EECS 563:

Project 1

Lynne Lammers

2124909

**Abstract**

In this project, Wireshark was used to examine various IP protocols on a home computer, including ICMP, ARP, and TCP. In Part I, ICMP packets were sent and received by sending ping requests. The structure and size of these packets were examined. In Part II, the address resolution protocol was investigated by clearing the ARP table and looking at ARP requests and replies in Wireshark to understand how an ARP table is built up. In Part III, a speed test was carried out and the resulting TCP packets were examined. Finally, in Part IV, the results were summarized and analyzed.

**Table of Contents**

Introduction …………………………………………………………………………… 4

Part I: ICMP Packets ……………………………………………………………….... 5

Part II: ARP Packets …………………………………………………………………. 7

Part III: TCP Packets ……………………………………………………………….... 9

Part IV: Summary & Analysis ……………………………………………………….. 10

References ……………………………………………………………………………. 11

**Introduction**

In this project, Wireshark, a network protocol analyzer, was installed and used to analyze various types of packets sent and received by a home computer. Wireshark allows the user to record packets being transmitted and filter the type of packets being recorded. It also allows the user to look at each packet in detail, examining headers and their contents as well as the data in the packet. This means that the user can see encapsulation up close and see exactly what information is being transmitted at each layer, allowing the user to get a better understanding of various protocols.

Specifically, in this project, several types of protocols were investigated. The first protocol was ICMP, or internet control message protocol. This protocol is used when a message needs to be sent back to the source of the packet. Both traceroute and ping use ICMP packets. Ping was used in the project and the resulting ICMP packets were examined to understand the encapsulation at each layer.

Another protocol examined was ARP. ARP, or address resolution protocol, is a protocol used to map between IP addresses and physical, or MAC, addresses. When a new message comes in with an IP address unknown by the ARP table, a packet will be sent out across the network to request information about the who owns that address. The owner will respond, and that information will be added to the table for future use. In this way, ARP tables make getting data to the correct device on a network faster and more efficient.

Finally, TCP or transport control protocol, was examined. This protocol essentially sets up a connection at the transport layer between two entities. When data is sent, an acknowledgement is returned. This feedback means that if a packet wasn’t successfully received, the recipient can notify the sender so the packet can be resent. In this project, we examined TCP packets that were sent and received as part of a speed test, looking at the types of flags set in each packet.

**Part I: Tracking ICMP Packets**

For this part of the project Wireshark was used to set up a capture filter allowing only ICMP packets to be captured. Next, after starting a capture, a single ping was sent to www.google.com, followed by a second ping with length set to 256 bytes. The capture was stopped, and the packets were examined. Several follow up questions were answered by examining the information contained in the packets. The questions, and their answers are listed below:

1) What is the IP address of destination: www.google.com?

The IP address of www.google.com is 74.125.227.113.

2) What is the “Time to live” set on each Echo (ping) requests? Why do these packets have these “Time to live” values?

The time to live from my computer to www.google.com was set to 64, and from www.google.com to my computer was 53. The two packets have the same time to live values because the size of the packet is unrelated to potential routing problems, and time to live is set to help prevent routing loops.

3) What is the length of the ICMP message?

The length varied between the two pings. For the first ping, the ICMP message had a total length of 98 bytes, which includes 14 bytes for the ethernet header, 20 bytes for the IPv4 header, 16 bytes for the ICMP header, and 48 bytes of data. For the second ping, the ICMP message had a total length of 298 bytes, which includes 14 bytes for the ethernet header, 20 bytes for IPv4 header, 16 bytes for the ICMP header, and 248 bytes of data.

4) What is the length of the IP packet in parts g) and i)?

In part g, the IP packet had a length of 84 bytes, with 20 bytes of IP header. In part i, the IP packet had a length of 284 bytes with 20 bytes of IP header.

5) What are the sent data bytes?

The sent data bytes for part g had a total of 48 bytes. Each byte counts up by one starting from eight (0x08) for the first byte to 55 (0x37). The sent data bytes for part i had a total of 248 bytes. The bytes count up by one starting from eight (0x08) for the first byte to 255 (0xff).

