Computer Networks
CL3001

LAB - 05
SMTP & FTP
Advanced Wireshark

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES,
KARACHI CAMPUS
FAST SCHOOL OF COMPUTING (AI & DS, CS, CY, SE)
SPRING 2025

Computer Networks Lab 05

Course: Computer Networks (CL3001) **Semester:** Spring 2025
Instructor: Sandesh Kumar **T.A:** N/A

Note:

- Maintain discipline during the lab.
- Listen and follow the instructions as they are given.
- Just raise hand if you have any problem.
- Completing all tasks of each lab is compulsory.
- Get your lab checked at the end of the session.

Lab Objective

- Introduction to SMTP & FTP in Cisco Packet Tracer.
- Network traffic analysis of HTTP/HTTPS protocol headers, cookies using Wireshark.
- Network traffic analysis of DNS using Wireshark.
- Network traffic analysis of TCP protocol using Wireshark
- Network traffic analysis of UDP protocol using Wireshark

SMTP

1. Introduction to SMTP

Simple Mail Transfer Protocol (SMTP) is an Internet standard for electronic mail (email) transmission. First defined by RFC 821in1982, it was last updated in 2008with Extended SMTP additions by RFC 5321, which is the protocol in widespread use today. Although electronic mail servers and other mail transfer agents use SMTP to send and receive mail messages, user-level client mail applications typically use SMTP only for sending messages to a mail server for relaying. For retrieving messages, client applications usually use either IMAP or POP3.

SMTP communication between mail servers uses port 25. Mail clients on the other hand, often submit the outgoing emails to a mail server on port 587. Despite being deprecated, mail providers sometimes still permit the use of nonstandard port 465 for this purpose.

SMTP runs over TCP.

Implementation:

Topology:

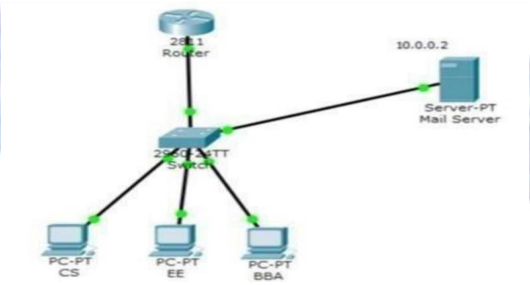

Figure 1

Configure and Verify Email Services:

- Click on Mail server.
- Go to services & then email services.
- Enable SMTP & POP3 Service.
- Set Domain name e.g. fast.com.
- Add following users:

Usernames	Passwords
Cs	**123**
Ee	**456**
Bba	**789**

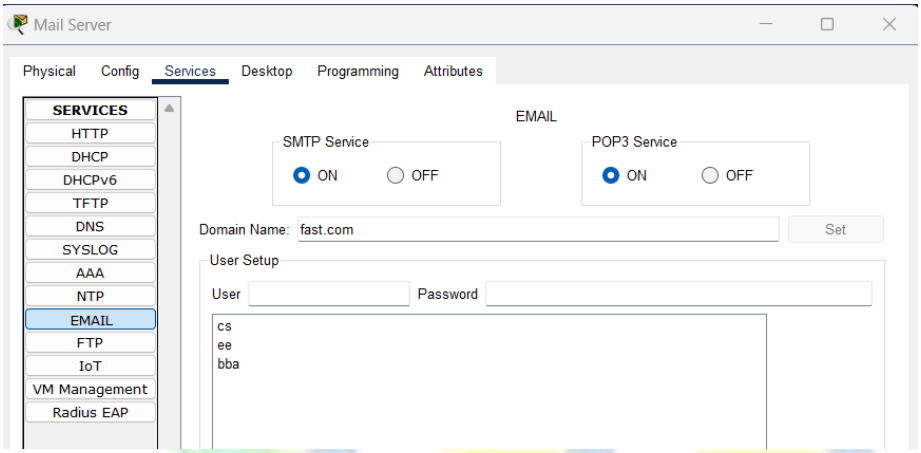

Figure 2

Now configure user email account.
Go to PC → Desktop → Email.
Fill the following fields as shown below.
Click "Save" to save the configurations and do the same for EE and BBA.

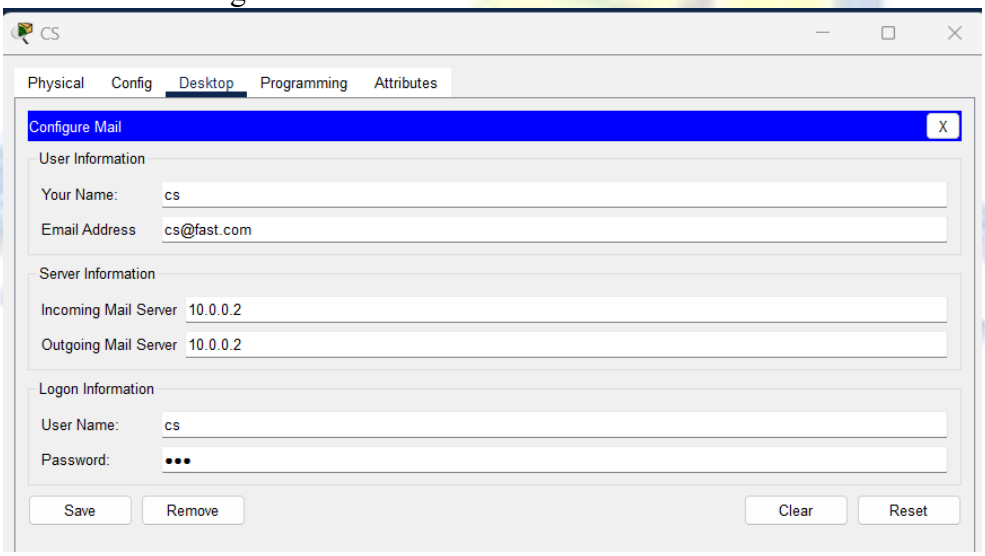

Figure 3

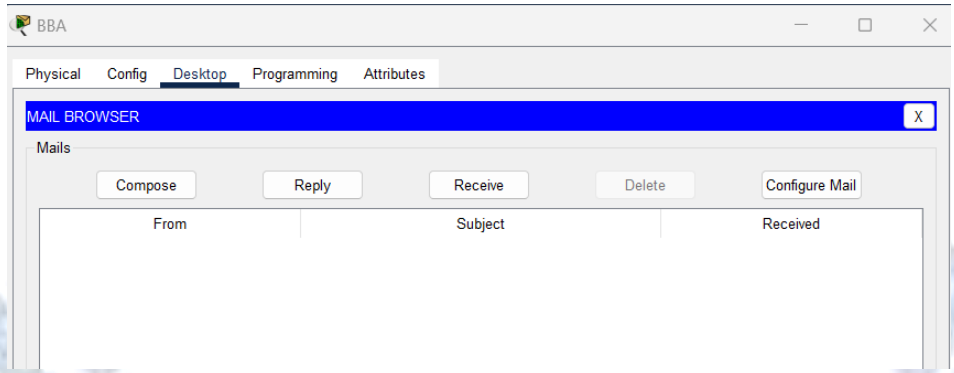

Figure 4

Now compose email to be sent to cs@fast.com from bba@fast.com

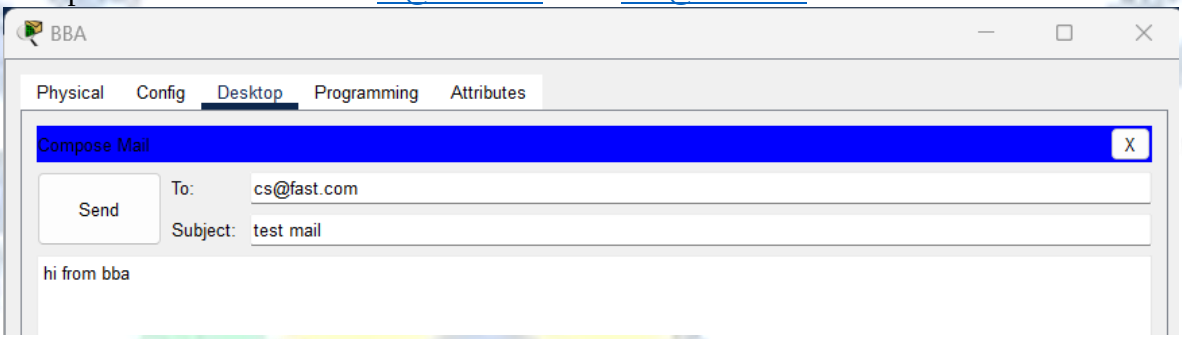

Figure 5

Click on "Send" to send Email.

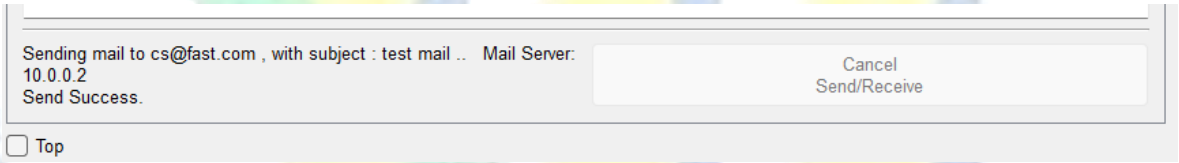

Figure 6

Now open email again on cs and click receive to see the email received from bba.

