

NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCE

Computer Network Lab (CL-307)

Lab Session 07

Awais Ahmed || Faizan Yousuf || Munim Ali

awais.ahmed@nu.edu.pk || faizan.yousuf@nu.edu.pk || munim.ali@nu.edu.pk

INTRODUCTION TO WIRESHARK

In this lab, we'll investigate the behavior of the celebrated TCP protocol in detail. We will do so by analyzing a trace of the TCP segments sent and received in transferring a 150KB file (containing the text of Lewis Carroll's *Alice's Adventures in Wonderland*) from your computer to a remote server. We'll study TCP's use of sequence and acknowledgement numbers for providing reliable data transfer.

OBJECTIVES

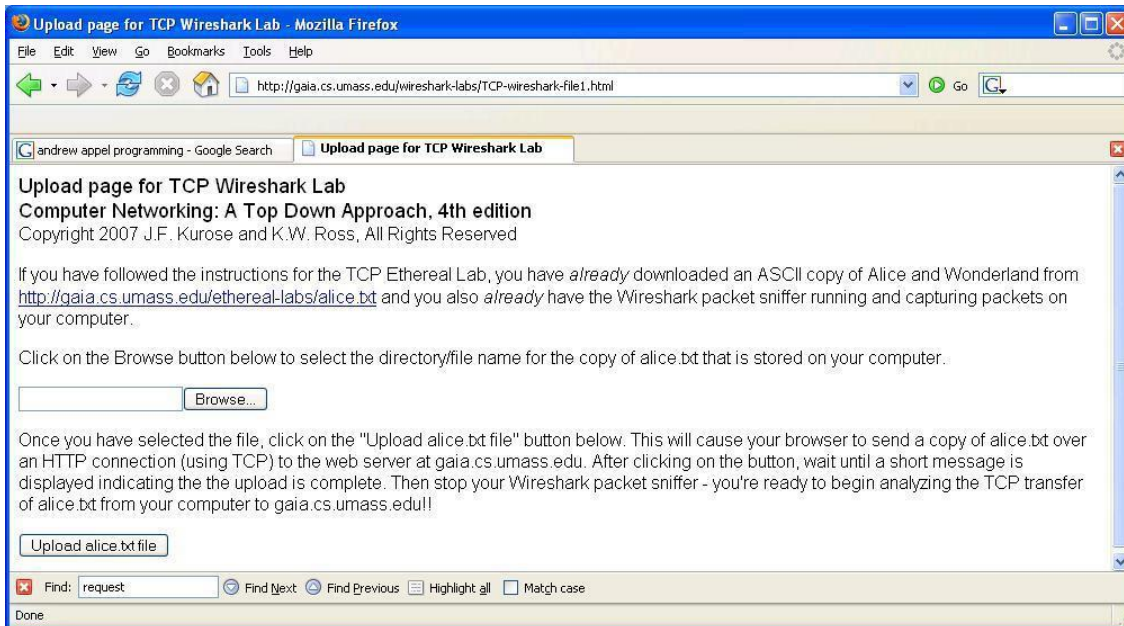
- Introduction to Wireshark Tool
- Capturing a bulk TCP transfer from your computer to a remote server.
- A first look at the captured trace.
- TCP Basics and activity.
- UDP Basics and activity.

CAPTURING A BULK TCP TRANSFER FROM YOUR COMPUTER TO A REMOTE SERVER

Before beginning our exploration of TCP, we'll need to use Wireshark to obtain a packet trace of the TCP transfer of a file from your computer to a remote server. You'll do so by accessing a Web page that will allow you to enter the name of a file stored on your computer (which contains the ASCII text of *Alice in Wonderland*), and then transfer the file to a Web server using the HTTP POST method. We're using the POST method rather than the GET method as we'd like to transfer a large amount of data *from* your computer to another computer. Of course, we'll be running Wireshark during this time to obtain the trace of the TCP segments sent and received from your computer.

