**SUBJECT CODE: BTITL505**

LAB MANUAL

**Computer Networks and Internetworking Protocols Lab**

**Third Year B.Tech., Semester – I, Academic Year 2022-2023**

|  |  |  |
| --- | --- | --- |
| **Teaching Scheme:-** | **Pract /week:2 Hrs** | |
| **Examination Scheme:-** | **CA: 60 Marks** | **ESE:40 Marks** |
| **Name of Faculty:** | **Dr.S.R.Jadhao** | |

**Institute Vision and Mission**

**Departmental Vision and Mission**

**Vision:**

*To provide quality IT education & nurture socially aware students in accordance with global IT trends.*

**Mission:**

***M1.*** *To build conducive environment for learning emerging IT skills.*

***M2****. To investigate complex problem in software design & development.*

***M3.*** *To imbibe the professional ethics & social values in the students to make them responsible citizens.*

**Program Educational Objectives (PEOs)**

***PEO1.*** *To instill the graduates with fundamentals in the field of information technology.*

***PEO2.*** *To assimilate the graduates with computational needs of society & emerging disciplines.*

***PEO3.*** *To follow the standard practices in entrepreneurship, research & development by fostering usage of modern IT tools.*

**Programme Specific Outcome (PSO's):**

***PS01:*** *An ability to design & develop algorithms, models & processes using logical problem solving strategies.*

***PSO2:*** *To apply knowledge of computing and mathematics in the field of data analytics.*

***PSO3:*** *To analyze & overcome network security issues & vulnerabilities to secure data computational needs of society & organizations.*

**Programme Outcomes (PO's):**

The programme is targeted at developing the following competencies, skills and abilities amongst students. They shall be able to:

● **PO1:** Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

●**PO2:** Problem Analysis - Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

●**PO3:** Design/ Development of solutions - Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

●**PO4:** Conduct Investigations of Complex problems - Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**●PO5:** Modern Tool Usage - Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

●**PO6:** The Engineer and Society - Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional, engineering practice.

●**PO7:** Environment and Sustainability - Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

●**PO8:** Ethics - Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

●**PO9:** Individual and Team Work - Function effectively as an individual, and as a member or leader in diverse teams, and in multi-disciplinary settings.

●**PO10:** Communication - Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

●**PO11:** Project Management and Finance - Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi-disciplinary environments.

●**PO12:** Life-Long Learning - Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**SUBJECT: Computer Networks and Internetworking Protocols Lab Course Code: BTITL505**

**SEM: V Academic term: Sep 2022-Dec 2022**

**Course Outcomes:**

After learning the course, the students should be able:

|  |  |  |
| --- | --- | --- |
| **CO** | **Statements** | **Cognitive level**  **of learning** |
| **C505.1** | To understand the basics about IP addressing schemes. | (Understand) |
| **C505.2** | To study and installation of Packet analyser Wireshark software. | (Apply) |
| **C505.3** | Demonstrate the concepts of data communication at physical layer and compare ISO - OSI model with TCP/IP model. | (Apply) |
| **C505.4** | To Analyse various routing algorithms and protocols at network layer. | (Analyze) |
| **C505.5** | To analyse and Explore networking protocols. | (Analyz) |

**SUBJECT: Computer Networks and Internetworking Protocols Lab Course Code: BTITL505**

**SEM: V Academic term: Sep 2022-Nov 2022**

**Experiment List**

|  |  |  |
| --- | --- | --- |
| **Sr.**  **No.** | **Title** | **CO Mapping** |
| 1 | Conversion of IP addresses (e.g. I/P: 10.24.164.254 O/P: 00001010.00011000.10000000.11111110 and I/P:binary dotted O/P: decimal dotted) | CO505.1 |
| 2 | Introduction to Wireshark | CO505.2 |
| 3 | Wireshark Lab: Ethernet and ARP | CO505.1, CO505.3  ,CO505.4 |
| 4 | Wireshark Lab: IP | CO505.1, CO505.3,  CO505.4 |
| 5 | Wireshark Lab: ICMP, study of ping and traceroute command | CO505.1, CO505.3,  CO505.4 |
| 6 | Wireshark Lab: UDP | CO505.1, CO505.3,  CO505.4 |
| 7 | Wireshark Lab: TCP | CO505.1, CO505.3,  CO505.4 |
| 8 | Study of ftp, telnet tools and network configuration files | CO505.1, CO505.3,  CO505.5 |
| 9 | DHCP server configuration | CO505.1,CO505.4 |
| 10 | Socket programming for UDP and TCP. | CO505.4,CO505.5 |
| 11 | Study and Implement basic networking commands in Linux (ipconfig,ipconfig/all,arp/a,netstat,netstat/an,ping,pathping, nslookup) | CO505.1 |
| 12 | Case Study: Study the detailed Networking Structure in SVKM-IOT Campus. | CO505.1, CO505.3,  CO505.4 |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (Subject Incharge)  Prof.(Dr.)S.R.Jadhao |  | (Head of I.T. Dept.)  Dr. Bhushan Chaudhari |
|  |  |  |

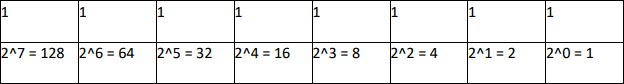
**Experiment 1**

Aim: -Conversion of IP addresses (e.g. I/P: 10.24.164.254 O/P: 00001010.00011000.10000000.11111110 and I/P:binary dotted O/P: decimal dotted)

Theory: - https://itexam24.com/7-1-2-9-lab-converting-ipv4-addresses-to-binary-answers/

IP enables hosts to communicate with each other at the Network layer. In IPv4 each packet contains a source and destination address. That’s how the routers on the network knows where the packet is coming from and where they must forward the packet. The IP addresses fields are represented in 32 bits. Routers know how to interpret those numbers, but for a human to understand them would be too difficult. From our point of view, we use what’s called a dotted decimal address. A dotted decimal address is the human representation of the binary address. For example, the address 192.168.10.1 is a dotted decimal address. In its binary form, the address is 11000000101010000000101000000001. IP addresses have 4 octets. For example, 192 is the first octet, 168 the second, 10 the third and 1 the last octet. In its binary form, 11000000 is the first octet, 10101000 the second, 00001010 the third and 00000001 the last octet. Every octet, in its decimal form, can get a value from 0 to 255.

In the binary system there are only 1s and 0s. Depending on their position in the octet, they get different values. Each position is a power of 2. To get the decimal number you have to sum up those number.



For example, we know that 10101000 is 168. But let’s see if we get the same number after we make the calculations. 2^7 \* 1 + 2^6 \* 0 + 2^5 \* 1 + 2^4 \* 0 + 2^3 \* 1 + 2^2 \* 0 + 2^1 \* 0 + 2^0\* 0 = 128 + 32 + 8 = 168 Let’s now learn how to convert those numbers from decimal to binary. The decimal to binary conversion is similar to binary to decimal conversion. Keep in mind those powers of 2. When you calculate the binary value, you take those powers of 2 and compare them with your number. If your number is greater, you write down 1 and you subtract that power of 2

from the number. If your number is lower than the power of 2, you write down 0. You continue to make the calculations until you reach 2^0. Class A addresses were the ones from 0.0.0.0 to 127.255.255.255.

A class A network is has a default netmask of 255.0.0.0 allowing for up to 16,777,214 hosts per network ( 2^24 – 2). However, there’s possible to create only 128 network from the whole class A space. Or at least it was, back in the days, when classless routing was not used.

Class B addresses are from 128.0.0.0 to 191.255.255.255. The whole class B was able to create 16,384 networks (2^14) with a maximum number of 65,534 hosts per network (2^16 – 2). The default netmask si 255.255.0.0.

Class C networks were found within the 192.0.0.0 – 223.255.255.255 range. This class allowed for more networks -2,097,150 ( 2^21 ) but the maximum hosts per network was only 254 (2^8 – 2).

The class D and the class E address blocks are the same used today for multicasting, respectively the experimental addresses

**Part 1: Convert IPv4 Addresses from Dotted Decimal to Binary**

**Part 2: Use Bitwise ANDing Operation to Determine Network Addresses**

**Part 3: Apply Network Address Calculations**

Every IPv4 address is comprised of two parts: a network portion and a host portion. The network portion of an address is the same for all devices that reside in the same network. The host portion identifies a specific host within a given network. The subnet mask is used to determine the network portion of an IP address. Devices on the same network can communicate directly; devices on different networks require an intermediary Layer 3 device, such as a router, to communicate.

To understand the operation of devices on a network, we need to look at addresses the way devices do—in binary notation. To do this, we must convert the dotted decimal form of an IP address and its subnet mask to binary notation. After this has been done, we can use the bitwise ANDing operation to determine the network address.

This lab provides instructions on how to determine the network and host portion of IP addresses by converting addresses and subnet masks from dotted decimal to binary, and then using the bitwise ANDing operation. You will then apply this information to identify

Steps: -

Part 1: Convert IPv4 Addresses from Dotted Decimal to Binary

In Part 1, you will convert decimal numbers to their binary equivalent. After you have mastered this activity, you will convert IPv4 addresses and subnet masks from dotted decimal to their binary form.

Step 1: Convert decimal numbers to their binary equivalent.

Fill in the following table by converting the decimal number to an 8-bit binary number. The first number has been completed for your reference. Recall that the eight binary bit values in an octet are based on the powers of 2, and from left to right are 128, 64, 32, 16, 8, 4, 2, and 1.

|  |  |
| --- | --- |
| **Decimal** | **Binary** |
| 192 | 11000000 |
| 168 | 10101000 |
| 10 | 00001010 |
| 255 | 11111111 |
| 2 | 00000010 |

#### Step 2: Convert the IPv4 addresses to their binary equivalent.

An IPv4 address can be converted using the same technique you used above. Fill in the table below with the binary equivalent of the addresses provided. To make your answers easier to read, separate the binary octets with a period.

|  |  |
| --- | --- |
| **Decimal** | **Binary** |
| 192.168.10.10 | 11000000.10101000.00001010.00001010 |
| 209.165.200.229 | 11010001.10100101.11001000.11100101 |
| 172.16.18.183 | 10101100.00010000.00010010.10110111 |
| 10.86.252.17 | 00001010.01010110.11111100.00010001 |
| 255.255.255.128 | 11111111.11111111.11111111.10000000 |
| 255.255.192.0 | 11111111.11111111.11000000.00000000 |

Part 2: Use Bitwise ANDing Operation to Determine Network Addresses

In Part 2, you will use the bitwise ANDing operation to calculate the network address for the provided host addresses. You will first need to convert an IPv4 decimal address and subnet mask to their binary equivalent. Once you have the binary form of the network address, convert it to its decimal form.