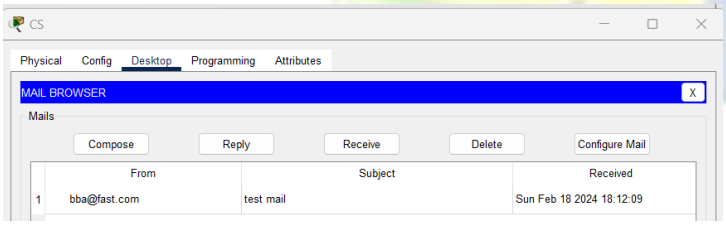

Figure 7

Simulation:

To note POP 3 header format information, go to simulation mode edit filters & check SMTP & POP 3 boxes.

After that click on capture/forward button. Now see how mail server works.

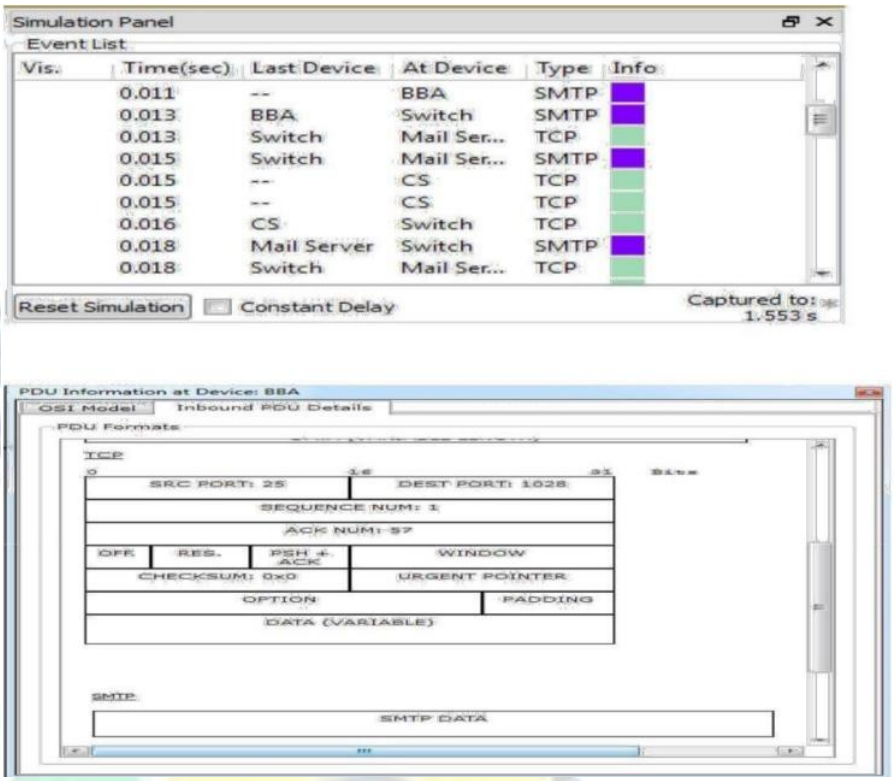

Figure 8

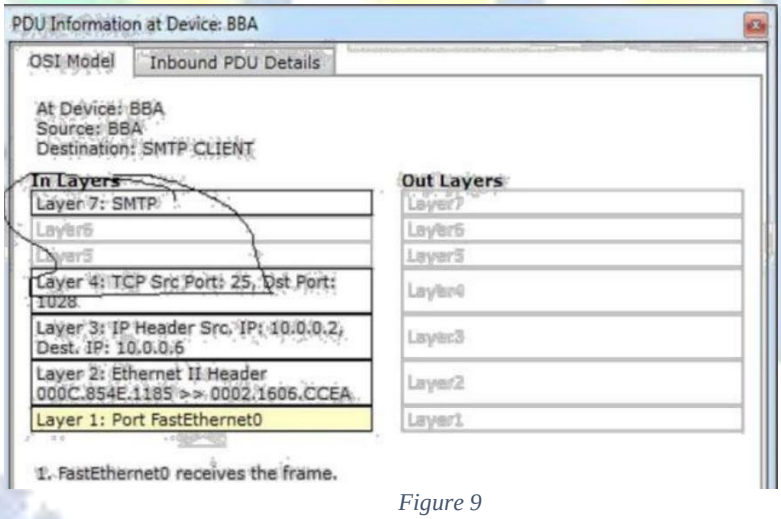

Figure 9

FTP

1. Introduction to FTP

The File Transfer Protocol (FTP) is a standard network protocol used to transfer computer files between a client and server on a computer network. FTP is built on client-server model architecture and uses separate control and data connections between the client and the server. FTP users may authenticate themselves with a clear-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it. For secure transmission that protects the username and password, and encrypts the content, FTP is often secured with SSL/TLS (FTPS) or replaced with SSH File Transfer Protocol (SFTP). FTP uses TCP asits under layer transport protocol for data reliability transfer. It uses port 21.

FTP may run in active or passive mode, which determines how the data connection is established.

- In active mode, the client starts listening for incoming data connections from the server on port M. It sends the FTP command PORT M to inform the server on which port it is listening. The server then initiates a data channel to the client from its port 20, the FTP server data port.
- In situations where the client is behind a firewall and unable to accept incoming TCP connections, passive mode may be used. In this mode, the client uses the control connection to send a PASV command to the server and then receives a server IP address and server port number from the server, which the client then uses to open a data connection from an arbitrary client port to the server IP address and server port number received.

Both modes were updated in September 1998 to support IPv6. Further changes were introduced to the passive mode at that time, updating it to extended passive mode.

Implementation:

In this activity, you will configure FTP server in Cisco Packet Tracer. After configuration you will transfer files between client & server. This activity is divided into 3 parts.

First Construct the topology & repeat all essential steps which we are done in pervious section.

Topology:

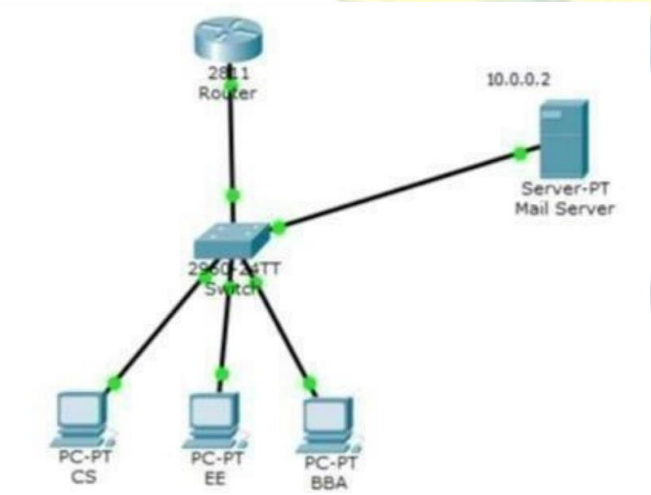

Figure 10

Part I: Configure FTP services on server

a) Click server > Config tab > FTP.

b) Click On to enable FTP service.

c) In User Setup, create the following user accounts. Click the + button to add the account:

Username	Password	Permissions
Fast	1234	Limited to Read, write & list.

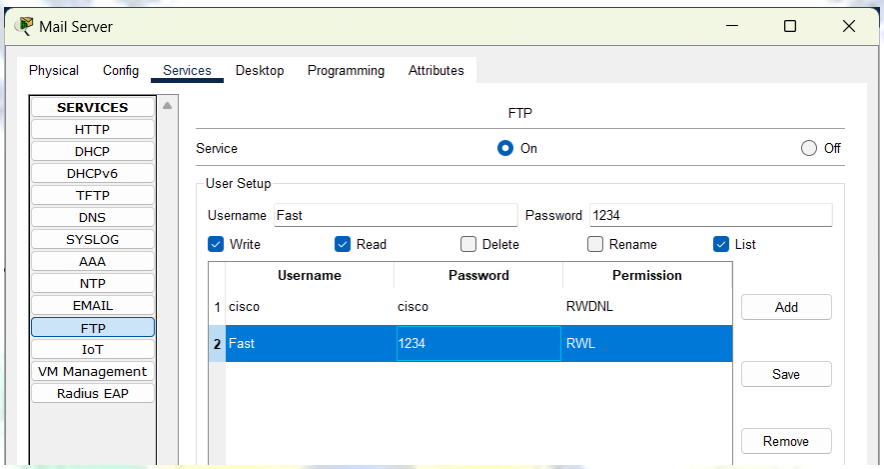

Figure 11

Now go to any PC Desktop command prompt. Connect with the FTP server using username & password assigned to FTP server.

