Data Communications Laboratory
Introduction to Wireshark

Objective

1. **Understand the role and functioning of packet sniffers**
2. **Understand how to use Wireshark for basic capture and analysis**
3. **Have a basic understanding of encapsulation**
4. **Examine Ethernet and a well-known application protocol HTTP as practice**

Reminder

As stated in section 1 last week, it is important that you **spend time preparing for the labs before your assigned two hours**. You should at least individually **read through the week's section** before the lab session since some sections require about an hour's reading. You should then **mentally prepare the steps** so that you can efficiently work through them when you come to the lab.
Also remember you **must wear enclosed footwear** as this is a university OH&S requirement for all labs on campus.

Exercise 1. Introduction to Wireshark

1.1 Motivation

Packet sniffer's (such as Wireshark) are applications that capture packets that are seen by a machine's network interface. You'll be using Wireshark in many of the remaining laboratories in this unit. Packet sniffers can be used to troubleshoot networks and also in application and network software development. In this course you will be using it to show you first hand that what is presented in lectures in theory is actually happening on networks. It is thus very important to learn Wireshark quickly.

You also do not need to wait for the lab before trying Wireshark – you can load it on Windows or macOS from:

http://www.wireshark.org/

where lots more information is available.

By default macOS disables programs like Wireshark from accessing all network traffic entering or leaving a host. To allow Wireshark access to all traffic you will need to run the following commands in Terminal:

```
sudo chgrp admin /dev/bpf*
sudo chmod g+rw /dev/bpf*
```

You might also need to set up sudo by adding your username to the sudoers file. Remember you should become familiar with using command-line interfaces – it's not hard, there's just a lot of it, and why would you want to provide a GUI for the above commands, when these are unlikely to be needed by average users..

1.2 Theory

When a sniffer runs on a system, it grabs all the packets that come into and goes out of the Network Interface Card (NIC) of the machine on which the sniffer is installed. This means that, if the NIC is set to the promiscuous mode, then it will receive all the packets sent to the network if that network is connected by a hub (if the network is switched, this won't happen – think about why). 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.

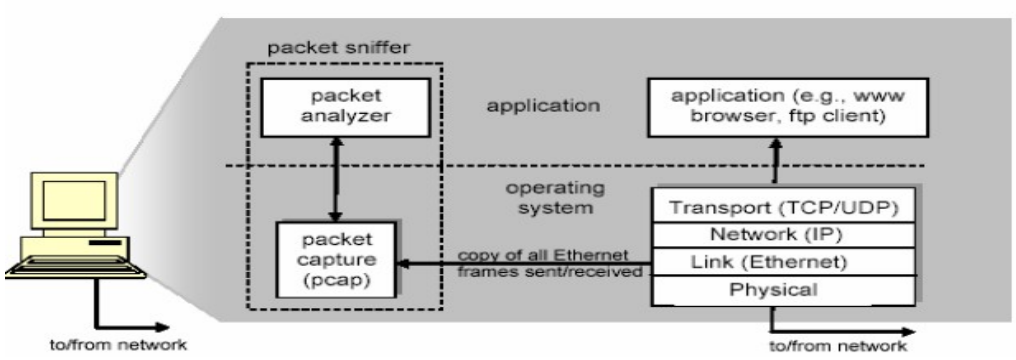


Figure 1: Packet Sniffer Structure

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. 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 network, 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.
The second component of a packet sniffer is the **packet analyser**, which displays the contents of all fields within a protocol message. In order to do so, the packet analyser must "understand" the structure of all messages exchanged by protocols. For example,

in the exercise that follows we are interested in displaying the various fields in messages exchanged by the application protocol in figure 1. The packet analyser understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. It understands the TCP segment structure, so it can extract the application protocol message contained in the TCP segment. Finally, if it understands the application protocol (typically if it is a common one such as HTTP) the packet analyser can extract some information from the application packet (such as reading the first bytes of an HTTP message, which must contain one of the strings "GET,""POST," or "HEAD").

