Assignment 3-1: Wireshark Viewing Protocols in Action

Name: _____

Objective

The purpose of this exercise is to introduce you to the TCP/IP applications and protocols by viewing the packets captured during a live session. It will also give you some idea of the volume activity that is constantly going on within our devices.

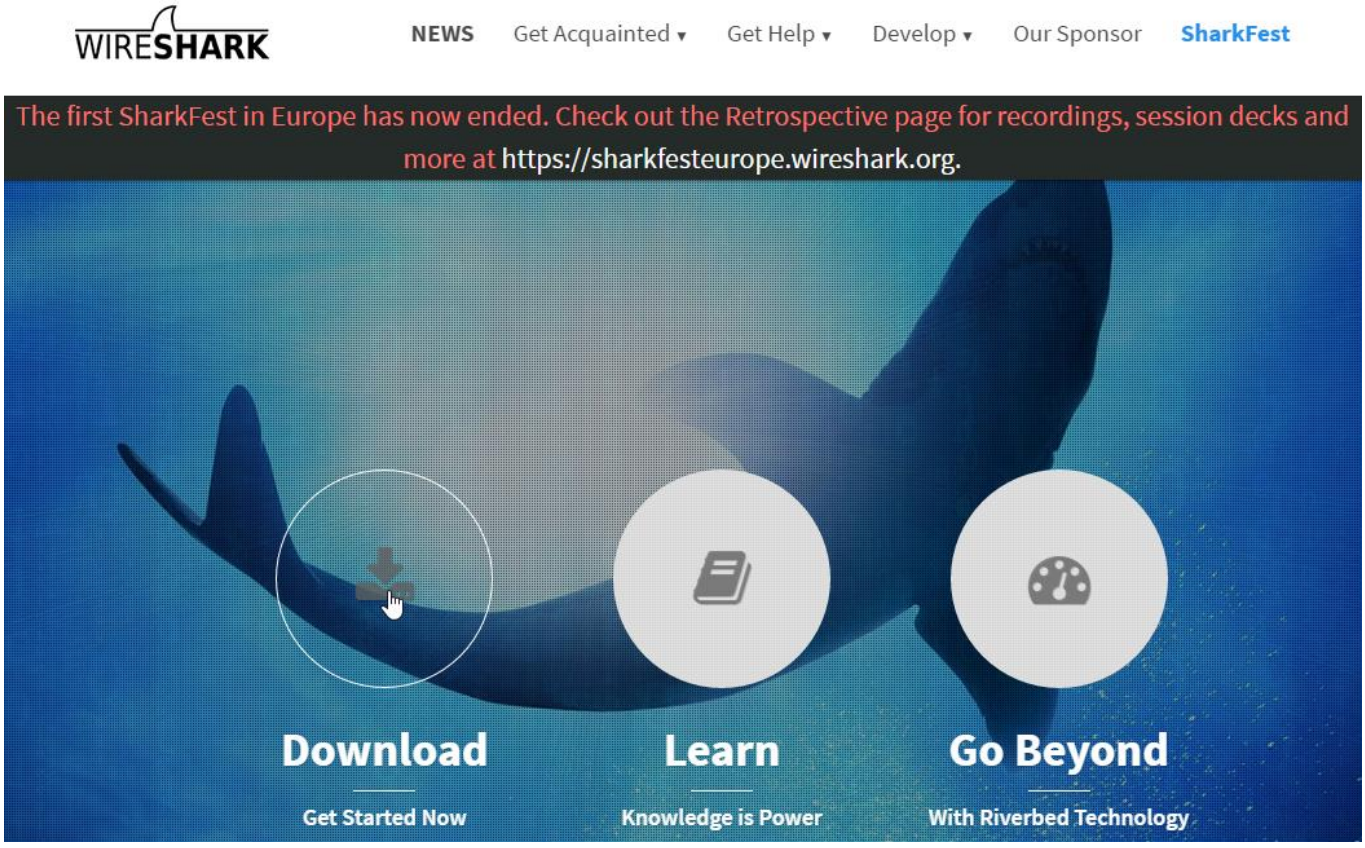
Background

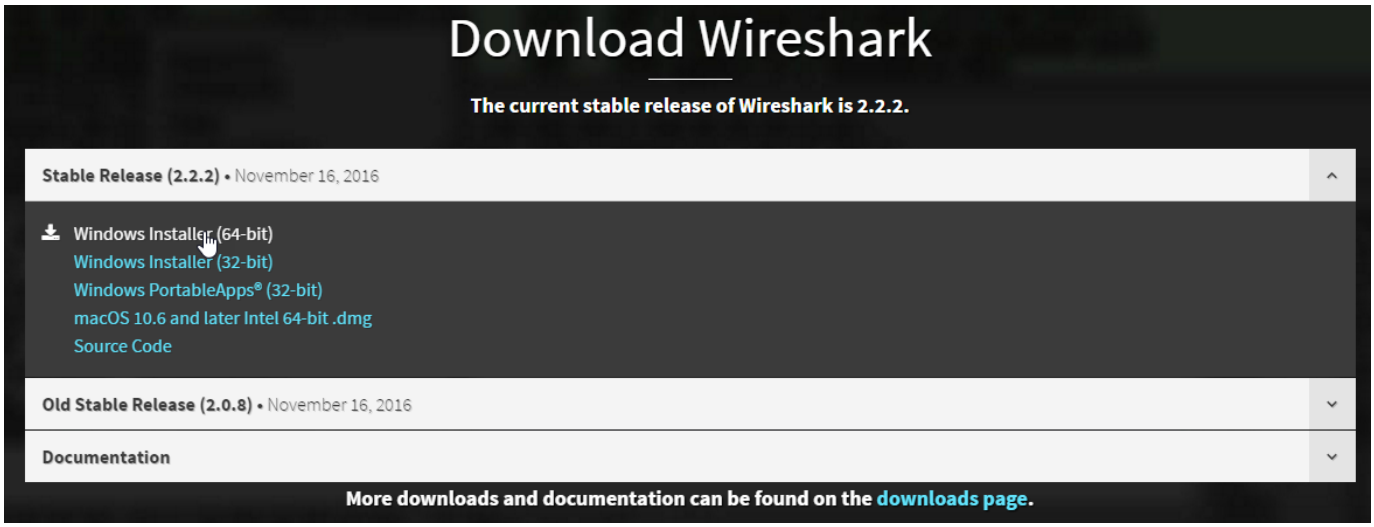
Wireshark is a free open source packet analyzer that is at home both in education and in the hands of professionals working in the field. It allows you to capture, view, and analyze IP traffic patterns as well as individual packets. There are a lot of resources on the Web to learn more about the tool and how it is used. Mastering it should be added to the to-do list of anyone planning to work in the networking or security industries.

A good place to start is the source, www.wireshark.org/. This site has many resources including videos and a page to download the latest version. Wireshark is also a very popular technology topic on YouTube where there are many videos, ranging from beginner tutorials to specialized uses of the tool.

Task 1: Acquiring and Installing Wireshark

Go to the Wireshark website (https://www.wireshark.org/) and look over the site and any of the learning resources. Or, go directly to the downloads link and choose the correct version for your computer: Note – the site changes periodically.