Do the following:

1. Start up your web browser. Go the <http://gaia.cs.umass.edu/wireshark-labs/alice.txt> and retrieve an ASCII copy of *Alice in Wonderland*. Store this file somewhere on your computer.
2. Next go to <http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html>. You should see a screen that looks like:

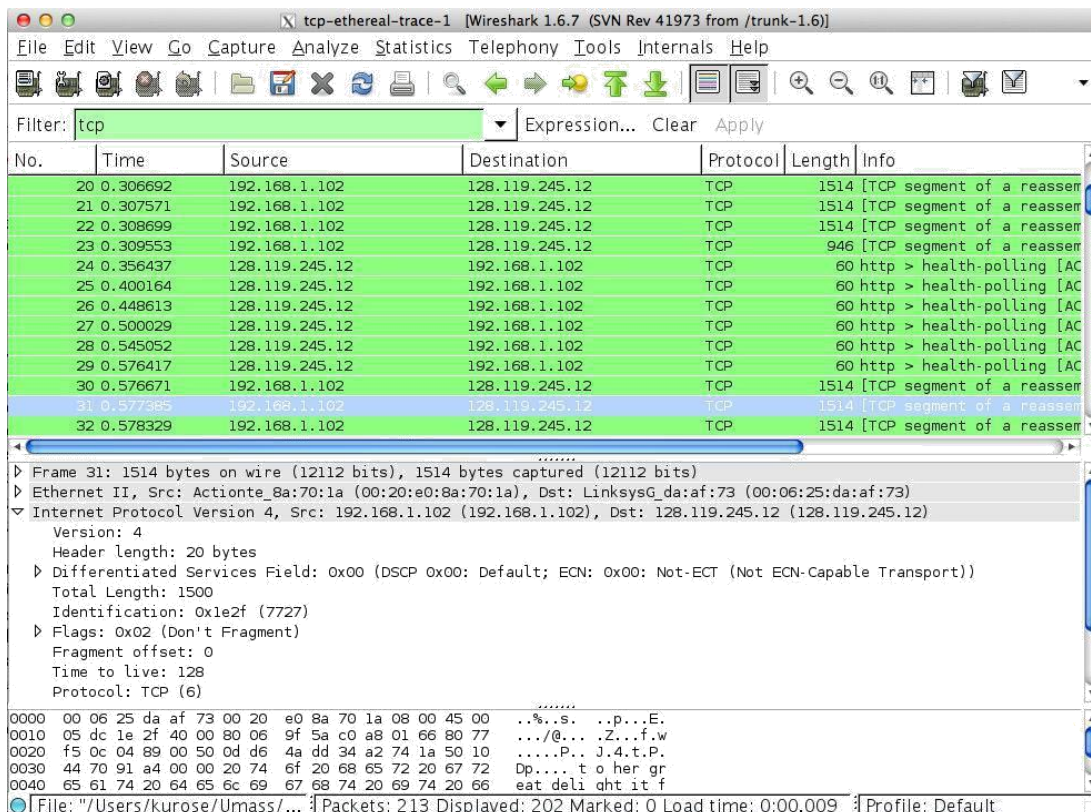


Use the **Browse** button in this form to enter the name of the file (full path name) on your computer containing *Alice in Wonderland* (or do so manually). Don't yet press the **"Upload alice.txt file"** button.

Now 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).

Returning to your browser, press the **"Upload alice.txt file"** button to upload the file to the **gaia.cs.umass.edu** server. Once the file has been uploaded, a short congratulations message will be displayed in your browser window.

Stop Wireshark packet capture. Your Wireshark window should look similar to the window shown below.



A FIRST LOOK AT THE CAPTURED TRACE

Before analyzing the behavior of the TCP connection in detail, let's take a high level view of the trace. First, filter the packets displayed in the Wireshark window by entering "**tcp**" (lowercase, no quotes, and don't forget to press return after entering!) into the display filter specification window towards the top of the Wireshark window.

