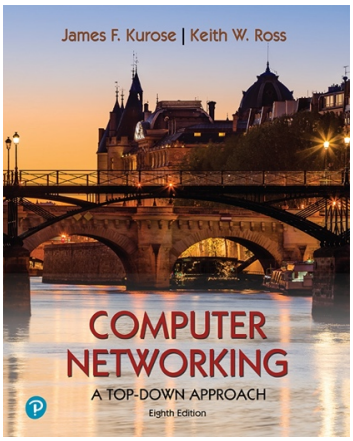
Wireshark Lab:
IP v8.0

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 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 text[1] 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 4.3 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 away to send an ICMP message back to the sender; the datagram sent with a TTL of 3

[1] References to figures and sections are for the 8th edition of our text, *Computer Networks, A Top-down Approach, 8th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2020.*

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.

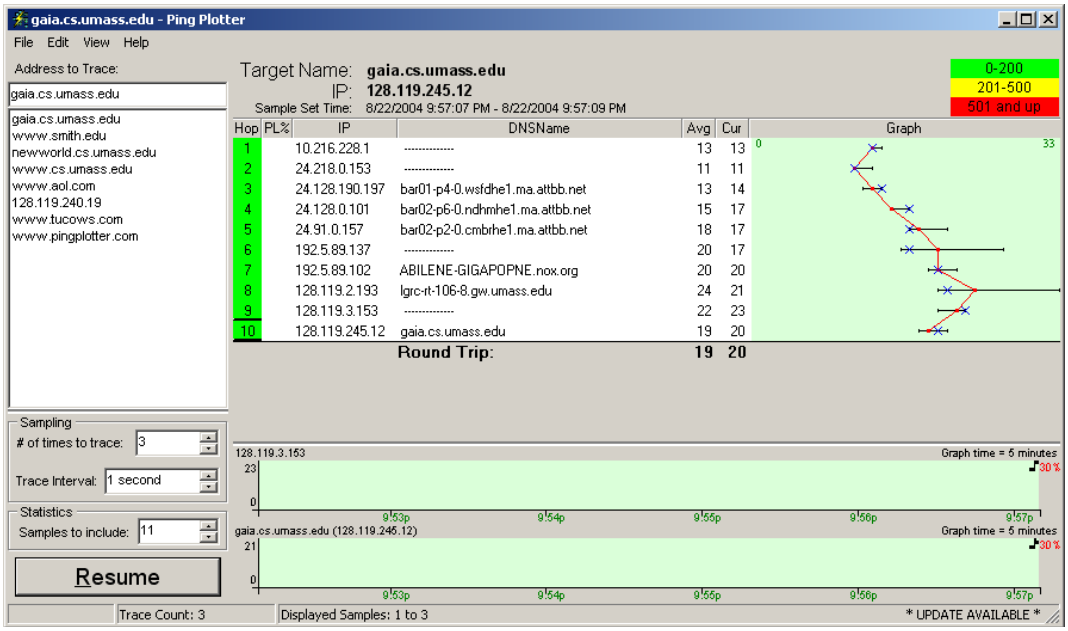
We'll want to run `traceroute` and have it send datagrams of various lengths.

- **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 `traceroute` datagrams of 2000 bytes towards gaia.cs.umass.edu, the command would be:

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

Do the following:

- 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 `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 Wireshark tracing.

