

# **ELEC3506/9506 Communication Networks**

## **– Lab 02**

### **Instruction**

This lab instruction comprises two phases, where each contains multiple questions that students need to answer them during the lab session to comprehend the main goal of the phase. Although, it is not mandatory to bring the answers into your Final Lab Report. Indeed, the Final Lab Report you need to submit to the Canvas should be within maximum of 7 pages, stating your overall learning experience from the lab experiments and the main results you have obtained. In the first phase, several aspects of the Internet control message protocol (ICMP) are explored. Indeed, students learn how to generate the ICMP messages through two different programs, namely Ping program and Traceroute program. Also, they get familiar with the format and contents of an ICMP message. In the second phase, students investigate a well-known protocol, namely Internet protocol (IP), focusing on the IP datagram. Indeed, students execute the Traceroute program on their systems to send and receive IP datagrams. Thereafter, students can analyze the various fields in the IP datagram, trace the IP datagrams, and study IP fragmentations in details.

A template has been provided for each Lab Report that you should follow it to prepare the Final Lab Report. You can find the template in the Canvas. Before starting the lab, students should follow the instructions provided for the lab session. The instructions are summarized as follows:

- You need to follow up the lab instruction, provided in the Canvas, regarding forming groups and submitting the Final Lab Reports.
- Complete the lab according to the instructions in the following pages.
- You should observe lab rules and behave properly in the lab.
- Relevant materials: ICMP - pages 621-630; Ping - page 627; Traceroute - page 628;  
Echo Request and Reply - page 625; IPv4 - pages 582-596;  
IP Higher-level Protocol Values - page 588.

Notation: Students can find the corresponding concepts in the main reference provided for this course:

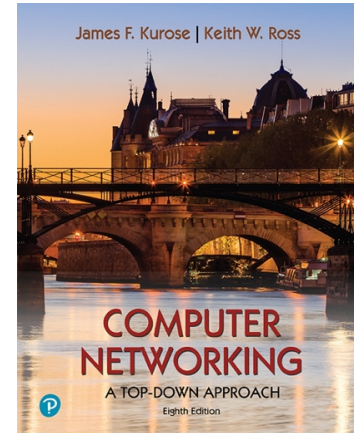
# Phase 1:

## ICMP

Supplement to *Computer Networking: A Top-Down Approach*, 8<sup>th</sup> 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

© 2005-2020, J.F Kurose and K.W. Ross, All Rights Reserved



In this lab, we'll explore several aspects of the ICMP protocol:

- ICMP messages generated by the Ping program;
- ICMP messages generated by the Traceroute program;
- the format and contents of an ICMP message.

Before attacking this lab, you're encouraged to review the ICMP material in section 5.6 of the text<sup>1</sup>. We present this lab in the context of the Microsoft Windows operating system. However, it is straightforward to translate the lab to a Unix or Linux environment.

## 1. ICMP and Ping

Let's begin our ICMP adventure by capturing the packets generated by the Ping program. You may recall that 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. As you might have guessed (given that this lab is about ICMP), both of these Ping packets are ICMP packets.

Do the following<sup>2</sup>:

---

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

<sup>2</sup> 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 *ICMP-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 *ICMP-ethereal-trace-1* trace file. You can then use this trace file to answer the questions below.

- 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 hostname*" or "*c:\windows\system32\ping -n 10 hostname*" in the MS-DOS command line (without quotation marks), where *hostname* is a host on another continent. If you're outside of Asia, you may want to enter `www.ust.hk` for the Web server at Hong Kong University of Science and Technology. 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.

