Computer Network: Lab3

The material you need to submit to the eLearning:

- Experimental report with the file name format "Student ID + Name"

Section	Details	Proportion
IP	Integrity and logic of the experimental process	10%
	Correctness of experimental results	10%
	Questions	20%
DHCP	Integrity and logic of the experimental process	10%
	Correctness of experimental results	10%
	Questions	10%
ICMP	Integrity and logic of the experimental process	10%
	Correctness of experimental results	10%
	Questions	10%

(1) IP

In this lab, we'll investigate the celebrated IP protocol, focusing on the IPv4 and IPv6 datagram. This lab has three parts. In the first part, we'll analyze packets in a trace of IPv4 datagrams sent and received by the traceroute program (the traceroute program itself is explored in more detail in the Wireshark ICMP lab). We'll study IP fragmentation in Part 2 of this lab, and take a quick look at IPv6 in Part 3 of this lab.

In order to generate a trace of IPv4 datagrams for the first two parts of this lab, we'll use the traceroute program to send datagrams of two different sizes to www.fudan.edu.cn. 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.
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 www.fudan.edu.cn 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 www.fudan.edu.cn

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 two traceroute commands, using gaia.cs.umass.edu as the destination, the first with a length of 56 bytes. Once that command has finished executing, enter a second traceroute command for the same destination, but with a length of 3000 bytes.
- Stop Wireshark tracing.

If you're unable to run Wireshark on a live network connection, you can use the packet trace file, *ip-wireshark-trace1-1.pcapng*[1], referenced in footnote 1. 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, as you explore the questions below.

Part1：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. For unification, we use the trace file in wireshark-traces-8.1.zip, please select only UDP and/or ICMP protocol packets.

Your screen:

Answer the following questions:

1. Select the first UDP 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. 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?

---

[1] You can 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*. This trace file can be used to answer these Wireshark lab questions without actually capturing packets on your own. The trace was made using Wireshark running on one of the author's computers, while performing the steps in this Wireshark lab. Once you've downloaded a trace file, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the trace file name.

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 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==192.168.86.61 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:

Answer the following questions:

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).

Your screen:

Answer the following questions:

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?

Part 2: Fragmentation

In this section, we'll look at a large (3000-byte) UDP segment sent by the traceroute program that is fragmented into multiple IP datagrams. Sort the packet listing from Part 1, with any display filters cleared, according to time, by clicking on the Time column.

13. 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. Has that segment been fragmented across more than one IP datagram?

14. What information in the IP header indicates that this datagram been fragmented?
15. What information in the IP header for this packet indicates whether this is the first fragment versus a latter fragment?
16. How many bytes are there in is this IP datagram (header plus payload)?
17. Now inspect the datagram containing the second fragment of the fragmented UDP segment. What information in the IP header indicates that this is not the first datagram fragment?
18. What fields change in the IP header between the first and second fragment?
19. 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?

Part 3: IPv6

In this final section we'll take a quick look at the IPv6 datagram using Wireshark. Because the Internet is still primarily at IPv4 network, and your own computer or your ISP may not be configured for IPv6, let's look at a trace of already captured packets that contain some IPv6 packets. To generate this trace, our web browser opened the youtube.com homepage. Youtube (and Google) provide fairly widespread support for IPv6. Open the file *ip-wireshark-trace2-1.pcapng* in the .zip file of traces given in footnote 1.

Let's start by taking a closer look at the 20th packet in this trace, sent at t=3.814489. This is a DNS request (contained in an IPv6 datagram) to an IPv6 DNS server for the IPv6 address of youtube.com. The DNS AAAA request type is used to resolve names to IPv6 IP addresses.

Answer the following questions:

20. What is the IPv6 address of the computer making the DNS AAAA request? This is the source address of the 20th packet in the trace. Give the IPv6 source address for this datagram in the exact same form as displayed in the Wireshark window.
21. What is the IPv6 destination address for this datagram? Give this IPv6 address in the exact same form as displayed in the Wireshark window.
22. What is the value of the flow label for this datagram?
23. How much payload data is carried in this datagram?
24. What is the upper layer protocol to which this datagram's payload will be delivered at the destination?