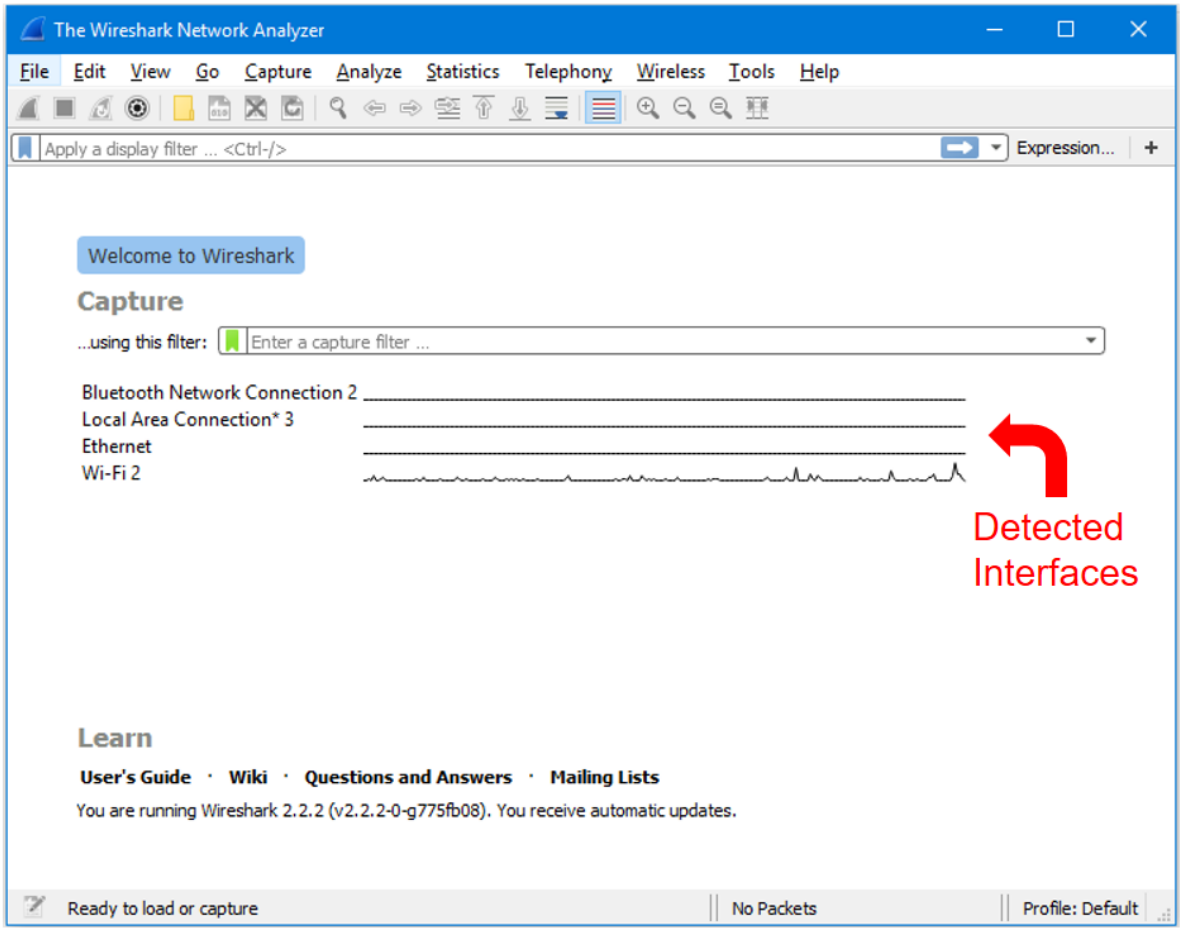




During installation, you will be advised that you also need to install a version of Pcap library - WinPcap is the Windows version; this library supports capturing packets on live network data. Accept the recommended options. The whole process should take about a minute.

Task 2: Capturing and Viewing IP Traffic

When you launch Wireshark, you will get a screen similar to this. Yours may vary a bit in appearance with newer versions or the operating system you are using, but you will recognize the features and options.



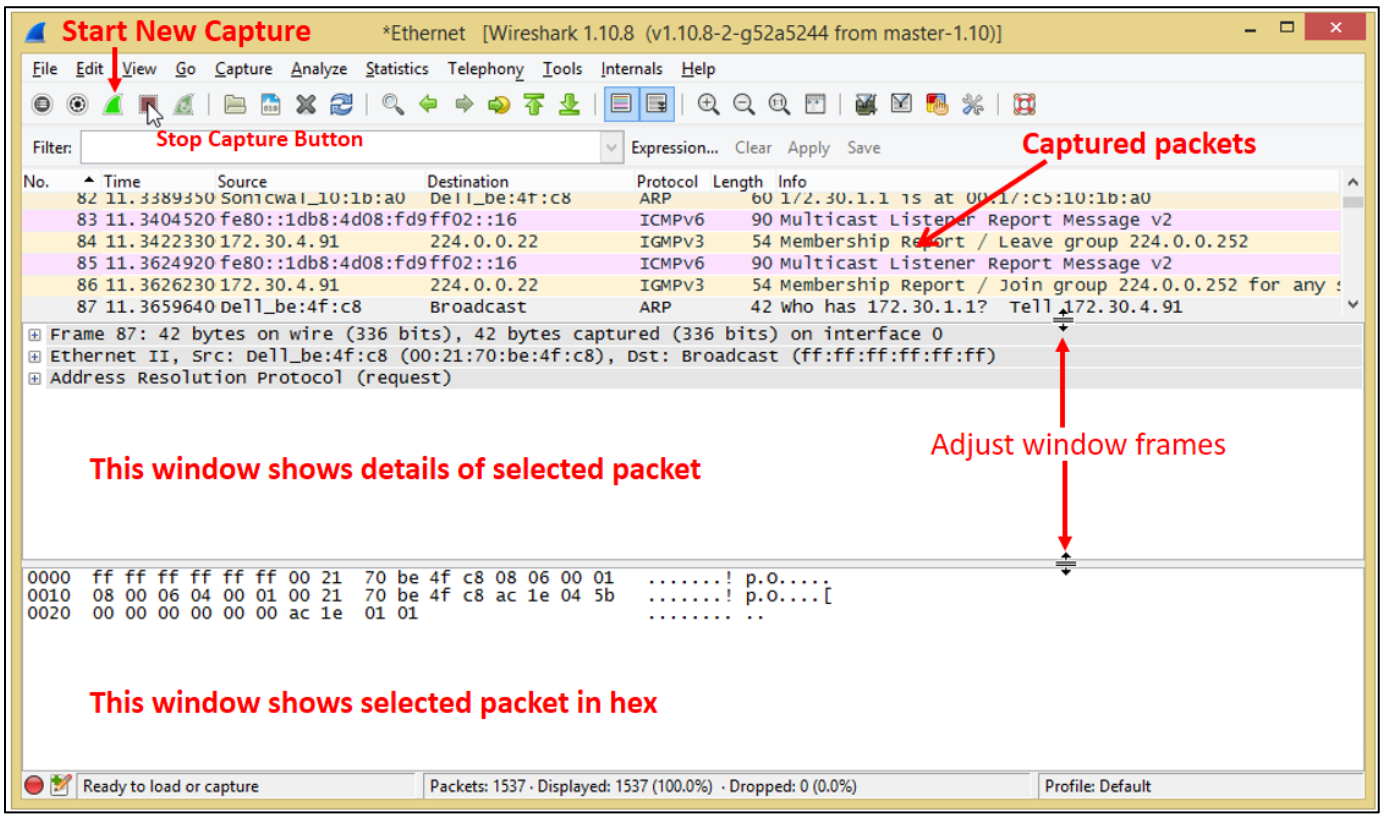
The detected interfaces will start to show if any traffic is present with a wavy line pattern, my WiFi 2 above. Double-clicking on it will select the capture interface and start a capture session.

