

## **Ex No: 14a    STUDY OF WIRESHARK TOOL FOR PACKET SNIFFING**

### **AIM:**

To study packet sniffing concepts using Wireshark Tool.

### **DESCRIPTION:**

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and display them in human-readable format. Wireshark includes filters, color coding, and other features that let you dig deep into network traffic and inspect individual packets. You can use Wireshark to inspect a suspicious program's network traffic, analyze the traffic flow on your network, or troubleshoot network problems.

### **What we can do with Wireshark:**

- Capture network traffic
- Decode packet protocols using dissectors
- Define filters – capture and display
- Watch smart statistics
- Analyze problems
- Interactively browse that traffic

### **Wireshark used for:**

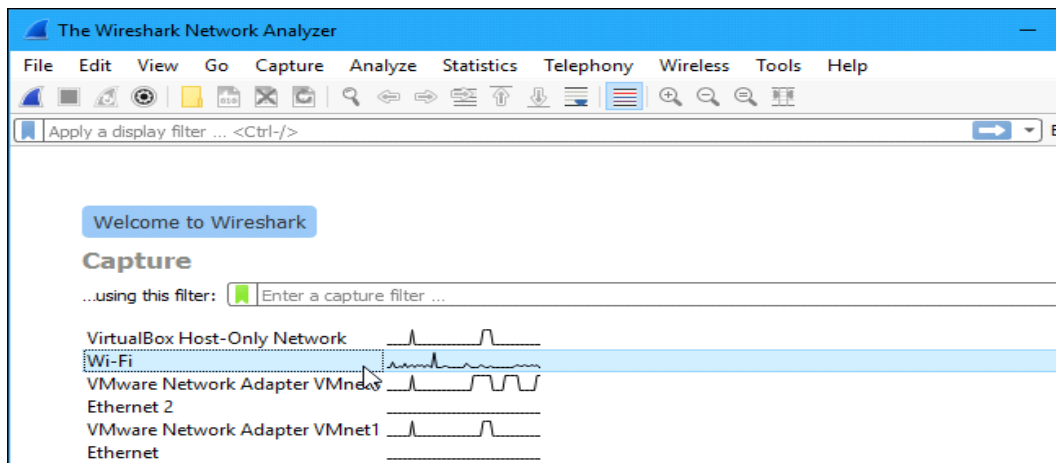
- Network administrators: troubleshoot network problems
- Network security engineers: examine security problems
- Developers: debug protocol implementations
- People: learn **network protocol internals**

### **Getting Wireshark**

Wireshark can be downloaded for Windows or macOS from [its official website](#). For Linux or another UNIX-like system, Wireshark will be found in its package repositories. For Ubuntu, Wireshark will be found in the Ubuntu Software Center.