Lastly, find the IPv6 DNS response to the IPv6 DNS AAAA request made in the 20th packet in this trace. This DNS response contains IPv6 addresses for youtube.com.

Answer the following questions:

25. How many IPv6 addresses are returned in the response to this AAAA request?

26. What is the first of the IPv6 addresses returned by the DNS for youtube.com? Give this IPv6 address in the exact same shorthand form as displayed in the Wireshark window.

(2) DHCP

Gathering a Packet Trace

On a Mac:

1. In a terminal window/shell enter the following command:

   % sudo ipconfig set en0 none

   Where en0 (in this example) is the interface on which you want to capture packets using Wireshark. You can easily find the list of interface names in Wireshark by choosing Capture->options. This command will de-configure network interface en0.
2. Start up Wireshark, capturing packets on the interface you de-configured in Step 1.
3. In the terminal window/shell enter the following command:

   % sudo ipconfig set en0 dhcp

   This will cause the DHCP protocol to request and receive an IP address and other information from the DHCP server.
4. After waiting for a few seconds, stop Wireshark capture.

On a Linux machine:

1. In a terminal window/shell, enter the following commands:

   sudo ip addr flush en0

   sudo dhclient -r

   where en0 (in this example) is the interface on which you want to capture packets using Wireshark. You can easily find the list of interface names in Wireshark by choosing Capture -> Options. This command will remove the existing IP address of the interface, and release any existing DHCP address leases.
2. Start up Wireshark, capturing packets in the interface you de-configured in Step 1.
3. In the terminal window/shell, enter the following command:

   sudo dhclient en0

   where, as with above, en0 is the interface on which you are currently capturing packets. This will cause the DHCP protocol to request and receive an IP address and other information from the DHCP server.
4. After waiting for a few seconds, stop Wireshark capture.

On a PC:

1. In a command-line window enter the following command:

   > ipconfig /release

This command will cause your PC to give up its IP address.

2. Start up Wireshark.
3. In the command-line window enter the following command:

   > ipconfig /renew

   This will cause the DHCP protocol to request and receive an IP address and other information from a DHCP server.
4. After waiting for a few seconds, stop Wireshark capture.
5. After stopping Wireshark capture in step 4, you should take a peek in your Wireshark window to make sure you've actually captured the packets that we're looking for. If you enter "dhcp" into the display filter field (as shown in the light green field in the top left of Figure 1), your screen (on a Mac) should look similar to Figure 1.

On a Mac:

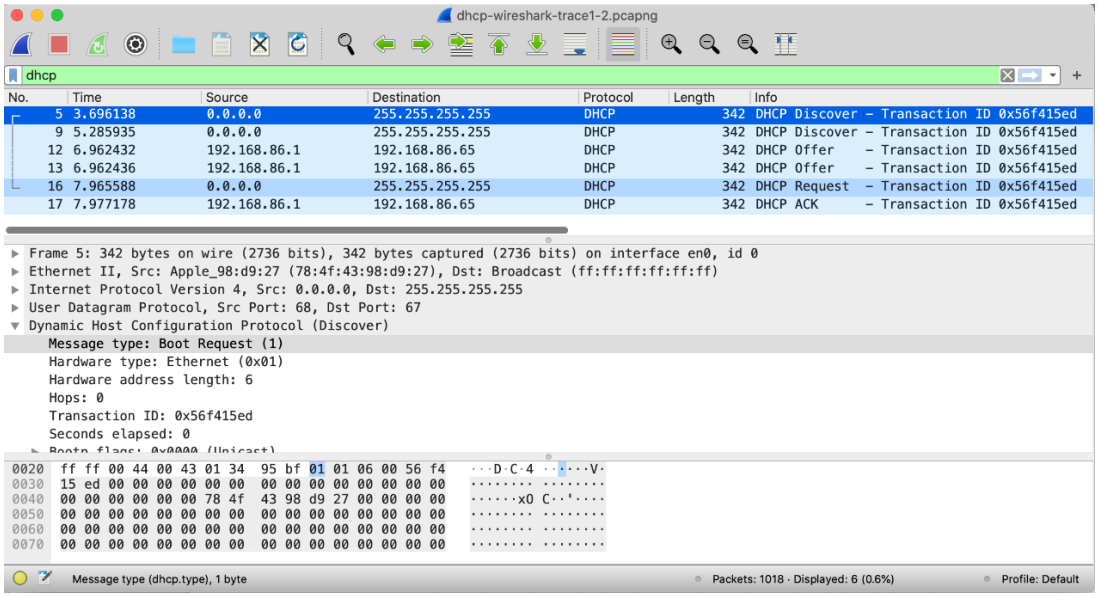

Figure 1 Wireshark display, showing the capture of DHCP Discover, Offer, Request and ACK messages

Your screen:

If you're unable to run Wireshark on a live network connection, are unable to capture all four DHCP messages, you can use the Wireshark trace file, dhcp-wireshark-trace1-1.pcapng[2] that we've gathered following the steps above on one of the author's

[2] You can download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip and extract the trace file *dhcp-wireshark-trace1-1.pcapng*. These trace files can be used to answer these Wireshark lab questions without actually capturing packets on your own. Each trace was made using Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark lab. Once you've downloaded a trace file, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the trace file name.

computers.

Let's start by looking at the DHCP Discover message. Locate the IP datagram containing the first Discover message in your trace.

Answer the following questions:

1. Is this DHCP Discover message sent out using UDP or TCP as the underlying transport protocol?
2. What is the source IP address used in the IP datagram containing the Discover message? Is there anything special about this address? Explain.
3. What is the destination IP address used in the datagram containing the Discover message. Is there anything special about this address? Explain.
4. What is the value in the transaction ID field of this DHCP Discover message?
5. Now inspect the options field in the DHCP Discover message. What are five pieces of information (beyond an IP address) that the client is suggesting or requesting to receive from the DHCP server as part of this DHCP transaction?

Now let's look at the DHCP Offer message. Locate the IP datagram containing the DHCP Offer message in your trace that was sent by a DHCP server in the response to the DHCP Discover message that you studied in questions 1-5 above.

Answer the following questions:

6. How do you know that this Offer message is being sent in response to the DHCP Discover message you studied in questions 1-5 above?
7. What is the *source* IP address used in the IP datagram containing the Offer message? Is there anything special about this address? Explain.
8. What is the *destination* IP address used in the datagram containing the Offer message? Is there anything special about this address? Explain.
9. Now inspect the options field in the DHCP Offer message. What are five pieces of information that the DHCP server is providing to the DHCP client in the DHCP Offer message?

It would appear that once the DHCP Offer message is received, that the client may have all of the information it needs to proceed. However, the client may have received OFFERs from multiple DHCP servers and so a second phase is needed, with two more mandatory messages – the client-to-server DHCP Request message, and the server-to-client DHCP ACK message is needed. But at least the client knows there is at least one DHCP server out there! Let's take a look at the DHCP Request message, remembering that although we've already seen a Discover message in our trace, that is not always the case when a DHCP request message is sent. Locate the IP datagram containing the first DHCP Request message in your trace.

Answer the following questions:

10. What is the UDP source port number in the IP datagram containing the first DHCP Request message in your trace? What is the UDP destination port number being used?

11. What is the source IP address in the IP datagram containing this Request message? Is there anything special about this address? Explain.
12. What is the destination IP address used in the datagram containing this Request message. Is there anything special about this address? Explain.
13. What is the value in the transaction ID field of this DHCP Request message? Does it match the transaction IDs of the earlier Discover and Offer messages?

Locate the IP datagram containing the first DHCP ACK message in your trace.

Answer the following questions:

14. What is the source IP address in the IP datagram containing this ACK message? Is there anything special about this address? Explain.
15. What is the destination IP address used in the datagram containing this ACK message. Is there anything special about this address? Explain.
16. What is the name of the field in the DHCP ACK message (as indicated in the Wireshark window) that contains the assigned client IP address?
17. For how long a time (the so-called "lease time") has the DHPC server assigned this IP address to the client?
18. What is the IP address (returned by the DHCP server to the DHCP client in this DHCP ACK message) of the first-hop router on the default path from the client to the rest of the Internet?