You should start to see activity on your screen. The volume and speed will depend on what all is going on with your computer. Closing applications and browsers will slow it down if you like.

After a few minutes, click on **stop capturing packets**  icon at the left end of the toolbar to stop your capture.

Task 3: Quick Tour

Like many user-friendly applications today, Wireshark offers several ways to use the tool. Whether you prefer menus, the toolbar buttons, or the startup screen, you should find a method familiar to you. In this exercise, you will use the menus and toolbars since they are common to all versions.



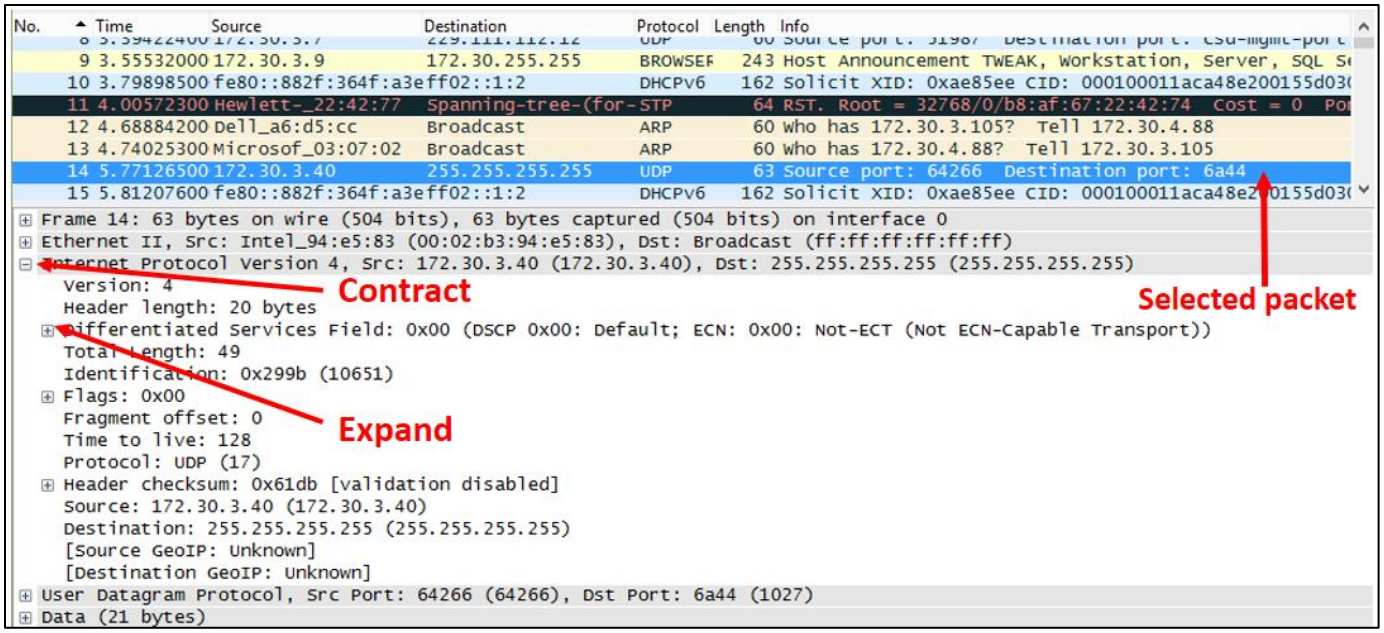
Take a moment to pause your mouse over each of the toolbar buttons; a description will appear telling you what each button does.

Open each of the first few menus to see that there are menu options to do the same things the toolbar buttons do.

Note there are three horizontal sections to the window.

The **upper section** is a list of the packets you captured. Note on the left end they are numbered sequentially – this is your capture order. The packets can be sorted on any of the columns by clicking on the column label at the top. Try a couple; you can always return to this order by clicking on the **No** at the top of the first column.

The **middle section** shows the details of any packet you select in the top section. Select a couple of rows in the top section and notice that the middle section shows what it is. Don't worry if it is all foreign to you today. Note some of the rows have a box with a plus in it (or a >), clicking on that will expand the line to show more details. A box with a minus (or ⌄) contracts the display back to one line.



The **bottom section** shows the actual bits in Hexadecimal that make up the packet selected in the top. If you select a row in the middle section, the bottom will highlight those specific bits. To the right of the hexadecimal values are any character equivalents to the Hex. As you select different types of packets, you will see some text appear in that area. In most cases, it will be just gibberish.

The horizontal bars that separate the sections can be dragged up or down to change the sizes of the sections. I find that I have little use for the bottom sections, so I tend to make it small, the middle section small, and thereby making my top section larger.

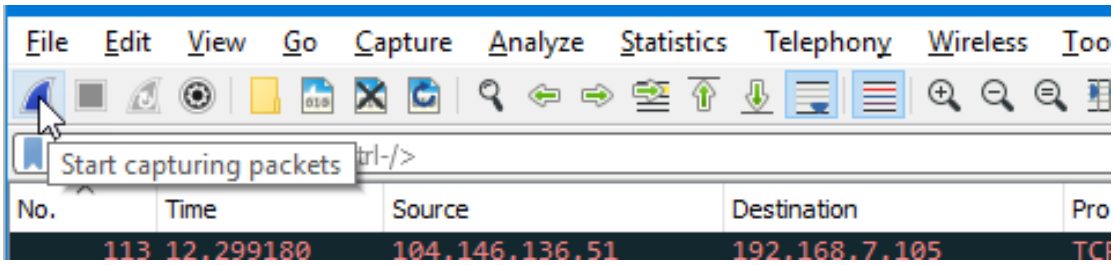
Experiment with these controls until you are comfortable with each – you can't hurt your data. These are just copies of data that passed through during our capture.

Look at any of the packets in your capture noting the source and destination, the protocol, and the size (Length). If you sort them by Protocol, you can easily see examples of some of the protocols discussed in the lectures and videos.

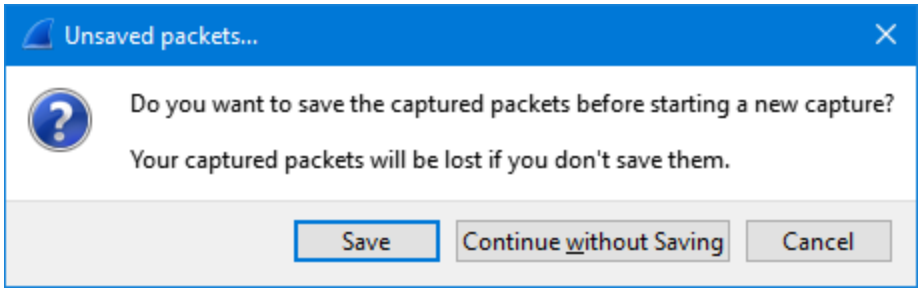
Task 4: Capturing Specific Traffic

Close any browsers and applications that may be running on your computer to reduce the amount of traffic captured. You can always come back later and experiment with other programs running.

Click on the **Start capturing packets** button.



You will be prompted to save your existing capture or discard it. Choose **Continue without saving**.



Open a command prompt (Start | Run | CMD) and clear your ARP table as we did in the last exercise (**arp -d**).

Release and renew your IP address information as we did in the last lab – **ipconfig /release** and **ipconfig /renew**.

After the interface gets a new address, ping Google (**ping www.google.com**).