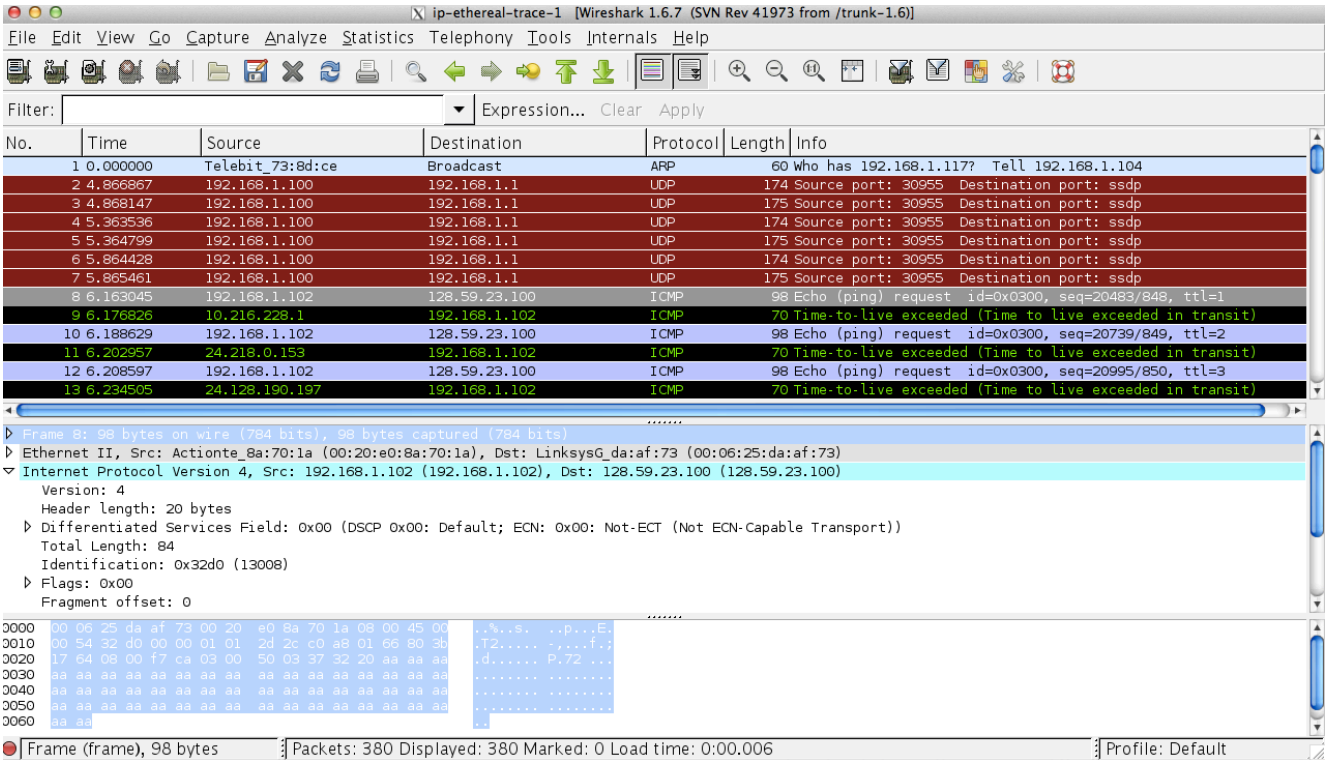
If you are unable to run Wireshark on a live network connection, you can download a packet trace file that was captured while following the steps above on one of the author's Windows computers[2]. You may well find it valuable to download this trace even if you've captured your own trace and use it, 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 *ip-ethereal-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. Whenever possible, when answering a question below you should hand in a printout of the packet(s) within the trace that you used to answer the question asked. When you hand in your assignment, annotate the output so that it's clear where in the output you're getting the information for your answer (e.g., for our classes, we ask that students markup paper copies with a pen, or annotate electronic copies with text in a colored font).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.



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, 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/wireshark-labs/wireshark-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.[3]]
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?

[3] The packets in the *ip-ethereal-trace-1* trace file in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.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.

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 the more fragments? How can you tell?
13. What fields change in the IP header between the first and second fragment?

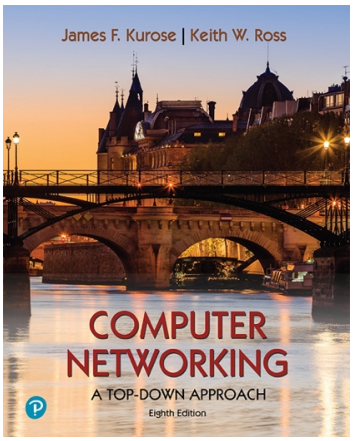
Now find the first ICMP Echo Request message that was sent by your computer after you changed the *Packet Size* in *pingplotter* to be 3500.

