Guide: Using Wireshark

I) Two ways to capture some packets:

i) A Simple capture

You are now ready to capture packets coming to and from your machine. Begin the capture process by selecting the “Capture” menu and then clicking “Start”.

Wireshark will immediately begin capturing data from the network adapter you selected earlier, or give an error message that no adapter is selected if you didn’t perform the pre-configuration.

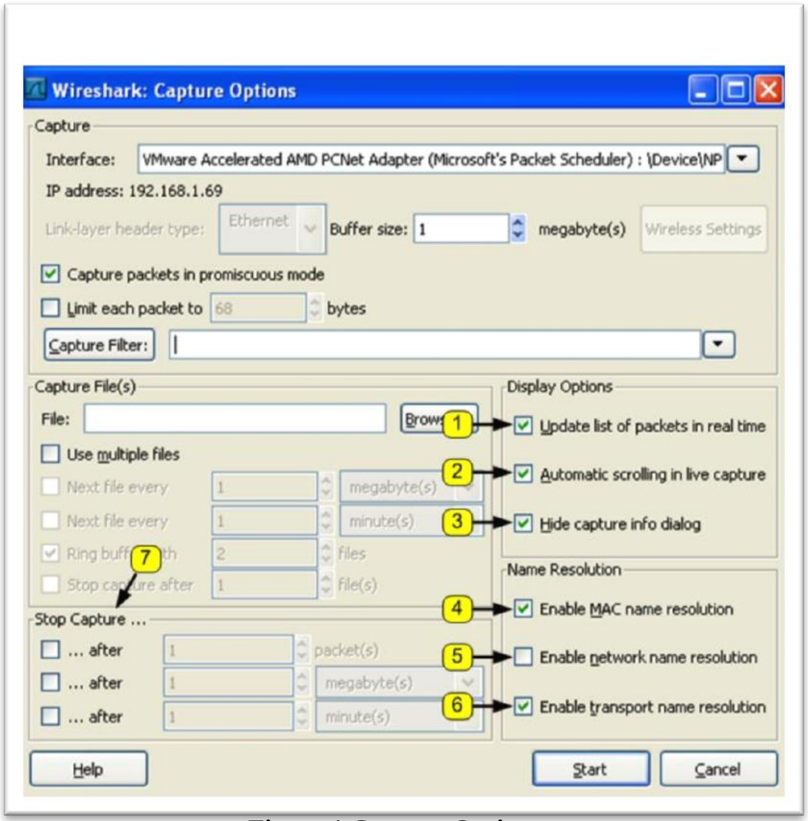
You can stop the capture by selecting “stop” from the capture menu.

ii) Selecting “Capture Options” before Capturing

Many people prefer to take an extra step before beginning the capture which lets a number of features be configured. Click the “Capture” menu then select “Options”. A number of options are available in this dialog. Some, such as “capture filter”, are for more advanced use. However, a number of options are available which are very useful even during basic captures. A number of these items are highlighted in Figure 1, including:

Update list of packets in real time: This tells Wireshark to displays packets as they captured rather than waiting until the capture is stopped (default is on).

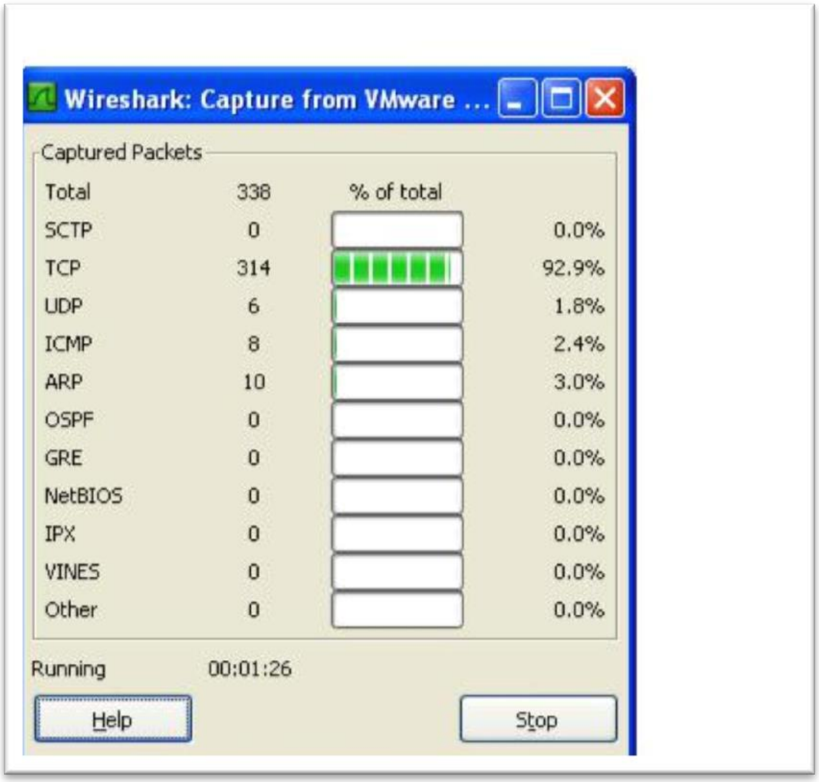
Automatic scrolling in live capture: If the previous item is selected, this tells Wireshark to scroll the packets so that you are viewing the most recent (default is on).