The following is a very brief introduction to Wireshark. For more information go to www.wireshark.org/docs

1.2 Introduction to Wireshark

Note: This information is extracted from Wireshark User's Guide: 27212 for Wireshark 1.0.0 By Ulf Lamping and Richard Sharpe.

Wireshark is a network packet analyser. A network packet analyser will capture network packets and display that packet data in as detailed a manner as possible.

You can think of a network packet analyser as a measuring device used to examine what's going on inside a network cable, just like a voltmeter is used by an electrician to examine what's going on inside an electric cable (but at a higher level, of course). In the past, such tools were either very expensive, proprietary, or both. Hoswever, opensource and freeware examples of these tools are now available. Wireshark is perhaps one of the best open source packet analysers available today.Wireshark is an open source software project, and is released under the **GNU General Public License** (GPL). You can freely use Wireshark on any number of computers you like, without worrying about license keys or fees or such. In addition, all source code is freely available under the GPL. Because of that, it is very easy for people to add new protocols to Wireshark, either as plugins, or built into the source, and they often do!

A brief history of Wireshark

In late 1997, Gerald Combs needed a tool for tracking down networking problems and wanted to learn more about networking, so he started writing Ethereal (the former name of the Wireshark project) as a way to solve both problems.

Ethereal was initially released, after several pauses in development, in July 1998 as version 0.2.0. Within days, patches, bug reports, and words of encouragement started arriving, so Ethereal was on its way to success.

Not long after that, Gilbert Ramirez saw its potential and contributed a low-level dissector to it. In October, 1998, Guy Harris of Network Appliance was looking for something better than tcpview, so he started applying patches and contributing dissectors to Ethereal.

In late 1998, Richard Sharpe, who was giving TCP/IP courses, saw its potential on such courses, and started looking at it to see if it supported the protocols he needed. While it didn't at that point, new protocols could be easily added. So he started contributing dissectors and contributing patches. The list of people who have contributed to Ethereal has become very long since then, and almost all of them started with a protocol that they needed that Ethereal did not already handle. So they copied an existing dissector and contributed the code back to the team.

In 2006 the project moved house and re-emerged under a new name: Wireshark.

Some intended purposes

Here are some examples people use Wireshark for:

- network administrators use it to **troubleshoot network problems**
- network security engineers use it to **examine security problems**
- developers use it to **debug protocol implementations**
- people use it to **learn network protocol** internals

Beside these examples, Wireshark can be helpful in many other situations too.

Features

The following are some of the many features Wireshark provides:

- Available for **UNIX** and **Windows**.
- **Capture** live packet data from a network interface.
- Display packets with **very detailed protocol information**.
- **Open and Save** packet data captured.
- **Import and Export** packet data from and to a lot of other capture programs.
- **Filter packets** on many criteria.
- **Search** for packets on many criteria.
- **Colorize** packet display based on filters.
- Create various **statistics**.
- ... and **a lot more!**

What Wireshark is not

Here are some things Wireshark does not provide:

- Wireshark isn't an intrusion detection system. It will not warn you when someone does strange things on your network that he/she isn't allowed to do. However, if strange things happen, Wireshark might help you figure out what is really going on.
- Wireshark will not manipulate things on the network, it will only "measure" things from it.
- Wireshark doesn't send packets on the network or do other active things (except for name resolutions, but even that can be disabled).

1.3 Using Wireshark – Understanding the “Capture Menu”

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 pane** 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 pane** provides details about the packet selected (highlighted) in the packet listing window. (To select a packet in the packet listing window, place the cursor over the packet's one-line summary in the packet listing window and click with the left mouse button.). These details include information about the Ethernet frame 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 right-pointing or down-pointing arrowhead 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 pane** 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 this exercise, we'll use the "Capture Menu" to capture packets that correspond to HTTP messages. Before we begin this exercise, we must first get familiar with the Capture Menu.

