

Lecture 4: Socket programming cont and Wireshark demo

Sample code explanation.

<i>udp server</i>	<i>udp client</i>
create server address struct	create server address struct
create server socket	create client address struct
bind server socket to server address	create client socket
loop	bind client socket to client address
receive from client address via	loop
server socket (also obtained the client	
address)	send to server address via
process info	client socket
end loop	receive from server address
	via client socket
	end loop

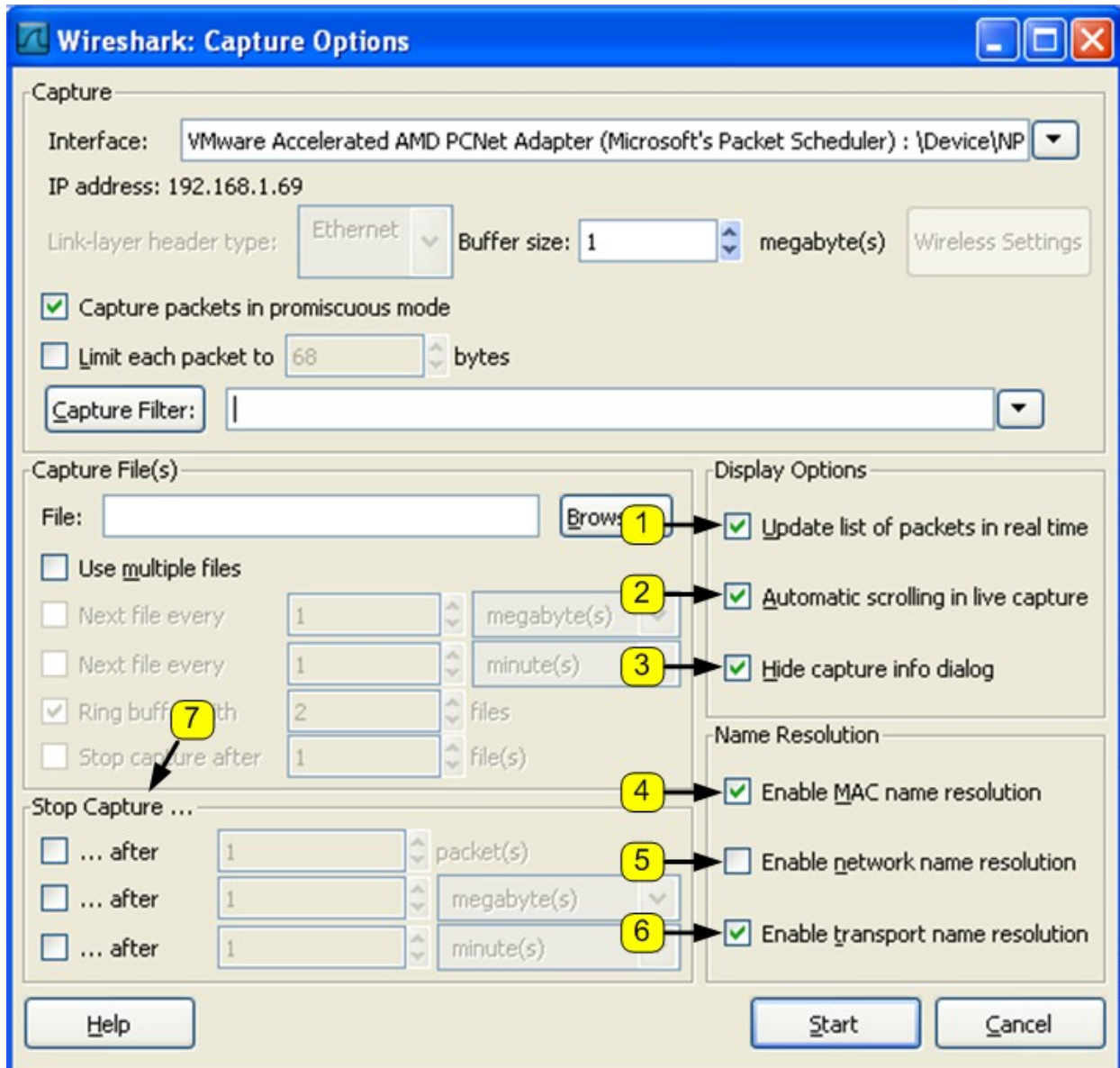
<i>TCP server</i>	<i>TCPclient</i>
create server address struct	create server address struct
create server socket	create client address struct
bind server socket to server address	create client socket
convert to a listening socket	connect client socket to host
loop	loop
accept client request (also get	write to the client socket
client address and a new file	
descriptor)	
process info	receive from client socket
end loop	end loop