**Note**: The ANDing process compares the binary value in each bit position of the 32-bit host IP with the corresponding position in the 32-bit subnet mask. If there two 0s or a 0 and a 1, the ANDing result is 0. If there are two 1s, the result is a 1, as shown in the example here.

#### Step 1: Determine the number of bits to use to calculate the network address.

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Decimal** | **Binary** | **Description** |
| IP Address | 192.168.10.131 | 11000000.10101000.00001010.10000011 | IP Address |
| Subnet Mask | 255.255.255.192 | 11111111.11111111.11111111.11000000 | Subnet Mask |
| Network Address | 192.168.10.128 | 11000000.10101000.00001010.10000000 | Network Address |

How do you determine what bits to use to calculate the network address?

The bits that are set to 1 in the binary subnet mask are used to calculate the network address.

In the example above, how many bits are used to calculate the network address?

 26 bits

#### Step 2: Use the ANDing operation to determine the network address.

1. Enter the missing information into the table below:

|  |  |  |
| --- | --- | --- |
| **Description** | **Decimal** | **Binary** |
| IP Address | 172.16.145.29 | 10101100.00010000.10010001.00011101 |
| Subnet Mask | 255.255.0.0 | 11111111.11111111.00000000.00000000 |
| Network Address | 172.16.0.0 | 10101100.00010000.00000000.00000000 |

Program:-

Int main()

{ char bin[4][8];

char bin2[4][9]; intipaddr[4]; if(!readIpAddress(ipaddr)) break;

convertDecToBin(ipaddr,bin2);

printf("The address converted to binary is :%8s %8s %8s %8s \n", bin2[0],bin2[1],bin2[2],bin2[3]);

break;

}

voidconvertDecToBin(intdec[],char bin[4][9])

{ inti,j; for(i=0;i=0;j--)

{ bin [i][j]=(dec[i]& 1)+'0'; dec[i]/=2; }

bin[i][8]=0; }

}

Conclusion.

Thus, we have studied the basic concepts of computer network deals Conversion of IP addresses and implement conversion from a dotted decimal IP form to 32 bit binary IP form

**Experiment 2**

Aim:- Introduction to Wireshark

Theory:-

In this first 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.

Figure 1 shows the structure of a packet sniffer. At the right of Figure 1 are the protocols (in this case, Internet protocols) and applications (such as a web browser or ftp client) that normally run on your computer. The packet sniffer, shown within the dashed rectangle in Figure 1 is an addition to the usual software in your computer, and consists of two parts.

The **packet capture library** receives a copy of every link-layer frame that is sent from or received by your computer. Recall from the discussion from section 1.5 in the text (Figure 1.241) that messages exchanged by higher layer protocols such as HTTP, FTP, TCP, UDP, DNS, or IP all are eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable. In Figure 1, the assumed physical media is an Ethernet, and so all upper-layer protocols are eventually encapsulated within an Ethernet frame. Capturing all link-layer frames thus gives you all messages sent/received from/by all protocols and applications executing in your computer.

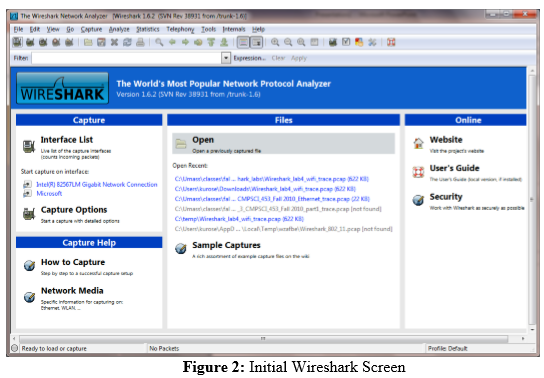
**Getting Wireshark**

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.

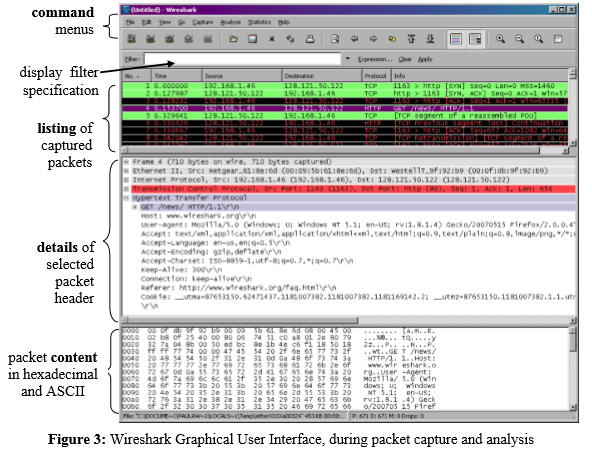
**Running Wireshark**

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



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 (“Microsoft”).

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.



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 packetlisting 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.

**Taking Wireshark for a Test Run**

The best way to learn about any new piece of software is to try it out! We’ll assume that your computer is connected to the Internet via a wired Ethernet interface. Indeed, I recommend that you do this first lab on a computer that has a wired Ethernet connection, rather than just a wireless connection. Do the following

1. Start up your favourite web browser, which will display your selected homepage.

2. Start up the Wireshark software. You will initially see a window similar to that shown in Figure 2. Wireshark has not yet begun capturing packets.

3. To begin packet capture, select the Capture pull down menu and select Interfaces. This will cause the “Wireshark: Capture Interfaces” window to be displayed

4. You’ll see a list of the interfaces on your computer as well as a count of the packets that have been observed on that interface so far. Click on Start for the interface on which you want to begin packet capture (in the case, the Gigabit network Connection). Packet capture will now begin - Wireshark is now capturing all packets being sent/received from/by your computer!

5. Once you begin packet capture, a window similar to that shown in Figure 3 will appear. This window shows the packets being captured. By selecting Capture pulldown menu and selecting Stop, you can stop packet capture. But don’t stop packet capture yet. Let’s capture some interesting packets first. To do so, we’ll need to generate some network traffic. Let’s do so using a web browser, which will use the HTTP protocol that we will study in detail in class to download content from a website.

6. While Wireshark is running, enter the URL: http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html and have that page displayed in your browser. In order to display this page, your browser will contact the HTTP server at gaia.cs.umass.edu and exchange HTTP messages with the server in order to download this page, as discussed in section 2.2 of the text. The Ethernet frames containing these HTTP messages (as well as all other frames passing through your Ethernet adapter) will be captured by Wireshark.

7. After your browser has displayed the INTRO-wireshark-file1.html page (it is a simple one line of congratulations), stop Wireshark packet capture by selecting stop in the Wireshark capture window. The main Wireshark window should now look similar to Figure 3.

8. Type in “http” (without the quotes, and in lower case – all protocol names are in lower case in Wireshark) into the display filter specification window at the top of the main Wireshark window. Then select Apply (to the right of where you entered “http”). This will cause only HTTP message to be displayed in the packet-listing window.

9. Find the HTTP GET message that was sent from your computer to the gaia.cs.umass.edu HTTP server. (Look for an HTTP GET message in the “listing of captured packets” portion of the Wireshark window (see Figure 3) that shows “GET” followed by the gaia.cs.umass.edu URL that you entered. When you select the HTTP GET message, the Ethernet frame, IP datagram, TCP segment, and HTTP message header information will be displayed in the packet-header window2. By clicking on ‘+’ and ‘-‘ right-pointing and down-pointing arrowheads to the left side of the packet details window, minimize the amount of Frame, Ethernet, Internet Protocol, and Transmission Control Protocol information displayed. Maximize the amount information displayed about the HTTP protocol. Your Wireshark display should now look roughly as shown in Figure 5. (Note, in particular, the minimized amount of protocol information for all protocols except HTTP, and the maximized amount of protocol information for HTTP in the packet-header window).

10. Exit Wireshark

Congratulations! You’ve now completed the first lab.

Conclusion: - Hence we have studied and installed Wireshark on computer.

**Experiment 3**

Aim:- Wireshark Lab: Ethernet and ARP

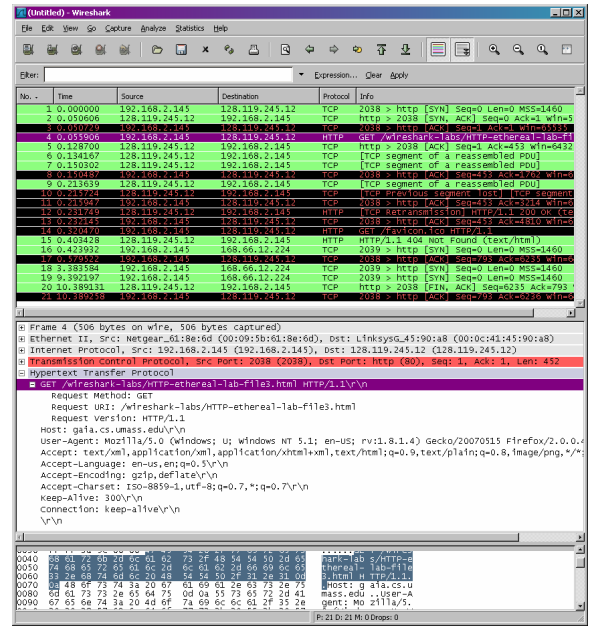
Theory:-

In this lab, we’ll investigate the Ethernet protocol and the ARP protocol. Before beginning this lab, you’ll probably want to review sections 5.4.1 (link-layer addressing and ARP) and 5.4.2 (Ethernet) in the book. RFC 826 (ftp://ftp.rfc-editor.org/innotes/std/std37.txt) contains the gory details of the ARP protocol, which is used by an IP device to determine the IP address of a remote interface whose Ethernet address is known.

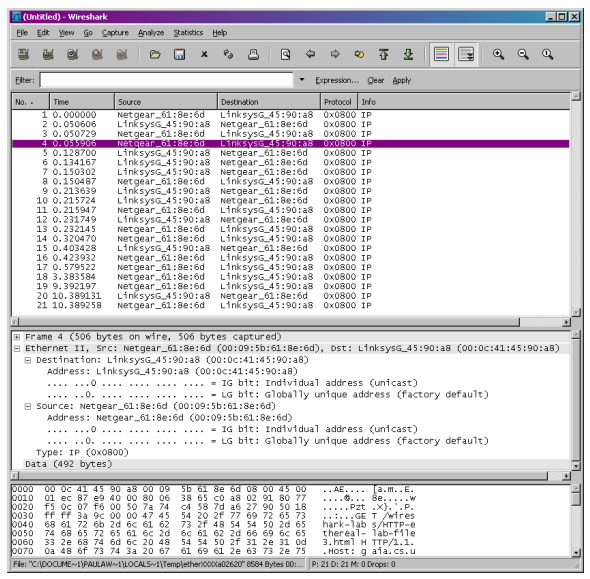
Steps:-

1. Capturing and analyzing Ethernet frames Let’s begin by capturing a set of Ethernet frames to study. Do the following1 : • First, make sure your browser’s cache is empty. (To do this under Netscape 7.0, select Edit->Preferences->Advanced->Cache and clear the memory and disk cache. For Internet Explorer, select Tools->Internet Options->Delete Files. For Firefox select Tools->Clear Private Data. • Start up the Wireshark packet sniffer • Enter the following URL into your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-ethereal-lab-file3.html Your browser should display the rather lengthy US Bill of Rights.