Capturing Packets Using Wireshark

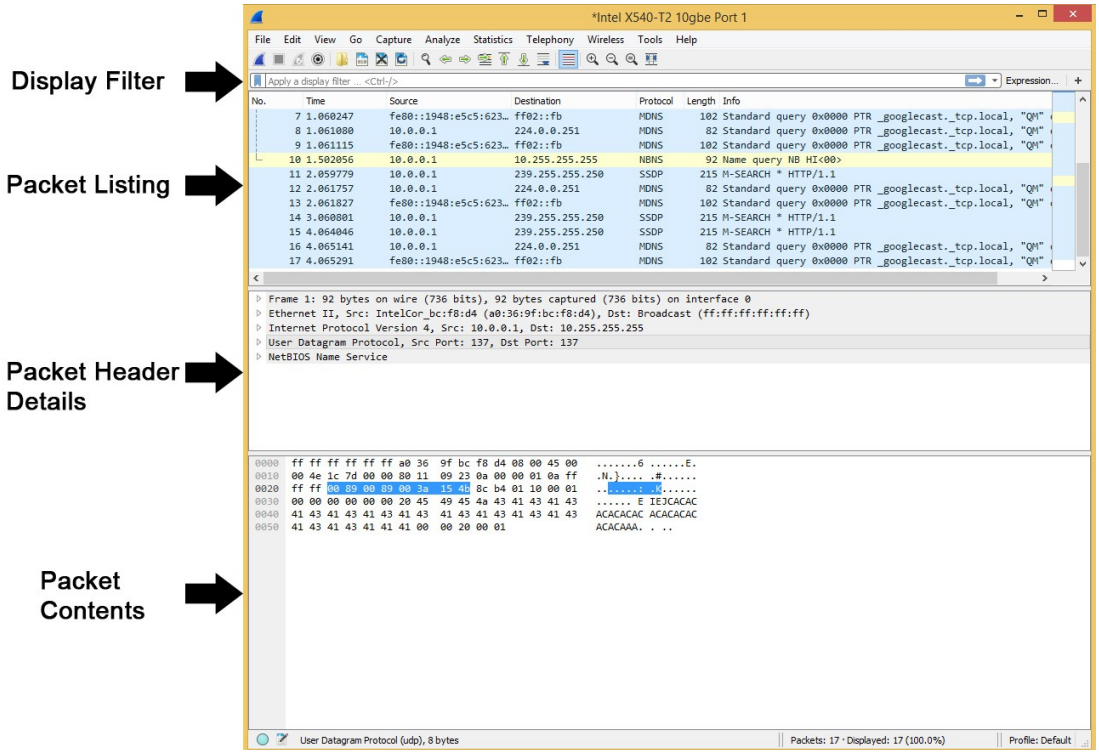


Figure 1: WireShark Graphical User Interface

Figure 1 shows the page of Wireshark once some packets have been captured.
The *packet listing* pane shows all the network packets that have been captured while you have been running Wireshark in capture mode – one packet per line. When you click on a packet, that packet is displayed in different ways in the lower two panes. The bottom *packet contents* pane shows the packet in raw format as it was captured. This information is very low level and mostly not very interesting. You can see three vertical sections. The first column gives you the offset of the octet. The next 16 two-digit columns show you the octet contents in hexadecimal (that is four bit numbers 0-15 are represented by 0-9, then A-F). You should get used to hexadecimal numbers for communications work since it represents four bit entities neatly and we will be using hexadecimal more in future labs. The next section shows the same data as ASCII characters. Wireshark does the best it can displaying the characters, but not all 8-bit patterns (hexadecimal codes) correspond to printable characters, especially codes < 0x20. Non-printable characters are simply represented by a dot '.'.
The middle *packet-header details* pane is where you will look most. This pane displays Wireshark's interpretation of the packet. The lines here represent different layers of the network stack, albeit upside down to how we usually think of the network stack. The top line represents the physical layer. The second line the data-link layer where you will mainly see Ethernet packets, but it could be another data-link protocol. The third line is the network layer, which is mainly IP. The fourth line is the transport layer, normally TCP, but it could be UDP. If Wireshark finds an application protocol like HTTP that it can interpret, it will also give you information about that on the fifth line.
In the toolbar, you will find some handy shortcuts for starting and stopping captures.