Interface: VMware Accelerated AMD PCNet Adapter (Microsoft's Packet Scheduler) : \Device\NP
IP address: 192.168.1.69
Link-layer header type: Ethernet Buffer size: 1 megabyte(s)
☑ Capture packets in promiscuous mode
☐ Limit each packet to 68 bytes
File:
☐ Use multiple files
☐ Next file every 1 megabyte(s)
☐ Next file every 1 minute(s)
☑ Ring buffer with 2 files
☐ Stop capture after 1 file(s)
☐ ... after 1 packet(s)
☐ ... after 1 megabyte(s)
☐ ... after 1 minute(s)
☑ Update list of packets in real time
☑ Automatic scrolling in live capture
☑ Hide capture info dialog
☑ Enable MAC name resolution
☐ Enable network name resolution
☑ Enable transport name resolution

Figure 1 Capture Options

Hide Capture Info dialog: The "Capture Info" dialog was always displayed in earlier versions of Wireshark and Ethereal but is now disabled by default. This dialog displays a bar-graph summary of the protocols during the capture, but disappears when the capture is stopped. You may find this useful in deciding whether you have captured enough of the packets of interest to you (default is on – i.e. hide)



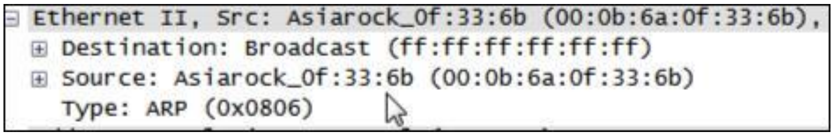
Captured Packets

Total	338	% of total	
SCTP	0		0.0%
TCP	314		92.9%
UDP	6		1.8%
ICMP	8		2.4%
ARP	10		3.0%
OSPF	0		0.0%
GRE	0		0.0%
NetBIOS	0		0.0%
IPX	0		0.0%
VINES	0		0.0%
Other	0		0.0%

Running 00:01:26

Figure 2 Capture Info Dialog

Enable MAC name resolution: This tells Wireshark to display the name of the manufacturer of the network card when it lists the MAC address. Figure 3 shows an example of MAC name resolution with a MAC address generated from an Asiarock network card (default is on).



```
⊟ Ethernet II, Src: Asiarock_0f:33:6b (00:0b:6a:0f:33:6b),
  ⊞ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ⊞ Source: Asiarock_0f:33:6b (00:0b:6a:0f:33:6b)
    Type: ARP (0x0806)
```

Figure 3 MAC name resolution

Enable network name resolution: Network Name Resolution (NNR) tells Wireshark to use names, such as cnn.com, in the summaries. If NNR is turned off, you will only see IP addresses in the summary. This setting only affects the summary. Even with names turned on, you can easily see the IP address by clicking on the packet and examining the packet details. However, it is easier to select packets if the names are available to identify network servers.

However, this requires Wireshark to perform a DNS lookup for every IP address. If you are connected to the internet, this may be trivial. But if you are working offline then you will need to wait for very DNS lookup to be attempted, and time-out and fail. This may take an exceptionally long time, and make Wireshark appear to freeze. Also, the DNS lookup will add extra packets into the capture. This adds an artificial component to the capture. This feature is turned off by default; you may prefer to turn it on if you are working on a computer with access to a DNS server.

