

UNI CS 3470

Networking

Project 5: Using Wireshark to Analyze Packet Traces¹²

Due Wednesday, 12/7, at 11:59:59 PM.

One's understanding of network protocols can often be greatly deepened by "seeing protocols in action" and by "playing around with protocols" – observing the sequence of messages exchanged between two protocol entities and delving down into the details of protocol operation. The project will give you experience observing packet data and frame headers of the following types: TCP, UDP, Telnet, SMTP, DHCP, DNS, and HTTP.

I will be asking you questions about particular traces in the following segments. In some places I will ask you to take screenshots, and in other places I will ask you to answer a question. Please figure out how to take a screenshot on your computer. Also, you may want to open up some kind of word processor to answer your questions. Create a new file called "**project5_answers.txt**" and type your name(s) at the top. Remember, being more specific in your answers is better than being less specific to get full points.

Grading will be as follows:

- Question 21 @6pts each = 6pts
- Screenshots and rest of questions 23@3pts each = 69pts
- Total = 75 pts

Once you are all done with this project, please zip up your screenshots and answer document and submit them to the usual place on eLearning.

You may complete this lab in groups of 1-3. I especially encourage groups of 2.

Part A: Introduction to Wireshark

The basic tool for observing the messages exchanged between executing protocol entities is called a **packet sniffer**. As the name suggests, a packet sniffer captures ("sniffs") messages being sent/received from/by your computer; it will also typically store and/or display the contents of the various protocol fields in these captured messages. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received

¹ This project is adapted from freely-available labs found at <http://www-net.cs.umass.edu/wireshark-labs/>.
© 2005-21012, J.F Kurose and K.W. Ross, All Rights Reserved

² Many traces referenced in this lab are freely-available sample captures located on the Wireshark sample captures wiki at http://wiki.wireshark.org/SampleCaptures#Sample_Captures .

packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer receives a *copy* of packets that are sent/received from/by application and protocols executing on other machines.

We will be using the Wireshark packet sniffer [<http://www.wireshark.org/>] for this project, allowing us to display the contents of messages being sent/received from/by protocols at different levels of the protocol stack. (Technically speaking, Wireshark is a packet analyzer that uses a packet capture library in your computer). Wireshark is a free network protocol analyzer that runs on Windows, Linux/Unix, and Mac computers. It's an ideal packet analyzer for our labs – it is stable, has a large user base and well-documented support that includes a user-guide (http://www.wireshark.org/docs/wsug_html_chunked/), man pages (<http://www.wireshark.org/docs/man-pages/>), and a detailed FAQ (<http://www.wireshark.org/faq.html>), rich functionality that includes the capability to analyze hundreds of protocols, and a well-designed user interface. It operates in computers using Ethernet, serial (PPP and SLIP), 802.11 wireless LANs, and many other link-layer technologies (if the OS on which it's running allows Wireshark to do so).

Getting Wireshark (if you are not using class laptops)

In order to run Wireshark, you will need to have access to a computer that supports both Wireshark and the *libpcap* or *WinPCap* packet capture library. The *libpcap* software will be installed for you, if it is not installed within your operating system, when you install Wireshark. See <http://www.wireshark.org/download.html> for a list of supported operating systems and download sites

Download and install the Wireshark software:

- Go to <http://www.wireshark.org/download.html> and download and install the Wireshark binary for your computer.

The Wireshark FAQ has a number of helpful hints and interesting tidbits of information, particularly if you have trouble installing or running Wireshark.

Running Wireshark

When you run the Wireshark program, you'll get a startup screen, as shown below:

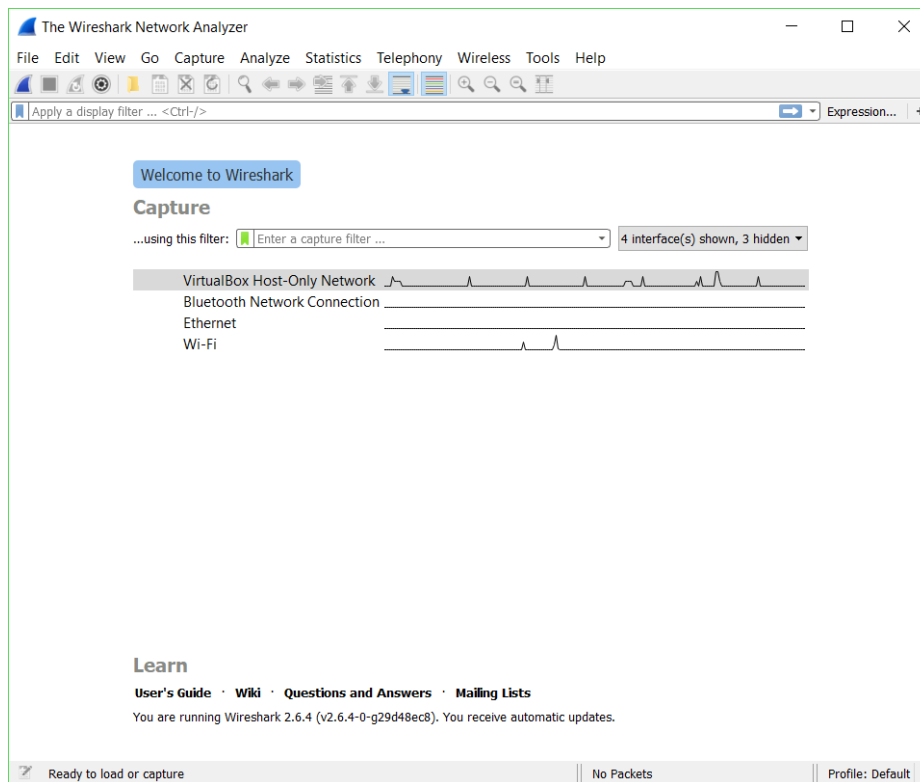


Figure 2: Initial Wireshark Screen

Once you start analyzing traces (either by creating or opening them), you will see a screen similar to the figure below.

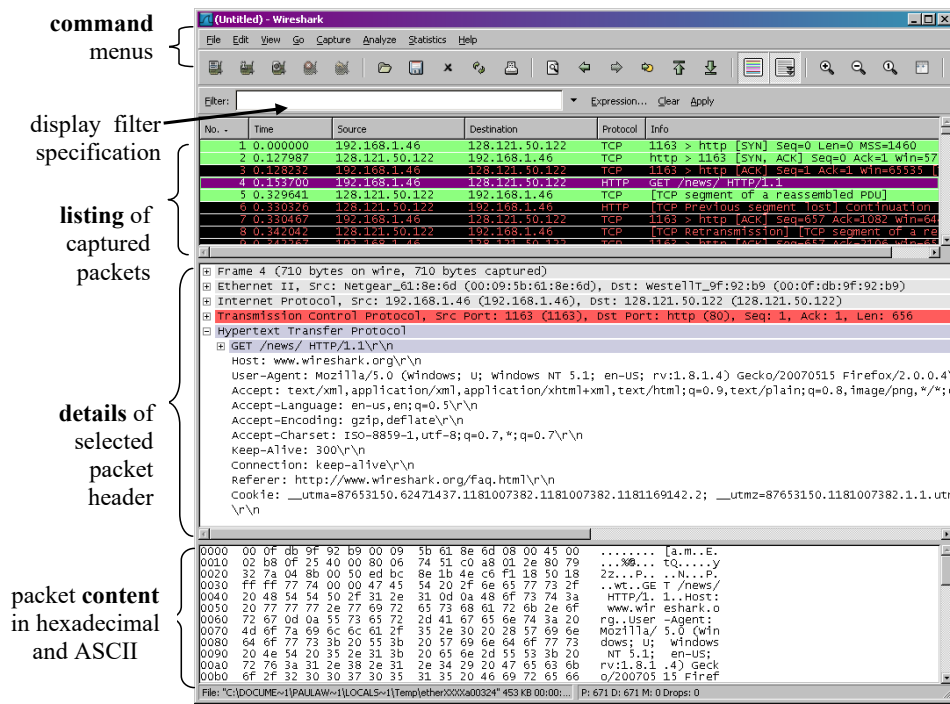


Figure 3: Wireshark Graphical User Interface, during packet capture and analysis

The Wireshark interface has five major components:

- The **command menus** are standard pulldown menus located at the top of the window. Of interest to us now are the File and Capture menus. The File menu allows you to save captured packet data or open a file containing previously captured packet data, and exit the Wireshark application. The Capture menu allows you to begin packet capture.
- The **packet-listing window** displays a one-line summary for each packet captured, including the packet number (assigned by Wireshark; this is *not* a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on a column name. The protocol type field lists the highest-level protocol that sent or received this packet, i.e., the protocol that is the source or ultimate sink for this packet.
- The **packet-header details window** provides details about the packet selected (highlighted) in the packet-listing window. (To select a packet in the packet-listing window, place the cursor over the packet's one-line summary in the packet-listing window and click with the left mouse button.). These details include information about the Ethernet frame (assuming the packet was sent/received over an Ethernet interface) and IP datagram that contains this packet. The amount of Ethernet and IP-layer detail displayed can be expanded or minimized by clicking on the plus minus boxes to the left of the Ethernet frame or IP datagram line in the packet details window. If the packet has been carried over TCP or UDP, TCP or UDP details will also be displayed, which can similarly be expanded or minimized. Finally, details about the highest-level protocol that sent or received this packet are also provided.
- The **packet-contents window** displays the entire contents of the captured frame, in both ASCII and hexadecimal format.
- Towards the top of the Wireshark graphical user interface, is the **packet display filter field**, into which a protocol name or other information can be entered in order to filter the information displayed in the packet-listing window (and hence the packet-header and packet-contents windows). In the example below, we'll use the packet-display filter field to have Wireshark hide (not display) packets except those that correspond to HTTP messages.

Part B: Analyzing Telnet

The best way to learn about any new piece of software is to try it out! Let's try to find some interesting information about a user's telnet session to a server. Telnet is a simple network protocol that allows a user on one computer to log onto another computer that is part of the same network. Do the following:

- Start up the Wireshark software. You will initially see a window similar to that shown in Figure 2.

- Click on the help menu, and then click on Sample Captures. That should launch your browser into the Wireshark packet trace repository.
- Click on the Telnet link
- Click on telnet-cooked.pcap . You may choose to have Wireshark directly open this file (e.g. Windows “open with” dialog), or else you can save this file to your computer and open it through Wireshark. Once the file is open, you should see something similar (although not exactly the same) as Figure 3.

Take a look at the Telnet trace and answer the following questions:

[Take screenshots in the following question(s).]

1. Find the TCP three-way handshake that establishes the connection. Adjust your Wireshark window so that only those three packets are visible, and then take a screenshot. Name this file “[screenshot01.png](#)”. Recall that a server and client use a certain combination of TCP flags to establish a connection. If you can’t quite remember what they are, take a look at your textbook or online presentation notes.
2. Find the TCP packet sequence that terminates the connection. Again, adjust your Wireshark window so that only those packets are visible, and then take a screenshot. Name this file “[screenshot02.png](#)”. Recall that a server and client use a certain combination of TCP flags to tear down a connection. If you can’t quite remember what they are, take a look at your textbook or online presentation notes.

[Copy and answer the following questions in your answer document.]

3. What is the IP address, MAC address, and port number of the client?
4. What is the IP address, MAC address, and port number of the telnet server?

In the Wireshark filter field towards the top left of the screen, please type “telnet” and hit the “Enter” key to filter out the extra TCP packets.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.004160	192.168.0.2	192.168.0.1	TELNET	93	Telnet Data ...
5	0.150335	192.168.0.1	192.168.0.2	TELNET	69	Telnet Data ...
7	0.150574	192.168.0.2	192.168.0.1	TELNET	69	Telnet Data ...
9	0.153657	192.168.0.1	192.168.0.2	TELNET	91	Telnet Data ...

Click on various telnet packets and look in the packet-header details window to find more information about what is being sent. The packet-header details window should look something like the following:

```
> Frame 27: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
> Ethernet II, Src: WesternD_9f:a0:97 (00:00:c0:9f:a0:97), Dst: Lite-OnU_3b:bf:fa (00:a0:cc:3b:bf:fa)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.2
> Transmission Control Protocol, Src Port: 23, Dst Port: 1550, Seq: 101, Ack: 198, Len: 32
> Telnet
```

We see that we can look at different headers and data for many levels of the ISO/OSI network stack. Of particular note is the application layer – Telnet! Expand the “>” beside Telnet on multiple packets to see what is being sent through the telnet protocol.

[Copy and answer the following questions in your answer document.]

5. What is the operating system of the telnet server? (Hint, you find this information transferred through the telnet protocol.)
6. What is the user’s username and password?
7. What commands does the user issue before exiting the server?