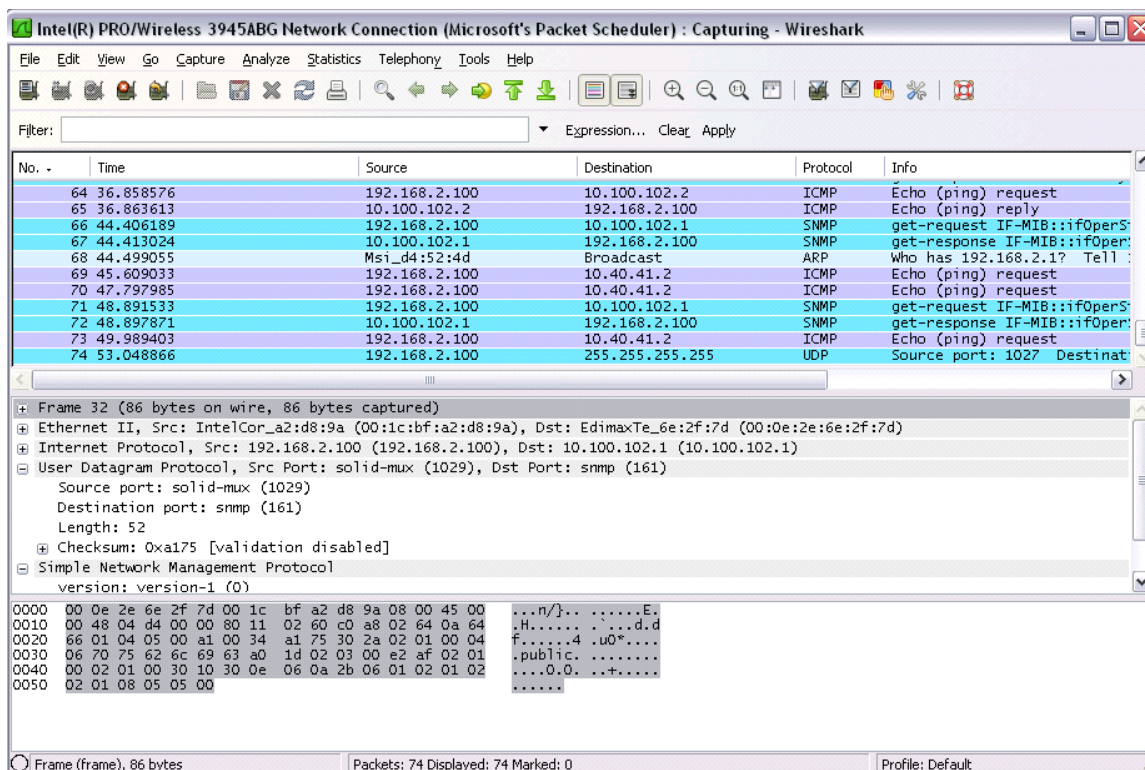
### **Capturing Packets**

After downloading and installing Wireshark, launch it and double-click the name of a network interface under Capture to start capturing packets on that interface



As soon as you click the interface's name, you'll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system.

If you have promiscuous mode enabled—it's enabled by default—you'll also see all the other packets on the network instead of only packets addressed to your network adapter. To check if promiscuous mode is enabled, click Capture > Options and verify the "Enable promiscuous mode on all interfaces" checkbox is activated at the bottom of this window.



Click the red "Stop" button near the top left corner of the window when you want to stop capturing traffic.

## The “Packet List” Pane

The packet list pane displays all the packets in the current capture file. The “Packet List” pane Each line in the packet list corresponds to one packet in the capture file. If you select a line in this pane, more details will be displayed in the “Packet Details” and “Packet Bytes” panes.

## The “Packet Details” Pane

The packet details pane shows the current packet (selected in the “Packet List” pane) in a more detailed form. This pane shows the protocols and protocol fields of the packet selected in the “Packet List” pane. The protocols and fields of the packet shown in a tree which can be expanded and collapsed.