Stop Wireshark packet capture. First, find the packet numbers (the leftmost column in the upper Wireshark window) of the HTTP GET message that was sent from your computer to gaia.cs.umass.edu, as well as the beginning of the HTTP response message sent to your computer by gaia.cs.umass.edu. You should see a screen that looks something like this (where packet 4 in the screen shot below contains the HTTP GET message)



Since this lab is about Ethernet and ARP, we’re not interested in IP or higherlayer protocols. So let’s change Wireshark’s “listing of captured packets” window so that it shows information only about protocols below IP. To have Wireshark do this, select Analyze->Enabled Protocols. Then uncheck the IP box and select OK. You should now see an Wireshark window that looks like:



In order to answer the following questions, you’ll need to look into the packet details and packet contents windows (the middle and lower display windows in Wireshark). Select the Ethernet frame containing the HTTP GET message. (Recall that the HTTP GET message is carried inside of a TCP segment, which is carried inside of an IP datagram, which is carried inside of an Ethernet frame; reread section 1.7.2 in the text if you find this nesting a bit confusing). Expand the Ethernet II information in the packet details window. Note that the contents of the Ethernet frame (header as well as payload) are displayed in the packet contents window.

Answer the following questions, based on the contents of the Ethernet frame containing the HTTP GET message. Whenever possible, when answering a question 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.

1. What is the 48-bit Ethernet address of your computer?

2. What is the 48-bit destination address in the Ethernet frame? Is this the Ethernet address of gaia.cs.umass.edu? (Hint: the answer is no). What device has this as its Ethernet address? [Note: this is an important question, and one that students sometimes get wrong. Re-read pages 468-469 in the text and make sure you understand the answer here.]

3. Give the hexadecimal value for the two-byte Frame type field. What do the bit(s) whose value is 1 mean within the flag field?

4. How many bytes from the very start of the Ethernet frame does the ASCII “G” in “GET” appear in the Ethernet frame?

5. What is the hexadecimal value of the CRC field in this Ethernet frame?

Next, answer the following questions, based on the contents of the Ethernet frame containing the first byte of the HTTP response message.

6. What is the value of the Ethernet source address? Is this the address of your computer, or of gaia.cs.umass.edu (Hint: the answer is no). What device has this as its Ethernet address?

7. What is the destination address in the Ethernet frame? Is this the Ethernet address of your computer?

8. Give the hexadecimal value for the two-byte Frame type field. What do the bit(s) whose value is 1 mean within the flag field?

9. How many bytes from the very start of the Ethernet frame does the ASCII “O” in “OK” (i.e., the HTTP response code) appear in the Ethernet frame?

10. What is the hexadecimal value of the CRC field in this Ethernet frame?

2. The Address Resolution Protocol

In this section, we’ll observe the ARP protocol in action. We strongly recommend that

you re-read section 5.4.2 in the text before proceeding.

ARP Caching

Recall that the ARP protocol typically maintains a cache of IP-to-Ethernet address translation pairs on your comnputer The arp command (in both MSDOS and Linux/Unix) is used to view and manipulate the contents of this cache. Since the arp command and the ARP protocol have the same name, it’s understandably easy to confuse them. But keep in mind that they are different - the arp command is used to view and manipulate the ARP cache contents, while the ARP protocol defines the format and meaning of the messages sent and received, and defines the actions taken on message transmission and receipt.

Let’s take a look at the contents of the ARP cache on your computer:

• MS-DOS. The arp command is in c:\windows\system32, so type either “arp” or “c:\windows\system32\arp” in the MS-DOS command line (without quotation marks).

• Linux/Unix. The executable for the arp command can be in various places. Popular locations are /sbin/arp (for linux) and /usr/etc/arp (for some Unix variants).

The arp command with no arguments will display the contents of the ARP cache on your

computer. Run the arp command.

11. Write down the contents of your computer’s ARP cache. What is the meaning of each column value?

In order to observe your computer sending and receiving ARP messages, we’ll need to clear the ARP cache, since otherwise your computer is likely to find a needed IP-Ethernet address translation pair in its cache and consequently not need to send out an ARP message.

• MS-DOS. The MS-DOS arp –d \* command will clear your ARP cache. The –d flag indicates a deletion operation, and the \* is the wildcard that says to delete all table entries.

• Linux/Unix. The arp –d \* will clear your ARP cache. In order to run this command you’ll need root privileges. If you don’t have root privileges and can’t

run Wireshark on a Windows machine, you can skip the trace collection part of this lab and just use the trace discussed in footnote 1.

Observing ARP in action

Do the following2:

• Clear your ARP cache, as described above.

• Next, make sure your browser’s cache is empty. (To do this under Netscape 7.0, select Edit->Preferences->Advanced->Cache and clear the memory and disk cache. For Internet Explorer, select Tools->Internet Options->Delete Files.)

• Start up the Wireshark packet sniffer

• Enter the following URL into your browser

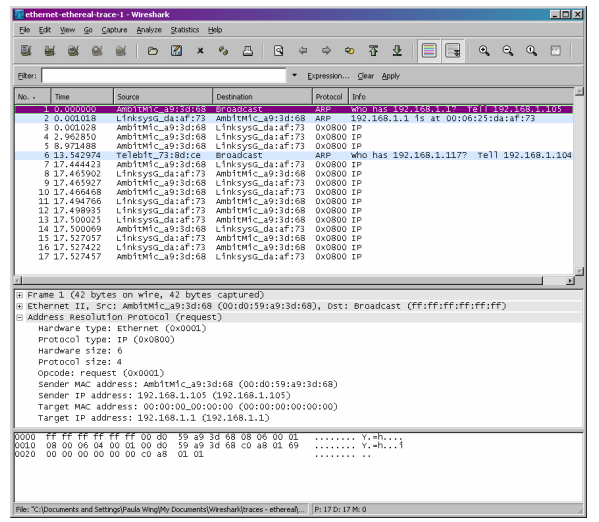
http://gaia.cs.umass.edu/wireshark-labs/ HTTP-wireshark-lab-file3.html

Your browser should again display the rather lengthy US Bill of Rights.

• Stop Wireshark packet capture. Again, we’re not interested in IP or higher-layer protocols, so change Wireshark’s “listing of captured packets” window so that it shows information only about protocols below IP. To have Wireshark do this,

select Analyze->Enabled Protocols. Then uncheck the IP box and select OK.

You should now see an Wireshark window that looks like:



Conclusion: - Traced and studied ARP and Ethernet details in Wireshark.

**Experiment 4**

Aim:- Wireshark Lab: IP

Theory:-

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.

Steps:-

**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

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 traceroute 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 traceroute 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 traceroute 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 traceroutes 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. With the Unix traceroute 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 traceroute command line immediately after the name or address of the destination. For example, to send traceroute datagrams of 2000 bytes towards gaia.cs.umass.edu, the command would be:

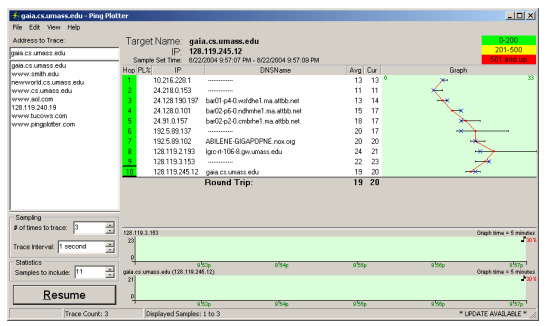
%traceroute gaia.cs.umass.edu 2000

Do the following:

• Start up Wireshark and begin packet capture (Capture->Option) 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 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 computers2

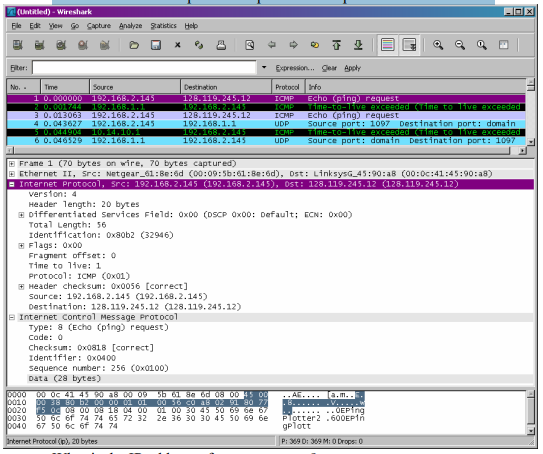
. You may well find it valuable to download this trace even if you’ve captured your own trace and use it, as well as your own trace, when you explore the questions below.

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. 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.

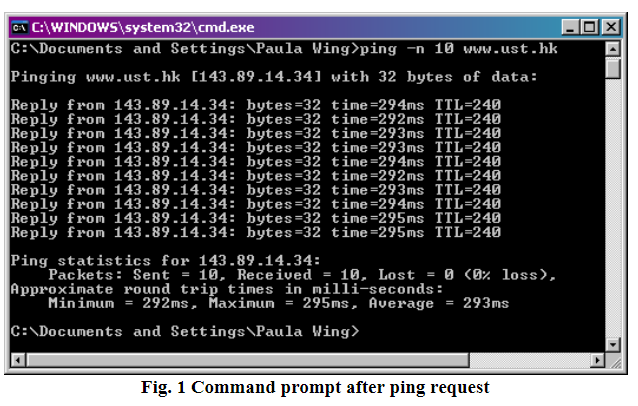


Conclusion: - Traced and studied IP details in Wireshark.

**Experiment 5**

Aim:- Wireshark Lab: ICMP, study of ping and traceroute command

Theory:-



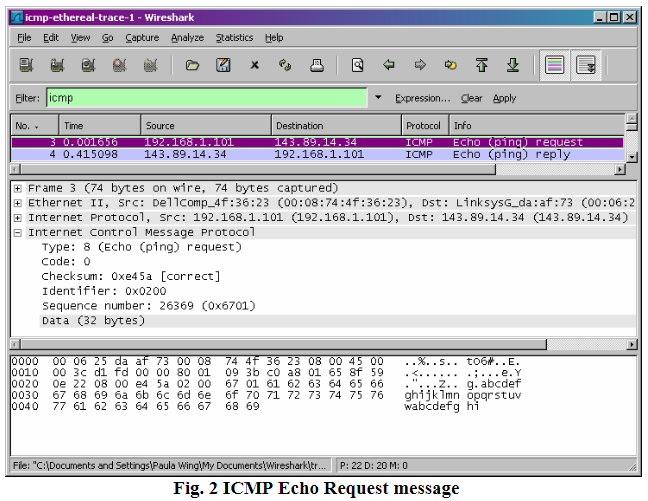
1.What is the IP address of your host? What is the IP address of the destination host?

