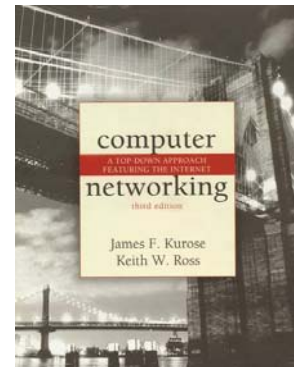


# Ethereal Lab: IP



*Computer Networking: A Top-down Approach Featuring the Internet, 3<sup>rd</sup> edition.*

Version: 1.0

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

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 Ethereal 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 1.63 in the text and section 3.4 of RFC 2151 [<ftp://ftp.rfc-editor.org/in-notes/rfc2151.txt>] to update yourself on the operation of the `traceroute` program. 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.<sup>1</sup>

## 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 away to send an ICMP message back to the sender; the datagram sent with a TTL of 3

---

<sup>1</sup> All references to the text in this lab are to *Computer Networking: A Top-down Approach Featuring the Internet*, 3<sup>rd</sup> edition.

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 `tracert` 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.

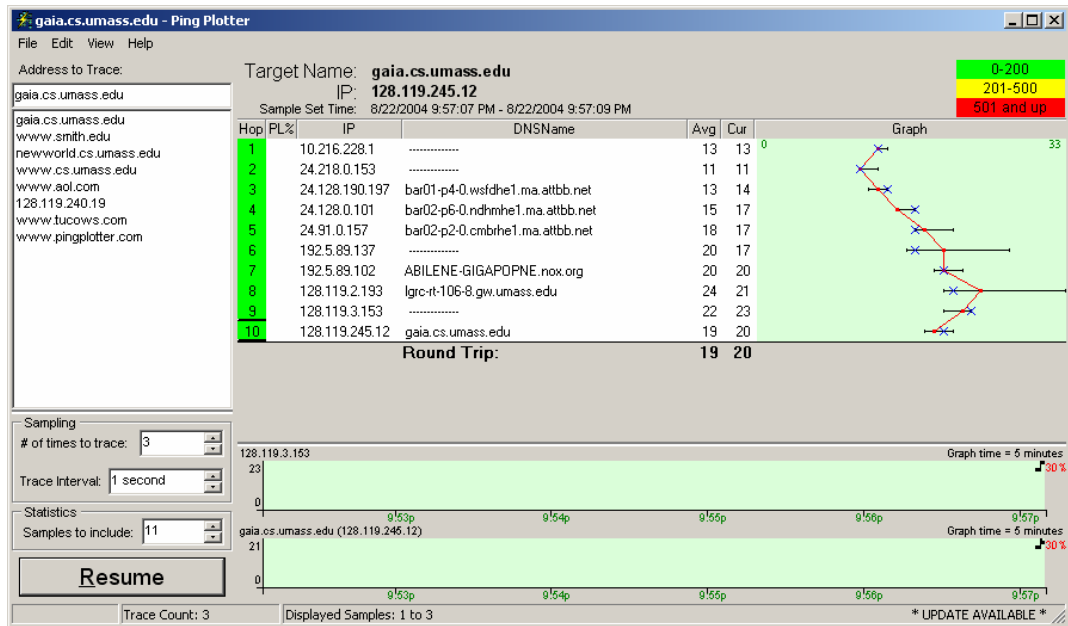
We'll want to run `tracert` and have it send datagrams of various lengths.

- **Windows.** The `tracert` program (used for our ICMP Ethereal 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 `tracert` 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 `tracert`s to your favorite sites. The size of the ICMP echo request message can be explicitly set in *pingplotter* by selecting the menu item *Edit->Advanced 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.** With the Unix `tracert` 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 `tracert` command line immediately after the name or address of the destination. For example, to send `tracert` datagrams of 2000 bytes towards `gaia.cs.umass.edu`, the command would be:

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

Do the following:

- Start up Ethereal and begin packet capture (*Capture->Start*) and then press *OK* on the Ethereal 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 Ethereal tracing.

- If you are using a Unix platform, enter three `traceroute` commands, one with a length of 56 bytes, one with a length of 2000 bytes, and one with a length of 3500 bytes.

Stop Ethereal tracing.