What you should see is series of TCP and HTTP messages between your computer and **gaia.cs.umass.edu**. You should see the initial three-way handshake containing a SYN message. You should see an HTTP POST message. Depending on the version of Wireshark you are using, you might see a series of "**HTTP Continuation**" messages being sent from your computer to **gaia.cs.umass.edu**. Recall from our discussion in the earlier HTTP Wireshark lab, that is no such thing as an HTTP Continuation message – this is Wireshark's way of indicating that there are multiple TCP segments being used to carry a single HTTP message. In more recent versions of Wireshark, you'll see "**[TCP segment of a reassembled PDU]**" in the Info column of the Wireshark display to indicate that this TCP segment contained data that belonged to an upper layer protocol message (in our case here, HTTP). You should also see TCP ACK segments being returned from **gaia.cs.umass.edu** to your computer.

Answer the following questions, by opening the Wireshark captured packet file **tcp-trace-1**.

To answer the question below, it's probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the "**details of the selected packet header window**".

TASK #1

What is the IP address and TCP port number used by the client computer (source) that is transferring the file to **gaia.cs.umass.edu**?

Answer:

TASK #2

1. What is the IP address of **gaia.cs.umass.edu**?
2. On what port number is it sending and receiving TCP segments for this connection?

Answer:

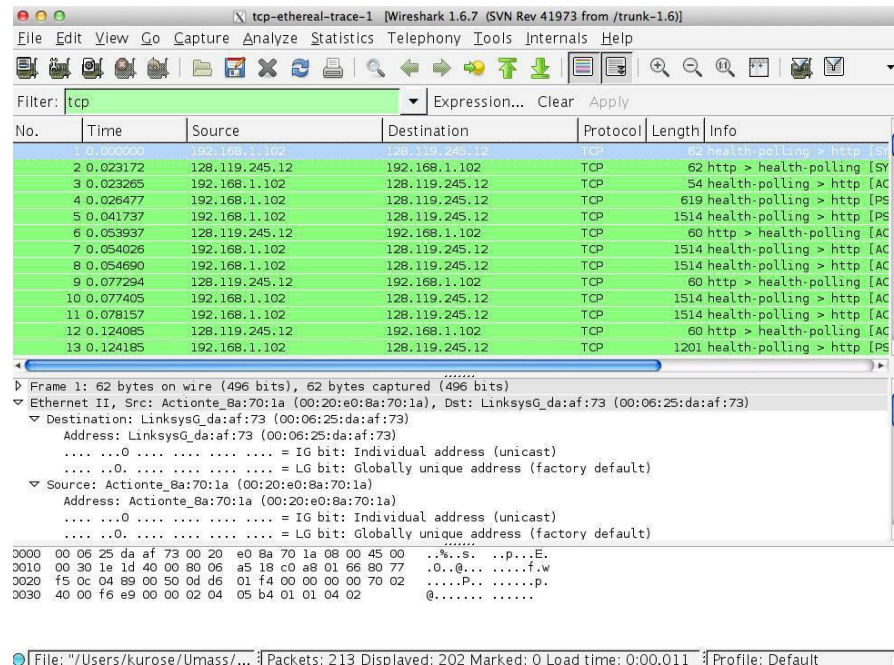
If you have been able to create your own trace, answer the following question:

TASK #3

What is the IP address and TCP port number used by your client computer (source) to transfer the file to **gaia.cs.umass.edu**?

Answer:

Since this lab is about TCP rather than HTTP, let's change Wireshark's "listing of captured packets" window so that it shows information about the TCP segments containing the HTTP messages, rather than about the HTTP messages. To have Wireshark do this, select **Analyze → Enabled Protocols**. Then uncheck the HTTP box and select **OK**. You should now see a Wireshark window that looks like:



This is what we're looking for a series of TCP segments sent between your computer and gaia.cs.umass.edu. We will use the packet trace that you have captured (and/or the packet trace tcp-trace-1) to study TCP behavior in the rest of this lab.

TCP BASICS

Answer the following questions for the TCP segments:

TASK #1

1. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and **gaia.cs.umass.edu**?
2. What is it in the segment that identifies the segment as a SYN segment?

Answer:

TASK #2

1. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN?
2. What is the value of the Acknowledgement field in the SYNACK segment?
3. How did gaia.cs.umass.edu determine that value?
4. What is it in the segment that identifies the segment as a SYNACK segment?