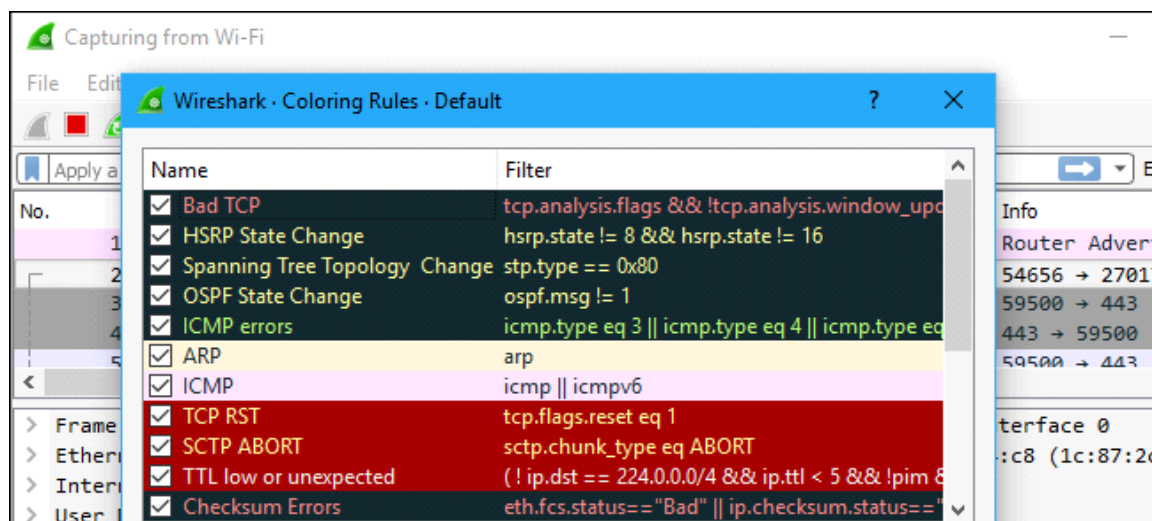
## The “Packet Bytes” Pane

The packet bytes pane shows the data of the current packet (selected in the “Packet List” pane) in a hexdump style.

## Color Coding

You’ll probably see packets highlighted in a variety of different colors. Wireshark uses colors to help you identify the types of traffic at a glance. By default, light purple is TCP traffic, light blue is UDP traffic, and black identifies packets with errors—for example, they could have been delivered out of order.

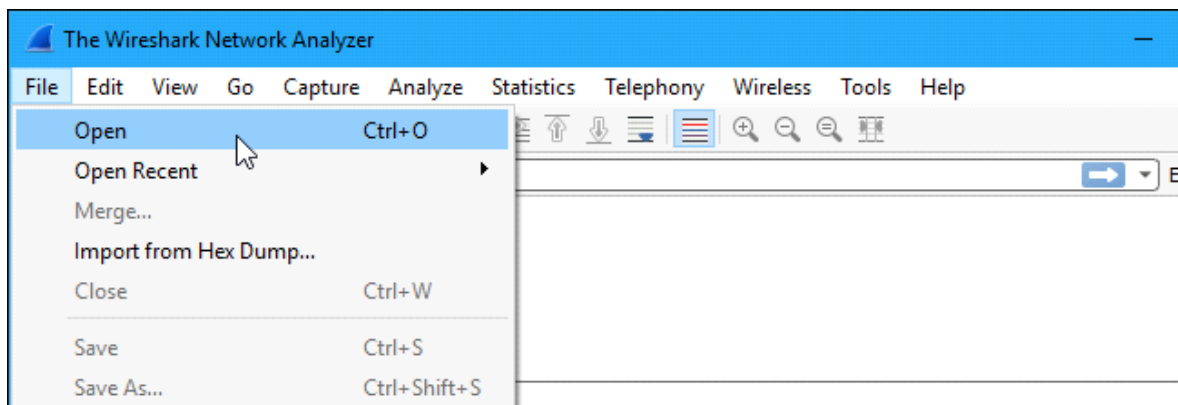
To view exactly what the color codes mean, click View > Coloring Rules. You can also customize and modify the coloring rules from here, if you like.



## Sample Captures

If there's nothing interesting on your own network to inspect, Wireshark's wiki has you covered. The wiki contains a [page of sample capture files](#) that you can load and inspect. Click File > Open in Wireshark and browse for your downloaded file to open one.

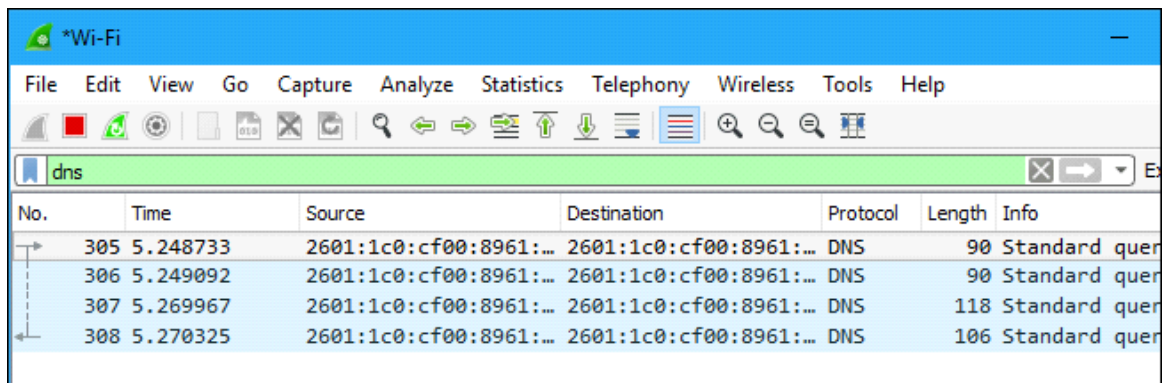
You can also save your own captures in Wireshark and open them later. Click File > Save to save your captured packets.