Since a capture can capture many packets, the filter text box allows you to enter criteria to display relevant packets.
Depending on how you would like to capture the packet, there are several ways to go about it.

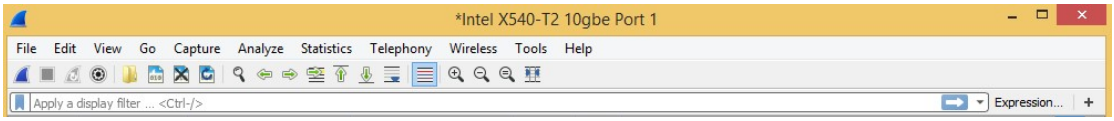


Figure 2: Wireshark Menu Bar

Log into your PC in the laboratory then start up Wireshark

To start with, we can choose the "Capture" menu (see figure 2) and see what is in there.

As the name implies, "Capture" menu is provided for the users to perform Packet Capture, and it also provides several options for suiting the situations and the conditions that the analysts have in the mind while performing the process of capturing the packets. Analysts could even set filters to avoid capturing unwanted traffic.

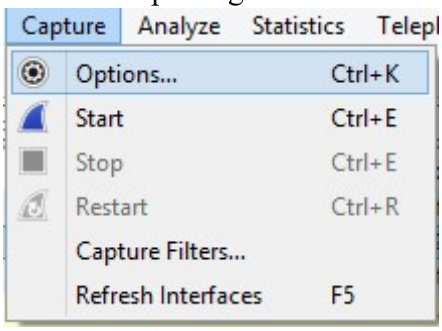


Figure 3: Wireshark Capture Tab

In Figure 3, choose the "Options" sub-menu in the "Capture" drop-down listing. Once you choose this, you will have the listing of various network interfaces in your machine that can be seen from your computer's NIC(s). An example is shown in Figure 4, where we have five interfaces.

Documentation Task 1.

What interfaces are available on your computer? What do they appear to be? Do they all have the same IP address (click on the arrow beside each interface for more information)? Record this in your documentation.