Answer:

TASK #3

What is the sequence number of the TCP segment containing the HTTP POST command?
(Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field).

Answer:

ANALYZING UDP

In this lab, we'll take a quick look at the UDP transport protocol. UDP is a streamlined, no-frills protocol. Because UDP is simple and sweet, we'll be able to cover it pretty quickly in this lab. So if you've another appointment to run off to in 30 minutes, no need to worry, as you should be able to finish this lab with ample time to spare.

Open the *udp-trace.pcap* and set your packet filter so that Wireshark only displays the UDP packets sent and received. Pick one of these UDP packets and expand the UDP fields in the details window.

TASK #1

Select one UDP packet from the trace. From this packet, determine:

1. How many fields there are in the UDP header?
2. Name these fields?

Answer:

TASK #2

By consulting the displayed information in Wireshark's packet content field for this packet, determine the length (in bytes) of each of the UDP header fields.

Answer:

TASK #3

1. The value in the Length field is the length of what?
2. Verify your claim with your captured UDP packet.

Answer:

TASK #4

What is the maximum number of bytes that can be included in a UDP payload? (Hint: the answer to this question can be determined by your answer to Task #3 above).

Answer:

TASK #5

What is the largest possible source port number? (Hint: see the hint in Task #4).

Answer:

TASK #6

What is the protocol number for UDP? Give your answer in both hexadecimal and decimal notation. (*To answer this question, you'll need to look into the Protocol field of the IP datagram containing this UDP segment*).

Answer:

ICMP PROTOCOL

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

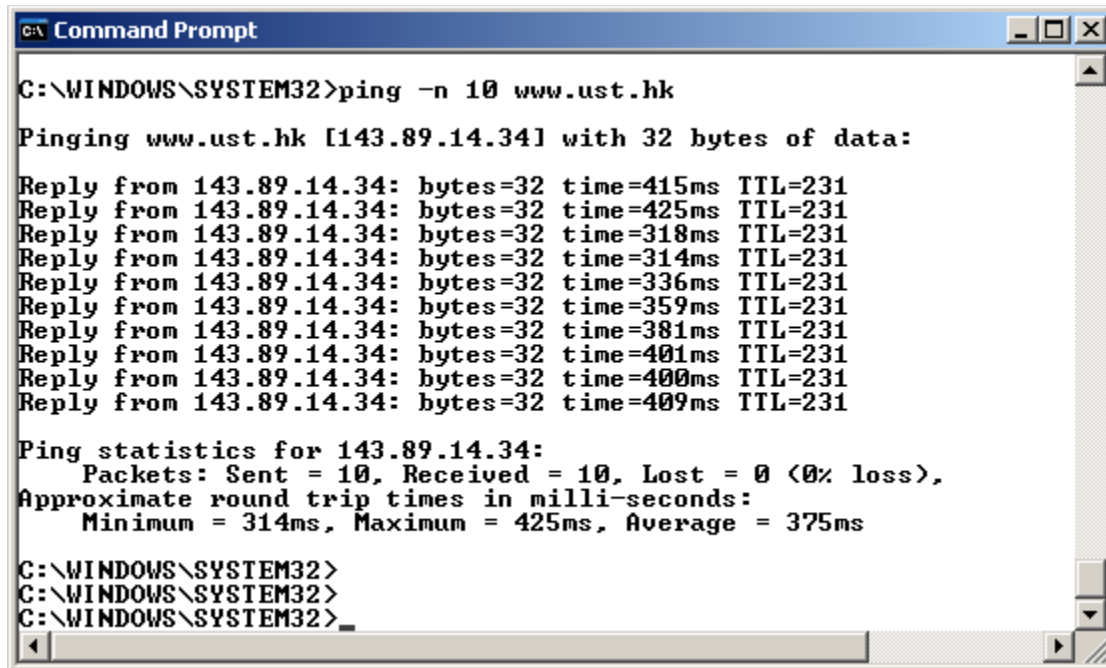
- ICMP messages generated by the Ping program;
- the format and contents of an ICMP message.