The IP address of my host is 192.168.1.101. The IP address of the destination host is 143.89.14.34.

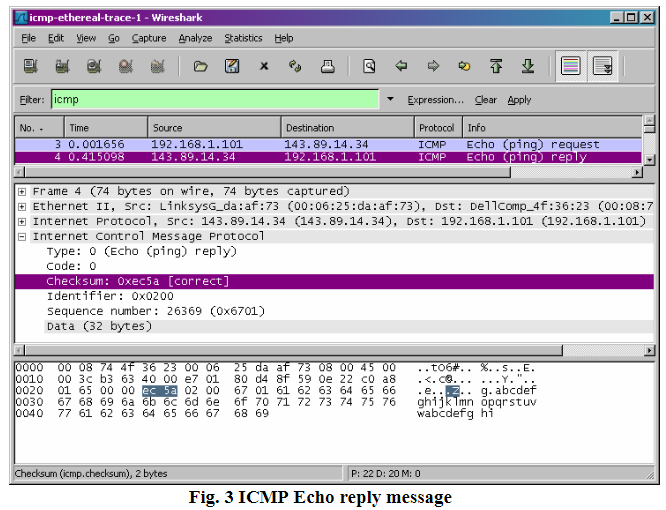
2.Why is it that an ICMP packet does not have source and destination port numbers?

The ICMP packet does not have source and destination port numbers because it was designed to communicate network k-layer information between hosts and routers, not between application layer processes. Each ICMP packet has a "Type" and a "Code". The Type/Code combination identifies the specific message being received. Since the network software itself interprets all ICMP messages, no port numbers are needed to direct the ICMP message to an

application layer process.



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? The ICMP type is 8, and the code number is 0. The ICMP packet also has checksum, identifier, sequence number, and data fields. The checksum, sequence number and identifier fields are two bytes each.



4.Examine the corresponding ping reply packet. What are the ICMP type and code numbers? What other fields does this ICMP packet have?

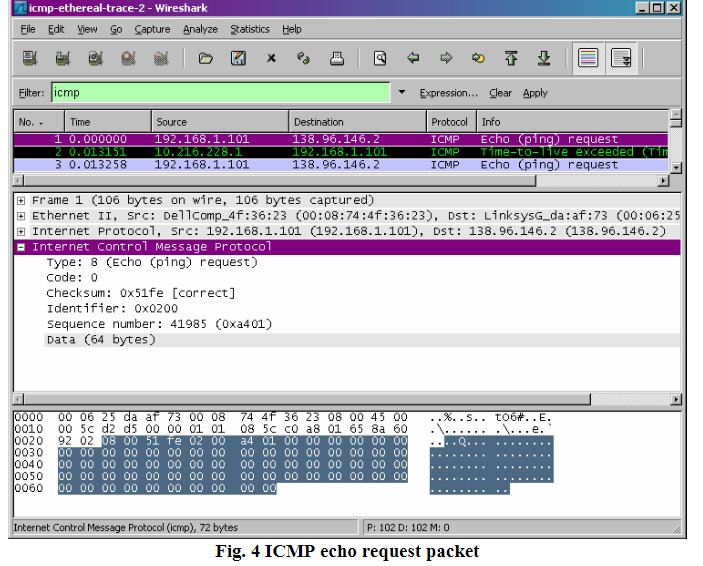
How many bytes are the checksum, sequence number and identifier fields? The ICMP type is 0, and the code number is 0. The ICMP packet also has checksum, identifier, sequence number, and data fields. The checksum, sequence number and identifier fields are two bytes each.

5.What is the IP address of your host?

What is the IP address of the target destination host? The IP address of my host is 192.168.1.101. The IP address of the destination host is 138.96.146.2.

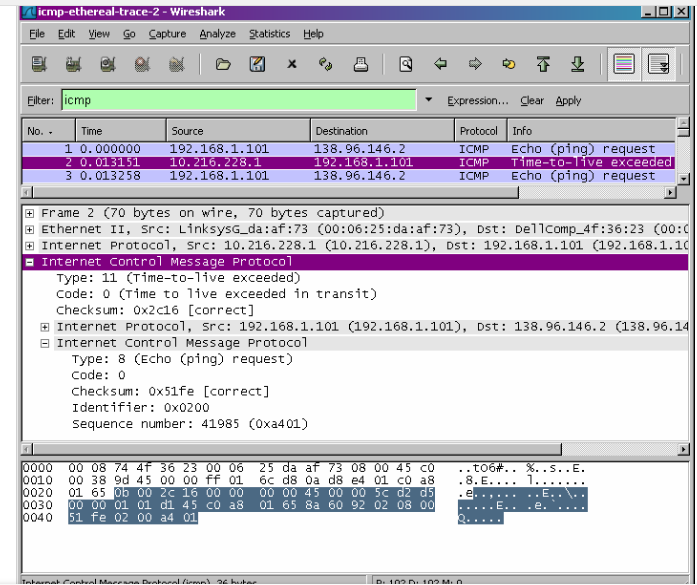
6.If ICMP sent UDP packets instead (as in Unix/Linux), would the IP protocol number still be 01 for the probe packets? If not, what would it be?

No. If ICMP sent UDP packets instead, the IP protocol number should be 0x11



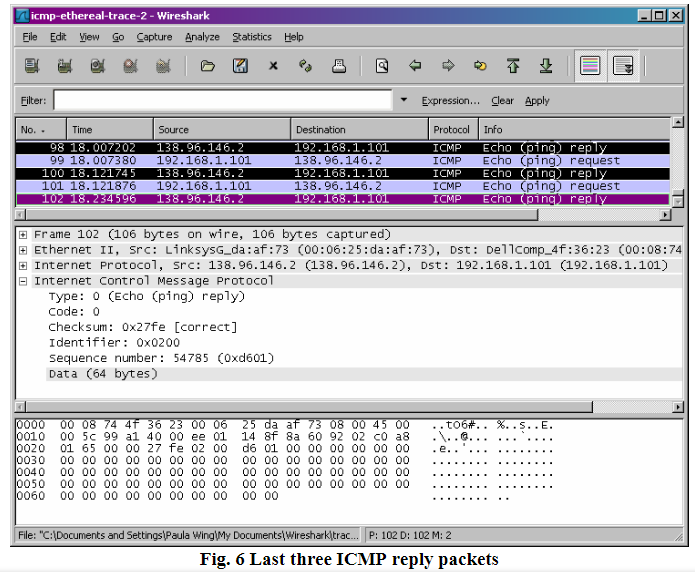
7.Examine the ICMP echo packet in your screenshot. Is this different from the ICMP ping query packets in the first half of this lab? If yes, how so?

The ICMP echo packet has the same fields as the ping query packets.



8.Examine the ICMP error packet in your screenshot. It has more fields than the ICMP echo packet. What is included in those fields?

The ICMP error packet is not the same as the ping query packets. It contains both the IP header and the first 8 bytes of the original ICMP packet that the error is for.



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?

The last three ICMP packets are message type 0 (echo reply) rather than 11 (TTL expired). They are different because the datagrams have made it all the way to the destination host before the TTL expired.

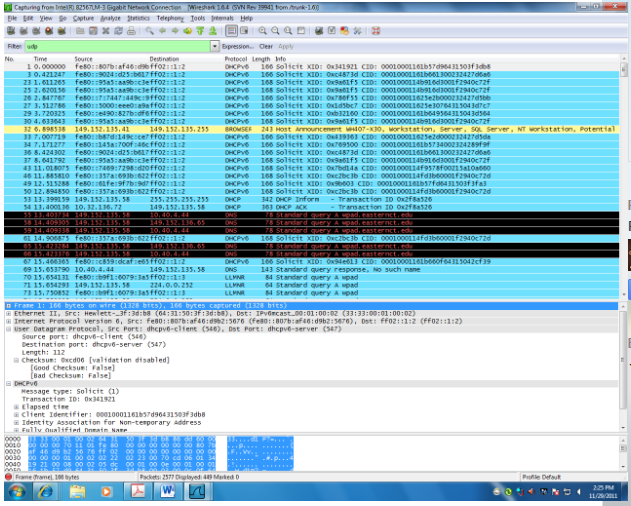
Conclusion: - Traced and studied ICMP, ping and traceroute command details in Wireshark.

**Experiment 6**

Aim: - Wireshark Lab: UDP

Theory: - https://maxwellsullivan.wordpress.com/2013/03/12/wireshark-lab-5-exploring-udp/

Steps: - http://wireshark16.blogspot.com/2011/11/wireshark-lab-udp.html



1. Select one packet. From this packet, determine how many fields there are in the

UDP header. (Do not look in the textbook! Answer these questions directly from

what you observe in the packet trace.) Name these fields.

The UDP header contains 4 fields. They are source Port, Destination port, Length and checksum.

2. From the packet content field, determine the length (in bytes) of each of the UDP

header fields.

The UDP has four fields at two bytes each so in total it is 8 bytes

Source Port is 2 bytes

Destination port is 2 bytes

Length is 2 bytes

Checksum is 2 bytes

3. The value in the Length field is the length of what? Verify your claim with your

captured UDP packet.

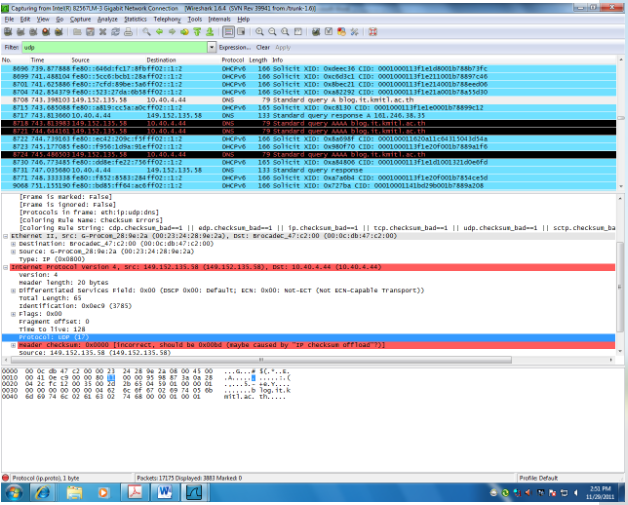
The value in the length field is the length of the header and the data inside in bytes.

4. What is the maximum number of bytes that can be included in a UDP payload.

The maximum length with the header included is 65535 but the actual maximum number of bytes with the header excluded is 65527

5. What is the largest possible source port number?

The largest possible source port number is 65535.



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 IP header.)