## Filtering Packets

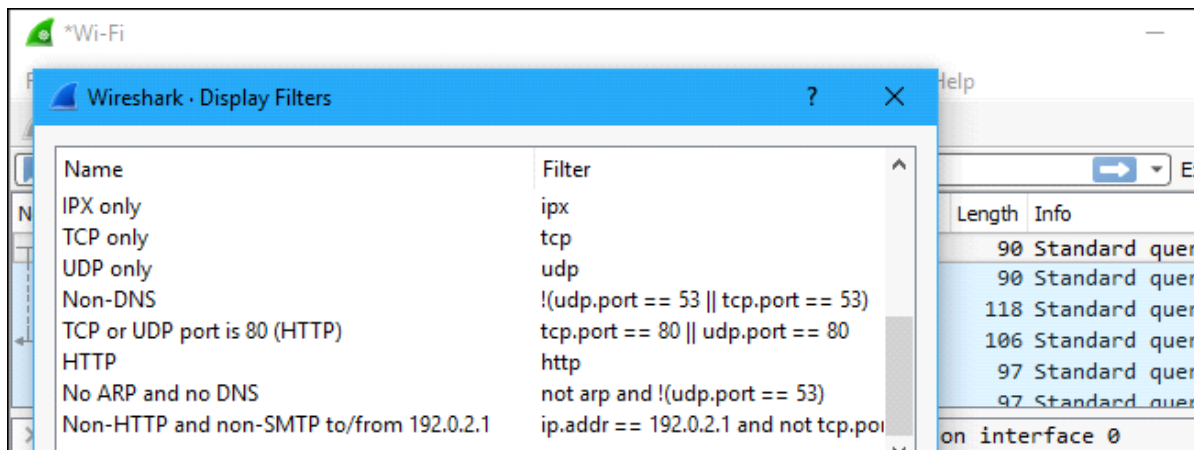
If you're trying to inspect something specific, such as the traffic a program sends when phoning home, it helps to close down all other applications using the network so you can narrow down the traffic. Still, you'll likely have a large amount of packets to sift through. That's where Wireshark's filters come in.

The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type "dns" and you'll see only DNS packets. When you start typing, Wireshark will help you autocomplete your filter.



