CSCI 5010 – Fundamentals of Data Communications

Lab 6

Wireshark

University of Colorado Boulder

Department of Computer Science

Network Engineering

Professor Levi Perigo, Ph.D.

# Objectives

* Learn the basic operations of Wireshark
* Learn how to capture and analyze ICMP traffic
* Demonstrate best practices for analyzer placement
* Differentiate Wireshark captures between switches and routers
* Display which NICs on analyzer are capturing traffic
* Display IPv4 and/or IPv6 addresses on NICs
* Learn how to perform continuous captures for HTTP requests
* Explain and display different ways to demonstrate top talkers on the network
* Learn how to create coloring rules in Wireshark
* Learn how to create graphs for visual representation
* Learn how to capture and analyze application specific traffic – DHCP, HTTP

# Summary

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, delving down into the details of protocol operation, and causing protocols to perform certain actions and then observing these actions and their consequences. This can be done in simulated scenarios or in a “real” network environment such as the Internet. In the Wireshark lab you’ll be doing in this course, you’ll be running various network applications in different scenarios using your own computer or in a virtual machine environment. You’ll observe the network protocols “in action,” interacting and exchanging messages with protocol entities executing elsewhere in the Internet. Thus, you and your computer will be an integral part of these “live” labs. You’ll observe, and you’ll learn, by doing.

In this Wireshark lab, you’ll get acquainted with Wireshark, and make some simple packet captures and observations.

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 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 your machine. The packet capture library receives a copy of every link-layer frame that is sent from or received by your computer.

The second component of a packet sniffer is the packet analyzer, which displays the contents of all fields within a protocol message. In order to do so, the packet analyzer must “understand” the structure of all messages exchanged by protocols. For example, suppose we are interested in displaying the various fields in messages exchanged by the HTTP protocol. The packet analyzer understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. Finally, it understands the TCP segment structure, so it can extract the HTTP message contained in the TCP segment. Finally, it understands the HTTP protocol and so, for example, knows that the first bytes of an HTTP message will contain the string “GET,” “POST,” or “HEAD.”

[We will be using the Wireshark packet sniffer [http://www.wireshark.org/](http://www.wireshark.org/)] for this lab, 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.

Prefer using VM1 - UBUNTU – 22.04

Part 1

Objective 1.1 - Downloading Wireshark and Navigation Overview

1. Download and install Wireshark:

For Ubuntu 22.04 refer - [How to Install and Configure Wireshark on Ubuntu 22.04 (linuxhint.com)](https://linuxhint.com/install-configure-wireshark-ubuntu-22-04/)

For Windows and MAC:

• Go to <http://www.wireshark.org/download.html>, download and install Wireshark.

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

• Helpful install video: <https://www.youtube.com/watch?v=flDzURAm8wQ>

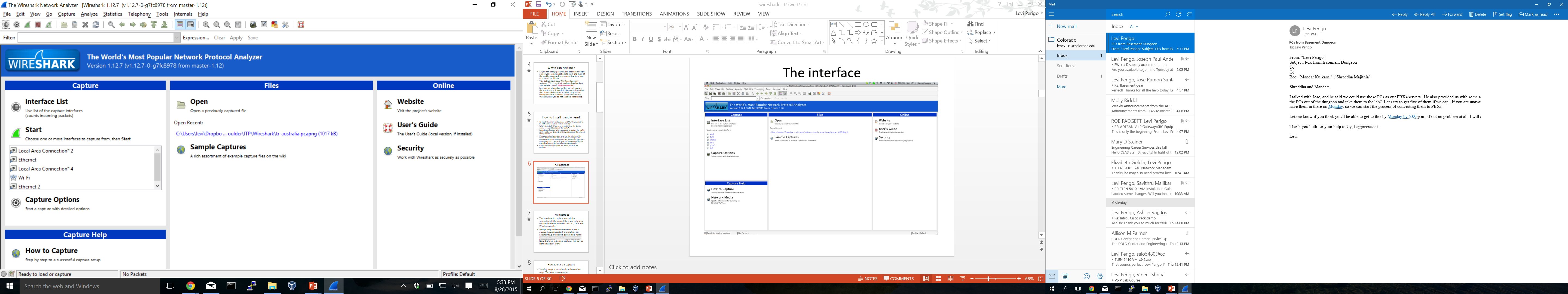
2. Wireshark Navigation

Helpful navigation video: <https://www.youtube.com/watch?v=PYrCS21sPbA>

Note: after installing you might need to reboot once.