Enable transport name resolution: This option tells Wireshark to display the typical name of a protocol rather than the port value. For example, a datagram with port 80 will be displayed as HTTP. However, you should remember that this is a simple lookup of a table. It is possible that some other, non-http, traffic may actually be using this port (default is on)

Stop Capture: The items in this section allow you to pre-select a stop condition for the capture. You may select to stop after a number of packets, an amount of data, or period of time. It is often interesting to close all applications, and then capture all traffic over a minute or two while your computer is "idle". This will show you the normal background traffic existing on your network (default is on). When you have selected the items which you prefer, click the "start" button.

II) Examining the Capture

Start a capture using either of the above methods. You may immediately see packets being saved to your machine. This traffic is most likely normal background activity.

Let's create some packets for Wireshark to capture. With Wireshark running and capturing packets, go to a web browser (e.g., Internet Explorer, Mozilla's Firefox, Opera, or Safari), and type in a web address, such as www.cnn.com.

When the web page finished loading, go back to Wireshark and through the menu click "Capture" then "Stop", or use the short-cut CTRL-E (for End). If you have changed the setting to display the "Capture Info" dialog box (Figure 2), you just need to click the "stop" button. Don't be surprised if Wireshark captures quite a few packets of information.

III) What if I can't find any packets?

If you don't see any packets while Wireshark is performing the capture, you may have de-selected the option to "Update packets in real time (item 1 in Figure 1). When the capture stops, you should see Wireshark process and load each packet which was captured. There are several things to check out if you don't see packets after you end the capture.

1) *When you were setting up Wireshark, did you select the network adapter that is being used to interface with the network?*

   Refer to section **Error! Reference source not found.**

2) *Are you using a wireless connection on a Windows machine?*

   Wireshark is not able to capture packets on some wireless connections within Windows. Refer to the section in Appendix for a possible workaround and more information.

3) *Are you using filters?*

   Wireshark can filter results so that only certain types of packets are captured. If the capture filter is set and no packets matched the filter then you will have captured no packets. There is nothing you can do except repeat the capture either without the capture filter or ensure that the specified packets are created. There is also a display filter that will hide any packet not meeting a specified condition. An example of a filter condition would be to only display packets sent to/from a specific IP address. If you set a filter, and then have no traffic that matches the filter, then you will not see any packets. Click the "clear" button next to the display filter to view all packets.

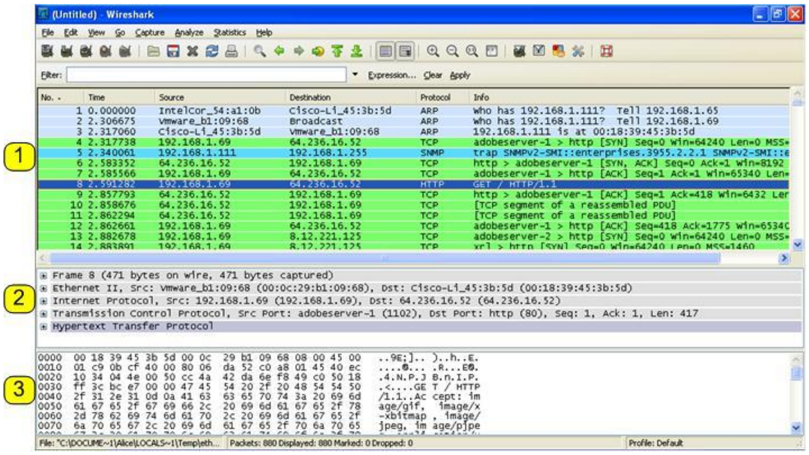
4) *Did you create any traffic for Wireshark to filter?*

After you go to the “Capture” menu and choose “Start”, you must leave Wireshark running. If the Capture Info dialog is displayed – do not click the “Stop” button. Then go to your web browser and enter a web address, such as www.cnn.com. Finally return to Wireshark and click on the “stop” button.

5) If none of these options worked, go to the Wireshark web site and check the FAQs, the documentation and the wiki at www.wireshark.com .

IV) Looking at Packets Captured by Wireshark

Once you have captured a set of packets, Wireshark should present you with a colourful window as shown in Figure 4 below.