Close the command window and return to the capture window and click **Stop**, or choose the **Stop Capture** on the main menu. Wireshark displays the capture window (top), a packet display window (middle), and a details window (hex) at the bottom.

Task 5: Sorting Tips

#1 – Sorting on the **Number** (No) and **Time** columns do the same thing. The number column is the packet sequence order, so the smaller numbers equal earlier times.

No.	Time	Source	Destination	Protocol	Length	Info
623	109.454191	192.168.7.105	35.164.191.79	TCP	54	58827→44
624	110.367098	40.97.143.146	192.168.7.105	TLSv1.2	139	Applicat
625	110.367141	40.97.143.146	192.168.7.105	TLSv1.2	1003	Applicat

#2 – Click on a column header to sort by that column in **ascending** order. If you click on it again, they will sort in reverse (**descending**) order. It is a two-way toggle. Note the arrow.

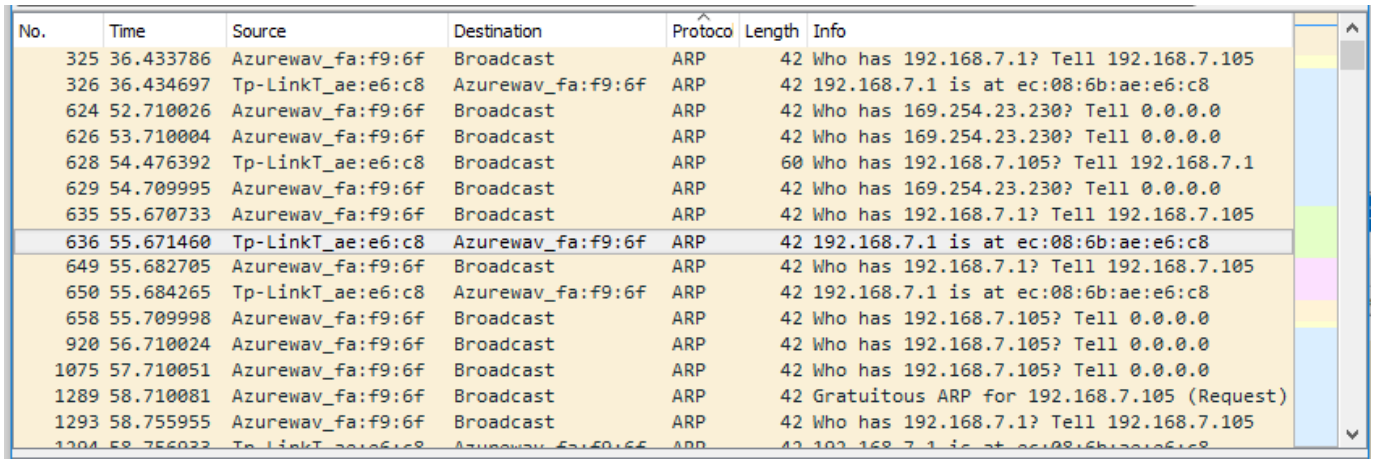
No.	Time	Source	Destination	Protocol	Length	Info
249	46.051369	WesternD_ec:54:ed	Azurewav_fa:f9:6f	ARP	60	Who
250	46.051398	Azurewav_fa:f9:6f	WesternD_ec:54:ed	ARP	42	192.
570	102.506665	Tp-LinkT_ae:e6:c8	Azurewav_fa:f9:6f	ARP	42	Who

No.	Time	Source	Destination	Protocol	Length	Info
966	148.621676	192.168.7.1	255.255.255.255	UDP	215	4111
949	142.478058	192.168.7.1	255.255.255.255	UDP	215	4111
921	139.405781	192.168.7.1	255.255.255.255	UDP	215	4111

#3 – Sorts are **NOT** cumulative. Any sort starts from the original sequence order. So, if you sort by (packet) **Length** and then by **Protocol** we might think they would be in Length order. It turns out that we would be wrong. They are in sequence order.

Task 6: Looking at your captured data

Click the **Protocol** column header. The packets are displayed in the order of their type. ARP packets should show up first, with others following in alphabetical order.