# Objective 1.2 - Running Wireshark

When you run the Wireshark program, you’ll get a startup screen, as shown below (note: this screenshot may be an older version of Wireshark):



**Figure 1:** Initial Wireshark Screen

Take a look at the upper left-hand side of the screen – you’ll see an “Interface list”. This is the list of network interfaces on your computer. Once you choose an interface, Wireshark will capture all packets on that interface. In the example above, there is an Ethernet interface (Gigabit network connection) and a wireless interface (“Wi-Fi”).

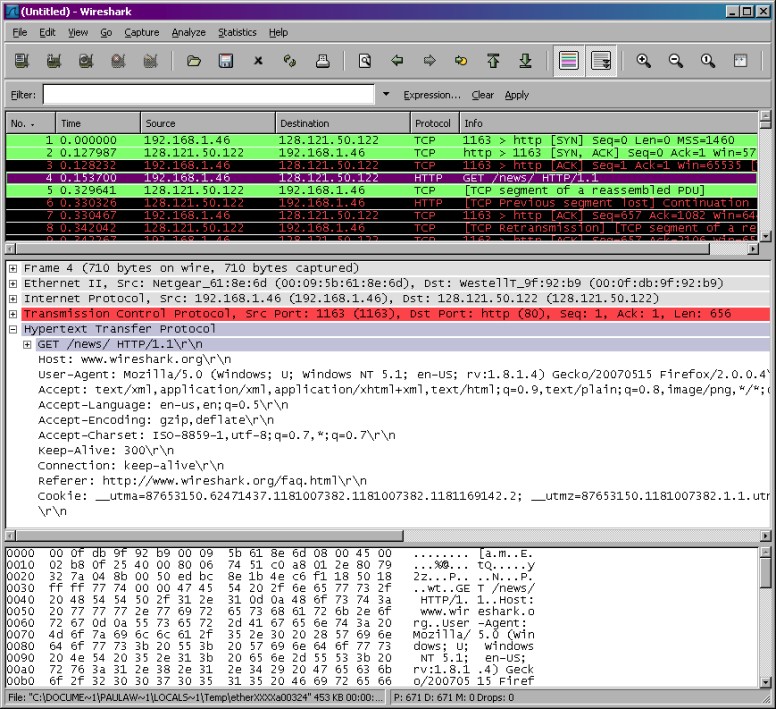
If you click on one of these interfaces to start packet capture (i.e., for Wireshark to begin capturing all packets being sent to/from that interface), a screen like the one below will be displayed, showing information about the packets being captured. Once you start packet capture, you can stop it by using the Capture pull down menu and selecting Stop.

**command**

menus

display filter specification

**listing** of captured packets



**details** of selected packet header

packet **content** in hexadecimal and ASCII

**Figure 2:** 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.

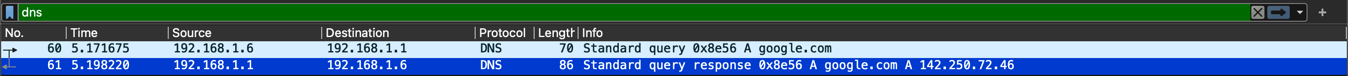
• The **packet-contents window** displays the entire contents of the captured frame, in both ASCII and hexadecimal format.

• 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 2

# Objective 2.1 – ICMP

1. Open command prompt/terminal (depending on the operating system)
2. Start Wireshark and begin capture
3. Ping any “hostname” (where the “hostname” is a URL, example: ping [www.google.com](http://www.google.com))
4. When the Ping finishes, stop the capture. [Press Ctrl + C in MAC to stop the ping]
5. Filter the capture to only display DNS traffic
   1. Provide a screenshot of the DNS reply from the server that shows the IP address of the URL. [**5 points**]



* 1. Explain why DNS would be in this capture when you pinged? [**5 points**]

DNS is in this capture because when I ping google.com, it needs to resolve its IP so I can reach the server. Without an IP, I cant get to google.com’s server.