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply
1
2
3

```
No. .  Time       Source              Destination         Protocol  Info
     1 0.000000   IntelCor_54:a1:0b   Cisco-Li_45:3b:5d   ARP       Who has 192.168.1.111?  Tell 192.168.1.65
     2 2.306675   Vmware_b1:09:68     Broadcast           ARP       Who has 192.168.1.111?  Tell 192.168.1.69
     3 2.317060   Cisco-Li_45:3b:5d   Vmware_b1:09:68     ARP       192.168.1.111 is at 00:18:39:45:3b:5d
     4 2.317738   192.168.1.69        64.236.16.52        TCP       adobeserver-1 > http [SYN] Seq=0 Win=64240 Len=0 MSS=
     5 2.340061   192.168.1.111       192.168.1.255       SNMP      trap SNMPv2-SMI::enterprises.3955.2.2.1 SNMPv2-SMI::e
     6 2.583352   64.236.16.52        192.168.1.69        TCP       http > adobeserver-1 [SYN, ACK] Seq=0 Ack=1 Win=8192
     7 2.585566   192.168.1.69        64.236.16.52        TCP       adobeserver-1 > http [ACK] Seq=1 Ack=1 Win=65340 Len=
     8 2.591282   192.168.1.69        64.236.16.52        HTTP      GET / HTTP/1.1
     9 2.857793   64.236.16.52        192.168.1.69        TCP       http > adobeserver-1 [ACK] Seq=1 Ack=418 Win=6432 Ler
    10 2.858676   64.236.16.52        192.168.1.69        TCP       [TCP segment of a reassembled PDU]
    11 2.862294   64.236.16.52        192.168.1.69        TCP       [TCP segment of a reassembled PDU]
    12 2.862661   192.168.1.69        64.236.16.52        TCP       adobeserver-1 > http [ACK] Seq=418 Ack=1775 Win=6534C
    13 2.882678   192.168.1.69        8.12.221.125        TCP       adobeserver-2 > http [SYN] Seq=0 Win=64240 Len=0 MSS=
    14 2.883891   192.168.1.69        8.12.221.125        TCP       xrl > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460

⊞ Frame 8 (471 bytes on wire, 471 bytes captured)
⊞ Ethernet II, Src: Vmware_b1:09:68 (00:0c:29:b1:09:68), Dst: Cisco-Li_45:3b:5d (00:18:39:45:3b:5d)
⊞ Internet Protocol, Src: 192.168.1.69 (192.168.1.69), Dst: 64.236.16.52 (64.236.16.52)
⊞ Transmission Control Protocol, Src Port: adobeserver-1 (1102), Dst Port: http (80), Seq: 1, Ack: 1, Len: 417
⊞ Hypertext Transfer Protocol

0000  00 18 39 45 3b 5d 00 0c  29 b1 09 68 08 00 45 00   ..9E;].. )..h..E.
0010  01 c9 0b cf 40 00 80 06  da 52 c0 a8 01 45 40 ec   ....@... .R...E@.
0020  10 34 04 4e 00 50 cc 4a  42 da 6e f8 49 c0 50 18   .4.N.P.J B.n.I.P.
0030  ff 3c bc e7 00 00 47 45  54 20 2f 20 48 54 54 50   .<....GE T / HTTP
0040  2f 31 2e 31 0d 0a 41 63  63 65 70 74 3a 20 69 6d   /1.1..Ac cept: im
0050  61 67 65 2f 67 69 66 2c  20 69 6d 61 67 65 2f 78   age/gif,  image/x
0060  2d 78 62 69 74 6d 61 70  2c 20 69 6d 61 67 65 2f   -xbitmap , image/
0070  6a 70 65 67 2c 20 69 6d  61 67 65 2f 70 6a 70 65   jpeg, im age/pjpe

File: "C:\DOCUME~1\Alice\LOCALS~1\Temp\eth...   Packets: 880 Displayed: 880 Marked: 0 Dropped: 0   Profile: Default
```

Figure 4: Packet Listing Window

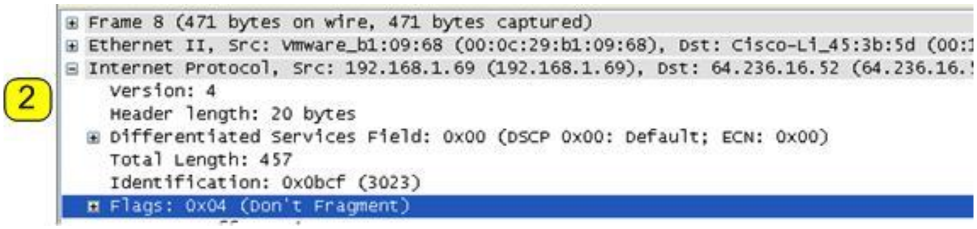
This window is divided into three areas.

i) Window Area 1: Summary

At the top is a colourful listing of all of the packets captured. Each line is a summary of a single frame or packet that was captured. The colours represent a coding scheme that can be used to quickly detect the type of packet. For example, the predominant colour in the graphic above is light green. Light green is the colour for HTTP packets.

ii) Window Area 2: Detail

When you click on a packet in area 1, the packet structure is shown in area 2. In the screenshot above, the packet shown in dark blue has been selected; therefore area 2 shows more details on that packet. In order to see more details, refer to Figure 5 below. This figure shows an enlarged version of area 2 from the previous figure.