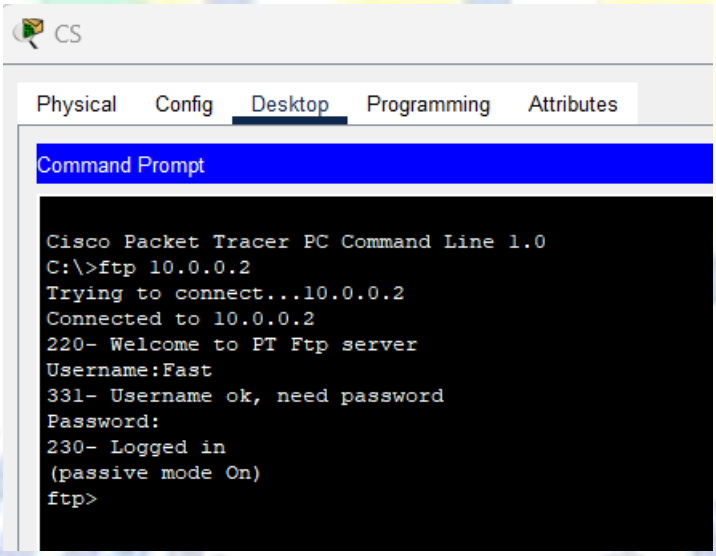

Figure 12

Part II: Upload file(s) to the FTP server

Go to PC Desktop text editor create file named test.bin

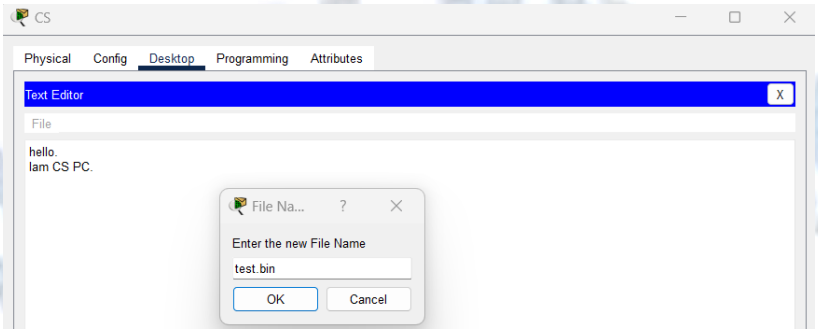

Figure 13

After creating the file go to PC Desktop command prompt and write the following command to transfer file from PC to FTP server.

put test.bin

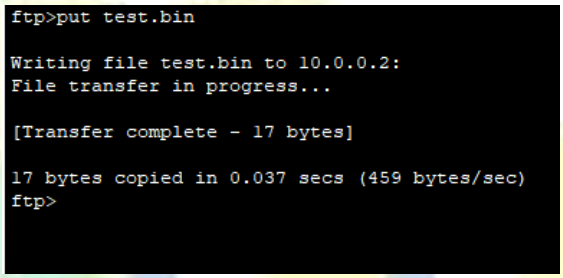

Figure 14

Part III: Upload file(s) to the FTP server

Now go to other PC desktop command prompt. Establish connection with FTP server and then write the dir command to see the files in FTP server.

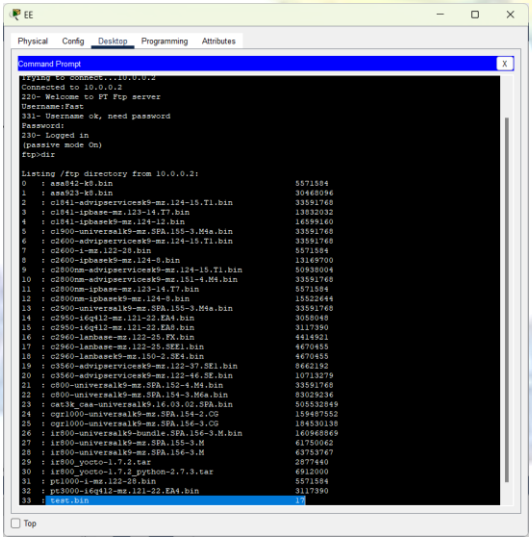

Figure 15

Simulation:

Select the simulation mode. Go to PC desktop command prompt again make connection with FTP server using its IP address.

Figure 16

Now to note the FTP header format information go to simulation mode edit filters and click on FTP check box then click on capture/forward button.

How FTP server resolves the login request.

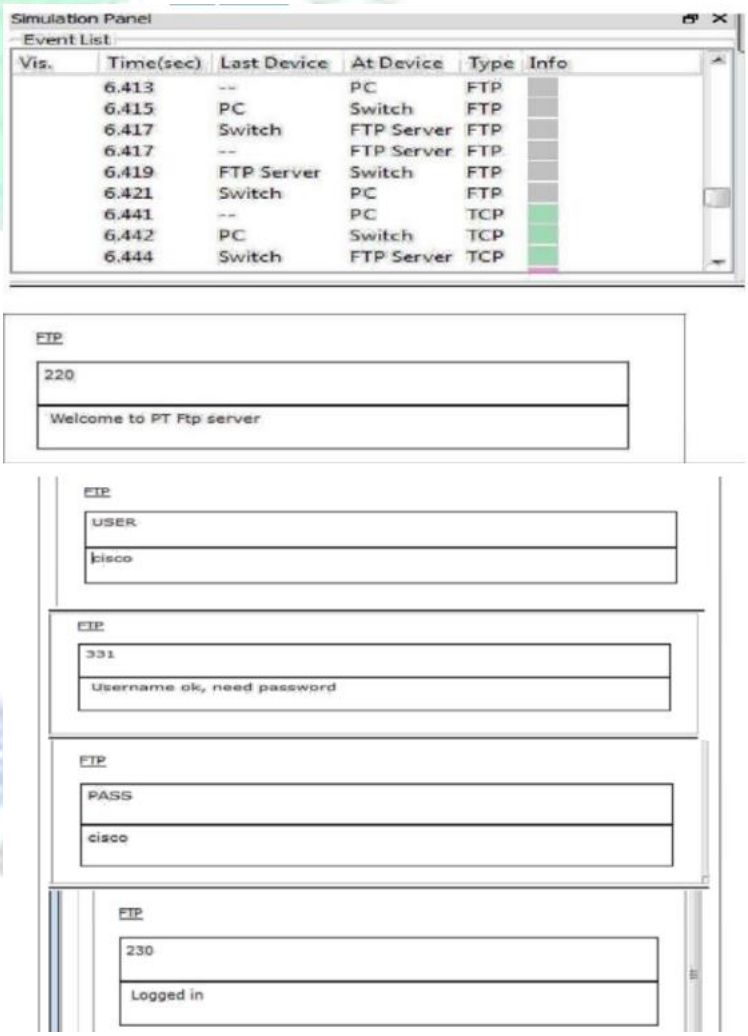

Figure 17

Now click on the FTP packet, you can note that the destination port is 21.