At the end of the experiment, your Command Prompt Window should look something like Figure 1. In this example, the source ping program is in Massachusetts and the destination Ping program is in Hong Kong. From this window we see that the source ping program sent 10 query packets and received 10 responses. Note also that for each response, the source calculates the round-trip time (RTT), which for the 10 packets is on average 375 msec.

```

C:\WINDOWS\SYSTEM32>ping -n 10 www.ust.hk

Pinging www.ust.hk [143.89.14.34] with 32 bytes of data:

Reply from 143.89.14.34: bytes=32 time=415ms TTL=231
Reply from 143.89.14.34: bytes=32 time=425ms TTL=231
Reply from 143.89.14.34: bytes=32 time=318ms TTL=231
Reply from 143.89.14.34: bytes=32 time=314ms TTL=231
Reply from 143.89.14.34: bytes=32 time=336ms TTL=231
Reply from 143.89.14.34: bytes=32 time=359ms TTL=231
Reply from 143.89.14.34: bytes=32 time=381ms TTL=231
Reply from 143.89.14.34: bytes=32 time=401ms TTL=231
Reply from 143.89.14.34: bytes=32 time=400ms TTL=231
Reply from 143.89.14.34: bytes=32 time=409ms TTL=231

Ping statistics for 143.89.14.34:
    Packets: Sent = 10, Received = 10, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 314ms, Maximum = 425ms, Average = 375ms

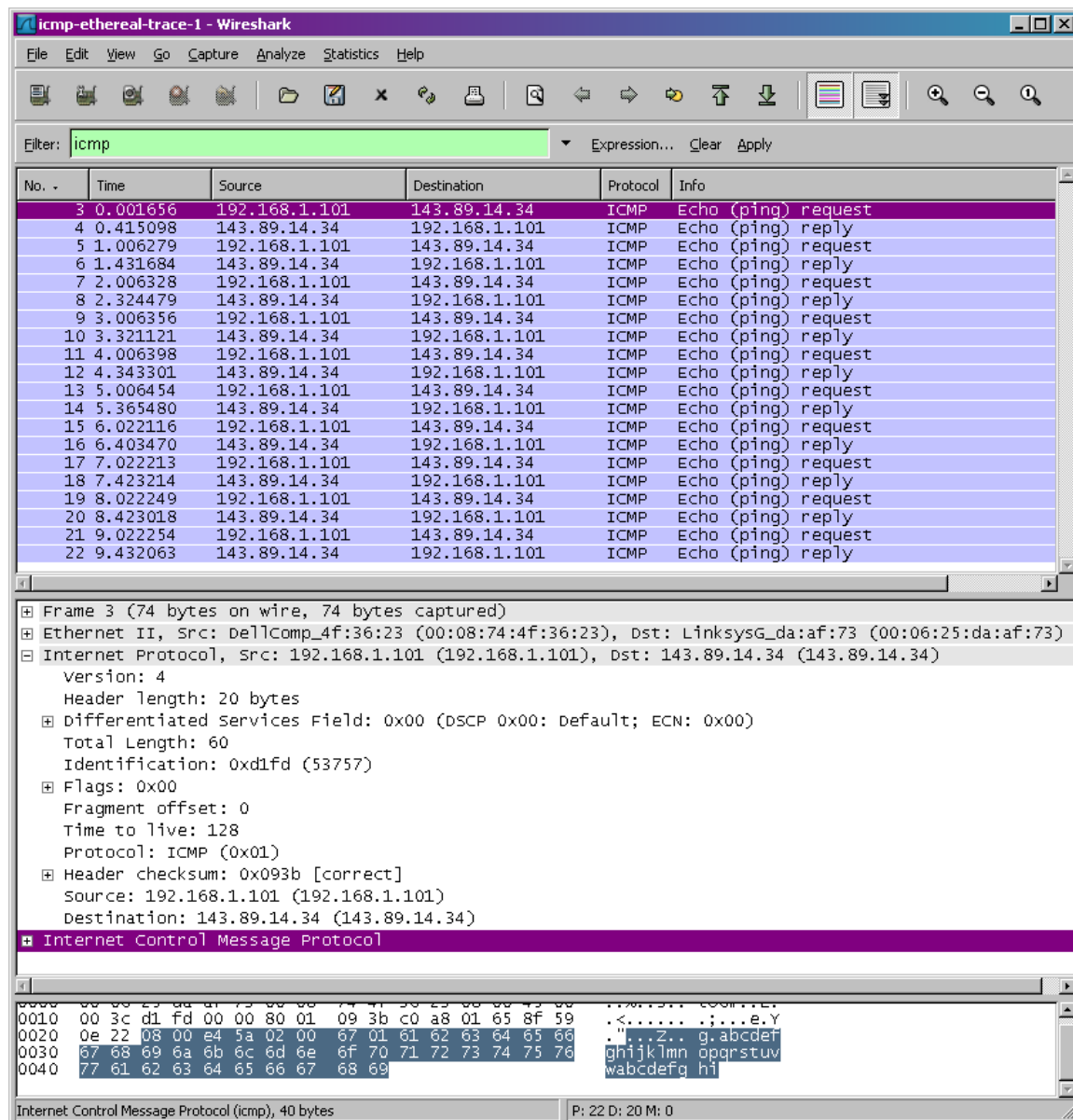
C:\WINDOWS\SYSTEM32>
C:\WINDOWS\SYSTEM32>
C:\WINDOWS\SYSTEM32>_