(3) ICMP

Part1: ICMP and Ping

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.

Do the following:

- Let's begin this adventure by opening the Windows Command Prompt application (which can be found in your Accessories folder).
- Start up the Wireshark packet sniffer, and begin Wireshark packet capture.
- The *ping* command is in c:\windows\system32, so type either "*ping –n 10 www.fudan.edu.cn*" or "*c:\windows\system32\ping –n 10 www.fudan.edu.cn*" in the MS-DOS command line (without quotation marks). The argument *"-n 10*" indicates that 10 ping messages should be sent. Then run the Ping program by typing return.
- When the Ping program terminates, stop the packet capture in Wireshark.

Your screen:

Command Prompt Window:

Wireshark:

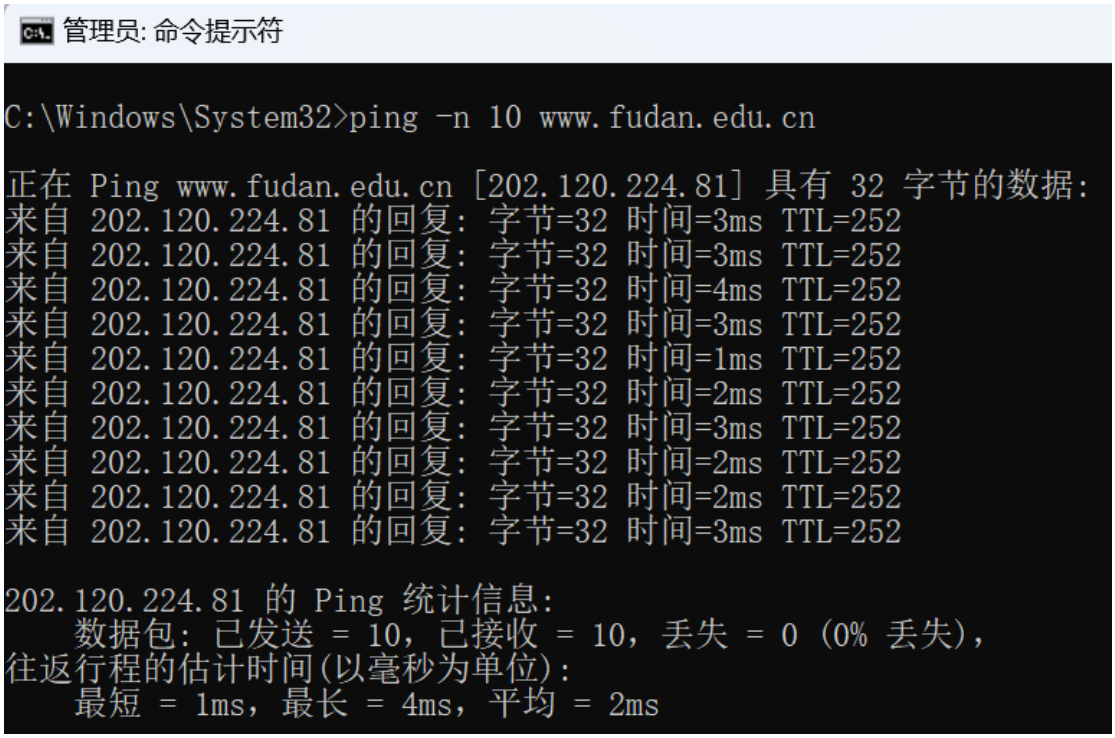

Figure 2 Command Prompt window after entering Ping command.

At the end of the experiment, your Command Prompt Window should look something like Figure 2. From this window we see that the source ping program sent 10 query packets and received 10 responses.