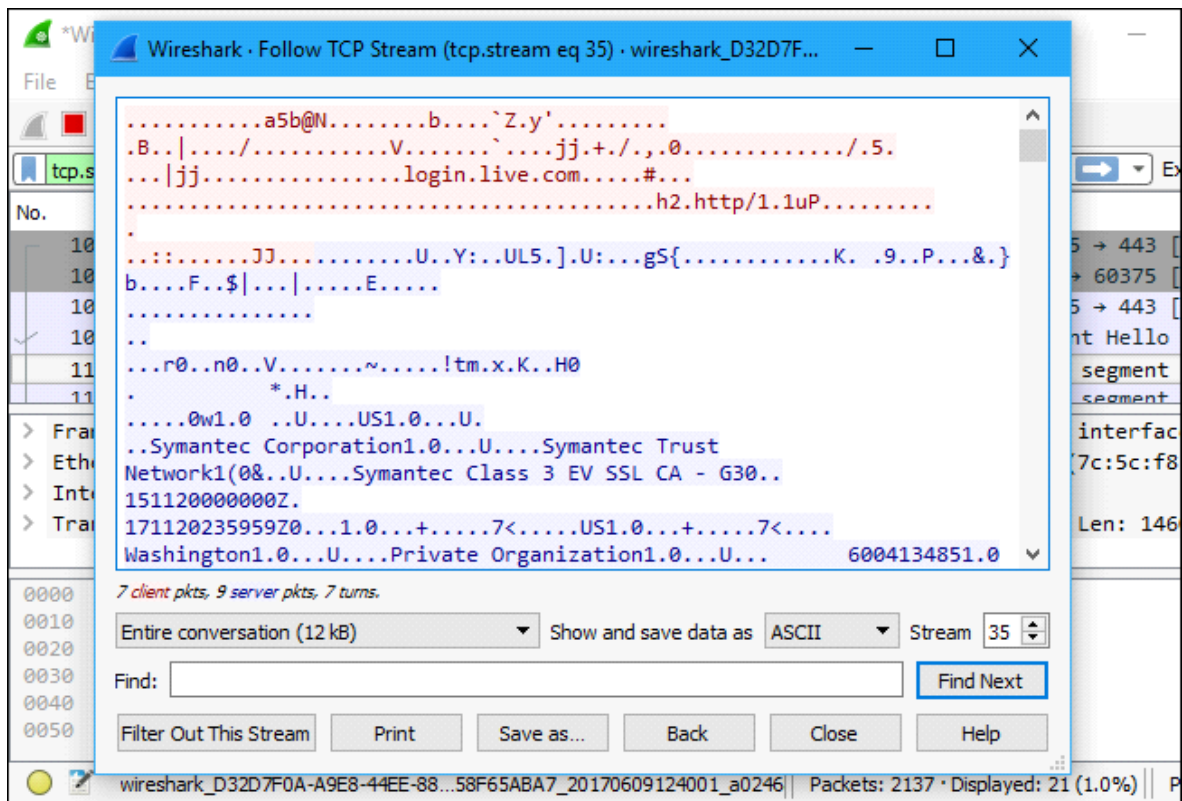
You can also click Analyze > Display Filters to choose a filter from among the default filters included in Wireshark. From here, you can add your own custom filters and save them to easily access them in the future.

For more information on Wireshark's display filtering language, read the [Building display filter expressions](#) page in the official Wireshark documentation.

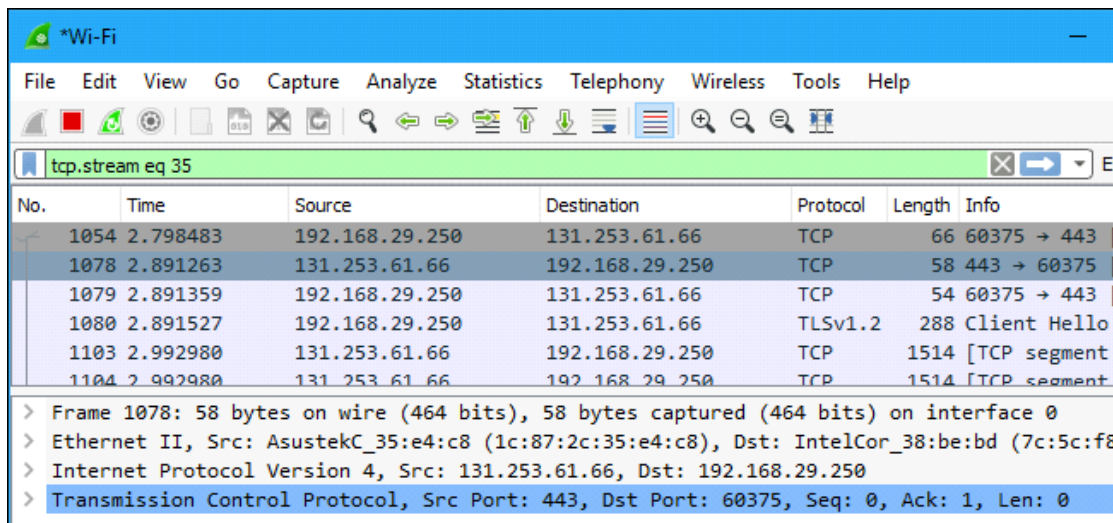


Another interesting thing you can do is right-click a packet and select Follow > TCP Stream.