```

**Figure 1** Command Prompt window after entering Ping command.

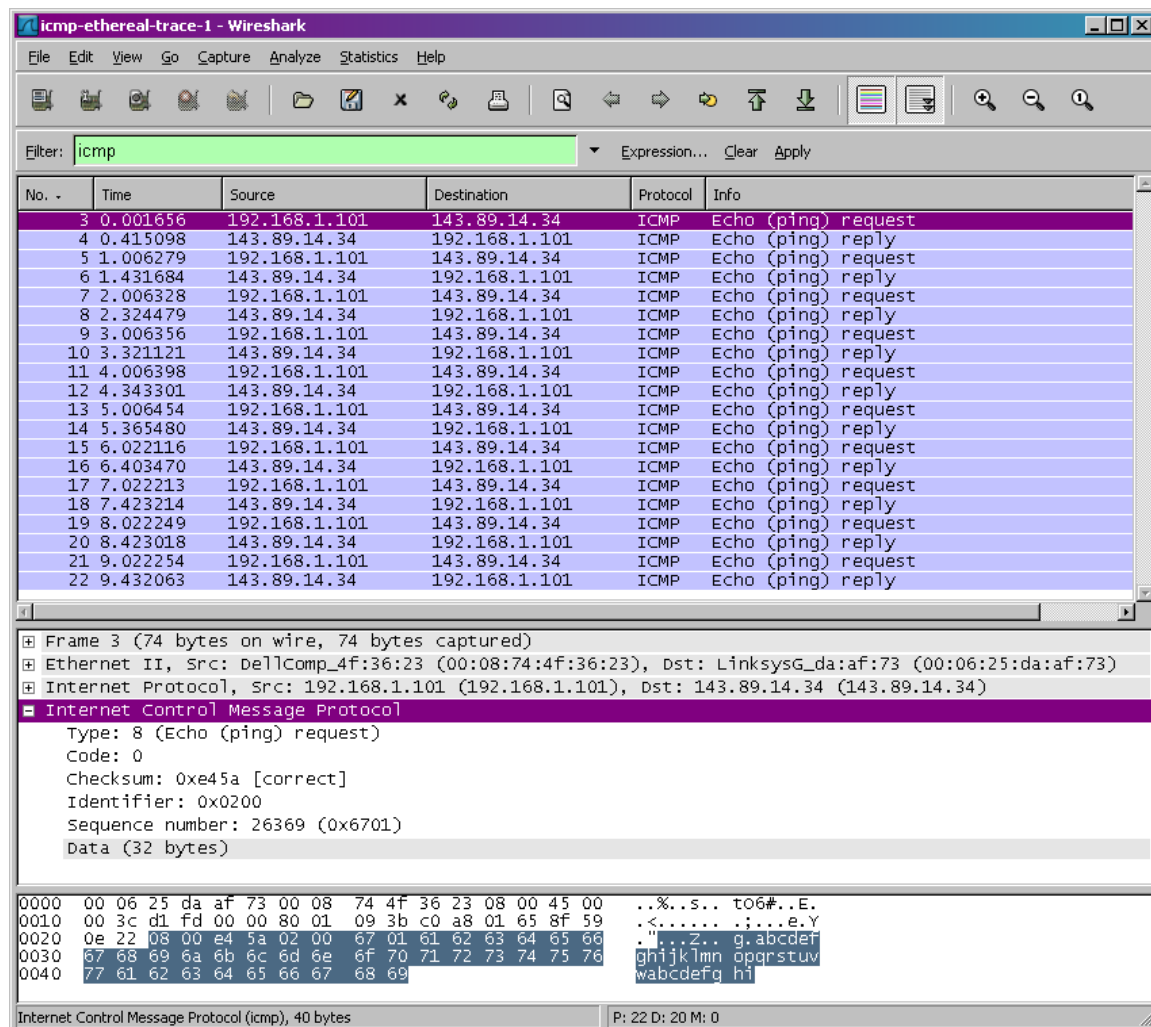
Figure 2 provides a screenshot of the Wireshark output, after "icmp" has been entered into the filter display window. Note that the packet listing shows 20 packets: the 10 Ping queries sent by the source and the 10 Ping responses received by the source. Also note that the source's IP address is a private address (behind a NAT) of the form 192.168/12; the destination's IP address is that of the Web server at HKUST. Now let's zoom in on the first packet (sent by the client); in the figure below, the packet contents area provides