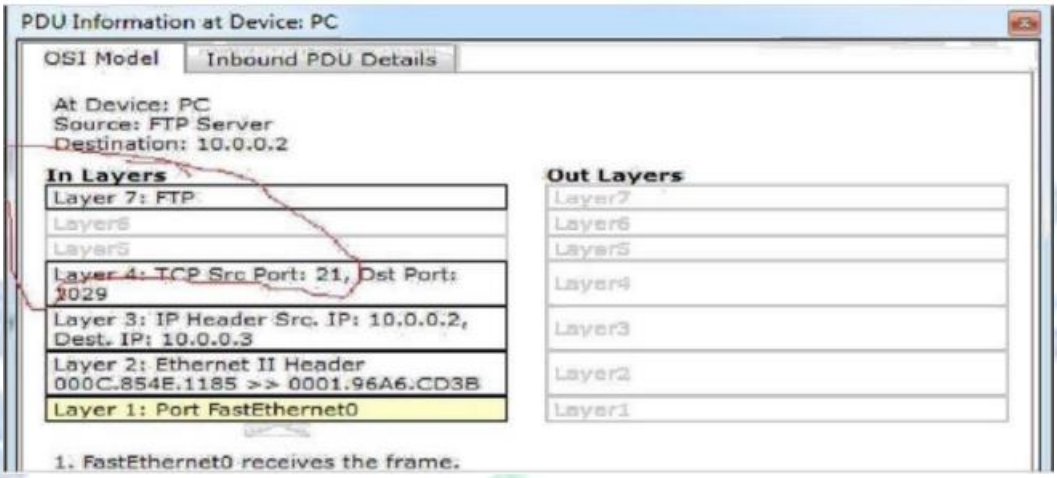

Figure 18

Now scroll the Outbound PDU Details, you can see the FTP PDU

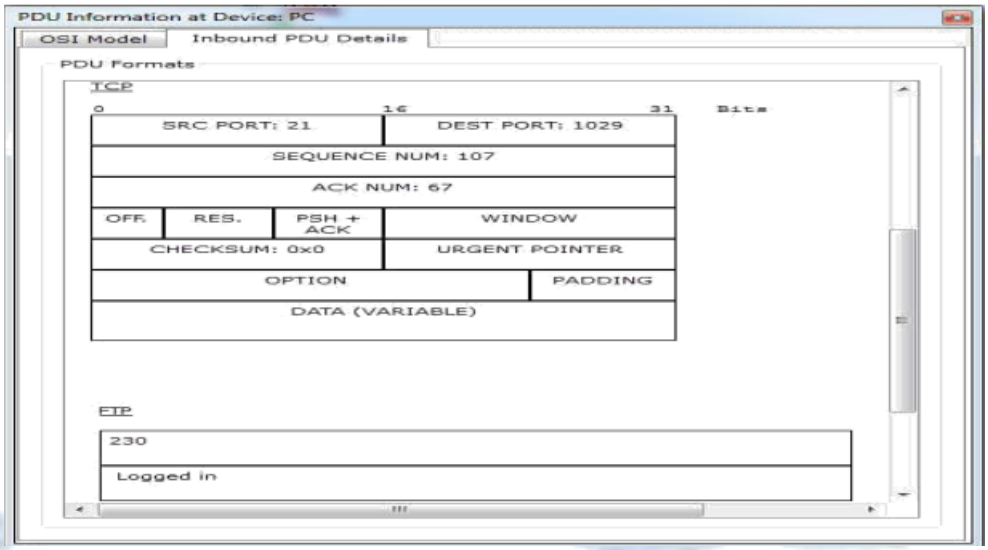

Figure 19

MORE ON WIRESHARK

1. The Basic HTTP GET/response interaction

Let's begin our exploration of HTTP by downloading a very simple HTML file - one that is very short, and contains no embedded objects.

Do the following:

1. Start up your browser.
2. Start up the Wireshark packet sniffer, as described in the Introductory lab (but don't yet begin packet capture). Enter "http" (just the letters, not the quotation marks) in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window. (We're only interested in the HTTP protocol here, and don't want to see the clutter of all captured packets).
3. Wait a bit more than one minute (we'll see why shortly), and then begin Wireshark packet capture.
4. Enter the following to your browser:
   http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html
   Your browser should display the very simple, one-line HTML file.

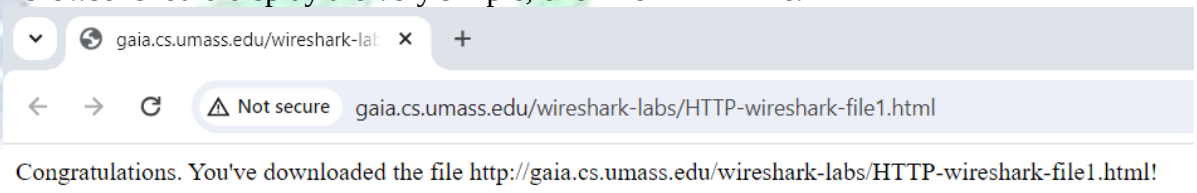

Figure 20

5. Stop Wireshark packet capture.

Your Wireshark window should look similar to the window shown below. If you are unable to run Wireshark on a live network connection, you can download a packet trace that was created when the steps above were followed properly.

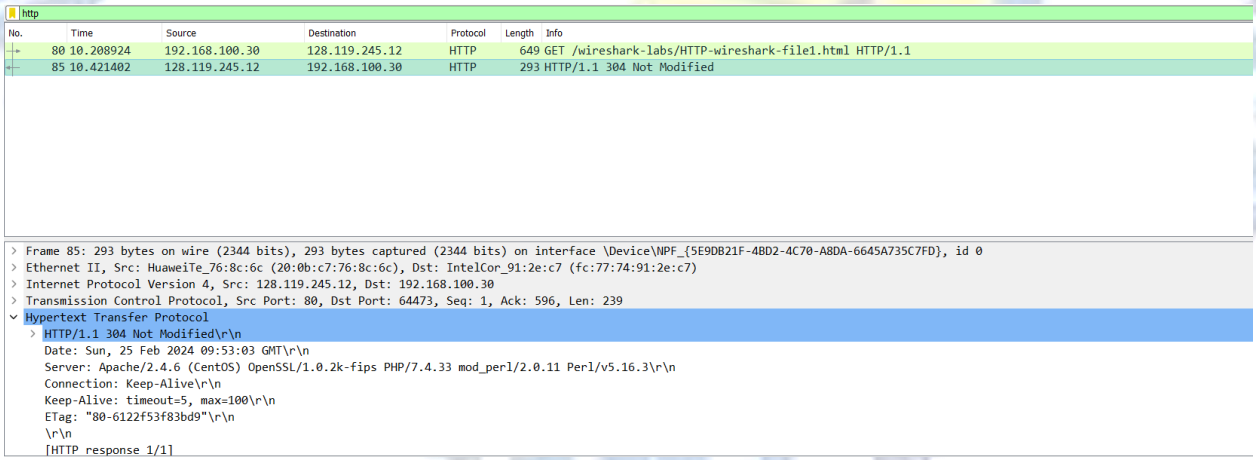

Figure 21

The example in the above picture shows in the packet-listing window that two HTTP messages were captured: the GET message (from your browser to the gaia.cs.umass.edu web server) and the response message from the server to your browser. The packet-contents window shows details of the selected message (in this case the HTTP OK message, which is highlighted in the packet-listing window). Recall that since the HTTP message was carried inside a TCP segment, which was carried inside an IP datagram, which was carried within an Ethernet frame, Wireshark displays the Frame, Ethernet, IP, and TCP packet information as well. We want to minimize the amount of non-HTTP data displayed (we're interested in HTTP here, and will be investigating these other protocols in later labs), so make sure the boxes at the far left of the Frame, Ethernet, IP and TCP information have a plus sign or a right-pointing triangle (which means there is hidden, undisplayed information), and the HTTP line has a minus sign or a down pointing triangle (which means that all information about the HTTP message is displayed).

Download the zip file:
http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip
Extract the file: **http-ethereal-trace-1**. The traces in this zip file were collected by Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark lab. Once you have downloaded the trace, you can load it into Wireshark and view the trace using the File pull down menu, choosing Open, and then selecting the http-ethereal-trace-1 trace file. The resulting display should look similar to previous.
(The Wireshark user interface displays just a bit differently on different operating systems, and in different versions of Wireshark).

(**Note**: You should ignore any HTTP GET and response for favicon.ico. If you see a reference to this file, it is your browser automatically asking the server if it (the server) has a small icon file that should be displayed next to the displayed URL in your browser. We'll ignore references to this pesky file in this lab.).
By looking at the information in the HTTP GET and response messages, answer the following questions. When answering the following questions, you should print out the GET and response messages (see the introductory Wireshark lab for an explanation of how to do this) and indicate where in the message you've found the information that answers the following questions. When you hand in your assignment, annotate the output so that it's clear where in the output you're getting the information for your answer (e.g., for our classes, we ask that students markup paper copies with a pen, or annotate electronic copies with text in a colored font).

1) Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?
2) What languages (if any) does your browser indicate that it can accept to the server?
3) What is the IP address of your computer? Of the gaia.cs.umass.edu server?
4) What is the status code returned from the server to your browser?
5) When was the HTML file that you are retrieving last modified at the server?
6) How many bytes of content are being returned to your browser?

In your answer to question 5 above, you might have been surprised to find that the document you just retrieved was last modified within a minute before you downloaded the document. That's because (for this particular file), the gaia.cs.umass.edu server is setting the file's last-modified time to be the current time, and is doing so once per minute. Thus, if you wait a minute between accesses, the file will appear to have been recently modified, and hence your browser will download a "new" copy of the document.

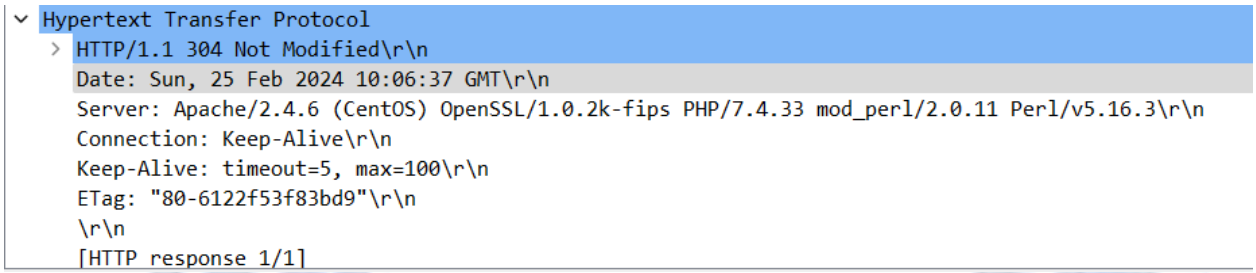

Figure 22

2. The HTTP Conditional GET/response interaction

Most web browsers perform object caching and thus perform a conditional GET when retrieving an HTTP object.

Before performing the steps below, make sure your browser's cache is empty. (To do this under Firefox, select Tools->Clear Recent History and check the Cache box, or for Internet Explorer, select Tools->Internet Options->Delete File; these actions will remove cached files from your browser's cache.)

Now do the following:

1. Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
2. Start up the Wireshark packet sniffer.
3. Enter the following URL into your browser:
   http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html
   Your browser should display a very simple five-line HTML file.