1. Filter the capture to only display the Ping traffic
   1. Were the Pings successful?
      1. yes
   2. Provide the filtered Wireshark screenshot, and explain how you know they were/were not successful? [**10 points**]

A screenshot of a computer

Description automatically generated

They were successful because there is a request and reply for each succeeding packet. It also displays to the right what packet was the reply in and request.

1. What is the IP address of your host? (show within Wireshark) [**5 points**]

A screenshot of a black box

Description automatically generated

1. What is the IP address of the destination host?
   1. Show within Wireshark, and explain how this address was selected? [**10 points**]

A screenshot of a phone

Description automatically generated

This address was selected according to whatever entry in the DNS server google.com resolves to.

1. 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? [**10 points**]

Type: 8 (Echo (ping) request)

Code: 0

It has checksum, Identifier fields, sequence number, and timestamp

Checksum: 11235 bytes (0x2be3)

Sequence Number (BE): 0 (0x0000)

Identifier (BE): 29139 (0x71d3)

Identifier (LE): 54129 (0xd371)

1. 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? [**10 points**]

Type: 0 (Echo (ping) reply)

Code: 0

The only field that is added to this reply packet is the response time in ms.

Checksum: 13283 bytes (0x33e3)

Sequence Number (BE): 0 (0x0000)

Identifier (BE): 29139 (0x71d3)

Identifier (LE): 54129 (0xd371)

1. Start a new Wireshark Capture. Ping a hostname or IP that gives you a “Request Timed Out” message. (e.g. You can try [www.wellsfargo.com or](http://www.wellsfargo.com/) any another website/IP of your choice.). Filter the ICMP traffic. Find the Type and Code of the packet in the above scenario. Paste the relevant screenshots. [**5 points**]

A black rectangular object with yellow lines

Description automatically generated

1. Do you see both ICMP Echo Request and Echo Reply messages? [**5 points**]

I do not see both. I only see Echo Request messages.

# Objective 2.2 – ICMP and Traceroute

1. Open command prompt/terminal (depending on the operating system)
2. Start Wireshark and begin capture
3. Traceroute to a “hostname” (where the “hostname” is a URL, example: tracert [www.google.com](http://www.google.com))
4. When trace completes, stop capture.
5. Provide a screenshot of the trace. Was it successful? How do you know? [**5 points**]

A screenshot of a computer

Description automatically generated

My traceroute was not successful. I am using a Mac and have tried doing ‘traceroute URL’ for multiple website and I always get Destination unreachable.

1. Filter the Wireshark capture to only show the relevant trace route data. Examine the ICMP traffic in Wireshark. What is different in the capture from the trace when compared to the capture of the Ping in previous objective? Explain what is different between the Ping and the trace route, and how this relates to how trace route works [**15 points**]

The difference is there are no request or reply type pings to the server and host. The only packets received are time to live indication packets from each server that leads to the destination hop. When I issue a traceroute, it sends a ping packet with a TTL of 1. Each hop increases the TTL by one and each router that intercepts this packet decreases it by 1 coming back to me.

# Part 3 – Wireshark NICs and IPv4/IPv6 addresses

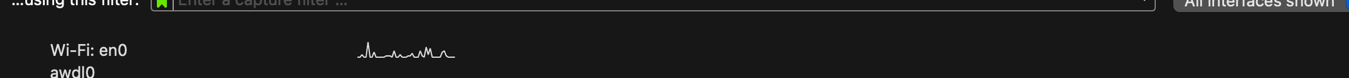
Objective 3.1

1. Provide a screenshot of the NICs that Wireshark has to choose from on the PC. **[10 points]**

**A blue line on a black background

Description automatically generated**

1. Which interface is currently capturing traffic? How do you know? (Provide a screenshot) **[10 points]**



The Wi-Fi: en0 interface is currently capturing traffic as you can see the little graph to the left recording what traffic is currently going to the machine.

1. What is the easiest way to determine the IPv4/IPv6 address of the NICs before a capture is started? Provide a screenshot where some NICs show IPv6 addresses and some show IPv4 addresses. **[10 points]**

A screenshot of a computer

Description automatically generated

The easiest way I found is simply click the cog next to the start capture blue shark fin button and it will bring up Capture Options. There you can see all the interface and associated IP information.