information about this packet. We see that the IP datagram within this packet has protocol number 01, which is the protocol number for ICMP. This means that the payload of the IP datagram is an ICMP packet.



**Figure 2** Wireshark output for Ping program with Internet Protocol expanded.

Figure 3 focuses on the same ICMP but has expanded the ICMP protocol information in the packet contents window. Observe that this ICMP packet is of Type 8 and Code 0 - a so-called ICMP “echo request” packet. (See Figure 5.19 of text.) Also note that this ICMP packet contains a checksum, an identifier, and a sequence number.



**Figure 3** Wireshark capture of ping packet with ICMP 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<sup>3</sup> 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.

You should answer the following questions:

<sup>3</sup> 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.

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?

## 2. 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. Traceroute is discussed in Section 1.4 and in Section 5.6 of the text.

Traceroute is implemented in different ways in Unix/Linux/macOS and in Windows. In Unix/Linux, the source sends a series of UDP packets to the target destination using an unlikely destination port number; in Windows, the source sends a series of ICMP packets to the target destination. For both operating systems, the program sends the first packet with TTL=1, the second packet with TTL=2, and so on. Recall that a router will decrement a packet's TTL value as the packet passes through the router. When a packet arrives at a router with TTL=1, the router sends an ICMP error packet back to the source. In the following, we'll use the native Windows *tracert* program. A shareware version of a much nicer Windows Traceroute program is *pingplotter* ([www.pingplotter.com](http://www.pingplotter.com)). We'll use *pingplotter* in our Wireshark IP lab since it provides additional functionality that we'll need there.

Do the following<sup>4</sup>:

- Let's begin 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 *tracert* command is in `c:\windows\system32`, so type either "*tracert hostname*" or "*c:\windows\system32\tracert hostname*" in the MS-DOS command line (without quotation marks), where *hostname* is a host on another continent.