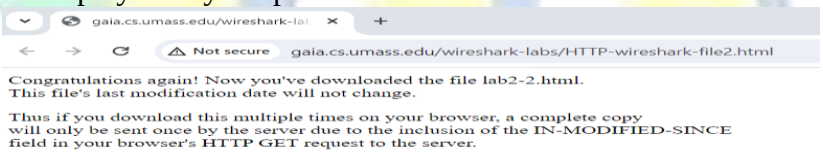

Figure 23

4. Quickly enter the same URL into your browser again (or simply select the refresh button on your browser).
5. Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

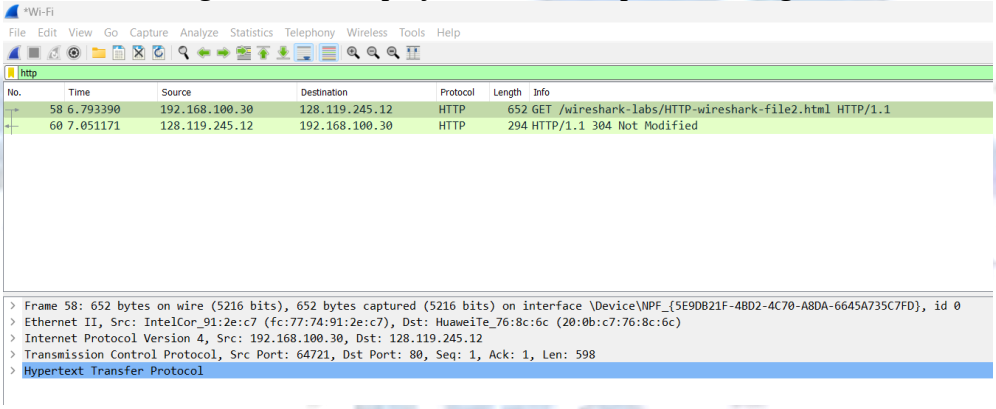

Figure 24

6. (**Note:** If you are unable to run Wireshark on a live network connection, you can use the http ethereal-trace-2 packet trace to answer the questions below; see footnote 1. This trace file was gathered while performing the steps above on one of the author's computers.).

Answer the following questions:

1) Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET?
2) Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?
3) Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE:" line in the HTTP GET? If so, what information follows the "IF-MODIFIED-SINCE:" header?
4) What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.

3. Retrieving Long Documents

In our examples thus far, the documents retrieved have been simple and short HTML files. Let's next see what happens when we download a long HTML file.
Do the following:

1. Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
2. Start up the Wireshark packet sniffer.
3. Enter the following URL into your browser:
   http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html
   Your browser should display the rather lengthy US Bill of Rights.

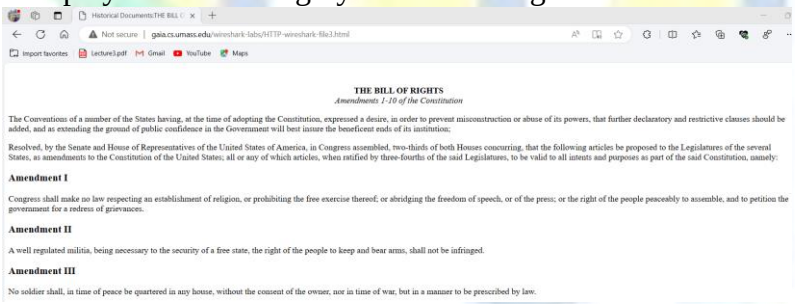

Figure 25

4. Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

5. (**Note**: If you are unable to run Wireshark on a live network connection, you can use the http-ethereal-trace-3 packet trace to answer the questions below; see footnote 1. This trace file was gathered while performing the steps above on one of the author's computers.)

   In the packet-listing window, you should see your HTTP GET message, followed by a multiple-packet TCP response to your HTTP GET request.
   This multiple-packet response deserves a bit of explanation.
   The HTTP response message consists of a status line, followed by header lines, followed by a blank line, followed by the entity body. In the case of our HTTP GET, the entity body in the response is

the entire requested HTML file. In our case here, the HTML file is rather long, and at 4500 bytes is too large to fit in one TCP packet. The single HTTP response message is thus broken into several pieces by TCP, with each piece being contained within a separate TCP segment. In recent versions of Wireshark, Wireshark indicates each TCP segment as a separate packet, and the fact that the single HTTP response was fragmented across multiple TCP packets is indicated by the "TCP segment of a reassembled PDU" in the Info column of the Wireshark display. Earlier versions of Wireshark used the "Continuation" phrase to indicated that the entire content of an HTTP message was broken across multiple TCP segments. We stress here that there is no "Continuation" message in HTTP!

Lab Exercise

1. How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill or Rights?
2. Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?
3. What is the status code and phrase in the response?
4. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

4. HTML Documents with Embedded Objects

Now that we've seen how Wireshark displays the captured packet traffic for large HTML files, we can look at what happens when your browser downloads a file with embedded objects, i.e., a file that includes other objects (in the example below, image files) that are stored on another server(s).
Do the following:

1. Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
2. Start up the Wireshark packet sniffer.
3. Enter the following URL into your browser
   http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html
   Your browser should display a short HTML file with two images.

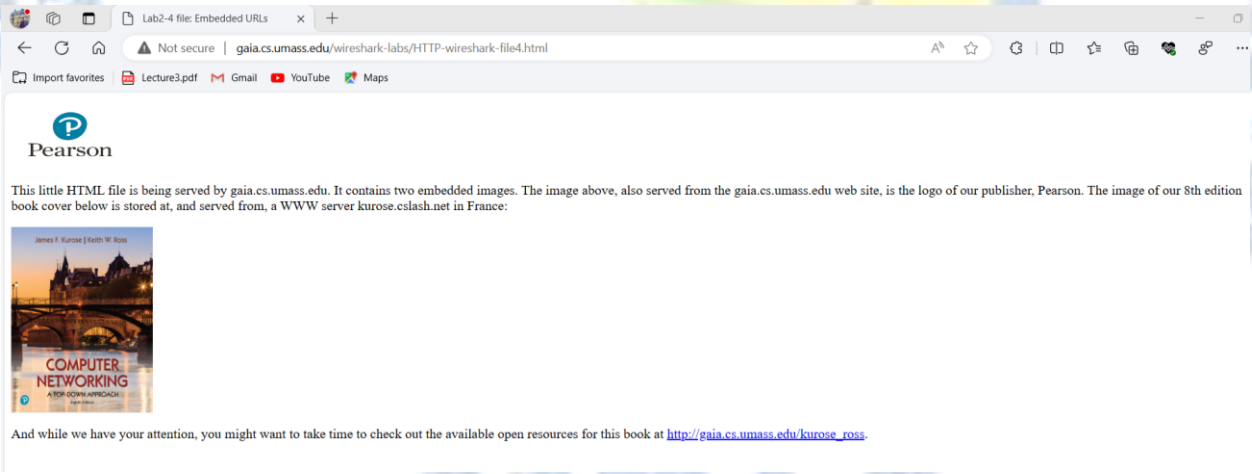

Figure 26

These two images are referenced in the base HTML file. That is, the images themselves are not contained in the HTML; instead the URLs for the images are contained in the downloaded HTML file. Your browser will have to retrieve these logos from the indicated web sites. Our publisher's logo is retrieved from the **gaia.cs.umass.edu web** site. The image of the cover for our 5th edition (one of our favorite covers) is stored at the **caite.cs.umass.edu server.** (These are two different web servers inside cs.umass.edu).

4. Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

(**Note:** If you are unable to run Wireshark on a live network connection, you can use the http-ethereal-trace-4 packet trace to answer the questions below; see footnote 1. This trace file was gathered while performing the steps above on one of the author's computers.)

Lab Exercise

1. How many HTTP GET request messages did your browser send? To which Internet addresses were these GET requests sent?
2. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.