You'll see the full TCP conversation between the client and the server. You can also click other protocols in the Follow menu to see the full conversations for other protocols, if applicable.



Close the window and you'll find a filter has been applied automatically. Wireshark is showing you the packets that make up the conversation.



## Inspecting Packets

Click a packet to select it and you can dig down to view its details.

\*Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.stream eq 35

No.	Time	Source	Destination	Protocol	Length	Info
1054	2.798483	192.168.29.250	131.253.61.66	TCP	66	60375 → 443
1078	2.891263	131.253.61.66	192.168.29.250	TCP	58	443 → 60375
1079	2.891359	192.168.29.250	131.253.61.66	TCP	54	60375 → 443
1080	2.891527	192.168.29.250	131.253.61.66	TLSv1.2	288	Client Hello

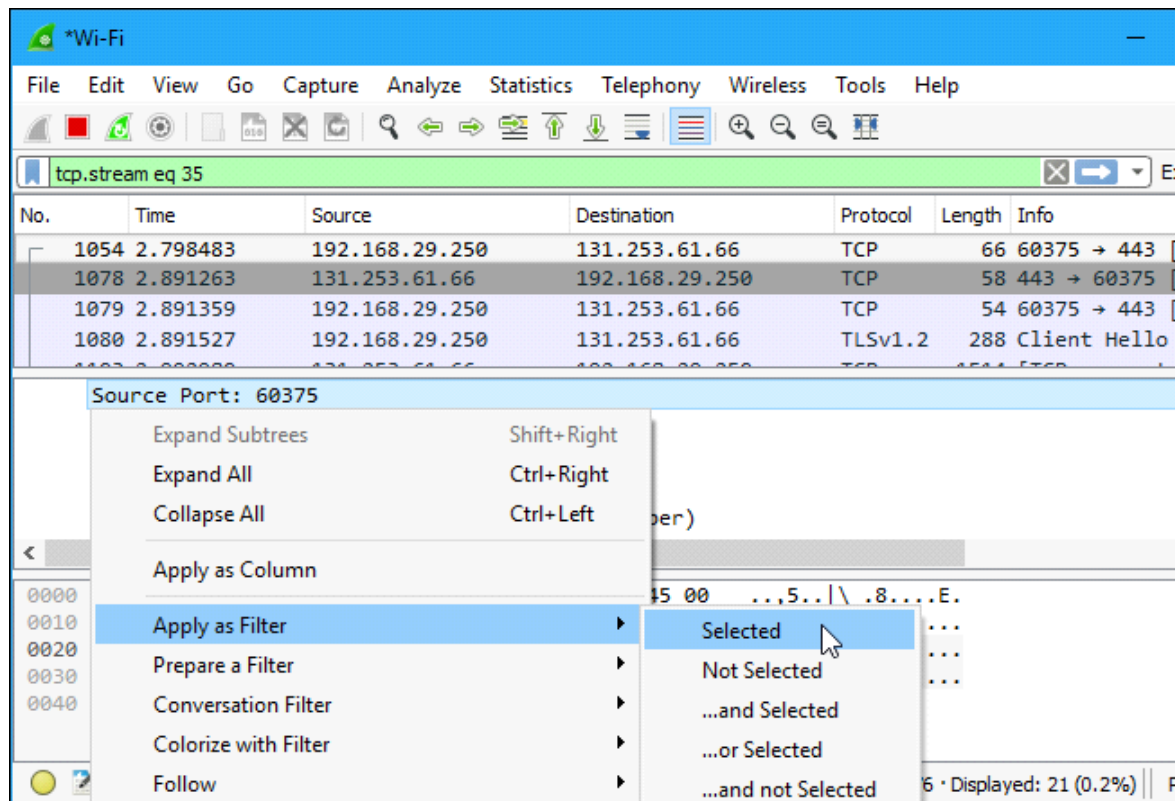
▼ Frame 1054: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0  
 Interface id: 0 (\Device\NPF\_{D32D7F0A-A9E8-44EE-88DC-DFD58F65ABA7})  
 Encapsulation type: Ethernet (1)  
 Arrival Time: Jun 9, 2017 12:40:04.140141000 Pacific Daylight Time  
 [Time shift for this packet: 0.000000000 seconds]  
 Epoch Time: 1497037204.140141000 seconds