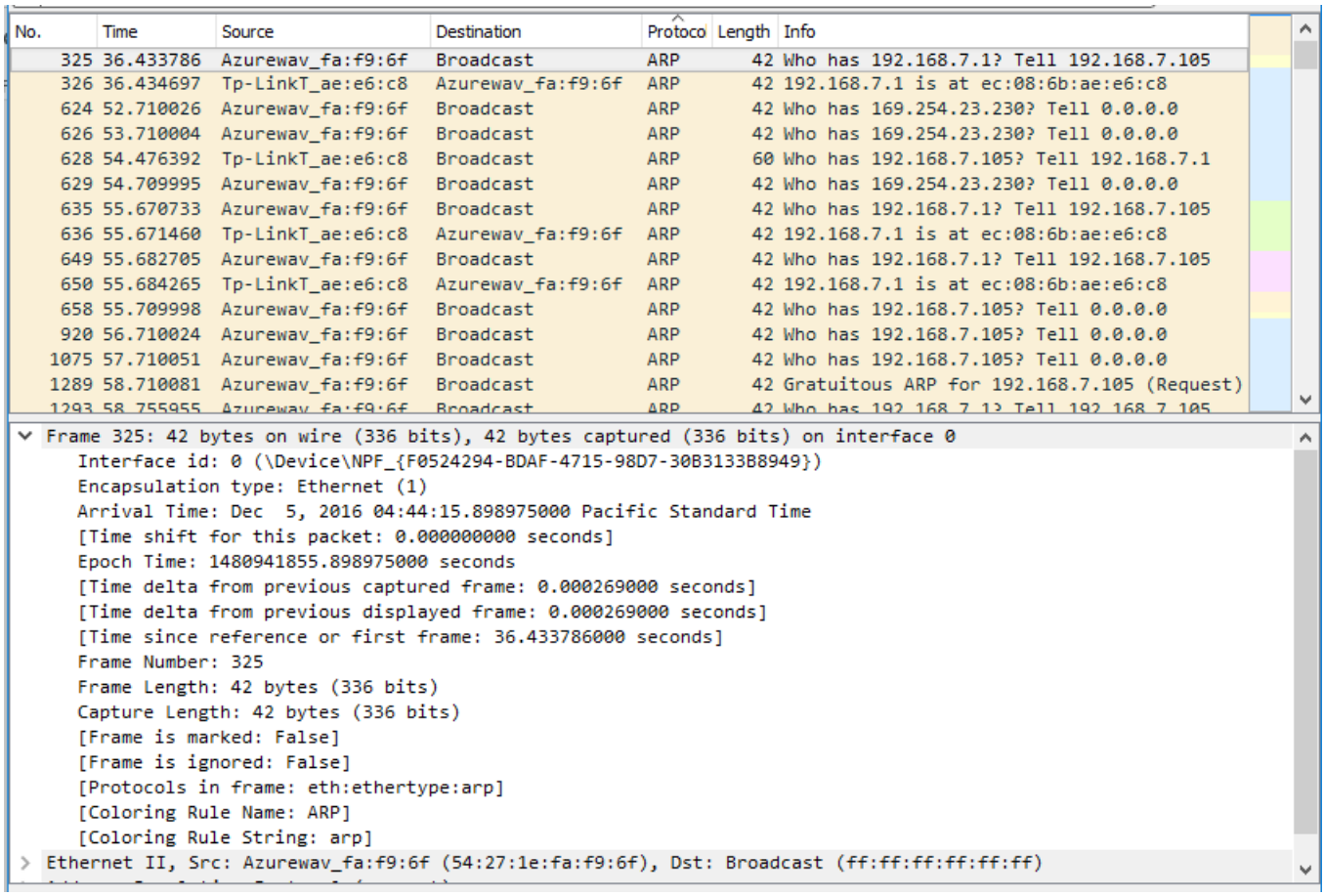


No.	Time	Source	Destination	Protoco	Length	Info
325	36.433786	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 192.168.7.1? Tell 192.168.7.105
326	36.434697	Tp-LinkT_ae:e6:c8	Azurewav_fa:f9:6f	ARP	42	192.168.7.1 is at ec:08:6b:ae:e6:c8
624	52.710026	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 169.254.23.230? Tell 0.0.0.0
626	53.710004	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 169.254.23.230? Tell 0.0.0.0
628	54.476392	Tp-LinkT_ae:e6:c8	Broadcast	ARP	60	Who has 192.168.7.105? Tell 192.168.7.1
629	54.709995	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 169.254.23.230? Tell 0.0.0.0
635	55.670733	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 192.168.7.1? Tell 192.168.7.105
636	55.671460	Tp-LinkT_ae:e6:c8	Azurewav_fa:f9:6f	ARP	42	192.168.7.1 is at ec:08:6b:ae:e6:c8
649	55.682705	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 192.168.7.1? Tell 192.168.7.105
650	55.684265	Tp-LinkT_ae:e6:c8	Azurewav_fa:f9:6f	ARP	42	192.168.7.1 is at ec:08:6b:ae:e6:c8
658	55.709998	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 192.168.7.105? Tell 0.0.0.0
920	56.710024	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 192.168.7.105? Tell 0.0.0.0
1075	57.710051	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 192.168.7.105? Tell 0.0.0.0
1289	58.710081	Azurewav_fa:f9:6f	Broadcast	ARP	42	Gratuitous ARP for 192.168.7.105 (Request)
1293	58.755955	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 192.168.7.1? Tell 192.168.7.105

Note these are Layer 2 exchanges – the source and destinations are MAC addresses not IP. Notice that the MAC addresses have substituted Manufacturer label for the first half of the MAC address.

In the above example, row 1 is my WiFi radio discovering the default gateway with a broadcast and asking the router to tell 192.168.7.105 (my WiFi) so that when the router replies (row 2), it doesn't have to do a broadcast. The request destinations are broadcasts, but all the replies are unicasted (to one address) reducing network traffic.

Select an ARP packet by clicking it once. In the middle window, you should see that all information about the packet is displayed, including the frame type, the protocol used, flags set, and addressing. Your display should look something like the following:

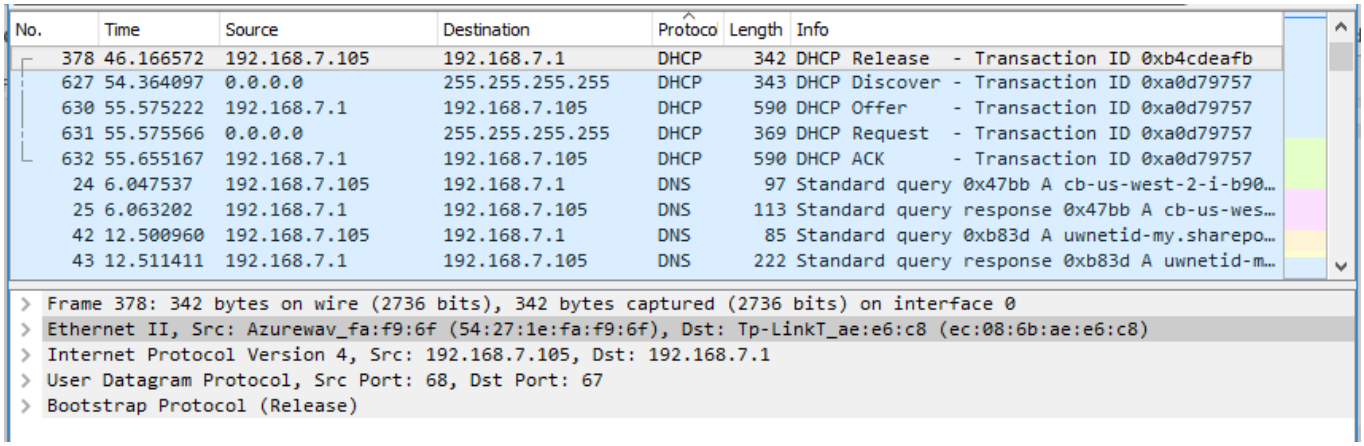


No.	Time	Source	Destination	Protoco	Length	Info
325	36.433786	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 192.168.7.1? Tell 192.168.7.105
326	36.434697	Tp-LinkT_ae:e6:c8	Azurewav_fa:f9:6f	ARP	42	192.168.7.1 is at ec:08:6b:ae:e6:c8
624	52.710026	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 169.254.23.230? Tell 0.0.0.0
626	53.710004	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 169.254.23.230? Tell 0.0.0.0
628	54.476392	Tp-LinkT_ae:e6:c8	Broadcast	ARP	60	Who has 192.168.7.105? Tell 192.168.7.1
629	54.709995	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 169.254.23.230? Tell 0.0.0.0
635	55.670733	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 192.168.7.1? Tell 192.168.7.105
636	55.671460	Tp-LinkT_ae:e6:c8	Azurewav_fa:f9:6f	ARP	42	192.168.7.1 is at ec:08:6b:ae:e6:c8
649	55.682705	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 192.168.7.1? Tell 192.168.7.105
650	55.684265	Tp-LinkT_ae:e6:c8	Azurewav_fa:f9:6f	ARP	42	192.168.7.1 is at ec:08:6b:ae:e6:c8
658	55.709998	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 192.168.7.105? Tell 0.0.0.0
920	56.710024	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 192.168.7.105? Tell 0.0.0.0
1075	57.710051	Azurewav_fa:f9:6f	Broadcast	ARP	42	Who has 192.168.7.105? Tell 0.0.0.0
1289	58.710081	Azurewav_fa:f9:6f	Broadcast	ARP	42	Gratuitous ARP for 192.168.7.105 (Request)

```
Frame 325: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
    Interface id: 0 (\Device\NPF_{F0524294-BDAF-4715-98D7-30B3133B8949})
    Encapsulation type: Ethernet (1)
    Arrival Time: Dec  5, 2016 04:44:15.898975000 Pacific Standard Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1480941855.898975000 seconds
    [Time delta from previous captured frame: 0.000269000 seconds]
    [Time delta from previous displayed frame: 0.000269000 seconds]
    [Time since reference or first frame: 36.433786000 seconds]
    Frame Number: 325
    Frame Length: 42 bytes (336 bits)
    Capture Length: 42 bytes (336 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:arp]
    [Coloring Rule Name: ARP]
    [Coloring Rule String: arp]
> Ethernet II, Src: Azurewav_fa:f9:6f (54:27:1e:fa:f9:6f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
```

Use the scroll bar to the right of the packet capture window to move down to the first DHCP packet. Notice that you have the entire DHCP process from release to DORA (Discover | Offer | Request | ACK – acknowledge).

After viewing the information in the middle window, simply arrow down to view the entire DHCP release and renew process.



Note that both DHCP and DNS are Layer 3 exchanges – source and destination addresses are IP. The DHCP **Discover** and **Request** are both broadcasts from a device without an IP address.