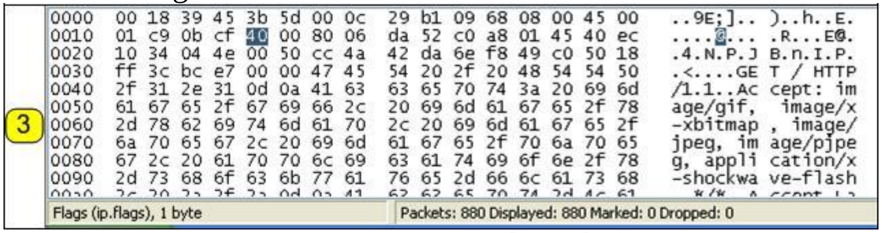
```
⊞ Frame 8 (471 bytes on wire, 471 bytes captured)
⊞ Ethernet II, Src: Vmware_b1:09:68 (00:0c:29:b1:09:68), Dst: Cisco-Li_45:3b:5d (00:1
⊟ Internet Protocol, Src: 192.168.1.69 (192.168.1.69), Dst: 64.236.16.52 (64.236.16.
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 457
    Identification: 0x0bcf (3023)
  ⊞ Flags: 0x04 (Don't Fragment)
```

Figure 5: Areas 2 Details (Extract from previous figure)

The first line of area two is created by Wireshark and contains statistical and informational data about the frame. It shows that this is the eighth frame (packet) that Wireshark captured. The next line in area 2 reveals that it was an Ethernet packet. Since the payload of this Ethernet packet was an Internet Protocol (IP) packet, the third line indicates that. You will also notice that there is a plus next to the first two lines and a minus next to the IP line. You can click on a plus to get more details on the packet contents. This has been done for the IP line so that the user can see the header information for the packet.

iii) Window Area 3: RAW Data

Clicking on a portion of the packet in area two changes the display in area 3. This was done in Figure 5 to select the IP flags field, in Figure 6 the hex of the flags field is selected. Area 3 has two parts. On the left are sixteen columns of two-characters each. This is the raw hexadecimal code that makes up the packet. On the right is the Unicode version of this hexadecimal code. If you click on an http line in window 2, you might notice English looking get commands or html commands in this right area.



```
0000  00 18 39 45 3b 5d 00 0c  29 b1 09 68 08 00 45 00   ..9E;].. )..h..E.
0010  01 c9 0b cf 40 00 80 06  da 52 c0 a8 01 45 40 ec   ....@... .R...E@.
0020  10 34 04 4e 00 50 cc 4a  42 da 6e f8 49 c0 50 18   .4.N.P.J B.n.I.P.
0030  ff 3c bc e7 00 00 47 45  54 20 2f 20 48 54 54 50   .<....GE T / HTTP
0040  2f 31 2e 31 0d 0a 41 63  63 65 70 74 3a 20 69 6d   /1.1..Ac cept: im
0050  61 67 65 2f 67 69 66 2c  20 69 6d 61 67 65 2f 78   age/gif,  image/x
0060  2d 78 62 69 74 6d 61 70  2c 20 69 6d 61 67 65 2f   -xbitmap , image/
0070  6a 70 65 67 2c 20 69 6d  61 67 65 2f 70 6a 70 65   jpeg, im age/pjpe
0080  67 2c 20 61 70 70 6c 69  63 61 74 69 6f 6e 2f 78   g, appli cation/x
0090  2d 73 68 6f 63 6b 77 61  76 65 2d 66 6c 61 73 68   -shockwa ve-flash
Flags (ip.flags), 1 byte      Packets: 880 Displayed: 880 Marked: 0 Dropped: 0
```

Figure 6 Hexadecimal View

V) Some Options to Analyse Captured Packets

Wireshark has several options to explore and analyse captured data. Feel free to explore the full set of options; however this section will discuss a few key capabilities.

i) Filters

Filters can be used to narrow in the focus on only important packets. See Appendix for a discussion of filters.

ii) Follow TCP Stream

Choose a TCP packet from the packet listing window (Area 1 in Figure 4). Right click on the chosen packet and select “Follow TCP Stream”. Wireshark will open a new window and display the set of data as it is seen by the application layer. For example, in the case of a HTTP response, this would be the HTTP data and the web page to be delivered to the browser.