5. HTTP Authentication

Finally, let's try visiting a web site that is password-protected and examine the sequence of HTTP message exchanged for such a site.
The URL **http://gaia.cs.umass.edu/wiresharklabs/protected_pages/HTTP-wireshark-file5.html** is password protected. The username is "wireshark-students" (without the quotes), and the password is "network" (again, without the quotes).
So let's access this "secure" password-protected site. Do the following:

1. Make sure your browser's cache is cleared, as discussed above, and close down your browser. Then, start up your browser.
2. Start up the Wireshark packet sniffer.
3. Enter the following URL into your browser:
   http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wiresharkfile5.html
   Type the requested user name and password into the pop up box:

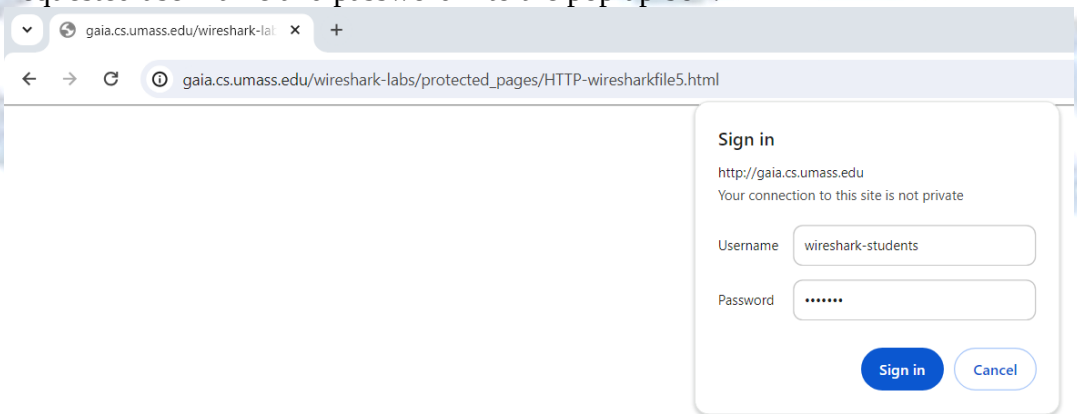

Figure 27

4. Stop Wireshark packet capture, and enter “http” in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

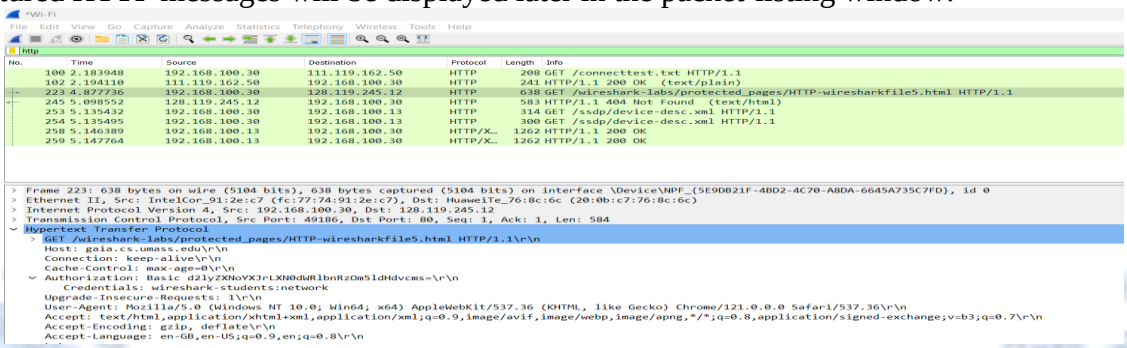

Figure 28

(**Note:** If you are unable to run Wireshark on a live network connection, you can use the http-ethereal-trace-5 packet trace to answer the questions below; see footnote 2. This trace file was gathered while performing the steps above on one of the author’s computers.)

Now let’s examine the Wireshark output. You might want to first read up on HTTP authentication by reviewing the easy-to-read material on “HTTP Access Authentication Framework” at http://frontier.userland.com/stories/storyReader$2159

Lab Exercise

1. What is the server’s response (status code and phrase) in response to the initial HTTP GET message from your browser?
2. When your browser’s sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?

6. Network Traffic Analysis of DNS Using Wireshark

Domain Name System (DNS) translates hostnames to IP addresses, fulfilling a critical role in the Internet infrastructure.

In this lab, we’ll take a closer look at the client side of DNS. Recall that the client’s role in the DNS is relatively simple – a client sends a query to its local DNS server, and receives a response back.

The hierarchical DNS servers communicate with each other to either recursively or iteratively resolve the client’s DNS query. From the DNS client’s standpoint, however, the protocol is quite simple – a query is formulated to the local DNS server and a response is received from that server.

1. nslookup

In this lab, we’ll make extensive use of the nslookup tool, which is available in most Linux/Unix and Microsoft platforms today.

To run nslookup in Linux/Unix, you just type the nslookup command on the command line. To run it in Windows, open the Command Prompt and run nslookup on the command line.

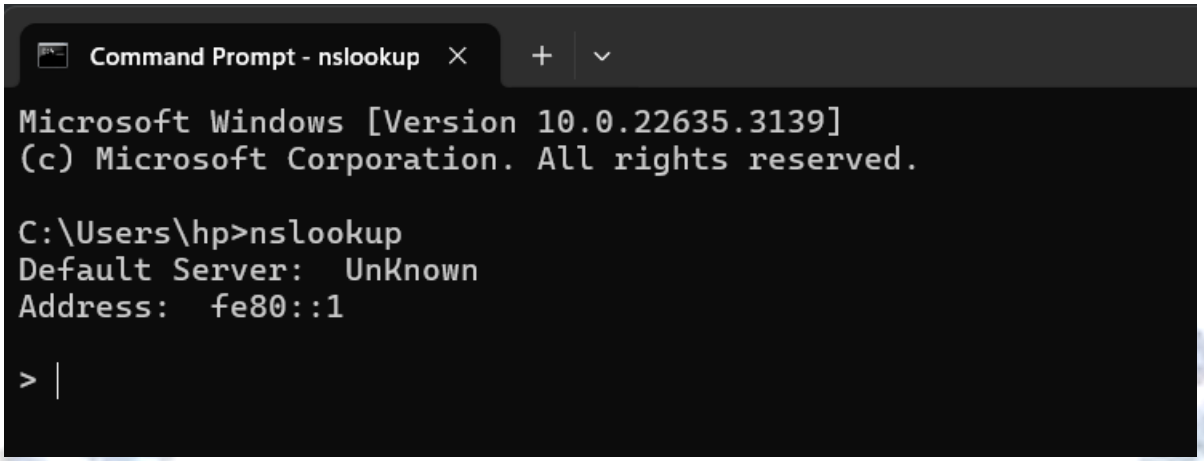

Figure 29

In it is most basic operation, nslookup tool allows the host running the tool to query any specified DNS server for a DNS record. The queried DNS server can be a root DNS server, a top-level-domain DNS server, an authoritative DNS server, or an intermediate DNS server (see the textbook for definitions of these terms).
To accomplish this task, nslookup sends a DNS query to the specified DNS server, receives a DNS reply from that same DNS server, and displays the result.

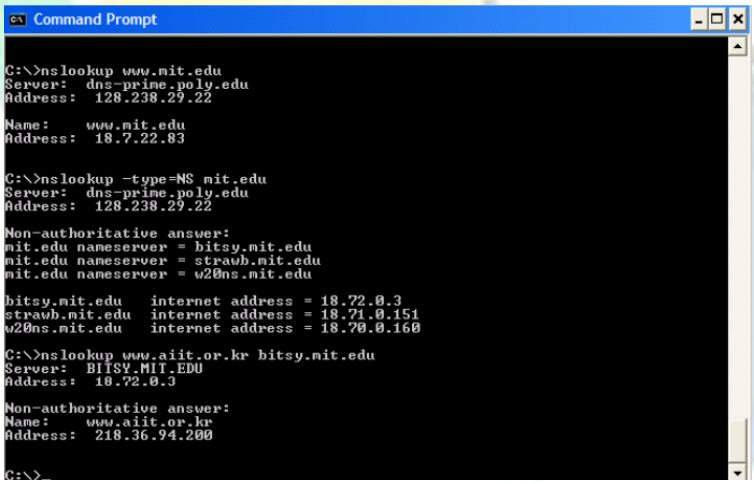

Figure 30

The above screenshot shows the results of three independent nslookup commands (displayed in the Windows Command Prompt).
In this example, the client host is located on the campus of Polytechnic University in Brooklyn, where the default local DNS server is dns-prime.poly.edu.
When running nslookup, if no DNS server is specified, then nslookup sends the query to the default DNS server, which in this case is dnsprime.poly.edu.
Consider the first command:

nslookup www.mit.edu

In words, this command is saying “please send me the IP address for the host www.mit.edu”.

As shown in the screenshot, the response from this command provides two pieces of information: (1) the name and IP address of the DNS server that provides the answer; and (2) the answer itself, which is the host name and IP address of www.mit.edu. Although the response came from the local DNS server at Polytechnic University, it is quite possible that this local DNS server iteratively contacted several other DNS servers to get the answer.
Now consider the second command:

nslookup –type=NS mit.edu

In this example, we have provided the option "-type=NS" and the domain "mit.edu". This causes nslookup to send a query for a type-NS record to the default local DNS server. In words, the query is saying, "please send me the host names of the authoritative DNS for mit.edu". (When the –type option is not used, nslookup uses the default, which is to query for type A records.) The answer, displayed in the above screenshot, first indicates the DNS server that is providing the answer (which is the default local DNS server) along with three MIT nameservers. Each of these servers is indeed an authoritative DNS server for the hosts on the MIT campus. However, nslookup also indicates that the answer is "non-authoritative," meaning that this answer came from the cache of some server rather than from an authoritative MIT DNS server. Finally, the answer also includes the IP addresses of the authoritative DNS servers at MIT.
(Even though the type-NS query generated by nslookup did not explicitly ask for the IP addresses, the local DNS server returned these "for free" and nslookup displays the result.).

