol specified in the IP datagrams returned from the routers? [Note: the answers for Linux/MacOS differ from Windows here].
11. Are the values in the Identification fields (across the sequence of all of ICMP packets from all of the routers) similar in behavior to your answer to question 9 above?
12. Are the values of the TTL fields similar, across all of ICMP packets from all of the routers?