However, the “Follow TCP Stream” command also does something that may confuse you – it automatically filters the packet display so that only packets relating to this stream are displayed. As a result, you may need to “Clear” (Appendix) the display filter after using “Follow TCP Stream” if you want to look at other packet data.

iii) Conversations and Endpoints

Under the statistics menu at the top of the main screen you can explore “Conversations” and “Endpoints”.

First, remember that the network traffic you capture may have traffic to/from more than one computer. There is a good chance that your LAN protocol is Ethernet, and Ethernet is designed to share a single network among many users. As a result, you may see packets for other users in your packet data. Even if your network is connected through a switch, you may see broadcast packets to other users.

Using endpoints lets you isolate traffic so that you are only looking at traffic to/from a specific machine. An endpoint can be defined by network layer. For example, a single MAC address on your machine is one endpoint. If you are running an email client and a web browser at the same time, all of that traffic will be consolidated through your computer’s MAC address. However, if at the TCP layer, an endpoint definition includes the port number of the application. Therefore, at the TCP layer, the traffic for the email client and the web browser will be separated. Wireshark's endpoint report lets you select the network layer of interest, and then to see the summarized endpoint traffic for that layer.

A conversation report is similar to an endpoint report. A conversation is defined as all of the traffic between two specific endpoints. As an example, consider packets at the TCP level. Let's say that you started capturing packets and then went to two web sites: www.cnn.com and www.usatoday.com. The endpoint report on your web browser will combine all traffic from your browser and both of these web sites. A conversation report between your browser and the www.cnn.com site would exclude the data from www.usatoday.com.

VI) Saving Captures

Wireshark also allows you to capture a set of packets and save it to a file that can be opened later. In addition to the obvious uses, this allows two unique capabilities.

- Instructors may wish to save one capture file and distribute it to all students. This allows instructors to pose a set of questions on a consistent data set, and to know that each student has appropriate data to answer the questions.
- In some circumstances, for example using a wireless network connection, students may have difficulty capturing packets. In these cases, Wireshark will still be able to analyse packets from saved files created on another platform or with other tools. These students can capture a set of packets on any accessible machine; save the captured packets; and transfer the saved file to their personal machine for analysis.

Appendix 1: Packets Captured: Explanation and Troubleshooting

Wireshark is designed to show you all packets that come into and out of your computer. You are probably using Ethernet for your LAN, and Ethernet is a shared-access protocol. As a result, Wireshark would theoretically allow you to see the following types of traffic:

- Packets sent to/from your computer.
- Broadcast packets sent to all computers on your local network.
- Packets sent to/from any other computers on your local network.

However, several factors may keep you from seeing some of the packets on your network.

I) Switches or Routers versus Hubs

Ethernet assumes that your local network looks like some variation of a bus, and that traffic to any computer on the local network will be seen by any other computer on that network.

In practice, Ethernet networks often use a star topology, wherein all of the computers are linked to a central unit. In the early days of Ethernet, this central unit was called a hub. A hub listens to each incoming port and repeats everything that it hears out to every port. Although a hub's physical network topology is a star, logically it acts like a bus topology – every station on the network sees all of the traffic on the network. Therefore, if your network uses a hub, your machine should be able to report both the traffic to your machine and also the traffic to all other machines on your network.