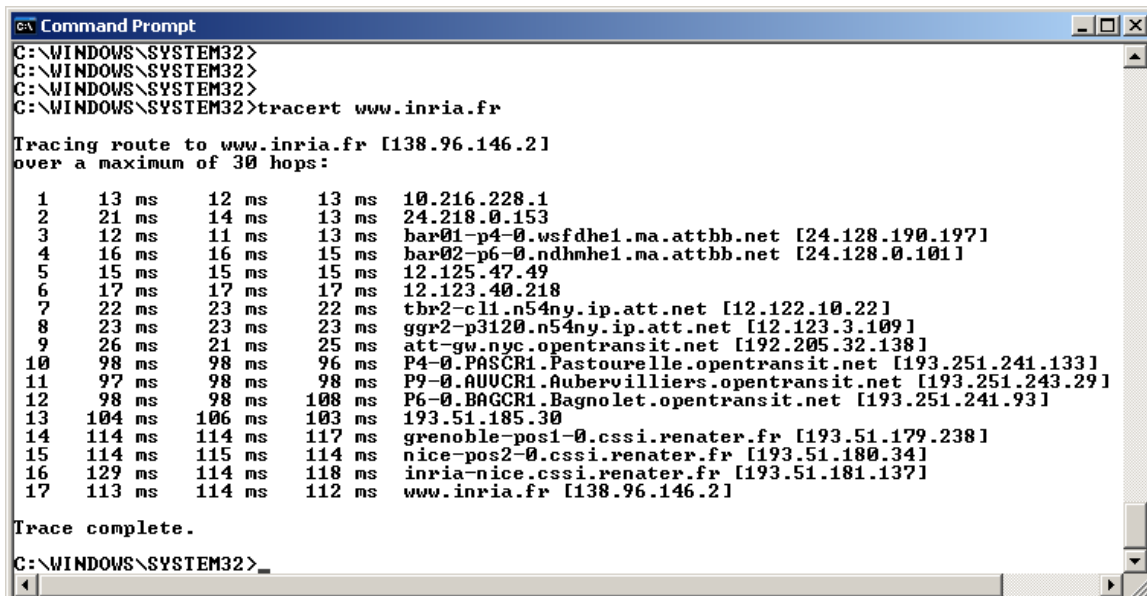
---

<sup>4</sup> 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 *ICMP-ethereal-trace-2*. 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 *ICMP-ethereal-trace-2* trace file. You can then use this trace file to answer the questions below.

(Note that on a Windows machine, the command is “*tracert*” and not “*traceroute*”). If you’re outside of Europe, you may want to enter `www.inria.fr` for the Web server at INRIA, a computer science research institute in France. Then run the Traceroute program by typing return.

- When the Traceroute program terminates, stop packet capture in Wireshark.