Now that we have provided an overview of nslookup, it is time for you to test drive it yourself. Do the following (and write down the results):

1. Run nslookup to obtain the IP address of a Web server in Asia. What is the IP address of that server?
2. Run nslookup to determine the authoritative DNS servers for a university in Europe?
3. Run nslookup so that one of the DNS servers obtained in Question 2 is queried for the mail servers for Yahoo! mail. What is its IP address?

2. ipconfig

ipconfig (for Windows) and ifconfig (for Linux/Unix) are among the most useful little utilities in your host, especially for debugging network issues.
Here we'll only describe ipconfig, although the Linux/Unix ifconfig is very similar. ipconfig can be used to show your Current TCP/IP information, including your address, DNS server addresses, adapter type and so on.
For example, if you all this information about your host simply by entering:

ipconfig \all

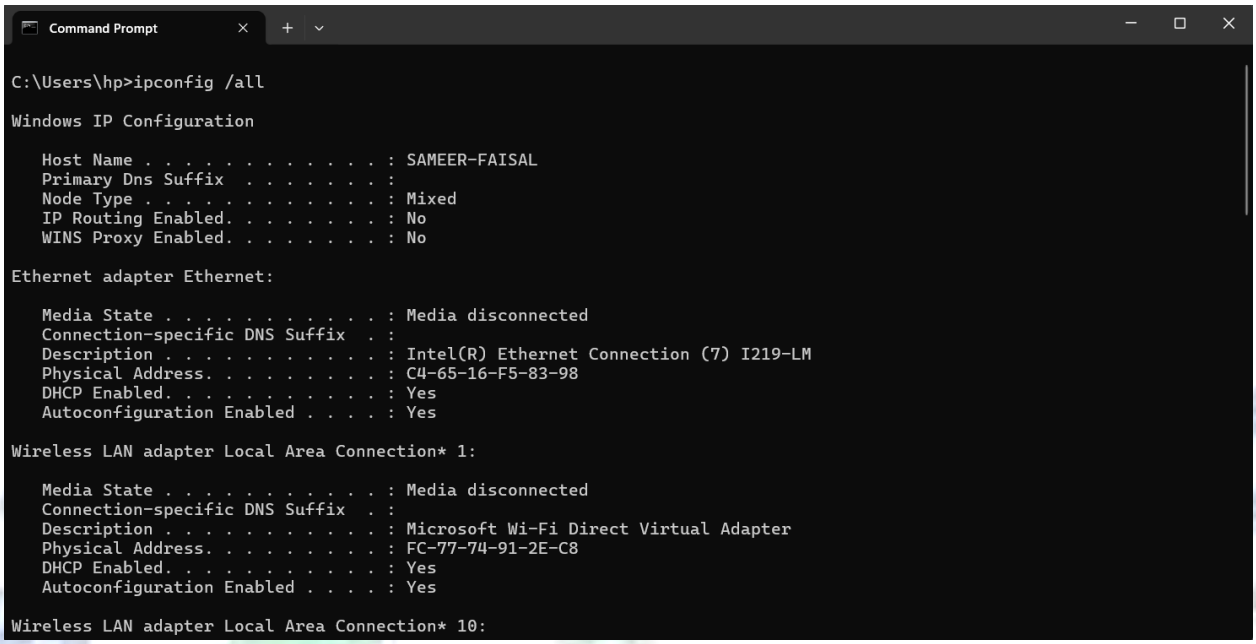

Figure 31

ipconfig is also very useful for managing the DNS information stored in your host. To see these cached records, after the prompt C:\> provide the following command:

ipconfig /displaydns

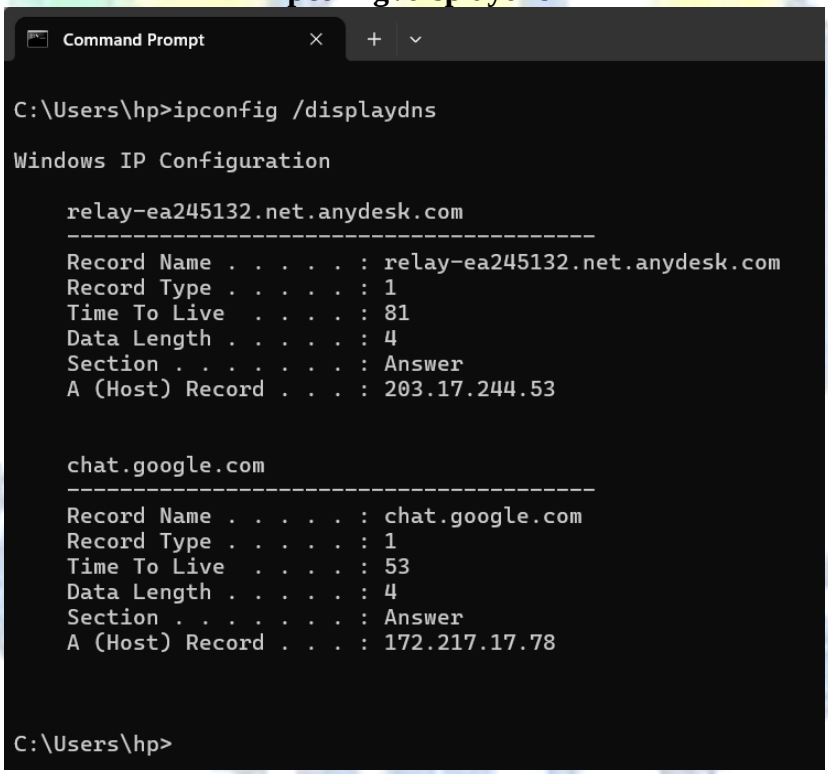

Figure 32

Each entry shows the remaining Time to Live (TTL) in seconds. To clear the cache, enter:

ipconfig /flushdns

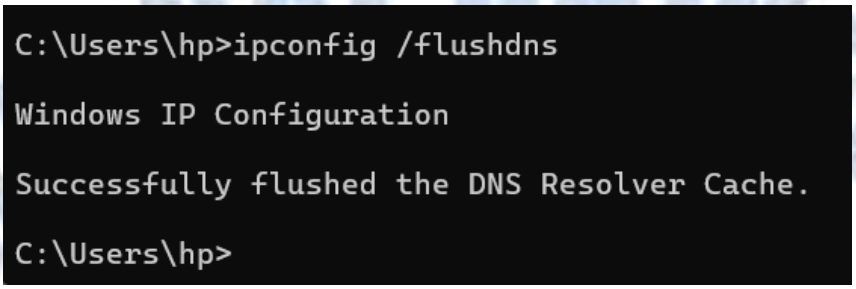

Figure 33

Flushing the DNS cache clears all entries and reloads the entries from the hosts file.

3. Tracing DNS with Wireshark

Now that we are familiar with nslookup and ipconfig, we're ready to get down to some serious business. Let's first capture the DNS packets that are generated by ordinary Websurfing activity.

1. Use ipconfig to empty the DNS cache in your host.
2. Open your browser and empty your browser cache. (With Internet Explorer, go to Tools menu and select Internet Options; then in the General tab select Delete Files.).
3. Open Wireshark and enter "ip.addr == your_IP_address" into the filter, where you obtain your_IP_address with ipconfig. This filter removes all packets that neither originate nor are destined to your host.
4. Start packet capture in Wireshark.
5. With your browser, visit the Web page: http://www.ietf.org
6. Stop packet capture.

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 computers.
Answer the following questions. Whenever possible, when answering a question below, 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) Locate the DNS query and response messages. Are then sent over UDP or TCP?
2) What is the destination port for the DNS query message? What is the source port of DNS response message?
3) To what IP address is the DNS query message sent? Use ipconfig to determine the IP address of your local DNS server. Are these two IP addresses the same?
4) Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?

5) Examine the DNS response message. How many "answers" are provided? What do each of these answers contain?

4. Playing with nslookup

Now let's play with nslookup.

1. Start packet capture.
2. Do an nslookup on www.mit.edu.
3. Stop packet capture.

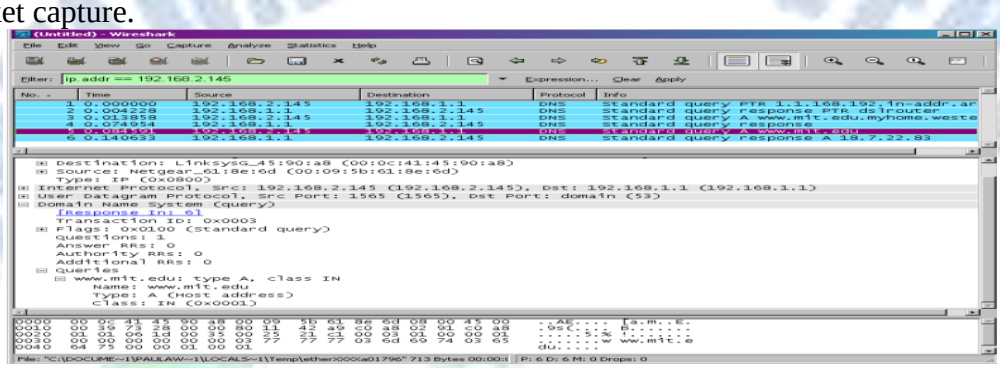

Figure 34

We see from the above screenshot that nslookup actually sent three DNS queries and received three DNS responses.