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

Wireshark Lab: NAT v8.0

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 behavior of the NAT protocol. This lab will be different from our other Wireshark labs, where we've captured a trace file at a single Wireshark measurement point. Because we're interested in capturing packets at both the input and output sides of the NAT device, we'll need to capture packets at *two* locations. Also, because many students don't have easy access to a NAT device or to two computers on which to take Wireshark measurements, this isn't a lab that is easily done "live" by a student. Therefore in this lab, you will use Wireshark trace files that we've captured for you. Before beginning this lab, you'll probably want to review the material on NAT section 4.3.4 in the text[1].

1. NAT Measurement Scenario

In this lab, we'll capture packets from a simple web request from a client PC in a home network to a www.google.com server. Within the home network, the home network router provides a NAT service, as discussed in Chapter 4. Figure 1 shows our Wireshark trace-collection scenario. As in our other Wireshark labs, we collect a Wireshark trace on the client PC in our home network. This file is called NAT_home_side[2]. Because we are also interested in the packets being sent by the NAT

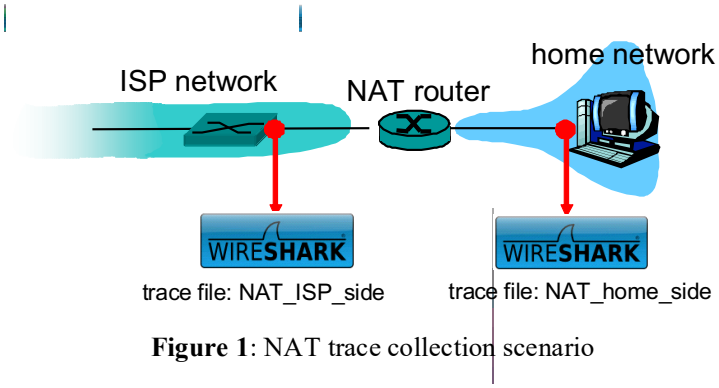


Figure 1: NAT trace collection scenario

[1] References to figures and sections are for the 8th edition of our text, *Computer Networks, A Top-down Approach, 8th ed.,* J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 202o.

[2] Download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip and extract the files need for this lab.

router into the ISP, we'll collect a second trace file at a PC (not shown) tapping into the link from the home router into the ISP network, as shown in Figure 1. (The hub device shown on the ISP side of the router is used to tap into the link between the NAT router and the first hop router in the ISP). Client-to-server packets captured by Wireshark at this point will have undergone NAT translation. The Wireshark trace file captured on the ISP side of the home router is called NAT_ISP_side.

Open the NAT_home_side file and answer the following questions. You might find it useful to use a Wireshark filter so that only frames containing HTTP messages are displayed from the trace file.

Whenever possible, when answering a question below, you should hand in a printout of the packet(s) within the trace that you used to answer the question asked. Annotate the printout[3] 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. What is the IP address of the client?
2. The client actually communicates with several different Google servers in order to implement "safe browsing." (See extra credit section at the end of this lab). The main Google server that will serve up the main Google web page has IP address 64.233.169.104. In order to display only those frames containing HTTP messages that are sent to/from this Google, server, enter the expression "http && ip.addr == 64.233.169.104" (without quotes) into the Filter: field in Wireshark .
3. Consider now the HTTP GET sent from the client to the Google server (whose IP address is IP address 64.233.169.104) at time 7.109267. What are the source and destination IP addresses and TCP source and destination ports on the IP datagram carrying this HTTP GET?
4. At what time[4] is the corresponding 200 OK HTTP message received from the Google server? What are the source and destination IP addresses and TCP source and destination ports on the IP datagram carrying this HTTP 200 OK message?
5. Recall that before a GET command can be sent to an HTTP server, TCP must first set up a connection using the three-way SYN/ACK handshake. At what time is the client-to-server TCP SYN segment sent that sets up the connection used by the GET sent at time 7.109267? What are the source and destination IP addresses and source and destination ports for the TCP SYN segment? What are the source and destination IP addresses and source and destination ports of the ACK sent in response to the SYN. At what time is this ACK received at the client? (Note: to find these segments you will need to clear the Filter expression you entered above in step 2. If you enter the filter "tcp", only TCP segments will be displayed by Wireshark).