At the end of the experiment, your Command Prompt Window should look something like Figure 4. In this figure, the client Traceroute program is in Massachusetts and the target destination is in France. From this figure we see that for each TTL value, the source program sends three probe packets. Traceroute displays the RTTs for each of the probe packets, as well as the IP address (and possibly the name) of the router that returned the ICMP TTL-exceeded message.



```
C:\WINDOWS\SYSTEM32>
C:\WINDOWS\SYSTEM32>
C:\WINDOWS\SYSTEM32>
C:\WINDOWS\SYSTEM32>tracert www.inria.fr

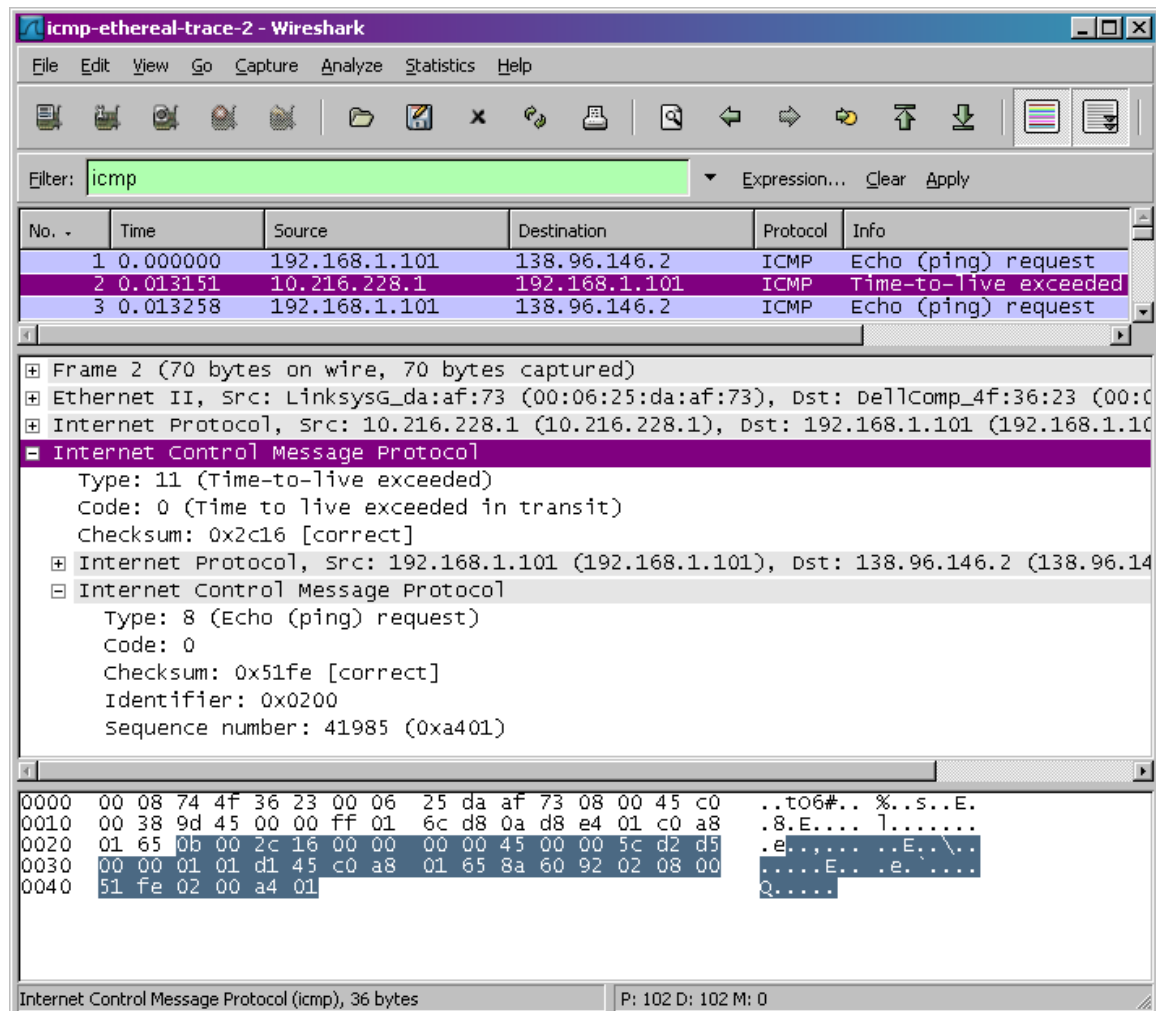
Tracing route to www.inria.fr [138.96.146.2]
over a maximum of 30 hops:
  1  13 ms  12 ms  13 ms  10.216.228.1
  2  21 ms  14 ms  13 ms  24.218.0.153
  3  12 ms  11 ms  13 ms  bar01-p4-0.wsfdhe1.ma.attbb.net [24.128.190.197]
  4  16 ms  16 ms  15 ms  bar02-p6-0.ndhmhe1.ma.attbb.net [24.128.0.101]
  5  15 ms  15 ms  15 ms  12.125.47.49
  6  17 ms  17 ms  17 ms  12.123.40.218
  7  22 ms  23 ms  22 ms  tbr2-cl1.n54ny.ip.att.net [12.122.10.22]
  8  23 ms  23 ms  23 ms  ggr2-p3120.n54ny.ip.att.net [12.123.3.109]
  9  26 ms  21 ms  25 ms  att-gw.nyc.opentransit.net [192.205.32.138]
 10 98 ms  98 ms  96 ms  P4-0.PASCR1.Pastourelle.opentransit.net [193.251.241.133]
 11 97 ms  98 ms  98 ms  P9-0.AUUCR1.Aubervilliers.opentransit.net [193.251.243.29]
 12 98 ms  98 ms  108 ms P6-0.BAGCR1.Bagnolet.opentransit.net [193.251.241.93]
 13 104 ms 106 ms 103 ms 193.51.185.30
 14 114 ms 114 ms 117 ms grenoble-pos1-0.cssi.renater.fr [193.51.179.238]
 15 114 ms 115 ms 114 ms nice-pos2-0.cssi.renater.fr [193.51.180.34]
 16 129 ms 114 ms 118 ms inria-nice.cssi.renater.fr [193.51.181.137]
 17 113 ms 114 ms 112 ms www.inria.fr [138.96.146.2]

Trace complete.
C:\WINDOWS\SYSTEM32>
```