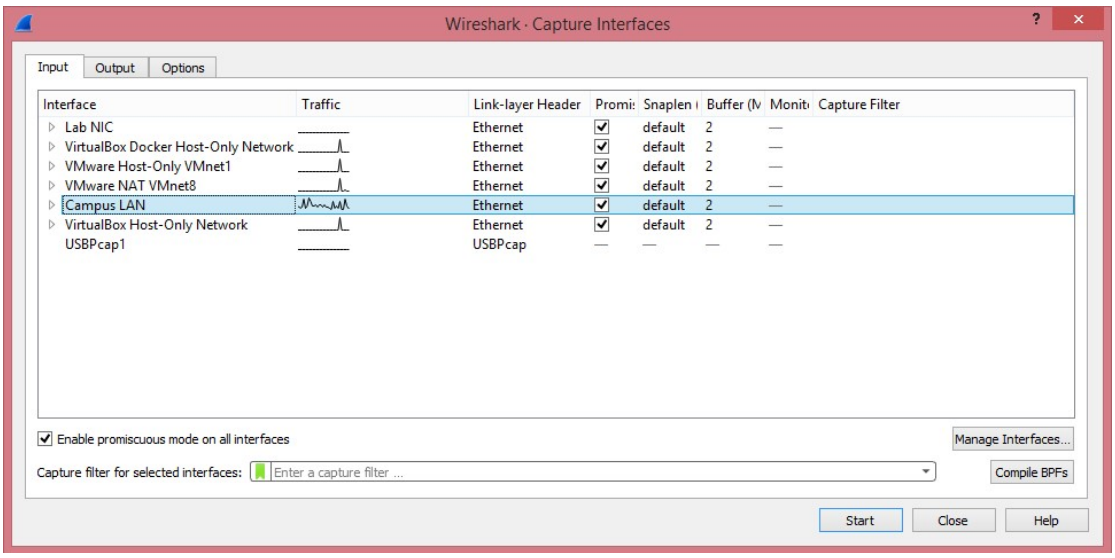


Figure 4: Wireshark Capture Interfaces

Once you have decided and clicked on the interface that you would like sniff, click the *Start* button. The interface titled "Ethernet" is the NIC that is used to connect your PC to the Internet. This is the interface we will be sniffing today.

Once the "Start" is clicked, a "Capture" window will open up showing the count of packets from each of the protocols. You should now see something similar to figure 1, but continuously changing as more and more packets are captured

Users can choose the option to view the packets when capturing them simultaneously. This is done by selecting "Update packets in real-time" in the Options tab from the Capture > Options menu, shown in Figure 5.

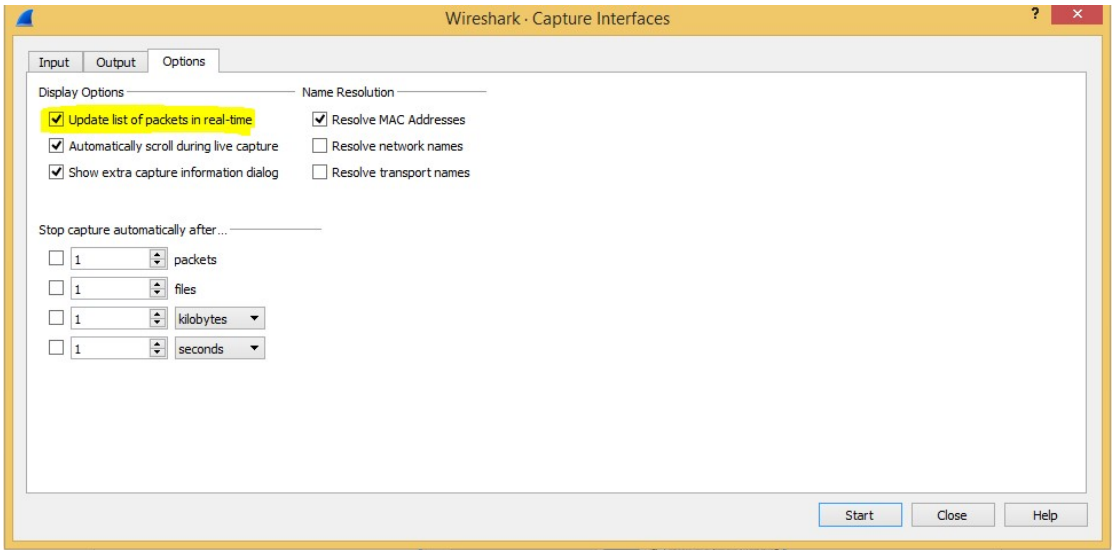


Figure 6: Capture Options

In Figure 5, there is a highlighted option that says "*Update list of packets in real time*". This is a checkbox option for users to choose based on what they would like to do. Checking this option would help the user to view the packet contents while the packets are being sniffed. This option may not be chosen by default, and hence the users would have to choose this option only when required and click on "Start" after choosing the option.

Now that, we have seen how to work with the Capture Menu of WireShark Packet sniffer you are now ready to work with this week's exercise.

Exercise 2. Using Wireshark to examine HTTP

Objective

1. Become more familiar with Wireshark by capturing data packets
2. Examine the Hyper Text Transfer Protocol in action
3. Analyse the results from capturing packets for a file download from a web server.