---

[3] What do we mean by "annotate"? If you hand in a paper copy, please highlight where in the printout you've found the answer and add some text (preferably with a colored pen) noting what you found in what you 've highlight. If you hand in an electronic copy, it would be great if you could also highlight and annotate.

[4] Specify time using the time since the beginning of the trace (rather than absolute, wall-clock time).

In the following we'll focus on the two HTTP messages (GET and 200 OK) and the TCP SYN and ACK segments identified above. Our goal below will be to locate these two HTTP messages and two TCP segments in the trace file (NAT_ISP_side) captured on the link between the router and the ISP. Because these captured frames will have already been forwarded through the NAT router, some of the IP address and port numbers will have been changed as a result of NAT translation.

Open the NAT_ISP_side. *Note that the time stamps in this file and in NAT_home_side are not synchronized since the packet captures at the two locations shown in Figure 1 were not started simultaneously.* (Indeed, you should discover that the timestamps of a packet captured at the ISP link is actually less that the timestamp of the packet captured at the client PC).

6. In the NAT_ISP_side trace file, find the HTTP GET message was sent from the client to the Google server at time 7.109267 (where t=7.109267 is time at which this was sent as recorded in the NAT_home_side trace file). At what time does this message appear in the NAT_ISP_side trace file? What are the source and destination IP addresses and TCP source and destination ports on the IP datagram carrying this HTTP GET (as recording in the NAT_ISP_side trace file)? Which of these fields are the same, and which are different, than in your answer to question 3 above?
7. Are any fields in the HTTP GET message changed? Which of the following fields in the IP datagram carrying the HTTP GET are changed: Version, Header Length, Flags, Checksum. If any of these fields have changed, give a reason (in one sentence) stating why this field needed to change.
8. In the NAT_ISP_side trace file, at what time is the first 200 OK HTTP message received from the Google server? What are the source and destination IP addresses and TCP source and destination ports on the IP datagram carrying this HTTP 200 OK message? Which of these fields are the same, and which are different than your answer to question 4 above?
9. In the NAT_ISP_side trace file, at what time were the client-to-server TCP SYN segment and the server-to-client TCP ACK segment corresponding to the segments in question 5 above captured? What are the source and destination IP addresses and source and destination ports for these two segments? Which of these fields are the same, and which are different than your answer to question 5 above?

Figure 4.25 in the text shows the NAT translation table in the NAT router.

10. Using your answers to 1-8 above, fill in the NAT translation table entries for HTTP connection considered in questions 1-8 above.

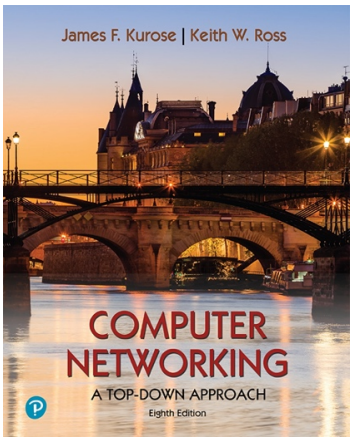
Extra Credit: The trace files investigated above have additional connections to Google servers above and beyond the HTTP GET, 200 OK request/response studied above. For example, in the NAT_home_side trace file, consider the client-to-server GET at time

1.572315, and the GET at time 7.573305. Research the use of these two HTTP messages and write a half page explanation of the purpose of each of these messages.

Wireshark Lab: DHCP v8.0

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 take a quick look at DHCP. DHCP is covered in Section 4.4.3 of the text[1]. Recall that DHCP is used extensively in corporate, university and home-network wired and wireless LANs to dynamically assign IP addresses to hosts (as well as to configure other network configuration information).

This lab is brief, as we'll only examine the DHCP packets captured by a host. If you also have administrative access to your DHCP server, you may want to repeat this lab after making some configuration changes (such as the lease time). If you have a router at home, you most likely can configure your DHCP server. Because many linux/Unix machines (especially those that serve many users) have a static IP address and because manipulating DHCP on such machines typically requires super-user privileges, we'll only present a Windows version of this lab below.

