© 2005-21012, J.F Kurose and K.W. Ross, All Rights Reserved

 Wireshark Lab: IP v6.0

Supplement to *Computer Networking: A Top-Down Approach, 6th 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



**SV: Please read all the instructions, but use the trace file from our website for answering the questions.**
In this lab, we’ll investigate the IP protocol, focusing on the IP datagram. We’ll do so by analyzing a trace of IP datagrams sent and received by an execution of the traceroute program (the traceroute program itself is explored in more detail in the Wireshark ICMP lab). We’ll investigate the various fields in the IP datagram, and study IP fragmentation in detail.

Before beginning this lab, you’ll probably want to review sections 1.4.3 in the text1 and section 3.4 of RFC 2151 [ftp://ftp.rfc-editor.org/in-notes/rfc2151.txt] to update yourself on the operation of the tracerouteprogram. You’ll also want to read Section 4.4 in the text, and probably also have RFC 791 [ftp://ftp.rfc-editor.org/in-notes/rfc791.txt] on hand as well, for a discussion of the IP protocol.

1. Capturing packets from an execution of traceroute

In order to generate a trace of IP datagrams for this lab, we’ll use the traceroute program to send datagrams of different sizes towards some destination, *X*. 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

1 References to figures and sections are for the 6th edition of our text, *Computer Networks, A Top-down Approach, 6th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2012.*

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 identities of the routers between itself and destination *X* by looking at the source IP addresses in the datagrams containing the ICMP TTL-exceeded messages.

We’ll want to run traceroute and have it send datagrams of various lengths.

* **Windows.** The tracert program (used for our ICMP Wireshark lab) provided with Windows does not allow one to change the size of the ICMP echo request (ping) message sent by the tracert program. A nicer Windows traceroute program is *pingplotter*, available both in free version and shareware versions at http://www.pingplotter.com. Download and install *pingplotter*, and test it out by performing a few traceroutes to your favorite sites. The size of the ICMP echo request message can be explicitly set in *pingplotter* by selecting the menu item *Edit-> Options->Packet Options* and then filling in the *Packet Size* field. The default packet size is 56 bytes. Once *pingplotter* has sent a series of packets with the increasing TTL values, it restarts the sending process again with a TTL of 1, after waiting *Trace Interval* amount of time. The value of *Trace Interval* and the number of intervals can be explicitly set in *pingplotter*.
* **Linux/Unix/MacOS.** With the Unix/MacOS traceroute command, the size of the UDP datagram sent towards the 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 traceroutedatagrams of 2000 bytes towards gaia.cs.umass.edu, the command would be:

%traceroute gaia.cs.umass.edu 2000

Do the following: **(SV: please read the instructions below, but you do not need to execute them.)**

* Start up Wireshark and begin packet capture *(Capture->Start)* and then press *OK* on the Wireshark Packet Capture Options screen (we’ll not need to select any options here).
* If you are using a Windows platform, start up *pingplotter* and enter the name of a target destination in the “Address to Trace Window.” Enter 3 in the “# of times to Trace” field, so you don’t gather too much data. Select the menu item *Edit >Advanced Options->Packet Options* and enter a value of 56 in the *Packet Size* field and then press OK. Then press the Trace button. You should see a *pingplotter* window that looks something like this:



Next, send a set of datagrams with a longer length, by selecting *Edit->Advanced Options->Packet Options* and enter a value of 2000 in the *Packet Size* field and then press OK. Then press the Resume button.

Finally, send a set of datagrams with a longer length, by selecting *Edit­>Advanced Options->Packet Options* and enter a value of 3500 in the *Packet Size* field and then press OK. Then press the Resume button.

Stop Wireshark tracing.

* • If you are using a Unix or Mac platform, enter three traceroutecommands, one with a length of 56 bytes, one with a length of 2000 bytes, and one with a length of 3500 bytes.

Stop Wireshark tracing.

**SV: For answering questions, please use the trace file provided at the website.**If you are unable to run Wireshark on a live network connection, you can download a packet trace file that was captured while following the steps above on one of the author’s Windows computers2. You may well find it valuable to download this trace even if you’ve captured your own trace and use it, as well as your own trace, when you explore the questions below.