Let's begin our ICMP adventure by capturing the packets generated by the Ping program. You may recall that the Ping program is a 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:

- 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.

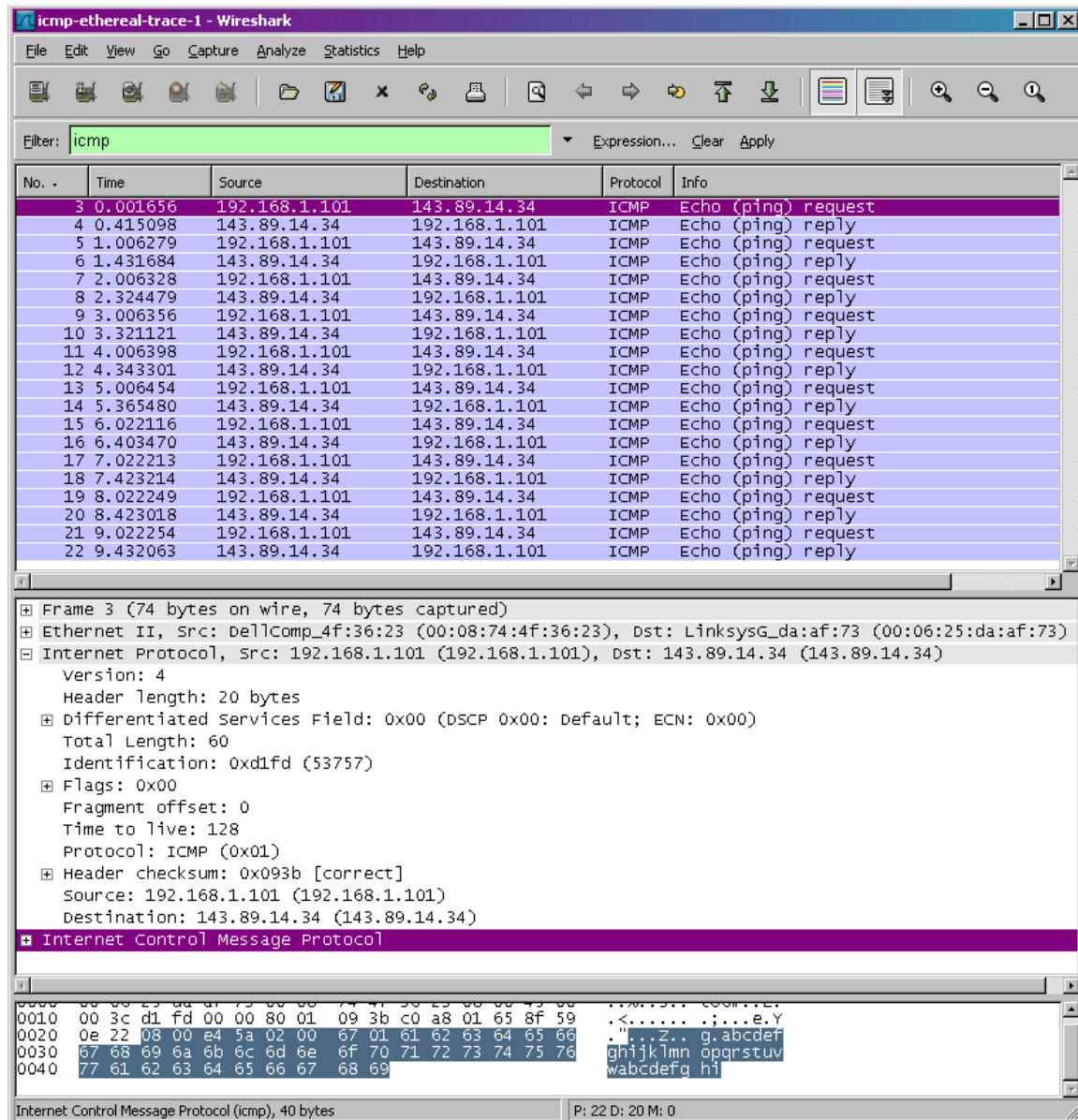


Figure 2 Wireshark output for Ping program with Internet Protocol expanded.

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?