The protocol number is 17 in decimal or 11 in hexadecimal

7. Search “UDP” in Google and determine the fields over which the UDP checksum is calculated.

After searching on google the checksum I found the Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets

8. Examine a pair of UDP packets in which the first packet is sent by your host and the second packet is a reply to the first packet. Describe the relationship between the port numbers in the two packets.

In the first packet sent by my host the source port of the UDP packet is the same as the destination port of the reply packet. Also the destination port of the UDP packet sent by my host computer matches the source port of the reply packet.

Conclusion: - Traced and studied UDP details in Wireshark.

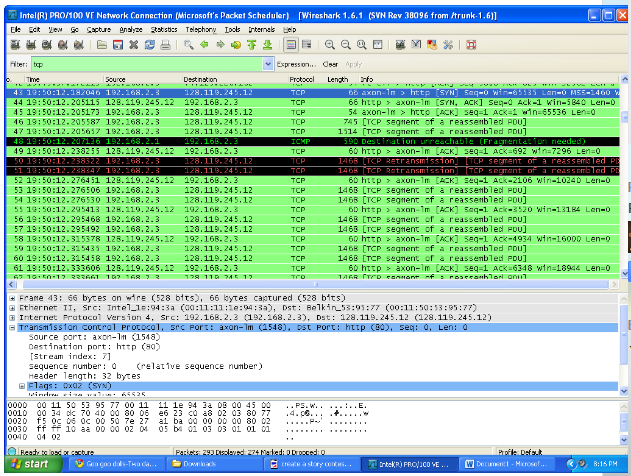
**Experiment 7**

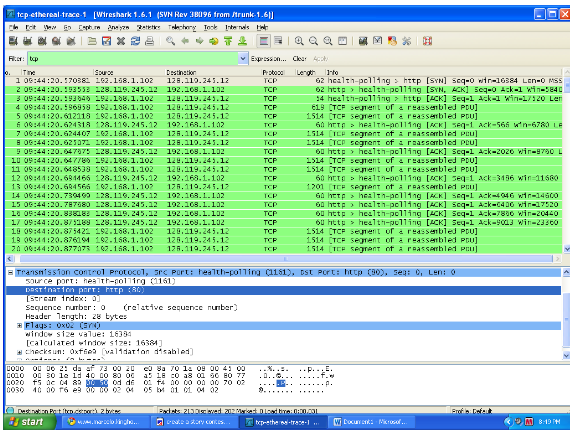
Aim:- Wireshark Lab: TCP

Theory:-

http://wireshark16.blogspot.com/2011/11/tcp-lab.html

Steps:- http://wireshark16.blogspot.com/2011/11/tcp-lab.html





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?

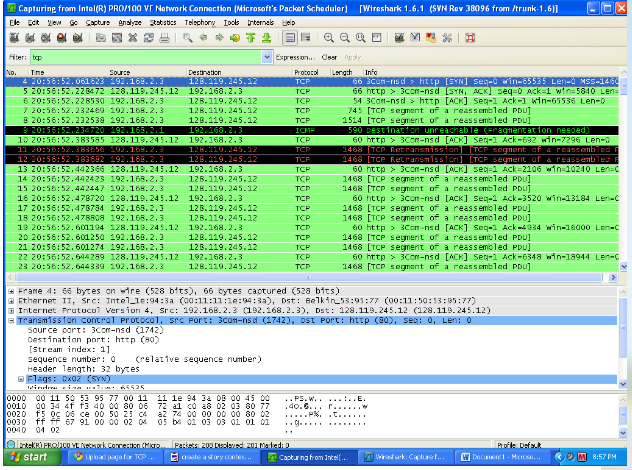
Client IP : 192.168.1.102

TCP port: health-polling (1161)

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

Ip address : 128.119.245.12

TCP Port: 80



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?

192.168.2.3

Source port:(1742)

4. 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? What is it in the segment that identifies the segment as a SYN segment?

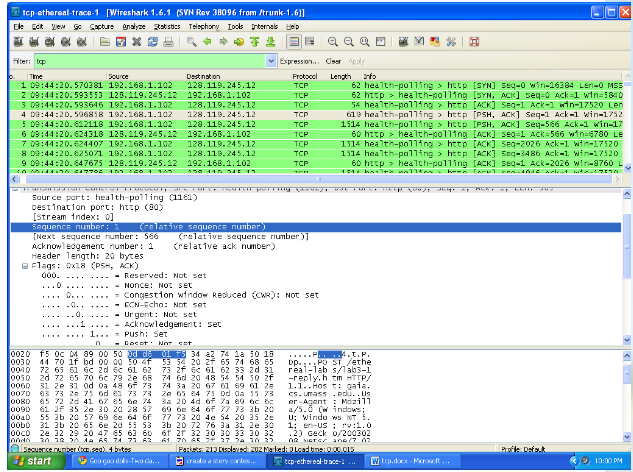
Sequence number: 0 (relative sequence number)

The syn segement can be seen to be set to 1 (.... .... ..1. = Syn: Set) This shows it is a syn segment

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

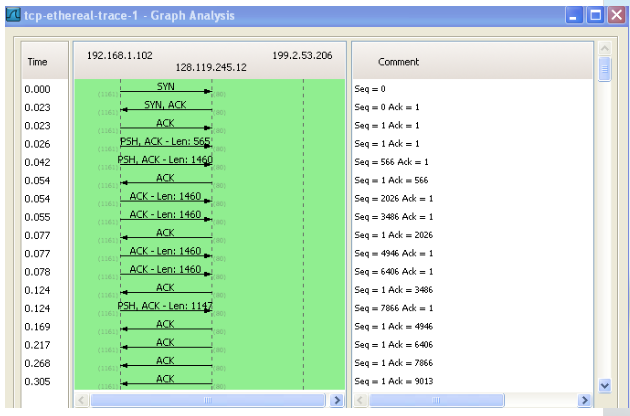
The sequence number: Acknowledgement number: 1 (relative ack number) value of the ACKnowledgement field is 1 gaia.cs.umass.edu determined that value by adding 1 to the sequence number of the previous segement.

This segment is identified as a synack segment acknowledgement and syn bits are both set.



6. 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.

The sequence number of the TCP segment containing the HTTP POST Command is 1



7. 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? What is the EstimatedRTT value (see page 249 in text) after the receipt of each ACK? Assume that the value of the EstimatedRTT is equal to the measured RTT for the first segment, and then is computed using the EstimatedRTT equation on page 249 for all subsequent segments.

Note: Wireshark has a nice feature that allows you to plot the RTT for each of the TCP segments sent. Select a TCP segment in the “listing of captured packets” window that is being sent from the client to the

gaia.cs.umass.edu server. Then select: Statistics->TCP Stream Graph-

>Round Trip Time Graph.

Conclusion: - Traced and studied TCP details in Wireshark.

**Experiment 8**

Aim:- Study of ftp, telnet tools and network configuration files.

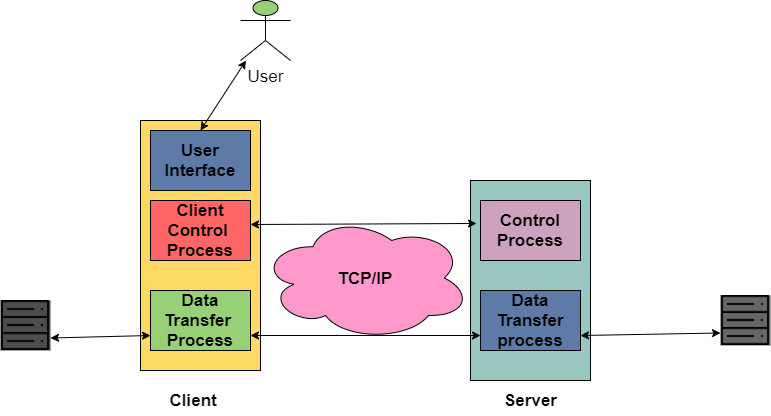
Theory:-

FTP means File Transfer Protocol and it is the standard mechanism provided by the TCP/IP in order to copy a file from one host to another.

* File Transfer Protocol is a protocol present at the Application layer of the OSI Model.
* FTP is one of the easier, simpler, and secure ways to exchange files over the Internet.
* FTP is different from the other client/server applications as this protocol establishes two connections between the hosts.
  + where one connection is used for the data transfer and is known as a **data connection.**
  + while the other connection is used to control information like commands and responses and this connection is termed as **control connection.**
* FTP is more efficient as there is the separation of commands.
* The File Transfer Protocol makes the use of two protocols; **Port 21** for the **Control connection**and **Port 20** is used for **Data connection.**
* The control connection in FTP makes the use of very simple rules of communication, we just need to transfer a line of command or a line of response at a time.
* On the other hand, the data connection needs more complex rules; and the reason behind this is there are a variety of types of data that needs to be transferred.
* The transferring of files from the client computer to the server is termed as "uploading", while the transferring of data from the server to the client computer is termed as "downloading".
* The types of files transferred using the FTP are ASCII files, EBCDIC files, or image files.

## **Working of FTP**

Given below figure shows the basic model of file Transfer Protocol, where the client comprises of three components: User Interface,Client control process, and client data transfer process. On the other hand, the server comprises of two components mainly the server control process and the server data transfer process.



1. Also, the control connection is made between the control processes while the data connection is made between the data transfer processes.
2. The control Connection remains connected during the entire interactive session of FTP while the data connection is opened and then closed for each file transferred.
3. In simple terms when a user starts the FTP connection then the control connection opens, while it is open the data connection can be opened and closed multiple times if several files need to be transferred.

## Data Structure

Given below are three data structures supported by FTP:

**1.File Structure** In the File data structure, the file is basically a continuous stream of bytes.

**2.Record Structure** In the Record data structure, the file is simply divided into the form of records.

**3.Page Structure** In the Page data structure, the file is divided into pages where each page has a page number and a page header. These pages can be stored and accessed either randomly or sequentially.

# **FTP Clients**

It is basically software that is designed to transfer the files back-and-forth between a computer and a server over the Internet. The FTP client needs to be installed on your computer and can only be used with the live connection to the Internet.

Some of the commonly used FTP clients are Dreamweaver, FireFTP, and Filezilla.

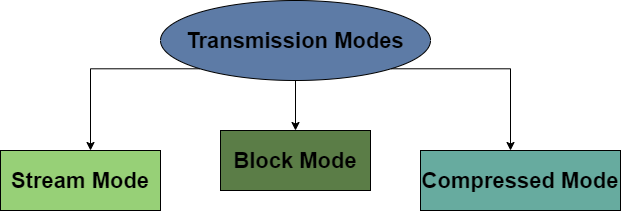
# **Features of FTP**

Following are the features offered by the File transfer protocol:

* FTP is mainly used to transfer one file at a time.
* Other actions performed by FTP are listing files, creating and deleting directories, deleting files, renaming files, and many more.
* FTP also hides the details of individual computer systems.
* FTP allows those files that have ownership and access restrictions.
* It is a connection-oriented protocol.
* FTP is a stateful protocol as in this the client establishes a control connection for the duration of an FTP session that typically spans multiple data transfers.