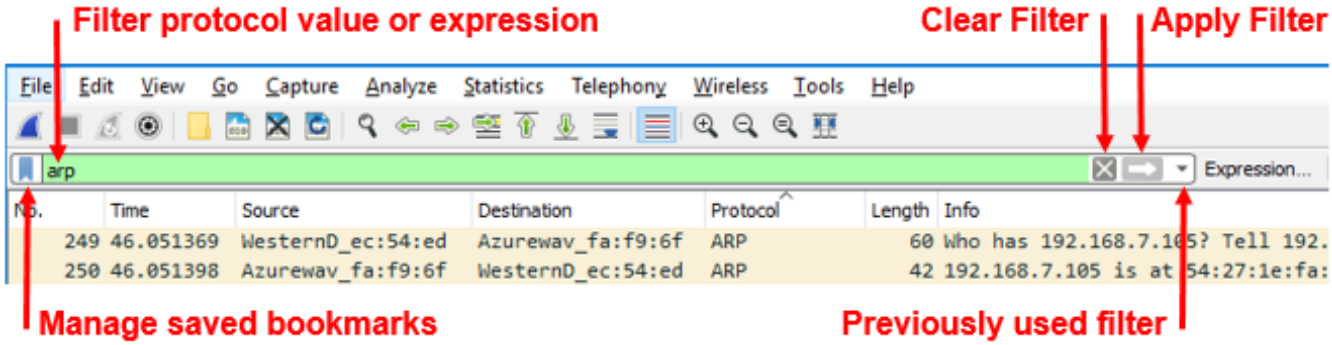
Repeat the above steps to view other types of packets and protocols. You may want to do additional captures with your browser running or while sending and receiving e-mails. We will return to this tool in future modules as we discuss other types of frames and packets.

Task 7: Filtering Output

You can filter both your display and your capturing. Let's start with the display once you have a capture. If you want to display a single protocol type, you can just type the protocol in the filter bar and press **Enter** or click on the **Apply Filter** button. The other rows are hidden (but not lost) until you click on the **Clear Filter** button.

Tip: as you type the filter bar will change colors. Green means it understands what you are saying, red means it does not. Don't panic if it starts out red – in my example; it was red until I typed the "p."

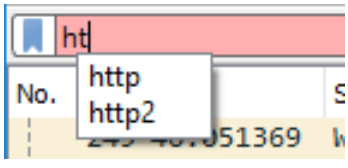
Type **arp** and press **Enter** or click on **Apply Filter** button. Filters must be in lower case.



To type another one, you can often type over the existing entry, but occasionally you have to click on the **Clear Filter** button to release the old filter. The tip-off is the bar stays red.

Clear the ARP filter and then type **DNS**.

Note: Wireshark was created by networkers, for networkers who typically are coders or scripters and love abbreviations. So more complex filters can at first seem rather cryptic. A drop-down list will appear as you type to show you possible options and their syntax. You can at any time click on the one you want to use.



Try **HTTP** and **UDP**. You will notice that UDP appears not to work because other protocols also appear. That is because they use UDP datagrams. The same is true with **TCP**. UDP and TCP are broader terms that define the container many other protocols use. While all HTTP packets are TCP packets, not all TCP packets are HTTP packets.

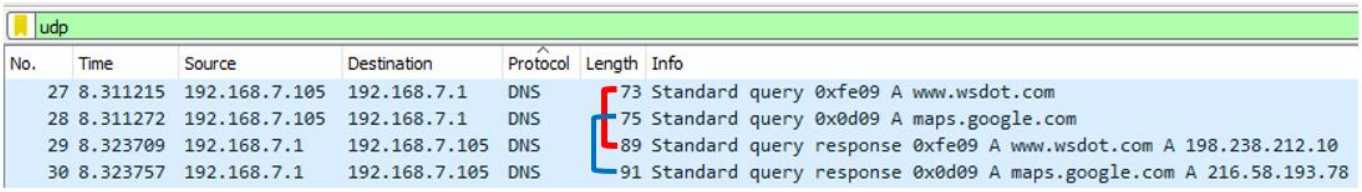
You can rerun any filter from this session by using the **Previously used filter** drop-down at the end of the filter bar.

Use the **Previously used filter** drop-down to rerun any of your earlier filters.

Note: If you do not clear your filters when you start a new packet capture, it will appear that you only captured packets that match the filter – until you clear the filter. We will look at filtering captures later.

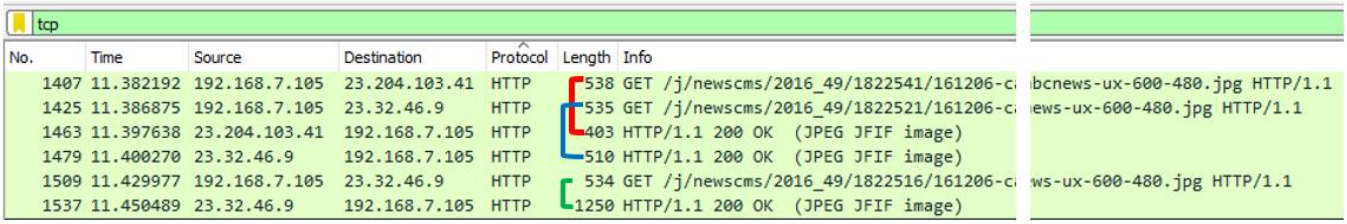
Looking at Packet Exchanges

DNS Requests: This first example from my capture includes two DNS requests from my computer to my router acting as a DNS server. One (#27) requesting the IP address for the Washington Department of Transportation (red line added) and the other (#29) is asking for Google map's IP address (blue line added). The linked packets (#28 and #30) are the replies – note the replies contain the requested IP addresses.



HTTP Web Requests: This is just part of a series of exchanges, packet numbers 1407, 1463, 1509 are requesting pictures (JPG files). I marked the replies with colored links so that you could see the request/reply nature of the exchanges.

Note the packet numbers; there were many other packets handled in the middle of these exchanges. The TCP filter and sorting on Protocol made the exchanges recognizable. *Part of output width of content omitted to fit the page.*



TCP Synchronization and Exchange: In the following exchange:

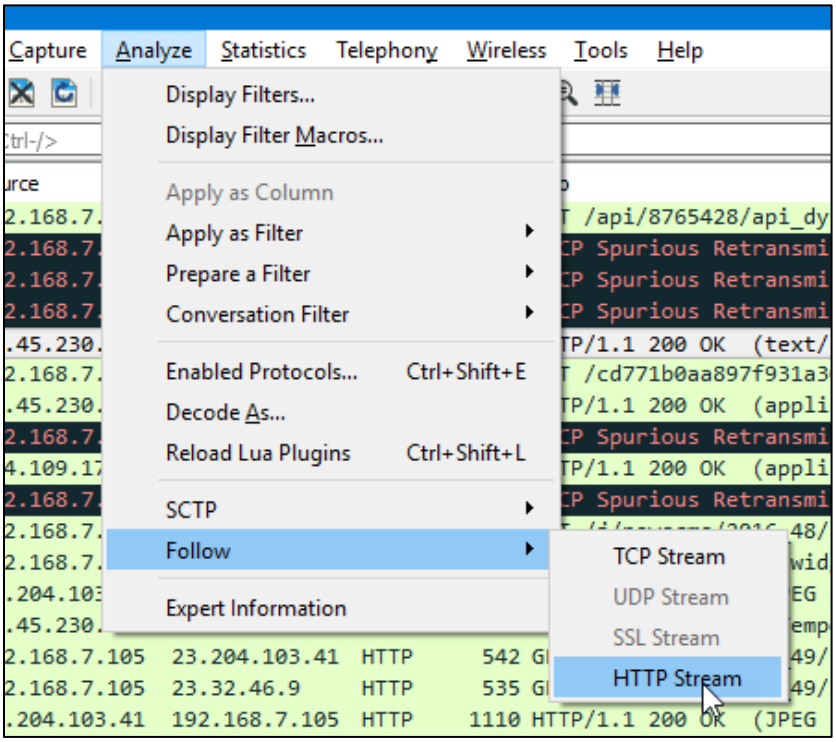
- The first two packets (300 and 311) are my computer asking to open a TCP session and computer 23.45.230.201 agreeing (SYN – synchronizing).
- Next, two packets (312 and 572) are my computer asking for certain conditions and computer 23.45.230.201 agreeing (ACK – acknowledging).
- Packets 573, 574, 575 are large (1,514 Bytes) that were each part of a larger data stream from the server that was reassembled on my computer.
- Packet 746 is my computer acknowledging (ACK) that the delivery was received without errors and in order.