6) What is the MAC address (physical address) of your NIC? Find the MAC address by examining the Echo (ping) request in the Packet Bytes Plane (the bottom window).

The MAC address for my NIC is e0:f8:47:33:30:a6.

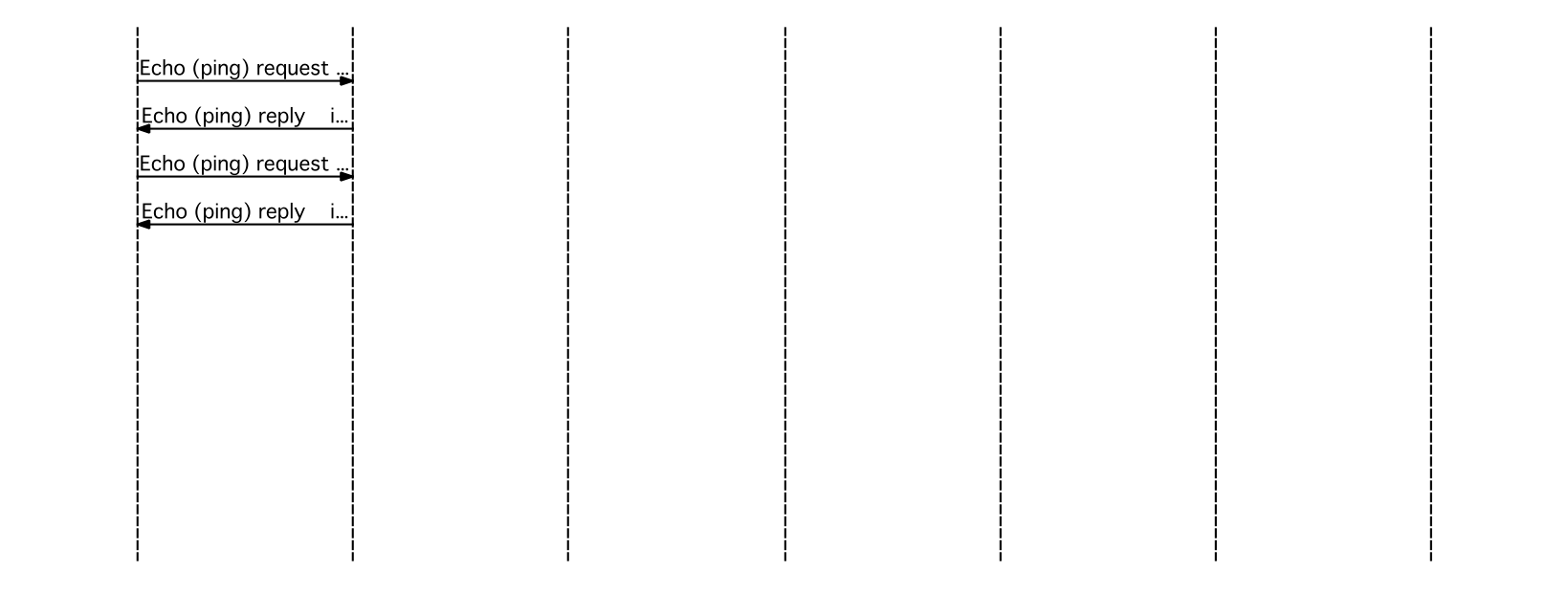
7) What is the source IP address (in Hex); look in the Packet Bytes Plane (the bottom window). Hint: use http://www.easycalculation.com/hex-converter.php.

The source IP address (for the initial sending of the packet) in hexadecimal is 0a000109.

8) In Wireshark, generate a flow graph under StatisticsFlow graph, select “All Packets”, “General flow”, and “Standard source destination addresses”. Discuss the resulting flow graph.

The flow chart shown below in figure 1 displays communication between two nodes, the source (my computer) and the destination (www.google.com). It displays an echo (or ping) request being sent, and the echo (or ping) reply returning. This happens twice, as two separate pings were sent.