Part C: Analyzing DHCP

Let’s take a look at a sample DHCP session. Do the following:

- Start up the Wireshark software. You will initially see a window similar to that shown in Figure 2.
- Click on the help menu, and then click on Sample Captures. That should launch your browser into the Wireshark packet trace repository.
- Find the link called dhcp.pcap and load it into Wireshark. This file is located towards the bottom of the webpage in the “General/Unsorted” category.

Take a look at the DHCP trace and answer the following questions:

[Copy and answer the following questions in your answer document.]

8. Please describe what is happening for each packet in the trace. Please include (a) which computer is sending the packet (client or dhcp server), (b) where the packet is directed, and (c) a description of the purpose of the dhcp packet. For (c), don't just copy what's in the info column (e.g. DHCP discover), but also tell me what that means with the dhcp protocol.
9. What is the transport protocol being used? How can you tell?
10. What is the IP address, MAC address, and the port of the DHCP server?
11. Why does the client initially have an IP address of 0.0.0.0?

[Take screenshots in the following question(s).]

12. Take a screenshot showing the IP address that the client *requests* of the DHCP server. Name the file “**screenshot03.png**”. Hint: The client may not actually get the IP address that it requests. Circle the IP address on the screenshot or else write the IP address in your answer document. You’ll have to do some digging to find this.

Part D: Analyzing SMTP

Let's take a look at a sample SMTP session. Do the following:

- Start up the Wireshark software. You will initially see a window similar to that shown in Figure 2.
- Click on the help menu, and then click on Sample Captures. That should launch your browser into the Wireshark packet trace repository.
- Find the link called smtp.pcap and load it into Wireshark. This file is located towards the bottom of the webpage in the "General/Unsorted" category.

Take a look at the SMTP trace and answer the following questions:

[Copy and answer the following questions in your answer document.]

13. The first two packets in the trace show a DNS lookup. What is the IP address of the DNS server, and what hostname is the query trying to resolve?
14. The client is attempting to send some mail. To which email address is mail being sent? From which email address is mail being sent?
15. The client enters some data into the body of the email message, but it is too large to fit into one packet. The packets are fragmented, and Wireshark shows the reassembled frame later in the trace. In which Wireshark frame number is the packet reassembled? How many data fragments are reassembled?
16. In which ways does the exchange between the client and SMTP server differ from the example commands to use an SMTP server in the presentation on 9/5 (session six) slide numbers 13-14?
17. (This question is tricky.) What kind of file is being *attached* to the email?

Part E: Analyzing TCP and HTTP

In the following trace we will analyze a trace of the TCP segments sent and received in transferring a 150KB file from a computer to a remote server using the HTTP protocol. Do the following:

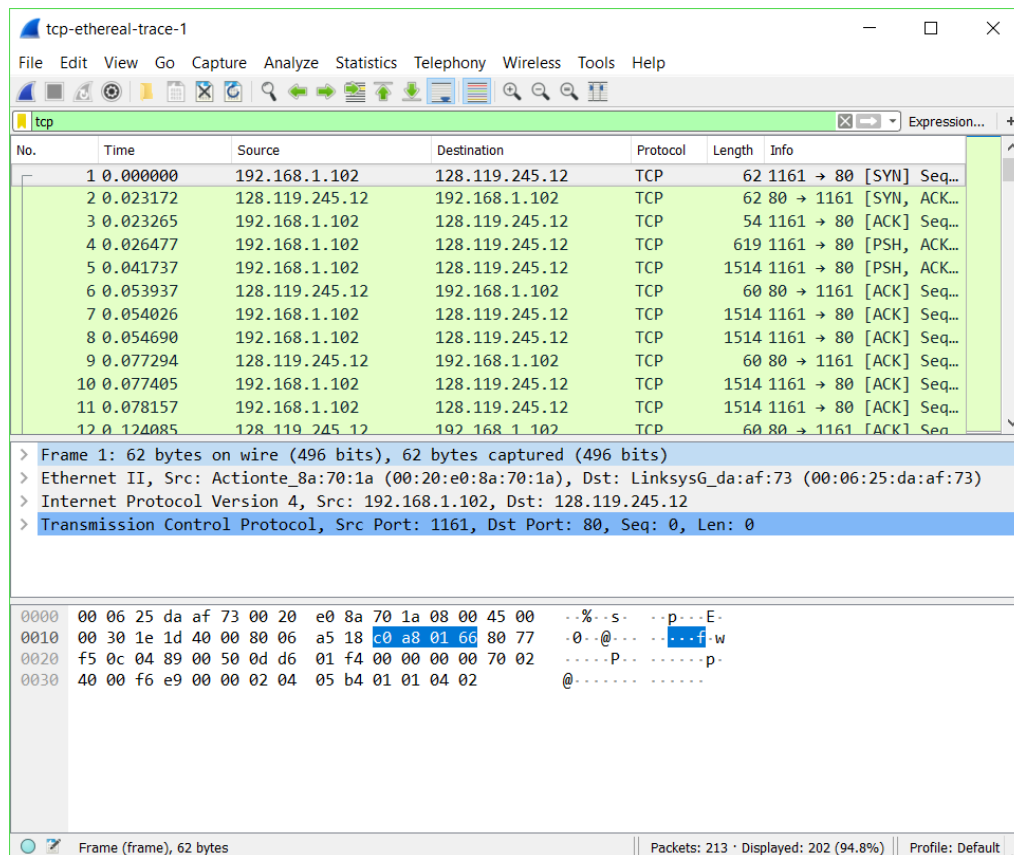
- Start up the Wireshark software. You will initially see a window similar to that shown in Figure 2.
- Download the file called tcp-ethereal-trace-1 from http://www.cs.uni.edu/~diesburg/courses/cs3470_fa18/resources/tcp-ethereal-trace-1.
- Open this file through the Wireshark interface.