DHCP Experiment

In order to observe DHCP in action, we'll perform several DHCP-related commands and capture the DHCP messages exchanged as a result of executing these commands. Do the following[2]:

1. Begin by opening the Windows Command Prompt application (which can be found in your Accessories folder). As shown in Figure 1, enter "*ipconfig /release*". The executable for *ipconfig* is in C:\windows\system32. This command releases your current IP address, so that your host's IP address becomes 0.0.0.0.

---

[1] References to figures and sections are for the 8th edition of our text, *Computer Networks, A Top-down Approach, 8th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2020.*

[2] If you are unable to run Wireshark live on a computer, you can download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip and extract the file *dhcp-ethereal-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 dhcp-ethereal-trace-1 trace file. You can then use this trace file to answer the questions below.

2. Start up the Wireshark packet sniffer, as described in the introductory Wireshark lab and begin Wireshark packet capture.
3. Now go back to the Windows Command Prompt and enter "*ipconfig /renew*". This instructs your host to obtain a network configuration, including a new IP address. In Figure 1, the host obtains the IP address 192.168.1.108
4. Wait until the "*ipconfig /renew*" has terminated. Then enter the same command "*ipconfig /renew*" again.
5. When the second *"ipconfig /renew"* terminates, enter the command "ipconfig/release" to release the previously-allocated IP address to your computer.
6. Finally, enter "*ipconfig /renew*" to again be allocated an IP address for your computer.
7. Stop Wireshark packet capture.