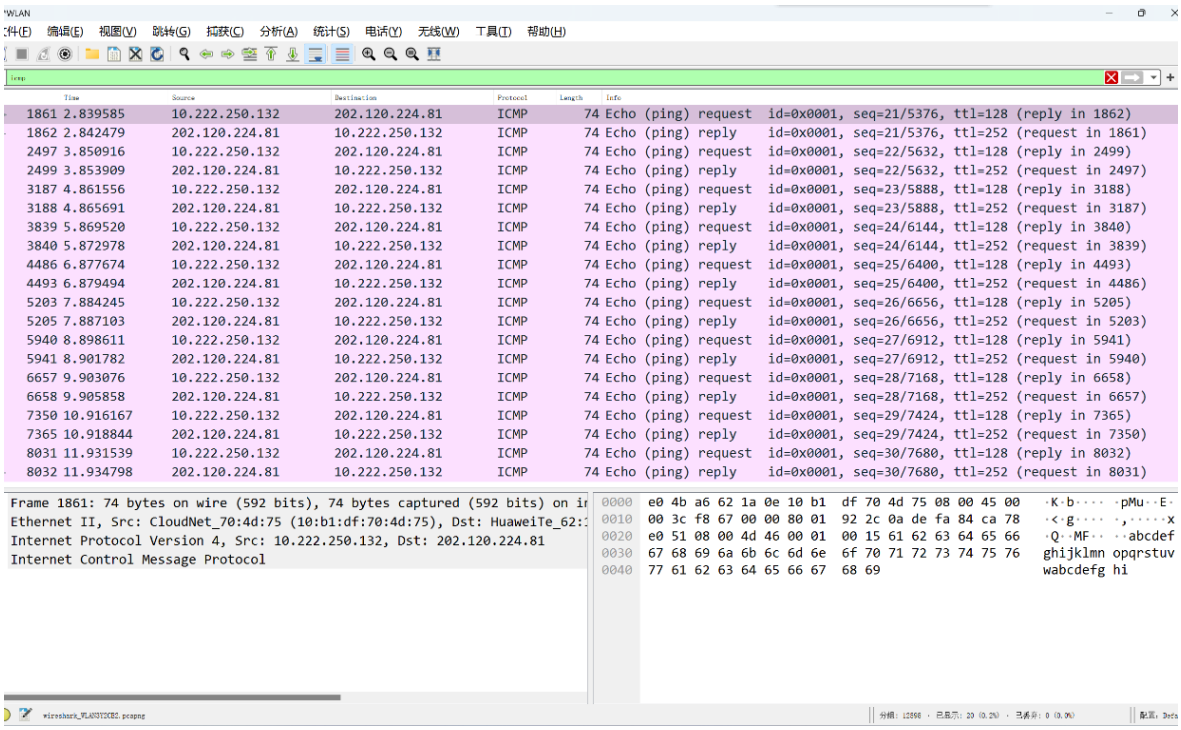

Figure 3 Wireshark output for Ping program with Internet Protocol expanded.

Figure 3 provides a screenshot of the Wireshark output, after "icmp" has been entered into the filter display window. Note that the packet listing shows 20 packets: the 10 Ping queries sent by the source and the 10 Ping responses received by the source.

Answer the following questions:

1. What is the IP address of your host? What is the IP address of the destination host?
2. Why is it that an ICMP packet does not have source and destination port numbers?
3. Examine one of the ping request packets sent by your host. 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?
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?

Part2: ICMP and Traceroute

Let's now continue our ICMP adventure by capturing the packets generated by the Traceroute program. You may recall that the Traceroute program can be used to figure out the path a packet takes from source to destination.

Do the following:

- Let's begin by opening the Windows Command Prompt application.
- Start up the Wireshark packet sniffer, and begin Wireshark packet capture.
- The *tracert* command is in c:\windows\system32, so type either "*tracert www.fudan.edu.cn*" or "*c:\windows\system32\tracert www.fudan.edu.cn*" in the MS-DOS command line (without quotation marks), where hostname is a host on another continent. (Note that on a Windows machine, the command is "*tracert*" and not "*traceroute*".) Then run the Traceroute program by typing return.
- When the Traceroute program terminates, stop packet capture in Wireshark.

Your screen:

Command Prompt Window:

Wireshark:

Answer the following questions:

5. What is the IP address of your host? What is the IP address of the target destination host?
6. If ICMP sent UDP packets instead (as in Unix/Linux), would the IP protocol number still be 01 for the probe packets? If not, what would it be?
7. 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?

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?