Figure 1: Flow graph between home computer and www.google.com for pings



[www.google.com](http://www.google.com)

Home computer

**Part II: Tracking ARP Packets**

In this part of the project, ARP packets were examined. First, the command arp -a was used in the terminal to view all entries in the current ARP table. After viewing the ARP table, Wireshark was set up to filter for only ARP packets, and a capture was begun. Next, the command arp -d -a was given in the terminal to delete all entries in the ARP table. After all entries were deleted, the ARP table’s contents were again examined using the command arp -a. The entries in the ARP table and the packets captured using Wireshark were examined through answering the following questions:

1) How big was your initial ARP table?

My initial ARP table had eight entries, shown below:

? (10.0.1.1) at 24:a2:e1:e7:e4:8d on en1 ifscope [ethernet]

? (10.0.1.2) at 9c:20:7b:84:2c:7b on en1 ifscope [ethernet]

? (10.0.1.4) at 34:c0:59:6e:a:a3 on en1 ifscope [ethernet]

? (10.0.1.5) at d4:4b:5e:79:d6:32 on en1 ifscope [ethernet]

? (10.0.1.8) at a8:86:dd:63:72:b3 on en1 ifscope [ethernet]

? (10.0.1.11) at 70:9e:29:4e:46:91 on en1 ifscope [ethernet]

? (10.0.1.15) at 34:af:2c:33:d3:b2 on en1 ifscope [ethernet]

? (10.0.1.255) at ff:ff:ff:ff:ff:ff on en1 ifscope [ethernet]

2) Look in the Ethernet packet and determine the Broadcast MAC address?

The broadcast MAC address is ff:ff:ff:ff:ff:ff

3) Why is a Broadcast destination MAC address used in ARP?

The broadcast MAC address is used because the sending computer does not know the MAC address for the requested IP address. To find it out, it must broadcast the request to all computers on the network, and thus it sends it to the broadcast MAC address.

4) Did the results of the arp –a command change as a result of executing the arp –d \* command? Explain your observation.

Yes. After running arp -d, only one entry remained:

? (10.0.1.1) at 24:a2:e1:e7:e4:8d on en1 ifscope [ethernet]

This IP address was requested very quickly, and thus was already present by the time the arp -a command was run. The others were deleted from the table and had not yet been restored.

5) Examine an ARP Request packet, why is the target MAC address all 0’s?

The target MAC address is all 0’s because it doesn’t yet know the correct MAC address. The goal of the request is to determine that information.

6) Examine an ARP Reply packet by changing the Filter to eth.dst==physical address, e.g., eth.dst==00:0E:0C:DA:85:8D. What new information is learned from the data contained in the ARP Reply packet?

The reply contains the MAC address of the sender, along with its IP address. Using these two pieces of info, the receiver can now fill in a line in its ARP table with the IP address and its associated MAC address.

7) Examine the arp exchange messages. Change the filter to eth.addr== physical address, e.g., eth.addr==00:0E:0C:DA:85:8D, and comment on the Flow Graph for these packets.

The flow graph shown in figure 2 displays ARP packets sent between my computer and the computer at IP address 10.0.1.1. First, my computer (represented by the first vertical line) sent out a broadcast message, represented by the second vertical line, requesting the owner of IP address 10.0.1.1 to send a reply back. Next, the computer with IP address 10.0.1.1 (represented by the third vertical line) sent back a reply to me containing its MAC address.

Figure 2: Flow graph for ARP packets



Home computer

Broadcast

Computer with IP address 10.0.1.1

**Part III: Tracking TCP Packets**

In the third part of this project, TCP packets were examined. To do this, Wireshark was set to filter for tcp packets, and a capture was started. A speed test at speed.sunflower.com was begun. After it completed, the capture was stopped and the packets were examined to answer the questions listed below:

1) Find the addresses of the hosts involved in the speed test. Under the StatisticsConversation Lists select TCP. From the list of TCP conversions find the addresses of the hosts involved in the speed test. Call the address of your local host Address A, and the address of your remote host Address B. Use a filter to focus on the packets involved in the speed test. In the filter blank at the top level of Wireshark type in: ip.addr= = Address A or ip.addr = = Address B then click “apply”. Describe the Flow Graph for these packets. Under StatisticsFlow Graph click

“Displayed Packets”, “TCP Flow”, and Standard source/destination addresses.