**Transmission Modes**

FTP can transfer a file across the data connection using one of the three given modes:



### 1.Stream Mode

Stream Mode is the default mode of transmission used by FTP. In this mode, the File is transmitted as a continuous stream of bytes to TCP.

If the data is simply in the form of the stream of bytes then there is no need for End-of-File, closing of data connection by the sender is considered as EOF or end-of-file. If the data is divided into records (that is the record structure), each record has an I-byte of EOR(end-of-record).

### 2.Block Mode

Block mode is used to deliver the data from FTP to TCP in the form of blocks of data. Each block of data is preceded by 3 bytes of the header where the first byte represents the block descriptor while the second and third byte represents the size of the block.

### 3.Compressed Mode

In this mode, if the file to be transmitted is very big then the data can be compressed. This method is normally used in Run-length encoding. In the case of a text file, usually, spaces/blanks are removed. While in the case of the binary file, null characters are compressed.

## **Advantages of FTP**

Following are some of the benefits of using File Transfer protocol:

* Implementation of FTP is simple.
* FTP provides one of the fastest ways to transfer files from one computer to another.
* FTP is a standardized protocol and is widely used.
* File Transfer protocol is more efficient as there is no need to complete all the operations in order to get the entire file,

## **Disadvantages of FTP**

Let us take a look at the drawbacks of FTP:

* File Transfer Protocol is not a secure way to transfer the data.
* FTP does not allow the copy from server to server and also not allows removal operations for the recursive directory.
* Scripting the jobs is hard using the FTP protocol.
* The spoofing of the server can be done in order to send data to a random unknown port on any unauthorized computer

**Introduction to Telnet**

In other chapters of this book, you have connected to other computers by traversing menus using gopher or by using one of the www browsers.  Often you might want to connect directly to another computer for services that are not provided on your computer.  For example, one of the authors does not have a mail server on his PC.  He needs to connect to another computer to read and send mail.  One of the ways of connecting to another computer on the Internet is using the Telnet protocol.  Telnet was one of the first protocols developed for the Internet.  It was designed to allow users around the world to use computers located at remote sites.  The program **telnet** is designed to make use of this Telnet protocol.

Today **telnet** is used to by individuals to log into other computers that they have accounts on.  **Telnet** is also used to log into computers that provide access to services such as library catalogs, Internet resources like **gopher** or **archie**, or other facilities that can be made available to others.  With the increased use of the World-wide Web, less emphasis is being placed on using **telnet** to connect to other computers.

**How to Use Telnet**

**Internet Addresses**

Before you can connect to another computer, you must know its Internet address.  All Internet hosts have two kinds of addresses.  Most of the time you will be able to use the domain address that is written in text, e.g. **www.jaring.my**.  Domain addresses usually have at least three parts.  The last part is either a two-letter geographic top-level domain or an organizational top-level domain (See Fig. 6.1). The next to last part of the address is a sub-domain that generally represents the institution.  The leftmost sub-domain of the address usually represents a computer at the institution.  So the address www.jaring.my represents the computer running the www server at the jaring institution in Malaysia.

Underlying the domain address is another address called the IP (Internet Protocol) address. This address is a numeric address represented as four numbers separated by periods. When you attempt to access a computer using its domain address, the address is converted into the corresponding IP address and then the connection is made.  For example, if you were to **telnet** to **www.jaring.my**, the domain address would be converted to the IP address **192.228.128.16**.  You can use the IP address anywhere that you would use the domain address.  There may be a few occasions where a domain address will not be recognized and you will have to use the IP address.

**The Telnet Program**

To use Telnet, you need to run a program often called telnet on your computer.  This program uses the Internet to connect to the computer you have specified.  You can either run the telnet program by typing telnet with no arguments, or by typing telnet followed by the name of the host you are connected to on the command line- telnet www.jaring.my.   If necessary, you may have to follow the domain address with a port number.

If you only type **telnet** on the command line, your computer's prompt will change to *telnet>* and you will need to use the *open* statement to connect to another computer on the Internet, e.g. 

***telnet>*** *open www.jaring.my*

The first few messages you see will be messages from your computer.  These will be followed by messages from the remote computer.  Usually you will be asked to enter your a user-id and password.  (See Fig. 7.2) Some of the public **telnet** sites will allow you to access them without a user-id and password.  Once this happens, you are logged onto the remote computer and can use all of the instructions that are appropriate for the remote computer.

When you are finished using the remote computer, you should log-out.  This often does not close the connection between your computer and the remote computer.  If you are still connected to the remote computer, you can break the connection by pressing CTRL-].  This will return you to the *telnet>* prompt.  You can then connect to another computer using the *open* statement or *quit* and return to using your own computer.  If you run into trouble, the question mark (?) will bring up a list of available commands.

-VE> **telnet ppp.itm.my**

Trying...192.228.170.5

Connected to PPP.ITM.MY.

Escape character is '^]'.

SunOS UNIX (pppitm)

login: **wayne**

Password: **\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Last login: Fri Jun 16 05:51:38 from venus.nmhu.edu

SunOS Release 4.1.3 (GENERIC-MAR17) #1: Fri Mar 17 09:03:22 SST 1995

Fig. 6.2 - Logging into another computer using **telnet**

Occasionally, you will not be able to connect to a remote site.  If you incorrectly type the domain address, you will see an error message as shown in Fig. 6.3. Sometimes the remote computer may not be running or all the lines may be busy.  If this happens, you will typically see a message like the one shown in Fig. 6.4.

-VE> telnet newton.dip.anl.gov

%UCX-E-TELNET\_GETHST, Error in getting host name

Fig. 6.3 - Incorrect address

-VE> telnet irc.nsysu.edu.tw

Trying...140.117.11.33

%UCX-E-TELNET\_CONECT, Failed to connect to remote host

Fig. 6.4 - Cannot connect to remote computer

When you connect to a remote computer, you will often be asked to enter a terminal type.  In the early days of computing, most users worked at terminals connected to the mainframe or minicomputer.  Today to connect to many servers, you need to emulate one of these types of terminals.  This is done using a terminal emulation software package.  The most common ones are DEC's (Digital Equipment Corporation) VT100 and VT220.  If the remote computer asks for a terminal type, try one of these first.  If neither works, try TTY which is the standard for teletypes.

In a few cases, **telnet** will not work.  If a computer is an IBM mainframe, it will expect you to use the program**tn3270.**  You can usually identify these by the *VM* or *MVS* in the introductory messages from the remote computer.  When you see this, try typing **tn3270** instead of **telnet**.

**Telnet Software**

Two of the more popular public domain / shareware packages for the PCs are TELNET and WinQVT (Fig. 6.5).  These both are available through common **ftp** sites discussed later in this chapter.  They are both easy to install under Windows.

Conclusion: Hence we have studied telnet and ftp.

**Experiment 9**

Aim:- DHCP server configuration

Theory:-

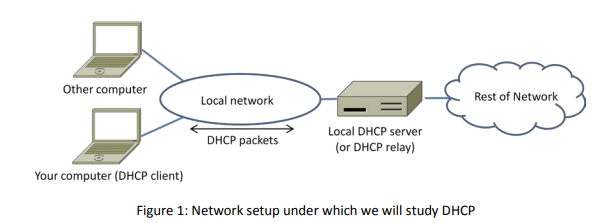
To see how DHCP (Dynamic Host Configuration Protocol) works. DHCP is an essential glue protocol that is used to configure your computer with an IP address, as well as other information.

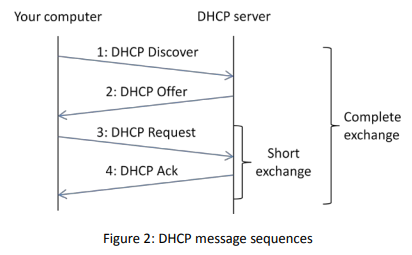
Wireshark: This lab uses the Wireshark software tool to capture and examine a packet trace. A packet trace is a record of traffic at a location on the network, as if a snapshot was taken of all the bits that passed across a particular wire. The packet trace records a timestamp for each packet, along with the bits that make up the packet, from the lower-layer headers to the higher-layer contents. Wireshark runs on most operating systems, including Windows, Mac and Linux. It provides a graphical UI that shows the sequence of packets and the meaning of the bits when interpreted as protocol headers and data. It color-codes packets by their type, and has various ways to filter and analyze packets to let you investigate the behavior of network protocols. Wireshark is widely used to troubleshoot networks. You can download it from www.wireshark.org if it is not already installed on your computer. We highly recommend that you watch the short, 5 minute video “Introduction to Wireshark” that is on the site. ipconfig (windows) / ifconfig (mac) / dhclient (linux): This lab uses a command-line utility to cause the computer renew its IP address lease using DHCP. The commands are installed on Windows/Mac/Linux computers, but there is a different command for each different operating system. The commands require administrative privileges to run when they change the state of the interface.

**Network Setup**

Recall that DHCP is normally used to assign a computer its IP address, as well as other parameters such as the address of the local router. Your computer, the client, uses the DHCP protocol to communicate with a DHCP server on the local network. Other computers on the local network also interact with the DHCP server. The setup is as shown in the example below. In deployments, there are several variations. For example, the local agent may be a DHCP relay that relays messages between local computers and a remote DHCP server. Or the DHCP server may be replicated for reliability, so that there are two or more local DHCP servers. For our purposes, it is sufficient to think about a single DHCP server.

The complete DHCP exchange for your computer to lease an IP address from a DHCP server is shown below. It involves four types of packets: Discover, for your computer to locate the DHCP server; Offer, for the server to offer an IP address; Request, for your computer to ask for an offered address; and Ack, for the server to grant the address lease. However, when a computer is re-establishing its IP address on a network that it has previously used, it may perform a short exchange involving only two types of DHCP packets: Request, to ask for the same IP address as from the same server as was used before; and ACK for the server to grant the address lease.