**Figure 4** Command Prompt window displays the results of the Traceroute program.



Figure 5 displays the Wireshark window for an ICMP packet returned by a router. Note that this ICMP error packet contains many more fields than the Ping ICMP messages.



**Figure 5** Wireshark window of ICMP fields expanded for one ICMP error packet.

## What to Hand In:

For this part of the lab, you should hand in a screen shot of the Command Prompt window. 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 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:

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?
10. Within the tracer measurements, is there a link whose delay is significantly longer than others? Refer to the screenshot in Figure 4, is there a link whose delay is significantly longer than others? On the basis of the router names, can you guess the location of the two routers on the end of this link?

### 3. Extra Credit

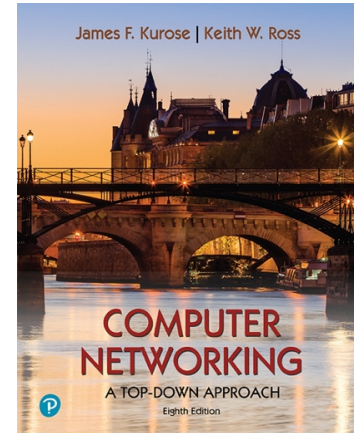
For one of the programming assignments you created a UDP client ping program. This ping program, unlike the standard ping program, sends UDP probe packets rather than ICMP probe packets. Use the client program to send a UDP packet with an unusual destination port number to some live host. At the same time, use Wireshark to capture any response from the target host. Provide a Wireshark screenshot for the response as well as an analysis of the response.

# Phase 2: IP

Supplement to *Computer Networking: A Top-Down Approach*, 8<sup>th</sup> 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

© 2005-2020, J.F Kurose and 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 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<sup>1</sup> and section 3.4 of RFC 2151 [[ftp://ftp.rfc-editor.org/in-notes/rfc2151.txt](http://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](http://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

---

<sup>1</sup> References to figures and sections are for the 8<sup>th</sup> edition of our text, *Computer Networks, A Top-down Approach*, 8<sup>th</sup> 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 `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.