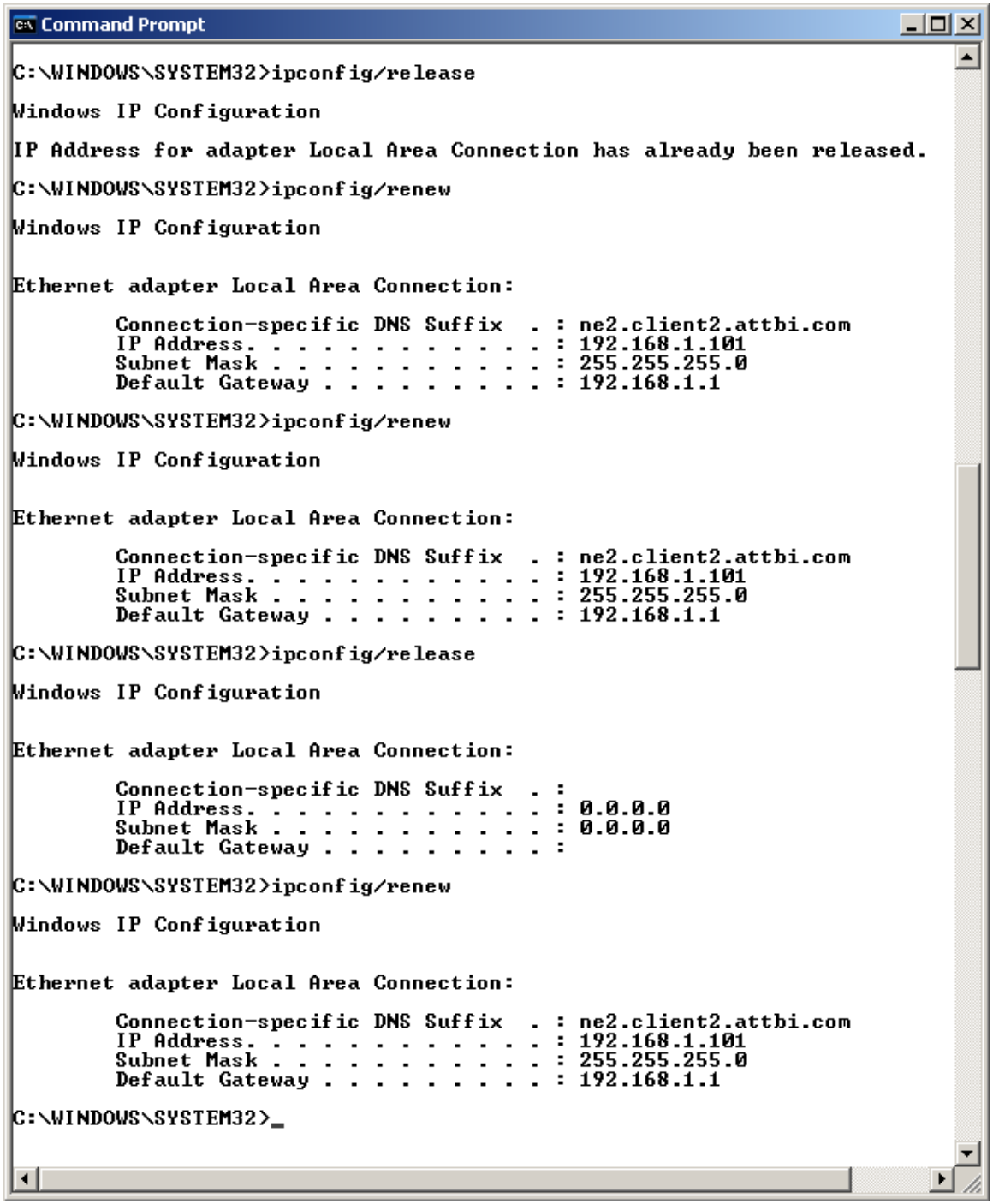


Figure 1 Command Prompt window showing sequence of *ipconfig* commands that you should enter.

Now let's take a look at the resulting Wireshark window. To see only the DHCP packets, enter into the filter field "bootp". (DHCP derives from an older protocol called BOOTP. Both BOOTP and DHCP use the same port numbers, 67 and 68. To see DHCP packets in the current version of Wireshark, you need to enter "bootp" and not "dhcp" in the filter.) We see from Figure 2 that the first *ipconfig* renew command caused four DHCP packets to be generated: a DHCP Discover packet, a DHCP Offer packet, a DHCP Request packet, and a DHCP ACK packet.