2.1 Check computer addresses and start up Wireshark Capture

1. Start up Wireshark, click ***Capture→Options*** and click on the "**Ethernet"** interface (first floor labs) or the "**Campus LAN**" interface (second floor lab) then click the ***Start*** button. Wireshark will begin running in Capture Mode and will open up a Wireshark Capture window showing you how many packets have been captured in real time.
2. Determine the MAC address corresponding to the IP address of the interface you are going to use (use the command prompt command **ipconfig** or view these details in the wireshark interface list)

Documentation Task 2.

Record the IP address and MAC address for the Ethernet interface of the computer you are using

2.2 Download a Web Page

1. Start up a web browser (Chrome, Firefox or whatever is available on the machine).
2. Start the capture in Wireshark
3. Enter the following address into your web browser (note the http, **not** https):

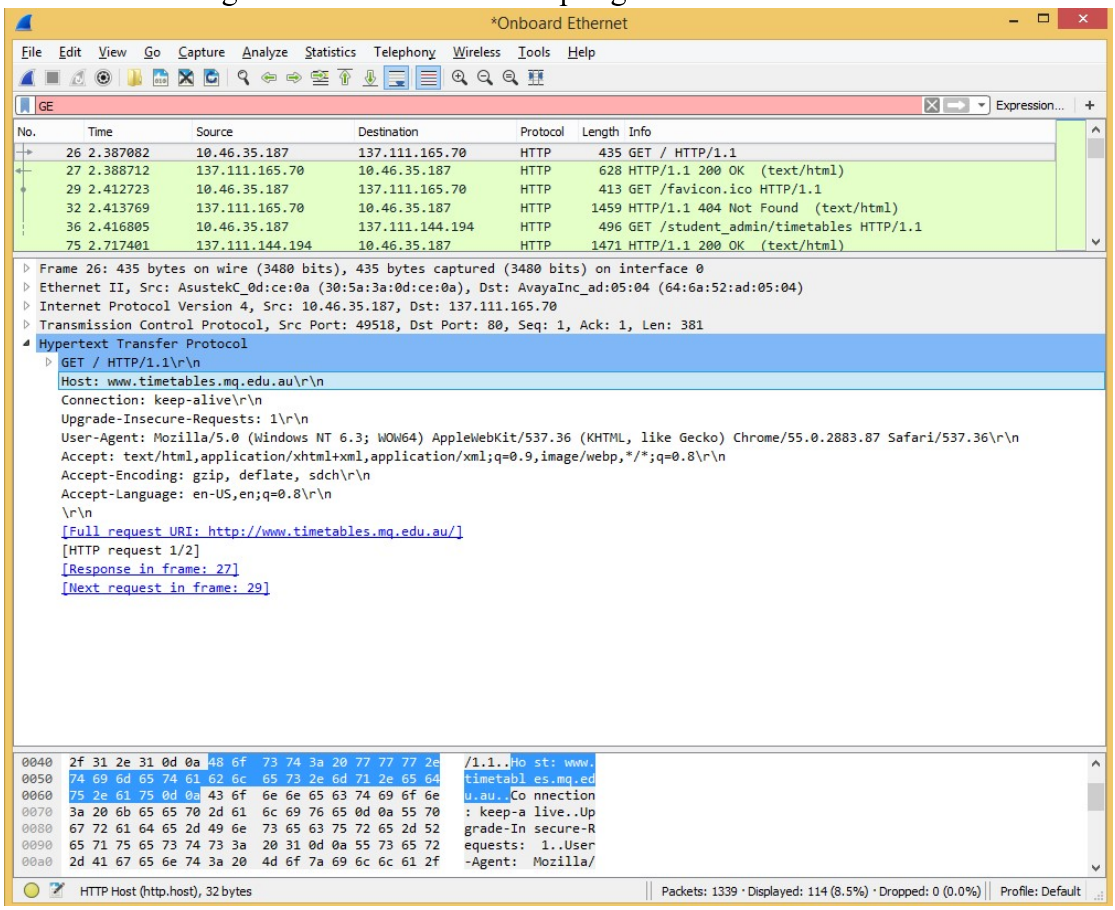
http://www.ntp.org/

Make sure type the whole address including the leading http.
If you don't you won't see anything in wireshark!