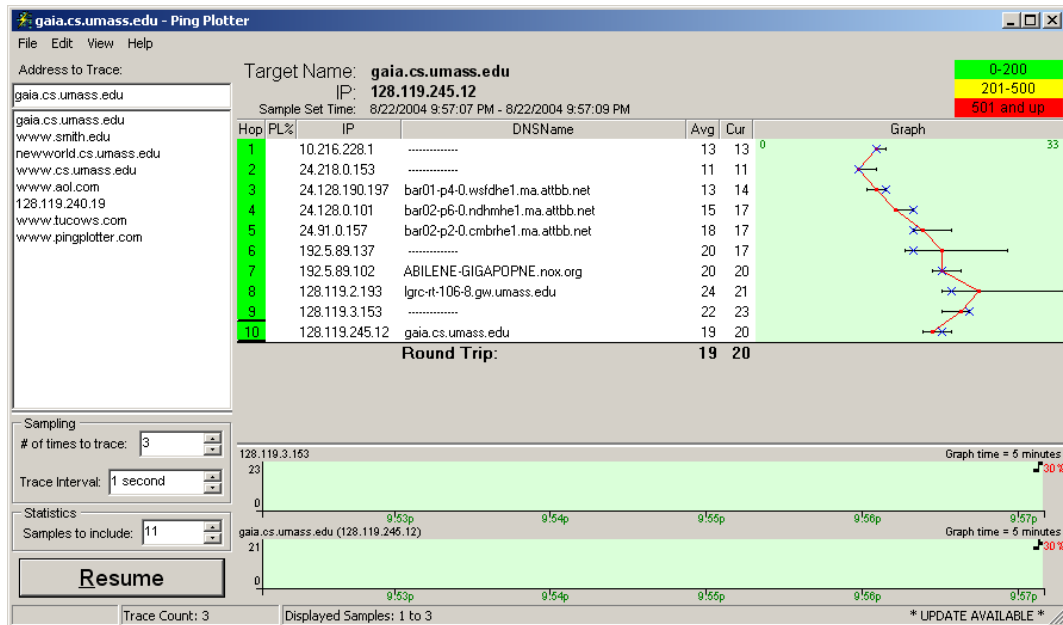
We'll want to run `tracert` and have it send datagrams of various lengths.

- **Windows.** The `tracert` program (used for our ICMP 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 `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->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 `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 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<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/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.

ip-ethereal-trace-1 [Wireshark 1.6.7 (SVN Rev 41973 from /trunk-1.6)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Telebit_73:8d:ce	Broadcast	ARP	60	Who has 192.168.1.117? Tell 192.168.1.104
2	4.866867	192.168.1.100	192.168.1.1	UDP	174	Source port: 30955 Destination port: ssdp
3	4.868147	192.168.1.100	192.168.1.1	UDP	175	Source port: 30955 Destination port: ssdp
4	5.363536	192.168.1.100	192.168.1.1	UDP	174	Source port: 30955 Destination port: ssdp
5	5.364799	192.168.1.100	192.168.1.1	UDP	175	Source port: 30955 Destination port: ssdp
6	5.864428	192.168.1.100	192.168.1.1	UDP	174	Source port: 30955 Destination port: ssdp
7	5.865461	192.168.1.100	192.168.1.1	UDP	175	Source port: 30955 Destination port: ssdp
8	6.163045	192.168.1.102	128.59.23.100	ICMP	98	Echo (ping) request id=0x0300, seq=20483/848, ttl=1
9	6.176826	10.216.228.1	192.168.1.102	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
10	6.188629	192.168.1.102	128.59.23.100	ICMP	98	Echo (ping) request id=0x0300, seq=20739/849, ttl=2
11	6.202957	24.218.0.153	192.168.1.102	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
12	6.208597	192.168.1.102	128.59.23.100	ICMP	98	Echo (ping) request id=0x0300, seq=20995/850, ttl=3
13	6.234505	24.128.190.197	192.168.1.102	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)

Frame 8: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)

Ethernet II, Src: Actionte\_8a:70:1a (00:20:e0:8a:70:1a), Dst: LinksysG\_da:af:73 (00:06:25:da:af:73)

Internet Protocol Version 4, Src: 192.168.1.102 (192.168.1.102), Dst: 128.59.23.100 (128.59.23.100)

Version: 4

Header length: 20 bytes

Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))

Total Length: 84

Identification: 0x32d0 (13008)

Flags: 0x00

Fragment offset: 0

0000 00 06 25 da af 73 00 20 e0 8a 70 1a 08 00 45 00 ... ..

0010 00 54 32 40 00 00 01 01 2d 2c c0 a8 01 66 80 3b ... ..

0020 17 64 08 00 f7 ca 03 00 50 03 37 32 20 aa aa aa ... ..

0030 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ... ..

0040 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ... ..

0050 aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ... ..

0060 aa aa

Frame (frame), 98 bytes | Packets: 380 Displayed: 380 Marked: 0 Load time: 0:00.006 | Profile: Default

- 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.<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/wireshark-labs/wireshark-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 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 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?