tcp.stream eq 28

No.	Time	Source	Destination	Protocol	Length	Info
300	10.594017	192.168.7.105	23.45.230.201	TCP	66	50638→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
311	10.602549	23.45.230.201	192.168.7.105	TCP	66	80→50638 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=32
312	10.602595	192.168.7.105	23.45.230.201	TCP	54	50638→80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
572	10.960537	23.45.230.201	192.168.7.105	TCP	60	80→50638 [ACK] Seq=1 Ack=446 Win=30272 Len=0
573	10.961936	23.45.230.201	192.168.7.105	TCP	1514	[TCP segment of a reassembled PDU]
574	10.962076	23.45.230.201	192.168.7.105	TCP	1514	[TCP segment of a reassembled PDU]
575	10.962542	23.45.230.201	192.168.7.105	TCP	1514	[TCP segment of a reassembled PDU]
746	11.085326	192.168.7.105	23.45.230.201	TCP	54	50638→80 [ACK] Seq=446 Ack=2921 Win=65536 Len=0
751	11.085717	192.168.7.105	23.45.230.201	TCP	54	50638→80 [ACK] Seq=446 Ack=5313 Win=65536 Len=0
774	11.097459	23.45.230.201	192.168.7.105	TCP	66	[TCP Dup ACK 572#1] 80→50638 [ACK] Seq=5313 Ack=446 Win=30272 Len=0 SLE=1 SRE=446

Note: Most TCP exchanges would be much larger than this and almost impossible to recognize because of the many unrelated packets received in between these. How I got this is not critical to understanding the TCP sequence of steps and process. But, if you want to play with it or see other examples, the steps are.

1. In filtered or unfiltered packets, select one TCP, UDP, SSL, or HTML packet.
2. On the Analyze menu choose Follow
3. Then choose a packet type. Your original packet selection determines which choice(s) are available. The image shows an HTTP packet's choices.



The HTTP exchanges can be very long with many TCP exchanges within delivering the actual content. HTTP is just an instruction language for data requests and formatting instructions. In the earlier HTTP example, some of the gaps between the requests and replies would have been TCP exchanges delivering the pictures.

Task 8: Filtering Logical Operators

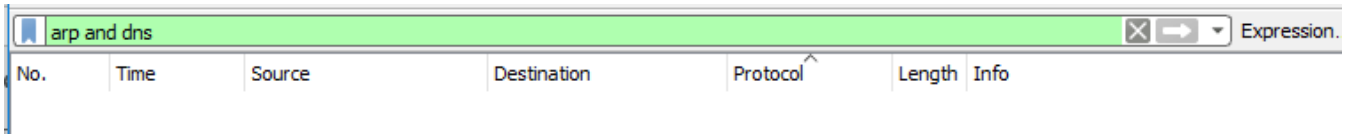
Occasionally, you may want to filter for two protocols or expressions. To do that you use the logical operators show on the table. You can use either the text or **English** forms or the more cryptic **C-like** operator.

Wireshark Logical Operators (use either)

English	C-like	Description and example
and	&&	Logical AND. `ip.src==10.0.0.5 and tcp.flags.fin`
or	||	Logical OR. `ip.scr==10.0.0.5 or ip.src==192.1.1.1`
xor	^^	Logical XOR. `tr.dst[0:3] == 0.6.29 xor tr.src[0:3] == 0.6.29`
not	!	Logical NOT. `not llc`
[…]		See "Substring Operator" below.
in		See "Membership Operator" below.

The AND, OR, and NOT operators and their C-like counterparts are the ones we will concern ourselves with at this point.

First, remember that OR and AND are easy to mix-up because they run almost the opposite of the way we often speak. Assume that we wanted to show both ARP and DNS packets. The following shows that entering **arp and dns** fails because no packet can be both. Instead, try **arp or dns** (or any two protocols in your capture set) and look over your results.



You could add a third protocol like **http or dns || ssdp** – just demonstrating multiple O operators and that you can mix the two forms of operators. Try it or one of your own – make sure all three types are in your capture.

Note: Once you filter, you can sort on the protocol column to make them easier to find.

Try **not tcp** and then **! udp** to see the equivalent to *everything but TCP* or UDP.

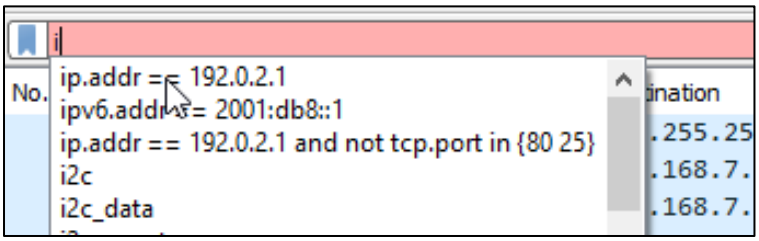
Task 9: Filtering Comparison Operators

So far, we have just looked at filtering on protocols, which is the default. But what we want packets with a specific IP address? To do that we are going to need to use Wireshark's somewhat cryptic keywords and the comparison operators shown below. As with the logical operators you can use either form.

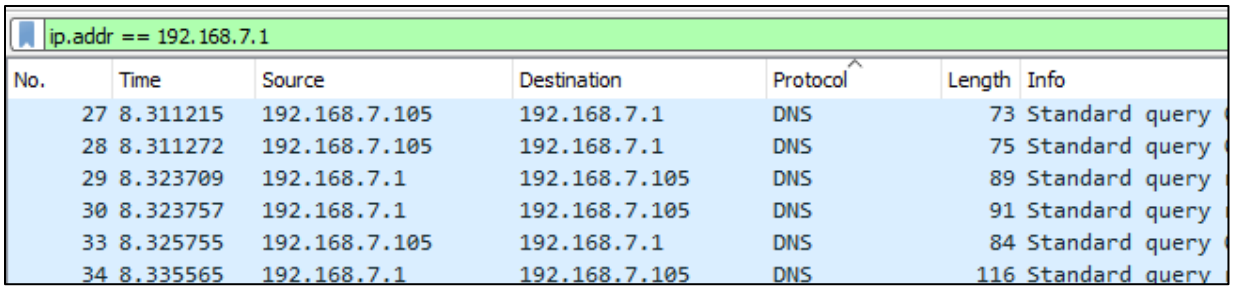
Wireshark Comparison Operators (use either)

English	C-like	Description and example
eq	==	Equal. `ip.src==10.0.0.5`
ne	!=	Not equal. `ip.src!=10.0.0.5`
gt	>	Greater than. `frame.len > 10`
lt	<	Less than. `frame.len < 128`
ge	>=	Greater than or equal to. `frame.len ge 0x100`
le	<=	Less than or equal to. `frame.len <= 0x20`

To see traffic involving the router (your default gateway) you need to know its IP address and use Wireshark's help with special keywords. When I typed the "i," the **ip.addr** and the syntax appeared in the list. Just select it and edit the IP address.