# Part 4 – Continuous Captures, Filtering, and Analysis

# Objective 4.1

1. Initiate a Wireshark capture that uses multiple files, where it creates a new file every 5 minutes, for a total of 15 minutes. (Provide a screenshot of the Capture Options you selected). Remember where you save this file, as we will use it in the future. Also, try to use the wireless NIC if possible, as there will be more traffic in the capture to analyze. **[15 points]**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Browse ten different websites, during this 15-minute continual capture time window.
2. What are THREE reasons why you would want to create multiple files? **[15 points]**
   1. You can divide the data into chunks which can be easier to handle than one big file
   2. If something happens to my computer or WIFI, then the file could corrupt or be lost. So, making multiple files will make sure you got most of the info needed.
   3. If an administrator knew the time of an incident, they could be able to access that capture file at that specific time, making it easier than having one large file.
3. How do you view the three files captured within Wireshark, and move between them, after they have been completed and saved? (Hint: File > File Set) **[10 points]**

You can go to File > File Set > List Files. There you can click between each capture file.

1. How do you see which websites you browsed during the capture? (Hint: Statistics > HTTP) **[5 points]**

You would go into Statistics > HTTP > Load Distribution and

# Objective 4.2

1. Create two Display filter buttons. One for traffic sourced from your machine’s IP address and one that only displays HTTP GET requests. (Hint: HTTP contains)
2. Provide a screenshot of the buttons you created, and the corresponding filtered capture. **[10 points]**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

(Buttons are showed to the right next to the search bar)

# Objective 4.3

1. Create a coloring rule for HTTP traffic. **[5 points]**

A screenshot of a computer

Description automatically generated

1. Provide a screenshot of your capture from above, showing where you changed the color of HTTP GET requests to Red background with White lettering. (Hint: Did you remember to move your color rule to the top?) **[10 points]**

A screenshot of a computer

Description automatically generated

# Part 5 - Top Talkers, Profiles, and Graphs

# Objective 5.1

1. Determine Top Talkers on the network. **[10 points]**
   1. Go to statistics > conversation and sort by bytes. You can then see a list of the top talkers with address A and address B talking with each other and how many bytes they transmitted.
2. What are two ways you can determine what device/IP address is transmitting the most traffic on the network? Provide a screenshot of one of those ways. (Hint: Protocol Hierarchy; Conversations; Endpoints) **[10 points]**
   1. One way is going to Endpoints in the IPV4 tab and sorting by bytes. You can then see how many total bytes was transmitted by one IP
   2. You can also see similar information in the Conversations tab with IPV4 devices.

A screenshot of a computer

Description automatically generated

# Objective 5.2

1. Create a graph that displays the top 4 protocols from the capture.
2. Provide screenshot of the graph. **[10 points]**

A screenshot of a graph

Description automatically generated

# Objective 5.3

1. Create three profiles that you will use for future analysis objectives. For example, Security, Troubleshooting, VoIP, etc. **[10 points]**

I created a Security profile, a TCP profile, and a DNS profile

1. Explain what you would use each profile for, what you changed, and provide a screenshot of one of them. **[10 points]**

I had some inspiration from this GitHub containing some really useful Wireshark profiles: https://github.com/amwalding/wireshark\_profiles.git

Security:

I changed the coloring rules for the Security profile. The goal here was to see where protocols fail or is not working as intended. I did some coloring for TCP error flags, changes in STP, and ICMP ping packet errors.

A screenshot of a computer

Description automatically generated

TCP Profile:

In the TCP profile, I also changed the color profiles to reflect certain flags. I made a certain color for SYN, SYNACK and ACK flags. It also checks if multiple ACK flags were sent and checks for bad TCP as well.

DNS Profile:

This is like the TCP profile as it monitors TCP traffic similarily, but now also looks for DNS flags and DNS server traffic. These are all displayed in the coloring rules.

# Part 6 - DHCP Release and Renew

# Objective 6.1

1. Start Wireshark and begin capture
2. Release the DHCP IP address your machine has obtained
3. Renew the DHCP IP address (for your machine to obtain a new address)
4. After your machine receives an IP address from the DHCP server, stop the capture
5. Filter the capture to only show the DHCP traffic. From the capture indicate the following:
   * 1. DHCP server address [**2 points**]
        1. 192.168.1.1
     2. The IP address your machine was offered and accepted [**2 points**]
        1. 192.168.1.6
     3. Explain the DHCP process, include a screenshot [**10 points**]
        1. DHCP process can be explained by the acronym: DORA. Discover, Offer, Request, Acknowledge. First the PC sends a Discover broadcast, looking for some server to offer an IP. My home router/DHCP server sees this broadcast and sends an offer back with an IP for my PC. Then my PC will look at this offer and send a request back to the server saying that I want to use this IP. And then the DHCP server will retrieve this and send an acknowledgment back to the host to confirm the IP.