Wireshark Demonstration

- History: Originally it was called Ethereal (name changed in 2006), a well known open source to monitor network activities.
- A packet sniffer (from the internet layer) for Ethernet
- The primary goal of this illustration is how to run wireshark to analyze the network activity.

1. How to capture packets

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”. You should see a dialog as in the following figure 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 including:



- 1) *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).
- 2) *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).
- 3) *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)
- 4) *Enable MAC name resolution*: This tells Wireshark to display the name of the manufacturer of the network card when it lists the MAC address. the following figure shows an example of MAC name resolution with a MAC address generated from an Asiarock network card (default is on).

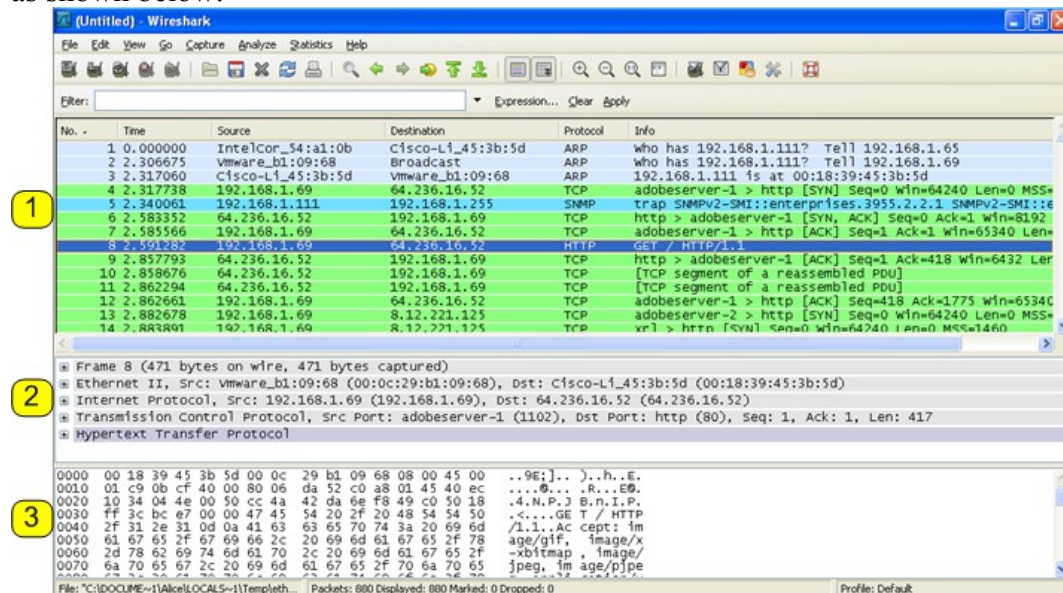
Ethernet II, Src: Asiarock_0f:33:6b (00:0b:6a:0f:33:6b),
 Destination: Broadcast (ff:ff:ff:ff:ff:ff)
 Source: Asiarock_0f:33:6b (00:0b:6a:0f:33:6b)
 Type: ARP (0x0806)

- 5) *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.

- 6) *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)
- 7) *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).