Answer the following questions.

[Copy and answer the following questions in your answer document.]

18. 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?
19. What is the IP address of host name of the server? On what port number is it sending and receiving TCP segments for this connection?

Type TCP into the filter window and hit enter. You should now see a Wireshark window that looks like:



This is what we're looking for - a series of TCP segments sent between a source computer and gaia.cs.umass.edu.

Answer the following questions for the TCP segments:

[Copy and answer the following questions in your answer document.]

20. 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. Is this a relative or actual sequence number? What does relative mean in this context?

21. (6pts) Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments? Finally, find the Estimated RTT and Timeout for segments 2-6 using the adaptive retransmission algorithm found on slide 23 of the powerpoint lecture on 11/14 where $\alpha=0.125$. You may want to make a table like this:

seq	time sent	time ack rcvd	RTT	EstRTT	Timeout
value	value	value	value	(same as RTT)	N/A
value	value	value	value	value	value
value	value	value	value	value	value
value	value	value	value	value	value
value	value	value	value	value	value
value	value	value	value	value	value

(Hint: What should we make our timeout value? It needs to be a bit longer than the EstRTT. For this example, let's make the timeout value be twice the Estimated RTT.)

Here is a sample table with the first row filled in for you:

seq	time sent	time ack rcvd	RTT	EstRTT	Timeout
1	0.026477	0.053937	0.02746	0.02746	N/A

22. (6 pts) What is the overall throughput for the TCP connection for the first 100KB of the upload? Throughput is the number of bytes transferred divided by the amount of time it took to transmit. (Hint: Find the time when the next packet after 100,584 bytes (~100KB) is transmitted and subtract the time when the first packet was transmitted to get the total transmit time. Divide the number of bytes by the number of seconds to get the throughput in bytes/second. Now find the number in Mb/sec.)

Let's now examine the amount of data sent per unit time from the client to the server. Rather than (tediously!) calculating this from the raw data in the Wireshark window, we'll use one of Wireshark's TCP graphing utilities - *Time-Sequence-Graph(Stevens)* - to plot out data.

Answer the following questions:

[Take screenshots in the following question(s).]

23. Select a TCP segment (*going from client to server*) in the Wireshark's "listing of captured-packets" window. Then select the menu : *Statistics->TCP Stream Graph-> Time-Sequence-Graph(Stevens)*. Here, each dot represents a TCP segment sent, plotting the sequence number of the segment versus the time at which it was sent. Note that a set of dots stacked above each other represents a series of packets that were sent back-to-back by the sender.

Use the *Throughput* plotting tool to view the sequence number versus time plot of segments being sent from the client to the server. Take a screenshot of this and name it "*screenshot04.png*".

[Copy and answer the following questions in your answer document.]

24. Please identify where TCP's slowstart phase begins and ends and where congestion avoidance takes over. Where congestion avoidance takes over, what is the average sending window size (where the sending window size is the same as the number of packets sent at one time)?