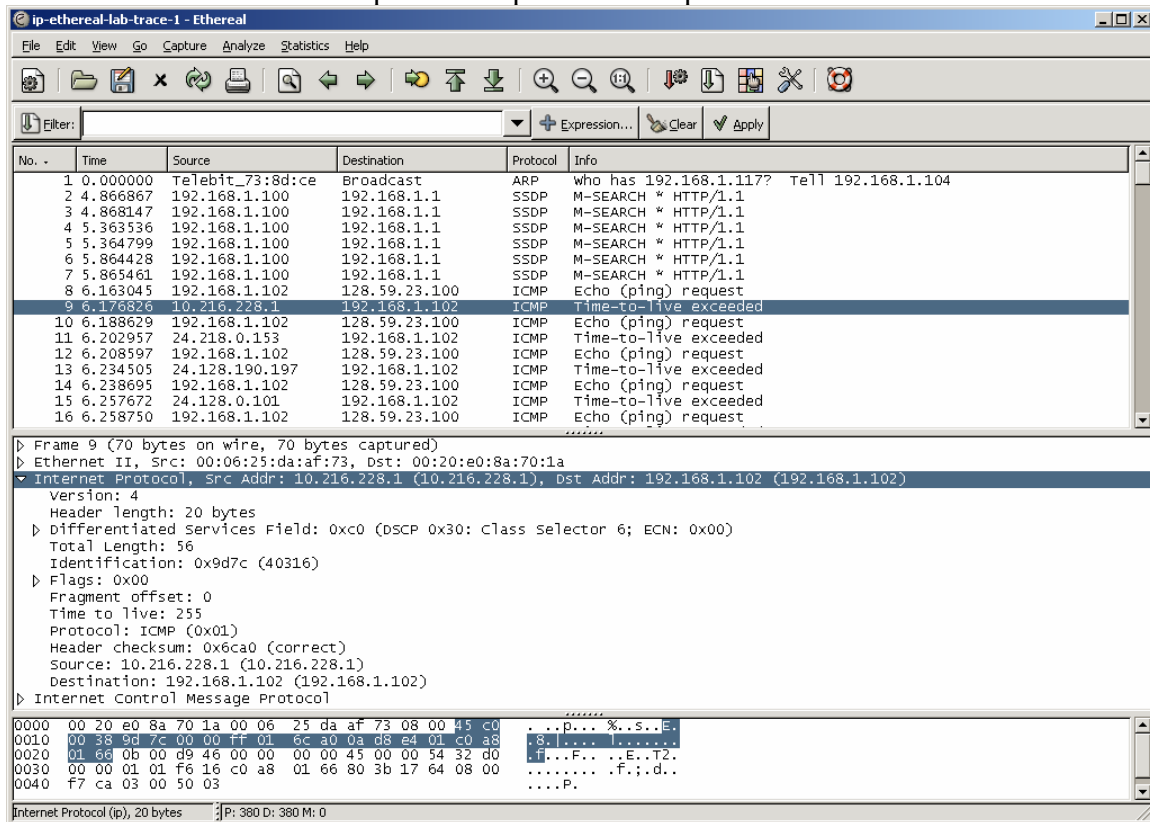
If you are unable to run Ethereal on a live network connection, you can download a packet trace file that was captured while following the steps above on one of the author's Windows computers<sup>2</sup>. You may well find it valuable to download this trace even if you've captured your own trace and use it, as well as your own trace, when you explore the questions below.

<sup>2</sup> Download the zip file <http://gaia.cs.umass.edu/ethereal-labs/ethereal-traces.zip> and extract the file *ip-ethereal-trace-1*. The traces in this zip file were collected by Ethereal running on one of the author's computers, while performing the steps indicated in the Ethereal lab. Once you have downloaded the trace, you can load it into Ethereal 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. Whenever possible, when answering a question you should hand in a printout of the packet(s) within the trace that you used to answer the question asked. Annotate 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.

1. 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.