Once you have captured a set of packets, Wireshark should present you with a colorful window as shown below.



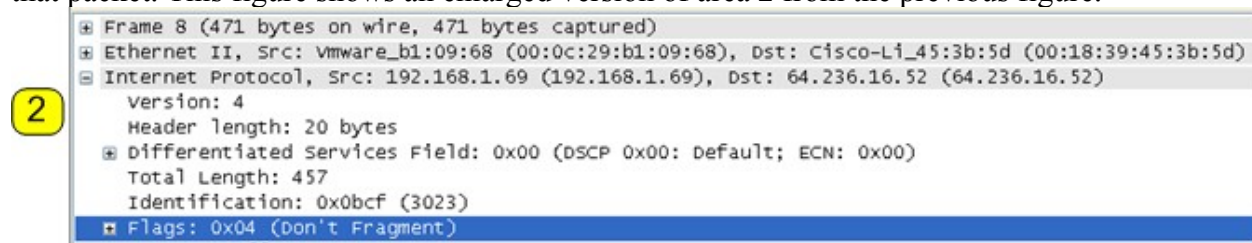
This window is divided into three areas.

Window Area 1: Summary

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

Window Area 2: Detail

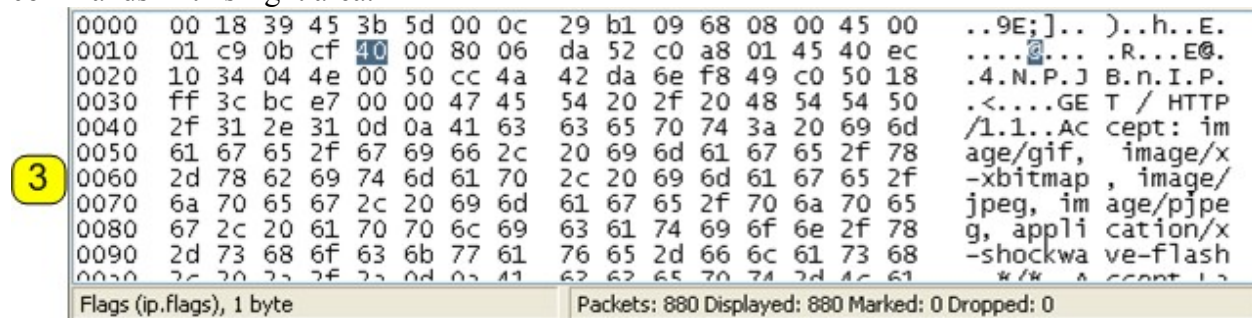
When you click on a packet in area 1, the packet structure is shown in area 2. In the screenshot above, the packet shown in dark blue has been selected; therefore area 2 shows more details on that packet. This figure shows an enlarged version of area 2 from the previous figure.



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.

Window Area 3: RAW Data

Clicking on a portion of the packet in area two changes the display in area 3. 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.



Wireshark has several options to explore and analyze captured data.


Filters

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. The figure below shows a display filter entered into the display filter dialog box at the top of the screen.



No.	Time	Source	Destination	Protocol	Ir
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

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 USA Today.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.

Follow TCP Stream

Choose a TCP packet from the packet listing window. 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” the display filter after using “Follow TCP Stream” if you want to look at other packet data.

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.ashland.edu and www.google.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 Ashland site would exclude the data from google.

2. A demonstration of HTTP packet catching

use browser to go to the host in our lab and apply filter to check the information sent over port 80

- Take the http server for example.
- Note on three-way hand shake (syn – **syn/ack** – ack)
- To close a connection, we need 4 packages. (FIN/ACK—**ACK** – **FIN/ACK** – ACK)
bold letters are server actions

Note: You can apply filters to see the transmissions of your interest.

3. Demonstrate the difference between TCP sockets and UDP sockets using the echo server code.

4. Exercise to analyze packets (see exercise sheet)