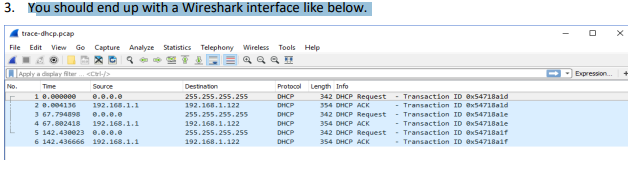


Steps:-

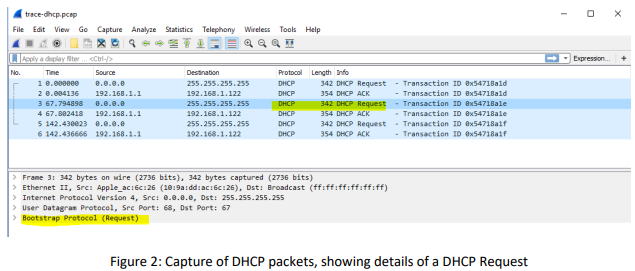
**Step 1: Capture a Trace**

**Step 1:** Capture a Trace Proceed as follows to renew your IP address and gather a trace of DHCP traffic. Note, however, that the following procedure will not work in the unlikely case that your computer’s IP address is statically assigned. Alternatively, you may use a supplied trace. Take care not to perform this lab remotely, since when you tell your computer to shut down and restart its network interface you will lose connectivity!

1. Launch Wireshark and start a capture with a filter of “(udp port 67) or (udp port 68)”. There is no shorthand to indicate DHCP, so we filter traffic using the UDP ports reserved for DHCP. Your capture window should be similar to the one pictured below, other than our highlighting. Select the interface from which to capture as the main wired or wireless interface used by your computer to connect to the Internet. If unsure, guess and revisit this step later if your capture is not successful. Uncheck “capture packets in promiscuous mode”. This mode is useful to overhear packets sent to/from other computers on broadcast networks. We only want to record packets sent to/from your computer. Leave other options at their default values. The capture filter, if present, is used to prevent the capture of other traffic your computer may send or receive. On Wireshark 1.8, the capture filter box is present directly on the options screen, but on Wireshark 1.9, you set a capture filter by double-clicking on the interface
2. Click on the Wireshark Trace at https://kevincurran.org/com320/labs/wireshark/trace-dhcp.pcap. OR if you do not wish to follow step 1, you can proceed manually by opening Wireshark as follows:
3. Start Wireshark by going to the bottom left of your Windows PC and typing “Wireshark” as shown in (a) on the left below. In some cases you are asked to upgrade, please click “skip this version” as highlighted below in (b). – OR you can start Wireshark by clicking on your download trace file.
4. You should end up with a Wireshark interface like below.



Step 2: Inspect the Trace Look for the short DHCP exchange (of a DHCP Request packet followed by a DHCP Ack packet) in your trace. Select the DHCP Request packet, and observe the protocol stack to see how DHCP messages are carried. The link protocol is likely Ethernet, and the next higher protocol is IP. Then comes UDP, so each DHCP message is carried in a UDP packet. On top of UDP, Wireshark is likely to say BOOTP (Bootstrap Protocol) instead of DHCP. This is a bit confusing, but DHCP is implemented as an extension of an older protocol called BOOTP. You can think of the BOOTP section as the DHCP header and message. An example window is shown below.



Expand the BOOTP (DHCP) section (using the “+” expander or icon) to look at the details of a DHCP Request message. There are many fields, and we will only point out a few rather than cover them all. These fields are carried in all DHCP messages, though they have different values in different messages.

• The message begins with a Message Type. It is a Boot Request, which is used for all DHCP messages sent from your computer to a DHCP server.

• After a few fields there is a Transaction ID field. All DHCP packets in a specific exchange between a client and server carry the same transaction ID; that is how both ends know that the packets belong to the exchange rather than another concurrent DHCP operation.

• There are several IP address fields. These fields are used to carry IP addresses such as the one that the computer is being assigned.

• There is a Magic Cookie field. It carries a value that indicates the rest of the message contains a series of DHCP Options. That is, this really is a DHCP message, not a BOOTP message.

• Each DHCP option is self-contained, with a type code saying what it represents, along with a length and value. The first option is DHCP Message Type, which says what kind of DHCP message is being carried. The other options vary with the type of DHCP message. For example, a DHCP Request will have a Requested IP Address option to ask for a specific address, which a DHCP Ack will have a IP Address Lease Time option to say for how long the IP address is being assigned.

Now select a DHCP Ack packet and compare the BOOTP fields. We will ask questions about these fields in the next section, but for now want you to observe that the DHCP Ack has the same overall format, but different values for the fields and carries different DHCP options. You can browse the options for DHCP Requests and Acks to learn about DHCP. You can see, for example, how long the IP address is assigned by the server, whether seconds, minutes, hours or days. You will also see the other configuration parameters that are assigned by the DHCP server, such as the IP address of the domain name server and router, the subnet mask, the domain name for the host, and more. You can also try to make out the whole sequence of DHCP messages that is exchanged for your network setup. It may be as simple as the short exchange of Request and Ack, or it may be the complete exchange of Discover, Offer, Request and Ack. It may have additional messages such as Release, and it may have multiple of the messages (e.g., two or more Offers or Acks) due to multiple local DHCP servers. Complicating the exchange with your computer is that the trace may capture concurrent DHCP traffic from other local computers. You can use the Transaction IDs to separate the different exchanges, and look at the Ethernet source address to see which DHCP messages were sent by your computer. It is likely that other DHCP traffic is mixed in with your exchange.

Step 3: Details of DHCP Messages Spend time understanding DHCP. Note the position of the Ethernet, IP, UDP, and BOOTP protocol block.

Answer the following questions based on your examination of the BOOTP/DHCP fields for both the DHCP Request and DHCP Ack. Answers on next page.

1. What are the two values of the BOOTP Message Type field?

2. How long is the Transaction ID field? Say whether it is likely that concurrent DHCP operations done by different computers will happen to pick the same Transaction ID.

3. What is the name of the field that carries the IP address that is being assigned to the client? You will find this field filled in on the DHCP Ack, as that message is completing the assignment.

4. The first DHCP option is DHCP Message Type. What option value stands for this type?

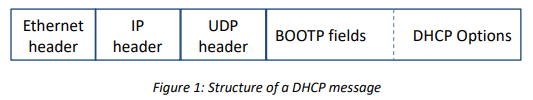
5. DHCP Requests will typically have a Client Identifier option. Look at the value of this option. How does it identify the client? Take a guess.

6. DHCP Acks will typically have a Server Identifier option. Look at the value of this option. How does it identify the server? Take a guess.

7. What option value stands for the Requested IP Address option? And for the IP Address Lease Time option?

8. How does the recipient of a DHCP message know that it has reached the last option?

Step 3: Answers to details of DHCP Messages



1. The two values are Boot Request (1) and Boot Reply (2).

2. The Transaction ID is 4 bytes long. Thus it is very unlikely that there will be collisions in a relatively small number of concurrent DHCP operations (until that number approaches 216!)

3. The “Your (client) IP address” field carries the IP address being leased to the client.

4. The option value of 53 stands for DHCP Message Type.

5. It is typical for the Client Identifier to carry the Ethernet address of the client, but possible to use some other kind of identifier (e.g., hostname, serial number).

6. It is typical for the Server Identifier to carry the IP address of the DHCP server, but possible to use some other kind of identifier.

7. The option value of 50 stands for Requested IP Address and the value of 51 stands for IP Address Lease Time.

8. The end of the DHCP options is identified with a DHCP option called End with value 255.

Step 4: DHCP Message Addressing

Now we will look at how DHCP messages are addressed to computers at the UDP, IP and Ethernet layers.This is interesting because DHCP is used to assign IP addresses – a computer requesting a DHCP address may neither have its own IP address nor know the IP address of the DHCP server.

Start by selecting a DHCP Request packet and looking at its UDP details in the middle Wireshark panel.We will only look at the DHCP Request message to keep things simple, as the details of addressing differfor other DHCP messages. Answers on next page.

1. What port number does the DHCP client use, and what port number does the DHCP server use?Ports matter because UDP messages are addressed using ports. Both of these port numbers areon the Request in the source and destination port fields (and you will also see them on the Ack).

Now look at the IP addresses in the IP protocol header of the packet for the next question. Do not look inside the BOOTP fields for the DHCP parameters, as we care about how DHCP messages are addressed at lower protocol layers. When the request is sent, your computer has no IP address and may not even know the IP address of the DHCP server, so the IP addressing differs from a routine IP packet.

2. What source IP address is put on the Request message? It is a special value meaning “this host on this network” used for initialization.

3. What destination IP address is put on the Request message? It is also a reserved value designed to reach the DHCP server wherever it is on the local network.

Look at the Ethernet addresses for the next question.

4. What source Ethernet address is put on the Request message, and what destination Ethernet address is put on the Request message? One of these addresses is a reserved address.

Looking at the addressing should help you to understand why your computer may record the DHCP traffic of other local computers in your trace. Since the IP addressing is not yet established, many DHCP messages are sent to all computers on the local network. This makes sure every computer receives DHCP messages intended for them, but it poses a difficulty: one computer may receive DHCP messages intended for another computer.

5. How does a computer work out whether a DHCP message it receives is intended as a reply to its DHCP Request message, and not a reply to another computer? Hint: if you are not sure then go over the fields you inspected previously in Step 2 above.

Step 4: Answers to DHCP Message Addressing

1. The DHCP client (your computer) uses UDP port 68 and the DHCP server uses UDP port 67.

2. The source IP address is 0.0.0.0. It is a special address used during address initialization.

3. The destination IP address is 255.255.255.255. It is the broadcast address, which means the message is intended for all computers on the network. (It is not possible to use a more restricted subnet broadcast, e.g., 192.168.255.255, as the subnet mask is not yet known by the client.)

4. The source Ethernet address is simply your own computer’s Ethernet address, since that is already assigned to your NIC. The destination Ethernet address is ff:ff:ff:ff:ff:ff, the reserved broadcast Ethernet address, so that the packet reaches all computers on the local network.

5. The DHCP messages in a single exchange carry the same Transaction ID. Thus a computer looks for a DHCP reply such as an Ack with a Transaction ID that matches the value it placed on the earlier DHCP message such as a Request. (This is in addition to any Ethernet address filtering: if the reply is unicast then it will have the computer’s Ethernet address as its destination.)

Conclusion: - Hence we have studied DHCP Server Configuration.

**Experiment 10**

Aim: - Socket programming for UDP and TCP.