2 Download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip and extract the file *ip­ethereal-trace-1*. The traces in this zip file were collected by Wireshark running on one of the author’s computers, while performing the steps indicated in the Wireshark lab. Once you have downloaded the trace, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the ip-ethereal-trace-1 trace file.

2. A look at the captured trace

In your trace, you should be able to see the series of ICMP Echo Request (in the case of Windows machine) or the UDP segment (in the case of Unix) sent by your computer and the ICMP TTL-exceeded messages returned to your computer by the intermediate routers. In the questions below, we’ll assume you are using a Windows machine; the corresponding questions for the case of a Unix machine should be clear.

Select the first ICMP Echo Request message sent by your computer, and expand the Internet Protocol part of the packet in the packet details window.



1. What is the IP address of your computer?
2. Within the IP packet header, what is the value in the upper layer protocol field?
3. How many bytes are in the IP header? How many bytes are in the payload *of the IP datagram*? Explain how you determined the number of payload bytes.
4. Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.

Next, sort the traced packets according to IP source address by clicking on the *Source* column header; a small downward pointing arrow should appear next to the word *Source*. If the arrow points up, click on the *Source* column header again. Select the first ICMP Echo Request message sent by your computer that shows up **(i.e. not the first that was sent chronologically, but the first that is listed with the source filter)**, and expand the Internet Protocol portion in the “details of selected packet header” window. In the “listing of captured packets” window, you should see all of the subsequent ICMP messages (perhaps with additional interspersed packets sent by other protocols running on your computer) below this first ICMP. Use the down arrow to move through the ICMP messages sent by your computer.

Next (with the packets still sorted by source address) find the series of ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router.**(you might want to briefly switch back to the time sorted view to identify the IP address of the first hop router by viewing which TTL exceeded packet was first )**

1. Which fields in the IP datagram *always* change from one datagram to the next within this series of ICMP messages sent by your computer?
2. Describe the pattern you see in the values in the Identification field of the IP datagram

Fragmentation

Sort the packet listing according to time again by clicking on the *Time* column.

9. Find the first ICMP Echo Request message that was sent by your computer after you changed the *Packet Size* in *pingplotter* to be 2000. Has that message been fragmented across more than one IP datagram? [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.zip and extract the *ip­ethereal-trace-1*packet trace. If your computer has an Ethernet interface, a packet size of 2000 *should* cause fragmentation.3]

10. Look at the first fragment of the fragmented IP datagram. What information in the IP header indicates that the datagram been fragmented? What information in the IP header indicates whether this is the first fragment versus a latter fragment? How long is this IP datagram?

1. What is the value in the Identification field and the TTL field for the TTL exceeded packet that is displayed at the top while the source filter is applied?
2. Do these values remain unchanged for all of the ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router? Why?

3 The packets in the *ip-ethereal-trace-1* trace file in http://gaia.cs.umass.edu/wireshark-labs/wireshark­traces.zip are all less that 1500 bytes. This is because the computer on which the trace was gathered has an Ethernet card that limits the length of the maximum IP packet to 1500 bytes (40 bytes of TCP/IP header data and 1460 bytes of upper-layer protocol payload). This 1500 byte value is the standard maximum length allowed by Ethernet. If your trace indicates a datagram longer 1500 bytes, and your computer is using an Ethernet connection, then Wireshark is reporting the wrong IP datagram length; it will likely also show only one large IP datagram rather than multiple smaller datagrams.. This inconsistency in reported lengths is due to the interaction between the Ethernet driver and the Wireshark software. We recommend that if you have this inconsistency, that you perform this lab using the *ip-ethereal-trace-1* trace file.

1. Look at the second fragment of the fragmented IP datagram. What information in the IP header indicates that this is not the first datagram fragment? Are the more fragments? How can you tell?
2. How many fragments were created from the original datagram?

Now find the first ICMP Echo Request message that was sent by your computer after you changed the *Packet Size* in *pingplotter* to be 3500**(looking at the reconstruction of the IP fragments in the final fragment will be helpful in finding which packets were fragmented from an original size of 3500. Look at the bottom of the Internet Protocol Version 4 tab and you will see a summary indicating the number of fragments and their total size)**.