4. A web page titled "NTP: The Network Time Protocol" should appear in your browser.
3. Close your browser window
4. Go back to the Wireshark Capture window and click the ***Stop*** button to stop the packet capture.

2.3 Verify that the Web Page Download has been Captured

1. Back in the Wireshark window, you should now see lots of packets in the top summary pane. You can filter out all packets except HTTP packets by typing the word "http" into the Filter box (click View→Filter Toolbar if you don't see a Filter box at the top). This will make things much easier to read.
2. Look for a sequence of packets starting with one with the info "Get / HTTP1.1, as in the figure. Expand the hypertext transfer protocol section as shown in the image and check the host is ntp.org



3. On the other hand, if the reply from the web server contains "HTTP/1.1 304 Not Modified", then this means Wireshark **did not capture** the packets from the web site because the web page was already stored (cached) in your browser. In this case you must clear your browser cache and then go back and re-do the capture as follows:

a. First, you must clear the web cache in your browser.

i. For Chrome and Firefox, press ***CTRL + SHIFT + DEL***

ii. Select "Everything" or "Beginning of time" and check all the boxes
iii. Click clear

b. Now in your Wireshark window, again click ***Capture→Options*** and click on the **Ethernet** interface then click the ***Start*** button. Go back to Part 2, step 1 above to download the web page again while Wireshark is capturing packets.

Documentation Task 3.

1. How many HTTP packets were received by your machine?

2. Which one contains the main source code for the web page? Could you tell this from the main capture window? How? *Hint: make sure the HTTP section is selected in the packet in the middle pane.*

3. What else can you tell about the web page and HTTP from the captured packets?

2.4 Encapsulation

In future sections you'll be looking in more detail at IP and TCP using Wireshark. For now, I want you to look at how Ethernet frames are used to carry IP, TCP and application packets. Have a look at the captured packets labelled HTTP. HTTP is an application layer protocol. In the information Wireshark displays about them you will also notice sections labelled IP and TCP. Packets (or frames) from one layer of the protocol stack are bundled up within (encapsulated) within the packet (frame) of the layer below. With a number of layers in the protocol stack you get a number of levels of bundling. I say "packets or frames" here because the messages of some layers are called "packets" (eg IP, TCP) and some frames (eg Ethernet).

Documentation Task 4.

1. Draw a diagram or table showing (in outline, don't worry about details such as how many bytes are used and fields in each packet) how the the IP, TCP and HTTP packets are contained within the Ethernet frame

2. What else can you observe about encapsulation from looking at your capture?

Exercise 3. Using Wireshark to examine Ethernet

Objective

In this exercise, we'll investigate the Ethernet protocol and the ARP protocol. Before beginning this lab, you'll probably want to review appropriate sections in the lecture notes/text book.

3.1 Theory

Ethernet is a family of frame-based computer networking technologies for local area networks (LANs). The name comes from the physical concept of the ether. It defines a number of wiring and signaling standards for the Physical Layer of the OSI networking model, through means of network access at the Media Access Control (MAC) /Data Link Layer, and a common addressing format.
Ethernet is standardized as IEEE 802.3. The combination of the twisted pair versions of Ethernet for connecting end systems to the network, along with the fiber optic versions for site backbones, is the most widespread wired LAN technology. It has been in use from around 1980 to the present, largely replacing competing LAN standards such as token ring, FDDI, and ARCNET.
You might want to revise the lecture notes on Ethernet.