Try it. The result will look something like the following. Note that it included packets where the router was the source and the destination.



No.	Time	Source	Destination	Protocol	Length	Info
27	8.311215	192.168.7.105	192.168.7.1	DNS	73	Standard query
28	8.311272	192.168.7.105	192.168.7.1	DNS	75	Standard query
29	8.323709	192.168.7.1	192.168.7.105	DNS	89	Standard query
30	8.323757	192.168.7.1	192.168.7.105	DNS	91	Standard query
33	8.325755	192.168.7.105	192.168.7.1	DNS	84	Standard query
34	8.335565	192.168.7.1	192.168.7.105	DNS	116	Standard query

Try editing the expression to show everything that does not include your router. You should be able to edit as little as just one character to accomplish it.

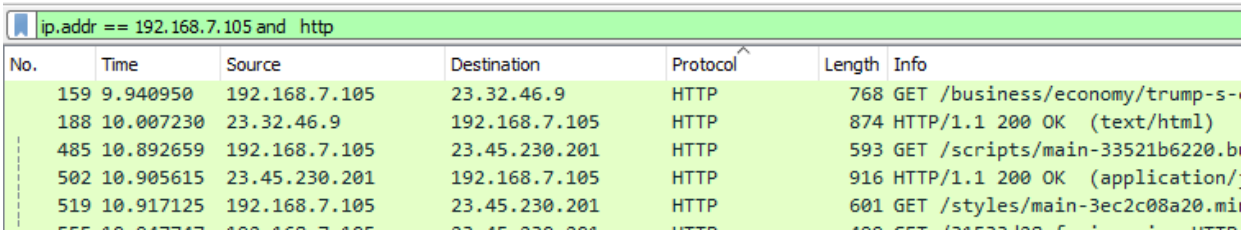
Any of the following should have worked (substituting an IP address of your choice):

- ip.addr != 192.168.7.1
- ip.addr ne 192.168.7.1
- not ip.addr == 192.168.7.1
- ! ip.addr == 192.168.7.1

To specify packets where an IP address is only the source or destination, we have to scroll down through the keywords to **ip.src** or **ip.dst** respectively in place of ip.addr.

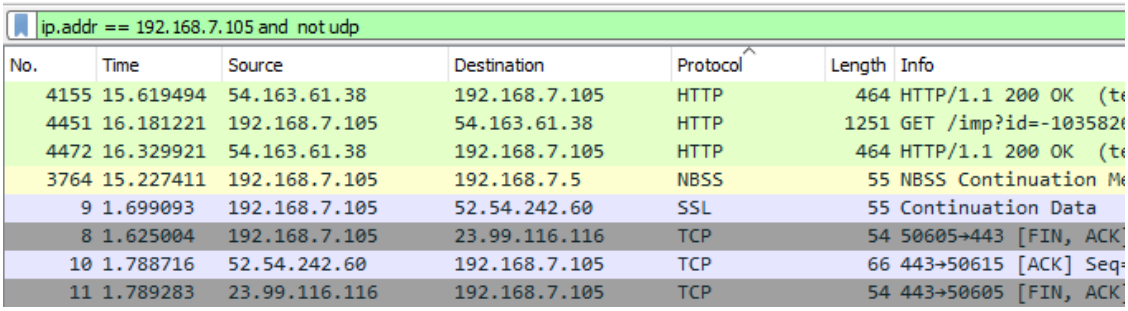
Try both with your computer's IP address.

The following is an example of using a complex expression with both types of operators. It is all HTTP traffic involving my IP address.



No.	Time	Source	Destination	Protocol	Length	Info
159	9.940950	192.168.7.105	23.32.46.9	HTTP	768	GET /business/economy/trump-s-
188	10.007230	23.32.46.9	192.168.7.105	HTTP	874	HTTP/1.1 200 OK (text/html)
485	10.892659	192.168.7.105	23.45.230.201	HTTP	593	GET /scripts/main-33521b6220.b
502	10.905615	23.45.230.201	192.168.7.105	HTTP	916	HTTP/1.1 200 OK (application/
519	10.917125	192.168.7.105	23.45.230.201	HTTP	601	GET /styles/main-3ec2c08a20.mi

The next one shows all packets involving my IP address but without any UDP packets.



No.	Time	Source	Destination	Protocol	Length	Info
4155	15.619494	54.163.61.38	192.168.7.105	HTTP	464	HTTP/1.1 200 OK (te
4451	16.181221	192.168.7.105	54.163.61.38	HTTP	1251	GET /imp?id=-103582
4472	16.329921	54.163.61.38	192.168.7.105	HTTP	464	HTTP/1.1 200 OK (te
3764	15.227411	192.168.7.105	192.168.7.5	NBSS	55	NBSS Continuation Me
9	1.699093	192.168.7.105	52.54.242.60	SSL	55	Continuation Data
8	1.625004	192.168.7.105	23.99.116.116	TCP	54	50605→443 [FIN, ACK]
10	1.788716	52.54.242.60	192.168.7.105	TCP	66	443→50615 [ACK] Seq=
11	1.789283	23.99.116.116	192.168.7.105	TCP	54	443→50605 [FIN, ACK]

Note the HTTP packets are because my browser is running with a website with many news articles that refresh every few minutes. Compare that to the next output.

ip.addr == 192.168.7.105 and not (udp or http)

No.	Time	Source	Destination	Protocol	Length	Info
3764	15.227411	192.168.7.105	192.168.7.5	NBSS	55	NBSS Continuation Me
9	1.699093	192.168.7.105	52.54.242.60	SSL	55	Continuation Data
8	1.625004	192.168.7.105	23.99.116.116	TCP	54	50605→443 [FIN, ACK]
10	1.788716	52.54.242.60	192.168.7.105	TCP	66	443→50615 [ACK] Seq=
11	1.789283	23.99.116.116	192.168.7.105	TCP	54	443→50605 [FIN, ACK]
12	1.789569	192.168.7.105	23.99.116.116	TCP	54	50605→443 [ACK] Seq=

The parenthesis makes the NOT apply to both UDP and HTTP. Now there are no HTTP packets either.

Experiment with the operators or anything that you learned from the videos.

We will return to Wireshark in other modules.

Capture Filters

We will not be using capture filters in this course, but here is a little information on them. For more information, consult the **Filtering while capturing a** section of the User guide.

Capture filters are not the same as display filters in several important ways.

- The packet capture syntax is like but not the same as the display filters. The syntax is the same as any program using the libpcap/WinPcap library to capture packets.
- Capture filters are more limited in options and used only to reduce the size of packet capture. Display filters are used to hide specific packets displayed from an existing packet list (the result of packet capture.)
- Capture filters are set before a packet capture is initiated and can't be modified during or after the capture. If certain packet types are excluded from the capture, only a new capture can include them. Display filters can be changed at any time, always returning to the full original capture set to start. Packets filtered out in one display, return when the filter is cleared or changed.