For the purpose of this assignment, in answering the following questions, ignore the first two sets of queries/responses, as they are specific to nslookup and are not normally generated by standard Internet applications.

You should instead focus on the last query and response messages.

Now, we'll investigate the behavior of the celebrated TCP protocol in detail. We'll do so by analyzing a trace of the TCP segments sent and received in transferring a 150KB file (containing the text of Lewis Carrol's Alice's Adventures in Wonderland) from your computer to a remote server. We'll study TCP's use of sequence and acknowledgement numbers for providing reliable data transfer; we'll see TCP's congestion control algorithm – slow start and congestion avoidance – in action; and we'll look at TCP's receiver-advertised flow control mechanism. We'll also briefly consider TCP connection setup and we'll investigate the performance (throughput and round-trip time) of the TCP connection between your computer and the server.

7. Capturing a Bulk TCP transfer from your computer to a Remote Server

Before beginning our exploration of TCP, we'll need to use Wireshark to obtain a packet trace of the TCP transfer of a file from your computer to a remote server. You'll do so by accessing a Web page that will allow you to enter the name of a file stored on your computer (which contains the ASCII text of Alice in Wonderland), and then transfer the file to a Web server using the HTTP POST method.

We're using the POST method rather than the GET method as we'd like to transfer a large amount of data from your computer to another computer.

Of course, we'll be running Wireshark during this time to obtain the trace of the TCP segments sent and received from your computer.

Do the following:

1. Start up your web browser. Go to the following URL:
   http://gaia.cs.umass.edu/wiresharklabs/alice.txt
   Retrieve an ASCII copy of Alice in Wonderland. Store this file somewhere on your computer.

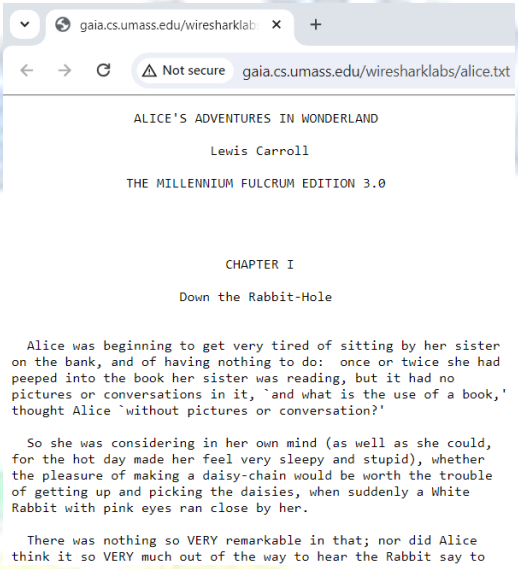

Figure 35

2. Next go to the following URL:
   http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html
   You should see a screen that looks like:

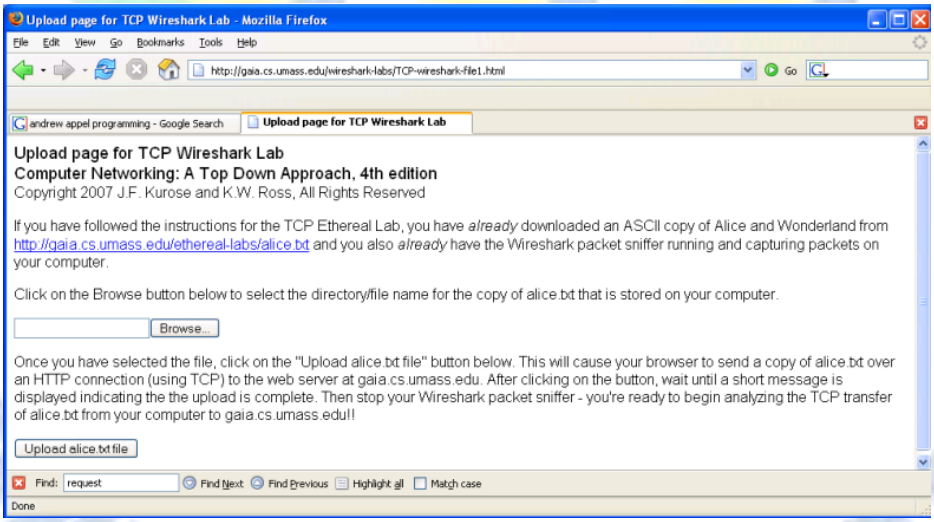

Figure 36

3. Use the Browse button in this form to enter the name of the file (full path name) on your computer containing Alice in Wonderland (or do so manually). Don't yet press the "Upload alice.txt file" button.
4. Now start up Wireshark and begin packet capture (Capture->Start) and then press OK on the Wireshark Packet Capture Options screen (we'll not need to select any options here).

5. Returning to your browser, press the "Upload alice.txt file" button to upload the file to the gaia.cs.umass.edu server. Once the file has been uploaded, a short congratulations message will be displayed in your browser window.
6. Stop Wireshark packet capture. Your Wireshark window should look similar to the window shown below.

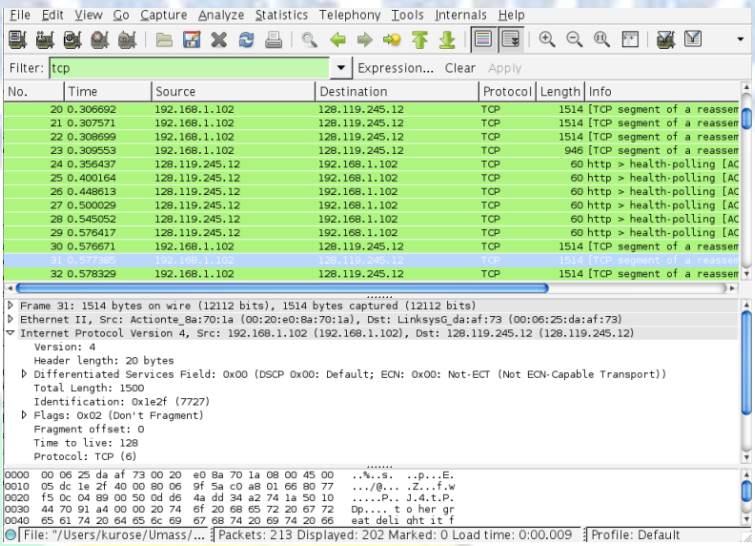

Figure 37

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 computers.
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.

8. A First Look at The Captured Trace

Before analyzing the behavior of the TCP connection in detail, let's take a high level view of the trace.

1. First, filter the packets displayed in the Wireshark window by entering "tcp" (lowercase, no quotes, and don't forget to press return after entering!) into the display filter specification window towards the top of the Wireshark window.

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

Answer the following questions, by opening the Wireshark captured packet file tcpethereal-trace-1 in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip (that is download the trace and open that trace in Wireshark; see footnote 2).
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 printout3 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.

Lab Exercise

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

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

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?

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

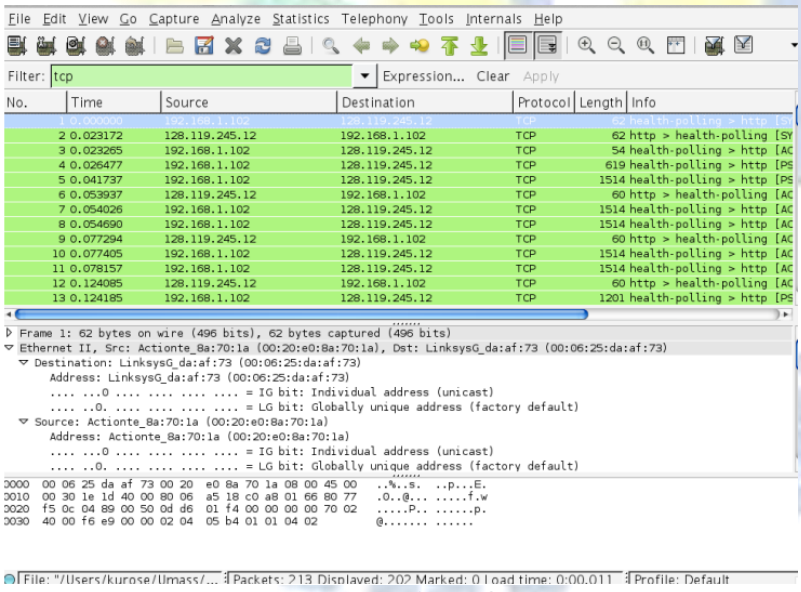

Figure 38

Lab Exercises

1. Let's suppose your organization need to create it's on small server (for provide some services) based network. With bellow mentioned topology and instructions:

a) Configure SMTP (create account with your last name along with last 3 digits roll number) send mail from PC A to PC-B.
b) PC A should be configured to have the SMTP account of Server 2 while PC B should be having an account of Server 1.
c) Configure FTP server create account with your first name, password with your roll number and filename with your last name (.bin extension) show all connection results. The FTP Server should be established on both Server 2 and Server 3.