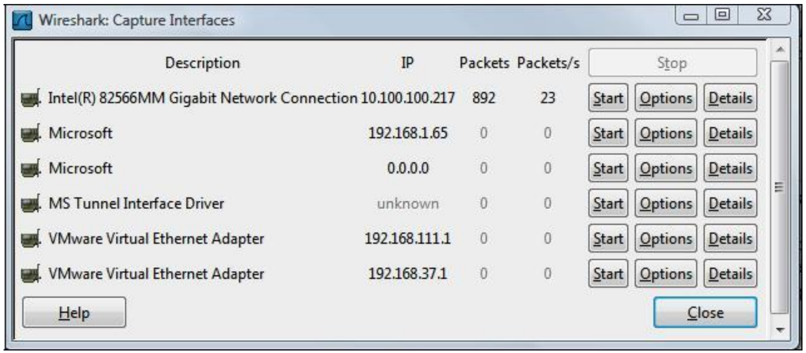
The problem with hubs is that they reduce capacity since each station must pick their packets out of a lot of irrelevant traffic for other stations. Today, it is more normal to build networks switches and routers. You can refer to your textbook for a description of the differences in these devices. However, the simple explanation is that they work to insure that each station only sees the traffic that it needs to see. It is likely that your network's central unit is a switch or a router. If this is the case, your computer (and Wireshark) will be able to see traffic that is addressed to/from your computer and broadcast traffic for all computers on the network, but you will not be able to see packets sent to/from other computers that are not addressed to your computer.

Some higher-end switches have the capability to duplicate all traffic passing through the switch and to send the copied traffic to a single port. This may be done by an administrator during a troubleshooting exercise and is normally disabled. This feature is known variously as "port mirroring" or "port spanning"

II) Your Network Adapter

Many computers today have more than one network adapter. For example, many laptops have both wireless network adapters (802.11 a/b/g) and wired adapters. You must make sure that Wireshark is listening to the correct adapter or it will not see any traffic. You can check which adapters are receiving data by clicking on the "Capture" menu then selecting "Interface". In Figure 7 you can see that Wireshark believes that there are six interfaces, but that only the first one is receiving packets. From this dialog you can select to:

- start a capture on a specific interface
- configure options before starting a capture on a specific interface
- view details of a particular interface



Description	IP	Packets	Packets/s
Intel(R) 82566MM Gigabit Network Connection	10.100.100.217	892	23
Microsoft	192.168.1.65	0	0
Microsoft	0.0.0.0	0	0
MS Tunnel Interface Driver	unknown	0	0
VMware Virtual Ethernet Adapter	192.168.111.1	0	0
VMware Virtual Ethernet Adapter	192.168.37.1	0	0

Figure 7 Captures/Interface dialog

The default adapter is setup in the menu “Edit/Preferences/Capture” – make sure you choose to save any changes using the dialog button at the bottom of the window. You can alter the selected interface for a single capture by going through the “Capture Options” dialog (see Figure 1)

One of the options in the capture settings is to set “promiscuous mode”. Typically, network adapters will screen out all traffic that is not destined for the computer. With this setting Wireshark will send a message to your network card telling it to pass through all traffic it sees. Even if you are on a broadcast, or hub-type network, Wireshark may not report traffic from/to other computers if promiscuous mode is not turned on.

III) Comment on Cable Modems

Typically, high-speed cable internet connections are shared connections. Theoretically, this means that you should be able to see the network traffic of your neighbours who have cable modems when you use Wireshark. The data entering your premises may include traffic from your neighbours. However, in many (most?) cases this neighbour-traffic is not visible inside your local network. Cable companies typically implement filtering and even authentication services inside their modems that eliminate packets that are not destined for the local system.

IV) Problem with Wireless LANs and Windows

Wireshark may not be able to report packets on a Windows computer using a wireless (802.11 a/b/g) adapter. One suggested workaround is to try turning off promiscuous mode. You can find this setting in the Edit menu under the Preferences menu choice. Once the resulting dialogue box appears, click on the “Capture” menu choice on the left side. Clear the check box so that “Capture packets in promiscuous mode:” is not checked. Click on the “Save” button at the bottom of the screen, and finally, click on the “OK” button at the bottom of the screen. On some monitors the OK button may be off of the bottom of the screen; your settings will ***not*** be saved if you click another button. Furthermore, your changes will be lost if you close the window by clicking on the x in the top right corner of the window.

As an alternative, Microsoft has a similar free product called “Network Monitor” which can analyze 802.11 packets (free, but not open source). For more information see http://blogs.technet.com/netmon/

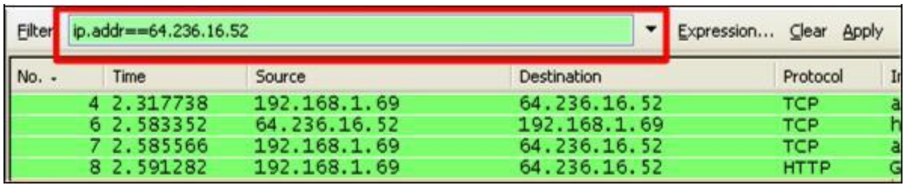
Appendix 2: Filters in Wireshark

Wireshark can filter results so that you only see certain packets. An example of a filter condition would be to only remember packets sent to/from a specific IP address.