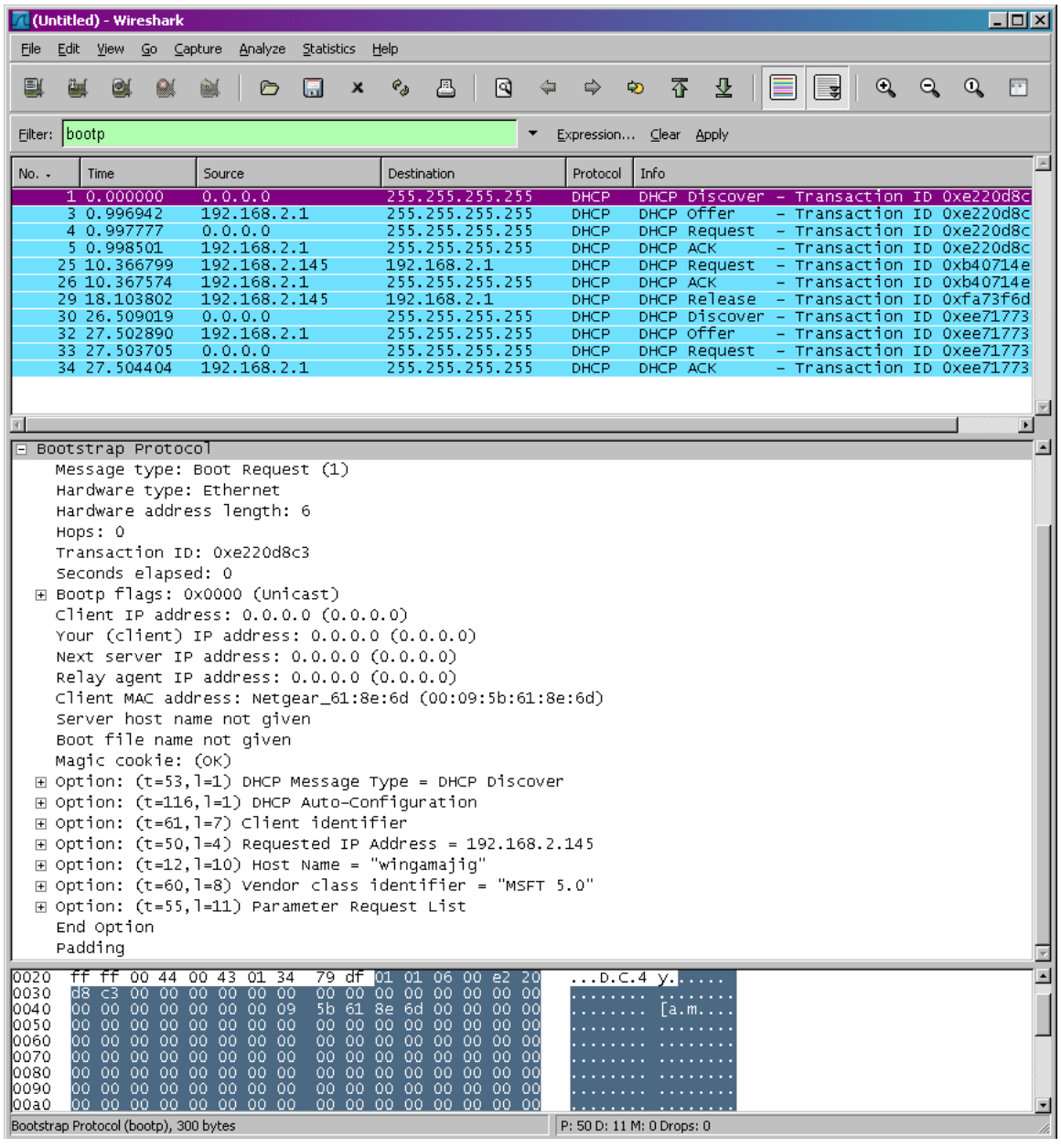


Figure 2 Wireshark window with first DHCP packet – the DHCP Discover packet – expanded.

What to Hand In:

You should hand in a screen shot of the Command Prompt window similar to Figure 1 above. Whenever possible, when answering a question below, you should hand in a printout of the packet(s) within the trace that you used to answer the question asked. Annotate the printout[3] 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 the following questions:

1. Are DHCP messages sent over UDP or TCP?
2. Draw a timing datagram illustrating the sequence of the first four-packet Discover/Offer/Request/ACK DHCP exchange between the client and server. For each packet, indicated the source and destination port numbers. Are the port numbers the same as in the example given in this lab assignment?
3. What is the link-layer (e.g., Ethernet) address of your host?
4. What values in the DHCP discover message differentiate this message from the DHCP request message?
5. What is the value of the Transaction-ID in each of the first four (Discover/Offer/Request/ACK) DHCP messages? What are the values of the Transaction-ID in the second set (Request/ACK) set of DHCP messages? What is the purpose of the Transaction-ID field?
6. A host uses DHCP to obtain an IP address, among other things. But a host's IP address is not confirmed until the end of the four-message exchange! If the IP address is not set until the end of the four-message exchange, then what values are used in the IP datagrams in the four-message exchange? For each of the four DHCP messages (Discover/Offer/Request/ACK DHCP), indicate the source and destination IP addresses that are carried in the encapsulating IP datagram.
7. What is the IP address of your DHCP server?
8. What IP address is the DHCP server offering to your host in the DHCP Offer message? Indicate which DHCP message contains the offered DHCP address.
9. In the example screenshot in this assignment, there is no relay agent between the host and the DHCP server. What values in the trace indicate the absence of a relay agent? Is there a relay agent in your experiment? If so what is the IP address of the agent?
10. Explain the purpose of the router and subnet mask lines in the DHCP offer message.
11. In the DHCP trace file noted in footnote 2, the DHCP server offers a specific IP address to the client (see also question 8. above). In the client's response to the

[3] What do we mean by "annotate"? If you hand in a paper copy, please highlight where in the printout you've found the answer and add some text (preferably with a colored pen) noting what you found in what you 've highlight. If you hand in an electronic copy, it would be great if you could also highlight and annotate.

first server OFFER message, does the client accept this IP address? Where in the client's RESPONSE is the client's requested address?

12. Explain the purpose of the lease time. How long is the lease time in your experiment?
13. What is the purpose of the DHCP release message? Does the DHCP server issue an acknowledgment of receipt of the client's DHCP request? What would happen if the client's DHCP release message is lost?
14. Clear the *bootp* filter from your Wireshark window. Were any ARP packets sent or received during the DHCP packet-exchange period? If so, explain the purpose of those ARP packets.