The flow graph for these packets begins with Address A sending a SYN, Address B sending a SYN ACK, and Address A sending an ACK. Next, AddressA sent a PSH ACK, and Address B responded with a PSH, ACK. This represented the ping portion of the test and was followed by FIN signals. After another set of SYN signals, the flow continued with Address B sending data along with ACKs and Address A responding intermittently with ACKs. This continued for the majority of the communication, representing the download portion of the test. At packet 57422, Address B sent a FIN PSH ACK signal and after several ACKS, Address A returned the FIN. Address A then sent several SYN until Address B responded with a SYN ACK, followed by an ACK from Address A. Following this, the flow essentially reversed, with Address A sending ACKs with data, and Address B responding intermittently with ACKs. This portion represented the upload speed test. Finally, Address B sent a FIN, ACK, Address A responded with a FIN, ACK, and Address B ended with an ACK.

2) What is the size of the TCP packets used in this application? Why this value was selected?

While a few were smaller, the vast majority of TCP packets used in the download and upload test had a length of 1448 bytes. This was likely used because the maximum length for transmission across ethernet is 1500 Bytes, so choosing a packet size that will be less than that once the IP additional headers are added helps reduce packet fragmentation and reassembly and thus reduces packet processing time.

3) What are the Seq and Ack numbers for the three-way hand shake that set up the TCP connection (Look at the packets at the start of the connection)?

The first three signals to carry out the handshake consisted first of a SYN signal from my computer to the destination, then a SYN ACK signal from the destination back to my computer, and finally an ACK from my computer to the destination. The first packet had sequence number 0 and Ack number 0. The second packet had Seq number 0 and Ack number 1. The third packet had Seq number 1 and Ack number 1.

4) Find the close connection packets for this application (go to the end of the trace), discuss what you observe.

At the end of the connection, Address B sent a FIN, ACK signal, representing a request to terminate the connection. Address A replied with a FIN, ACK signal, and finally Address B sent an ACK. This allowed the TCP connection to close at the end of the test.

**Part IV: Analysis and Summary**

In this project Wireshark was used to investigate network protocols. Wireshark is an excellent tool for this, since it presents a detailed view of the packets. Specifically, it allows the user to look at each of the header portions individually. For example, the user can look at the ethernet header, which is the deepest level of encapsulation occurring on the link layer. Here, the user can see the source and destination MAC addresses. Encapsulated within that layer is the next higher layer of encapsulation, represented by the IP header. Here, the user can look at the source and destination IP addresses, protocol, header checksum, and time to live among other things. The next higher layer is the transport layer, which may be a TCP header, an ICMP header, an ARP header, or the header for a different transport protocol. Each of these has unique information. Finally, the user can look at the actual data being sent. When combined, the user can see exactly how each packet is wrapped up, layer by layer, and has a better view of how each layer going down adds a header and each layer moving up removes one to get to the information relevant to its protocol.

Wireshark is also valuable for examining how a protocol is used. For example, in part II, the methodologies and uses of the ARP protocol were seen in detail. After clearing the ARP table, ARP packets broadcasting in search of the owner of an IP address could be seen in the packet capture. In part III, examining the flow of the TCP packets revealed how flags such as SYN and FIN are used in that protocol to start and end connections. In the above examples and in many other ways, Wireshark can provide great examples of how protocols are implemented and used.

Wireshark could be a valuable tool for more than just learning about protocols. For example, it could help a network administrator determine the cause of various network issues. If a user is having trouble maintaining a TCP connection, for example, Wireshark could reveal exactly what problems are occurring by examining how packets are sent, and which are successfully received. If a developer were creating a new protocol or updating a current one, Wireshark may also help them in debugging that process. They could look at what values are being set in the header, where and when each packet is sent, and what packets are being received. Wireshark would be an invaluable tool for any person trying to understand, improve, or repair networks.

**References**

Peterson, Larry L. and Davie, Bruce S. (2012). Computer Networks: A Systems Approach,

Fifth Edition. Burlington, MA: Elsevier, Inc.

Wireshark Frequently Asked Questions. Retrieved from http://www.wireshark.org/faq.html

Frost, V. Technical Report Format. Retrieved from

<http://www.ittc.ku.edu/~frost/EECS_563/Technical%20Report%20Format%20for%20EECS%20563_2014.pdf>