Wireshark uses two types of filters, capture filters and display filters. Capture filters are used to decide which packets should be kept. Only packets that meet filter criteria will be kept. Display filters work after the capture is completed. They restrict which packets are shown, but they don't actually discard any information. Capture filters would be more useful on very busy networks when you need to limit the amount of data your machine needs to process. On the other hand, display filters don't actually save any memory; display filters let you temporarily focus an analysis without losing any underlying information.

Capture filters can be set in two different places. Go to the Capture menu and select "Options" and you will find a selection for capture filters. Alternatively, Go to the Capture menu and select "Capture Filters". From the "Capture Filters" dialog box you will see a help menu that will explain how the function works.

Display filters can be entered at the top of the display screen. Figure 8 below shows a display filter entered into the display filter dialog box at the top of the screen.



Filter: ip.addr==64.236.16.52

```
No. -   Time       Source          Destination     Protocol  I
    4 2.317738   192.168.1.69    64.236.16.52    TCP       a
    6 2.583352   64.236.16.52    192.168.1.69    TCP       h
    7 2.585566   192.168.1.69    64.236.16.52    TCP       a
    8 2.591282   192.168.1.69    64.236.16.52    HTTP      G
```

Figure 8: Using Display Filters

The display filter shown in the image above will only display packets if they are from/to IP address 64.236.16.52. This specific filter limited packets to those involved with CNN.com. If you also captured traffic to USAToday.com, you would not be able to see it until you clicked on "Clear" to the right of the filter area. A more specific filter to restrict the display of packets within a single session would be "(ip.addr eq 64.236.16.52 and ip.addr eq 192.168.1.69) and (tcp.port eq 80 and tcp.port eq 1102)". In this case both endpoints are explicitly selected (both IP and ports used in the session). Some commands, such as "Follow TCP Stream" automatically enter values in the filter field. After you use a command like this, you may need to "Clear" the filter to see the complete set of packets.

Appendix 3: Hits Versus Page Views

It may take more effort than you realize to deliver a web page to your computer. The first step is to get the raw HTML code for the page. Getting this code takes several sets of packets – the details will be left to an exercise to be completed later, but suffice it to say that retrieval includes setup and control packets as well as query and response packets. Furthermore, in most cases the response will be a multi-packet data burst that must be reassembled into a complete http response.

However, once the page is delivered to the application, the system has only completed the first step required to display the web page. Let's consider a simplified web page in HTML, as shown in the box below (Figure 9).

```
<HTML>
<Body>
Look at this pretty Christmas tree.<br> <img src=tree.jpg>
</Body>
</HTML>
```

Figure 9: Simplified Web Page

This web page will display a short sentence (Look at this pretty Christmas tree.), followed by a line break, and then a picture of a tree. Notice that the picture of the tree is not part of the HTML page that is delivered. All that gets delivered with the page is a placeholder that tells the browser to get the picture called tree.jpg and to put it into a specific spot on the page.

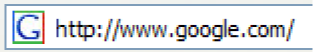
So, once the browser deciphers the web page, it knows it must make another request of the web server. Now the browser asks for the picture tree.jpg. As a result, displaying this page takes two hits on the browser. One hit (or request) was for the original web page, and the second hit was for the picture to be embedded into the web page. Each additional picture or external page element is another hit on the web page.

How many pictures are on a single page? 10? 20? A recent analysis of the CNN front page indicated over one hundred and fifty separate files were required to display the page. A lot of these files are graphic files. This includes tiny graphic arrows, almost invisible lines, menu choices, and advertisements. In addition, javascript files, stylesheets, and iFrames can all be external links, and thus can be additional sources of hits.

Especially in the case of advertisements, these hits may not come from the original web site. Therefore, at the packet level there may be many packets from many different sources that have to be considered as part of the same web page. Increasingly, developers are making dynamic web pages. This means that some portion of the web page may be continuously updated through interaction between the user and the server. This dynamic process requires ongoing hits on the server, even after the web page is initially 'complete'.

Since each of these hits results in a new request from the server, the number of packets required to assemble a web page is larger than many people realize.

Hint: a favicon.ico is a small graphic that can be used as an icon to identify a web page. In the following graphic the colorful "G" to the left is a favicon.ico.



http://www.google.com/