Theory: -

**ALGORITHM:**

**Server**

1. Create a server socket and bind it to port.

2. Listen for new connection and when a connection arrives, accept it.

3. Send server‟s date and time to the client.

4. Read client‟s IP address sent by the client.

5. Display the client details.

6. Repeat steps 2-5 until the server is terminated.

7. Close all streams.

8. Close the server socket.

9. Stop.

**Client**

1. Create a client socket and connect it to the server‟s port number.

2. Retrieve its own IP address using built-in function.

3. Send its address to the server.

4. Display the date & time sent by the server.

5. Close the input and output streams.

6. Close the client socket. 7. Stop.

**PROGRAM: //TCP Date Server--tcpdateserver.java**

import java.net.\*;

import java.io.\*;

import java.util.\*;

class tcpdateserver {

public static void main (String arg[])

{

ServerSocket ss = null;

Socket cs;

PrintStream ps;

BufferedReader dis;

String inet;

try {

ss = new ServerSocket(4444);

System.out.println("Press Ctrl+C to quit");

while(true)

{

cs = ss.accept();

ps = new PrintStream(cs.getOutputStream());

Date d = new Date();

ps.println(d);

dis = new BufferedReader(new InputStreamReader(cs.getInputStream()));

inet = dis.readLine();

System.out.println("Client System/IP address is :"+ inet);

ps.close(); dis.close(); } }

catch(IOException e)

{ System.out.println("The exception is :" + e); } } }

**// TCP Date Client--tcpdateclient.java**

import java.net.\*;

import java.io.\*;

class tcpdateclient

{ public static void main (String args[])

{ Socket soc;

BufferedReader dis;

String sdate;

PrintStream ps;

try { InetAddress ia = InetAddress.getLocalHost();

if (args.length == 0) soc = new Socket(InetAddress.getLocalHost(),4444);

else soc = new Socket(InetAddress.getByName(args[0]),4444);

dis = new BufferedReader(new InputStreamReader(soc.getInputStream())); sdate=dis.readLine();

System.out.println("The date/time on server is : " +sdate);

ps = new PrintStream(soc.getOutputStream());

ps.println(ia);

ps.close();

catch(IOException e)

{ System.out.println("THE EXCEPTION is :" + e);

} } }

**OUTPUT Server:**

$ javac tcpdateserver.java

$ java tcpdateserver

Press Ctrl+C to quit

Client System/IP address is : localhost.localdomain/127.0.0.1

**Client:**

$ javac tcpdateclient.java

$ java tcpdateclient

The date/time on server is: Wed Jul 06 07:12:03 GMT 2011

Every time when a client connects to the server, server‟s date/time will be returned to the client for synchronization.

Conclusion.

Thus the program for implementing to display date and time from client to server using TCP Sockets was executed successfully and output verified using various samples. We implemented a Socket Server and Client program for TCP

**Socket programming for UDP**

Theory: -

UDP is a connectionless protocol and the socket is created for client and server to transfer the data. Socket connection is achieved using the port number. Domain Name System is the naming convention that divides the Internet into logical domains identified in Internet Protocol version 4 (IPv4) as a 32-bit portion of the total address.

**ALGORITHM:**

**Server**

1. Create two ports, server port and client port.

2. Create a datagram socket and bind it to client port.

3. Create a datagram packet to receive client message.

4. Wait for client's data and accept it.

5. Read Client's message.

6. Get data from user.

7. Create a datagram packet and send message through server port.

8. Repeat steps 3-7 until the client has something to send.

9. Close the server socket.

10. Stop.

**Client**

1. Create two ports, server port and client port.

2. Create a datagram socket and bind it to server port.

3. Get data from user.

4. Create a datagram packet and send data with server ip address and client port.

5. Create a datagram packet to receive server message.

6. Read server's response and display it.

7. Repeat steps 3-6 until there is some text to send.

8. Close the client socket.

9. Stop.

**PROGRAM // UDP Chat Server--udpchatserver.java**

import java.io.\*;

import java.net.\*;

class udpchatserver {

public static int clientport = 8040,serverport = 8050;

public static void main(String args[]) throws Exception {

DatagramSocket SrvSoc = new DatagramSocket(clientport);

byte[] SData = new byte[1024];

BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); System.out.println("Server Ready");

while (true) {

byte[] RData = new byte[1024];

DatagramPacket RPack = new DatagramPacket(RData,RData.length); SrvSoc.receive(RPack);

String Text = new String(RPack.getData());

if (Text.trim().length() == 0) break;

System.out.println("\nFrom Client <<< " + Text );

System.out.print("Msg to Cleint : " );

String srvmsg = br.readLine();

InetAddress IPAddr = RPack.getAddress();

SData = srvmsg.getBytes();

DatagramPacket SPack = new DatagramPacket(SData,SData.length,IPAddr, serverport); SrvSoc.send(SPack); }

System.out.println("\nClient Quits\n"); SrvSoc.close(); } }

**// UDP Chat Client--udpchatclient.java**

import java.io.\*;

import java.net.\*;

class udpchatclient {

public static int clientport = 8040,serverport = 8050;

public static void main(String args[]) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader (System.in)); DatagramSocket CliSoc = new DatagramSocket(serverport);

InetAddress IPAddr;

String Text;

if (args.length == 0)

IPAddr = InetAddress.getLocalHost();

else IPAddr = InetAddress.getByName(args[0]);

byte[] SData = new byte[1024];

System.out.println("Press Enter without text to quit");

while (true)

{ System.out.print("\nEnter text for server : ");

Text = br.readLine();

SData = Text.getBytes();

DatagramPacket SPack = new DatagramPacket(SData,SData.length, IPAddr, clientport ); CliSoc.send(SPack);

if (Text.trim().length() == 0) break;

byte[] RData = new byte[1024];

DatagramPacket RPack = new DatagramPacket(RData,RData.length); CliSoc.receive(RPack);

String Echo = new String(RPack.getData()) ;

Echo = Echo.trim();

System.out.println("From Server <<< " + Echo); }

CliSoc.close(); } }

**OUTPUT**

**Server**

$ javac udpchatserver.java

$ java udpchatserver

Server Ready

From Client <<< are u the SERVER

Msg to Cleint : yes

From Client <<< what do u have to serve

Msg to Cleint : no eatables

Client Quits

**Client**

$ javac udpchatclient.java

$ java udpchatclient

Press Enter without text to quit

Enter text for server : are u the SERVER

From Server <<< yes

Enter text for server : what do u have to serve

From Server <<< no eatables

Enter text for server : Ok

**Conclusion: -**Thus both the client and server exchange data using UDP sockets. We implemented a Socket Server and Client program for UDP.

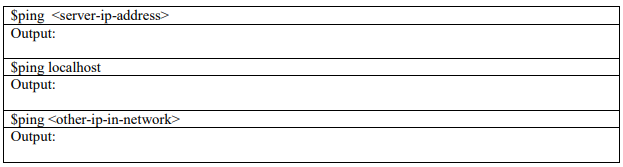
**Experiment 11**

Aim: - Study and Implement basic networking commands in Linux (ipconfig,ipconfig/all,arp/a,netstat,netstat/an,ping,pathping, nslookup)

Theory: -

Execute the following commands and write their output

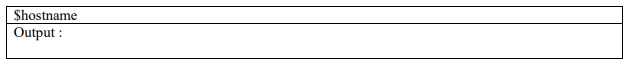
**1. ping:** This command is used to test connectivity between two nodes. Ping use ICMP (Internet Control Message Protocol) to communicate to other devices. You can ping host name or ip address using below command. example: ping 201.54.100.1 or ping www.google.com



**2. hostname**

Gives the host name of the computer they are logged into. To set the hostname permanently

use /etc/sysconfig/network file.



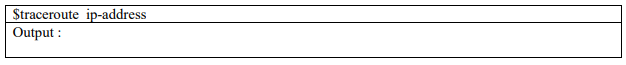
**3. traceroute**

traceroute is a network troubleshooting utility which shows number of hops taken to reach

destination also determine packets traveling path.

**4. netstat**

Netstat (Network Statistic) command displays interfaces, connection information, routing table information etc.



Execute it with the following options and write the output:

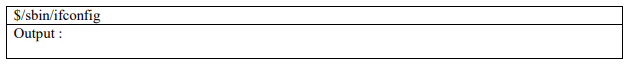
netstat –t

netstat –s –t

netstat –i

**5. ifconfig**

ifconfig is used for displaying network interface information.



**6. who**

Displays information of all users who are logged in



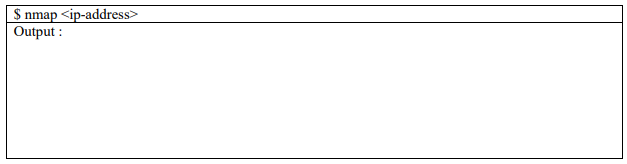
**7. whoami**

The whoami command writes the user name (i.e., login name) of the owner of the current login session to standard output.



**8. nmap**

Network mapper tool to discover hosts and services on a computer network.



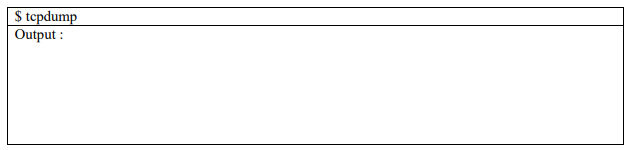


**9. tcpdump**

Tcpdump prints out a description of the contents of packets on a network interface that match the boolean expression; the description is preceded by a time stamp, printed, by default, as hours, minutes, seconds, and fractions of a second since midnight. Sample output for ARP protocol:

arp who-has 128.3.254.6 tell 128.3.254.68

arp reply 128.3.254.6 is-at 02:07:01:00:01:c4



Conclusion: - Hence we have studied the basic networking commands and observed the output.

**Experiment 12**

Aim: - Case Study: Study the detailed Networking Structure in SVKM-IOT Campus.

Theory: -

Find out information about the network in your lab and fill in details below:

* 1. Total Number of computers in your lab:
  2. Find details of any 5 computers:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | MAC address | IP address | LAN speed | Default  mask | hostname |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |

* 1. Are the IP addresses assigned to the machines statically or dynamically?
  2. Does the network have a DHCP server?
  3. If yes, what is the address of the server ?
  4. How many servers are configured? :

Details of servers:

|  |  |  |  |
| --- | --- | --- | --- |
|  | IP address | MAC address | Purpose |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |

* 1. Cables
     1. Type :
     2. Is it coaxial / twisted pair or fiber optic cable ?
     3. Cable bandwidth
     4. Maximum cable length limit
     5. Connector used
  2. Switches:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Company Name | MAC address | No. of ports | Managed / Unmanaged | IP’s of Machines  connected to the switch |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |

* 1. Routers:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Company  Name | No. / Types of  ports | Port speed | IP address |  |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |

1. Is there wi-fi capability in the LAN? If yes,
2. What is the Wi-fi access point address?
3. How many devices / IP addresses does it support?
4. What is the bandwidth? If no,
5. What additional devices are needed?
6. Where will you connect them?
7. What will be its IP address?
8. Is there internet access in the lab?

If not, what changes to the hardware / software must be made ?

If yes, what is the IP address of the router / gateway ?

1. Draw the Network Topology (show how machines and servers are connected using connectivity devices)
2. If 20 more machines have to be added to the network, what changes must be made to the network?
3. If the network is to be divided into four subnetworks having 50 machines each, give a plan to do so. What additional devices will be needed? Give the IP address of each subnetwork and the address ranges for hosts in each subnetwork.

Conclusion: - Hence we have studied the Networking Structure in SVKM-IOT Campus.