A screen shot of a table

Description automatically generated

Part 7 – Web traffic (HTTP) and TCP Connection

# Objective 7.1

1. Start Wireshark and select the appropriate interface to begin capturing packets.

2. Go to <http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html> in your browser and capture the web session on Wireshark.

3. How is the TCP connection established? Explain the process. What is it called? Locate it in your Wireshark capture. Paste relevant screenshots. **[5 points]**

A screen shot of a computer

Description automatically generated

The process establishes a TCP connection. Here’s how it works:

My PC sends a SYN packet to open the other host or the website.

When the website host receives it, it sends a SYN acknowledgement, also known as a SYN ACK

Then my PC sends an ACK back to acknowledge the SYN ACK.

4. Inspect information within the first packet of the TCP connection process.

i. What is the destination port number? How would you classify it? **[1 point]**

51024, I would classify it as the random port number that was opened by the website to establish the TCP connection.

ii. Which control flag (or flags) is set? What does it imply? **[1 point]**

SYN flag is set. This is a flag to try to reach the website host and establish a potential connection.

iii. What is the relative sequence number set to? **[1 point]**

It is set to Seq=0

5. Inspect the next packet in the TCP connection process.

i. Which control flag (or flags) is set? What do they imply? **[1 point]**

SYN,ACK flag is set. This implies that the website host got the original SYN packet and is sending back a request to establish the connection

ii. What is the relative sequence number and relative acknowledgement number set to? **[1 point]**

Seq=0 Ack=1

6. Finally, inspect the third packet of the connection process.

1. Which control flag (or flags) is set? What do they imply? **[1 point]**

ACK flag is set. This implies that my PC got the SYN ACK and wants to finally establish that TCP connection.

1. What is the relative sequence number and relative acknowledgement number set to? What do they imply? **[1 point]**

Seq=1 Ack=1. They imply that my PC got the SYN packet and increases the Seq number.

1. In your Wireshark Capture display only the HTTP (Web) traffic. (Paste screenshot). [**5 points**]



1. Examine the HTTP request packet. What is the destination IP address and destination port number? Which TCP control flag (or flags) is set, and what do they mean? Paste relevant screenshots. **[5 points]**

IP: 128.119.245.12

Destination Port: 51024

Flags: PSH, ACK

PSH means to just send the data to the host and the ACK flag means that my PC acknowledges that the server is there for the connection.

A computer screen shot of a black and white screen

Description automatically generated

9. Examine the HTTP packets and answer the following questions -

i. What HTTP version is running on the client? What version of HTTP is the server running? **[2 points]**

HTTP 1.1 on both sides

ii. What is the status code returned from the server to your browser? **[1 point]**

304

iii. When was the HTML file that you are retrieving last modified at the server? **[1 point]**

Oct 30th @ 5:59

iv. How many bytes of content are being returned to your browser? **[1 point]**

239 Bytes

10. Can you see the text displayed on the browser in your Wireshark packets as well? Why/why not? Paste relevant screenshots. [**5 points**]

I cannot see the text displayed on the browser in my Wireshark packet. I can see the URL and how the text was sent to my PC, but I cannot see the text itself.

Part 8 – Parsing .pcap using Python [Extra Credit]

# Objective 8.1

1. Start a new capture in Wireshark using the capture filter of ‘icmp’. Open the command prompt/terminal and execute these commands -

ping -4 google.com

ping wellsfargo.com

[Use -c 4 option if pinging from MAC.]

1. Stop the capture and save the file as .pcap.
2. Write a script using Python that parses the saved .pcap file and prints out only the source and destination IPs of each packet of the file sequentially. You can use the Python library pcapfile for this purpose [<https://pythonhosted.org/pypcapfile/installing.html>]. [**20 points**]

# Total Score = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/291 [+20 Extra Credit]