The screenshot shows the Wireshark interface with a captured trace. The packet list pane displays 16 packets. Packet 9 is selected, and its details are expanded in the packet details pane. The details pane shows the Internet Protocol header for packet 9, including fields like Version, Header Length, Total Length, Identification, Flags, Fragment Offset, Time to Live, Protocol, Header Checksum, Source, and Destination. The packet bytes pane shows the raw data of the IP header and payload.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	Telebit_73:8d:ce	Broadcast	ARP	who has 192.168.1.117? Tell 192.168.1.104
2	4.866867	192.168.1.100	192.168.1.1	SSDP	M-SEARCH * HTTP/1.1
3	4.868147	192.168.1.100	192.168.1.1	SSDP	M-SEARCH * HTTP/1.1
4	5.363536	192.168.1.100	192.168.1.1	SSDP	M-SEARCH * HTTP/1.1
5	5.364799	192.168.1.100	192.168.1.1	SSDP	M-SEARCH * HTTP/1.1
6	5.864428	192.168.1.100	192.168.1.1	SSDP	M-SEARCH * HTTP/1.1
7	5.865461	192.168.1.100	192.168.1.1	SSDP	M-SEARCH * HTTP/1.1
8	6.163045	192.168.1.102	128.59.23.100	ICMP	Echo (ping) request
9	6.176626	10.216.228.1	192.168.1.102	ICMP	Time-to-live exceeded
10	6.188629	192.168.1.102	128.59.23.100	ICMP	Echo (ping) request
11	6.202957	24.218.0.153	192.168.1.102	ICMP	Time-to-live exceeded
12	6.208597	192.168.1.102	128.59.23.100	ICMP	Echo (ping) request
13	6.234505	24.128.190.197	192.168.1.102	ICMP	Time-to-live exceeded
14	6.238695	192.168.1.102	128.59.23.100	ICMP	Echo (ping) request
15	6.257672	24.128.0.101	192.168.1.102	ICMP	Time-to-live exceeded
16	6.258750	192.168.1.102	128.59.23.100	ICMP	Echo (ping) request

Frame 9 (70 bytes on wire, 70 bytes captured)  
Ethernet II, Src: 00:06:25:da:af:73, Dst: 00:20:e0:8a:70:1a  
Internet Protocol, Src Addr: 10.216.228.1 (10.216.228.1), Dst Addr: 192.168.1.102 (192.168.1.102)  
Version: 4  
Header length: 20 bytes  
Differentiated Services Field: 0xc0 (DSCP 0x30: Class Selector 6; ECN: 0x00)  
Total Length: 56  
Identification: 0x9d7c (40316)  
Flags: 0x00  
Fragment offset: 0  
Time to live: 255  
Protocol: ICMP (0x01)  
Header checksum: 0x6ca0 (correct)  
Source: 10.216.228.1 (10.216.228.1)  
Destination: 192.168.1.102 (192.168.1.102)  
Internet Control Message Protocol

```
0000  00 20 e0 8a 70 1a 00 06 25 da af 73 08 00 45 c0  ..P...%.s..E.
0010  00 58 9f 7e 00 00 ff 01 0c 80 0a 3e e4 01 c0 a8  .8. ....l.....
0020  01 68 0b 00 09 46 00 00 00 00 45 00 00 54 32 d0  .F...F...E..T.
0030  00 00 01 01 f6 16 c0 a8 01 66 80 3b 17 64 08 00  ....f.;d..
0040  f7 ca 03 00 50 03  ..P.
```

- What is the IP address of your computer?
- Within the IP packet header, what is the value in the upper layer protocol field?
- 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.
- 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, 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.

5. Which fields in the IP datagram *always* change from one datagram to the next within this series of ICMP messages sent by your computer?
6. Which fields stay constant? Which of the fields *must* stay constant? Which fields must change? Why?
7. Describe the pattern you see in the values in the Identification field of the IP datagram

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.

8. What is the value in the Identification field and the TTL field?
9. Do these values remain unchanged for all of the ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router? Why?

## Fragmentation

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

10. 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/ethereal-labs/ethereal-traces.zip> and extract the *ip-ethereal-trace-1* packet trace. If your computer has an Ethernet interface, a packet size of 2000 *should* cause fragmentation.<sup>3</sup>]
11. Print out 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?

---

<sup>3</sup> The packets in the *ip-ethereal-trace-1* trace file in <http://gaia.cs.umass.edu/ethereal-labs/ethereal-traces.zip> are all less than 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 than 1500 bytes, and your computer is using an Ethernet connection, then Ethereal 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 Ethereal software. We recommend that if you have this inconsistency, that you perform this lab using the *ip-ethereal-trace-1* trace file.

12. Print out the second fragment of the fragmented IP datagram. What information in the IP header indicates that this is not the first datagram fragment? Are there more fragments? How can you tell?
13. What fields change in the IP header between the first and second fragment?

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.

14. How many fragments were created from the original datagram?
15. What fields change in the IP header among the fragments?