3.2 Capturing and analyzing Ethernet frames

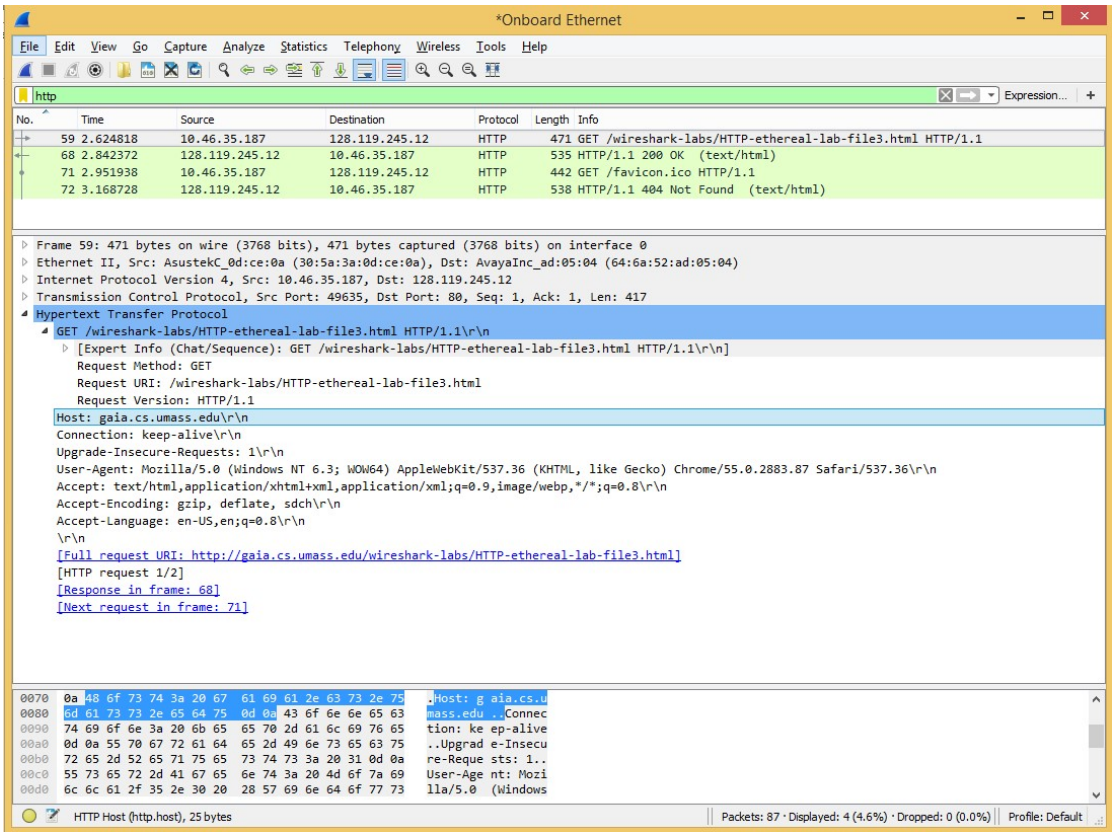
Let's begin by capturing a set of Ethernet frames to study. Do the following[1]:

1. First, make sure your browser's cache is empty. Check the instructions a few pages back on how to do this or ask your instructor.

2. Start up the Wireshark packet sniffer

3. 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.

4. 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 59 in the screen shot below contains the HTTP GET message)



5. 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).

6. 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. 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.

Documentation Task 5.

Answer the following questions, based on the contents of the Ethernet frame containing the HTTP GET message.

1. How many bytes long is the packet?

2. What is the 48-bit MAC address of your computer?

3. 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.]

4. Give the hexadecimal value for the two-byte Type field.

5. How many bytes from the very start of the Ethernet frame does the ASCII “G” in “GET” appear in the 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 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. Is the OK in the HTTP message actually contained in the HTTP packet shown to you by Wireshark when you filter for HTTP packets? If not, where is it?

10. Compare any Ethernet packet you have captured to the structure shown in lectures. Are they the same or, if there are differences, what are they?