0000	1c 87 2c 35 e4 c8 7c 5c f8 38 be bd 08 00 45 00	..,5.. \ .8....E.
0010	00 34 0b 5d 40 00 80 06 4f 85 c0 a8 1d fa 83 fd	.4.]@... 0.....
0020	3d 42 eb d7 01 bb 22 52 7b 69 00 00 00 00 80 02	=B...."R {i.....
0030	fa f0 48 ef 00 00 02 04 05 b4 01 03 03 08 01 01	..H.....
0040	04 02	..

Encapsulation type (frame.encap\_type) | Packets: 8136 · Displayed: 21 (0.3%)

You can also create filters from here — just right-click one of the details and use the Apply as Filter submenu to create a filter based on it.





Wireshark is an extremely powerful tool, and this tutorial is just scratching the surface of what you can do with it. Professionals use it to debug network protocol implementations, examine security problems and inspect network protocol internals.

**Flow Graph:** Gives a better understanding of what we see.



Example 001.pcap - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter:

No. - Time

1	0.000000
2	2.183304
3	3.430100
4	3.457181
5	3.461602
6	3.623867
7	3.728365
8	3.728429
9	3.728839
10	3.768896
11	3.770703
12	3.772411

Statistics

- Summary
- Protocol Hierarchy
- Conversations
- Endpoints
- Packet Lengths...
- IO Graphs
- Conversation List
- Endpoint List
- Service Response Time
- BOOTP-DHCP...
- Compare...
- Flow Graph...
- HTTP
- IP Addresses...
- IP Destinations...
- IP Protocol Types...
- ONC-RPC Programs
- TCP Stream Graph
- UDP Multicast Streams
- WLAN Traffic...

Expression... Clear Apply

Destination	Protocol	Info
10.40.41.2	ICMP	Echo (ping) request
10.40.41.2	ICMP	Echo (ping) request
212.150.49.10	DNS	Standard query A www.ynet
192.168.2.100	DNS	Standard query response C
212.150.49.10	DNS	Standard query A www.lenovo
212.143.162.157	TCP	dzdaemon > http [SYN] Seq
192.168.2.100	TCP	http > dzdaemon [SYN, ACK
212.143.162.157	TCP	dzdaemon > http [ACK] Seq
212.143.162.157	HTTP	GET / HTTP/1.1
192.168.2.100	TCP	http > dzdaemon [ACK] Seq
192.168.2.100	HTTP	HTTP/1.0 301 Moved Perman
212.143.162.157	HTTP	GET /home/0.7340.L=8.00.h

Frame 5 (74 bytes on wire, 74 bytes captured on interface 0) on interface 0

Ethernet II, Src: IntelCor\_a2:8b:00:00:00:00, Dst: 192.168.2.100

Internet Protocol Version 4, Src: 192.168.2.100, Dst: 212.150.49.10

User Datagram Protocol, Src Port: 53, Dst Port: 53

Domain Name System (query)

0000 00 0e 2e 6e 2f 7d 00 1c bf a2 d8 9a 08 00 45 00 ...n/}.. ..E.

0010 00 3c 7f ea 00 00 80 11 f2 19 c0 a8 02 64 d4 96 ...<... ..d..

0020 31 0a 0b 4f 00 00 35 00 28 f5 df 9e d7 01 00 00 01 ...0.5.( .....w

0030 00 00 00 00 00 00 03 77 77 77 06 6c 65 6e 6f 76 .....w ww.lenov

0040 6f 03 63 6f 6d 00 00 01 00 01 .....o.com... ..

File: "D:\Courses\Freeware\Example 001.pcap" ... Packets: 1303 Displayed: 1303 Marked: 0



Example 001.pcap - Graph Analysis

Time 192.168.2.100 10.40.41.2 212.150.49.10 212.143.162.157 212.143.162.141 Comment

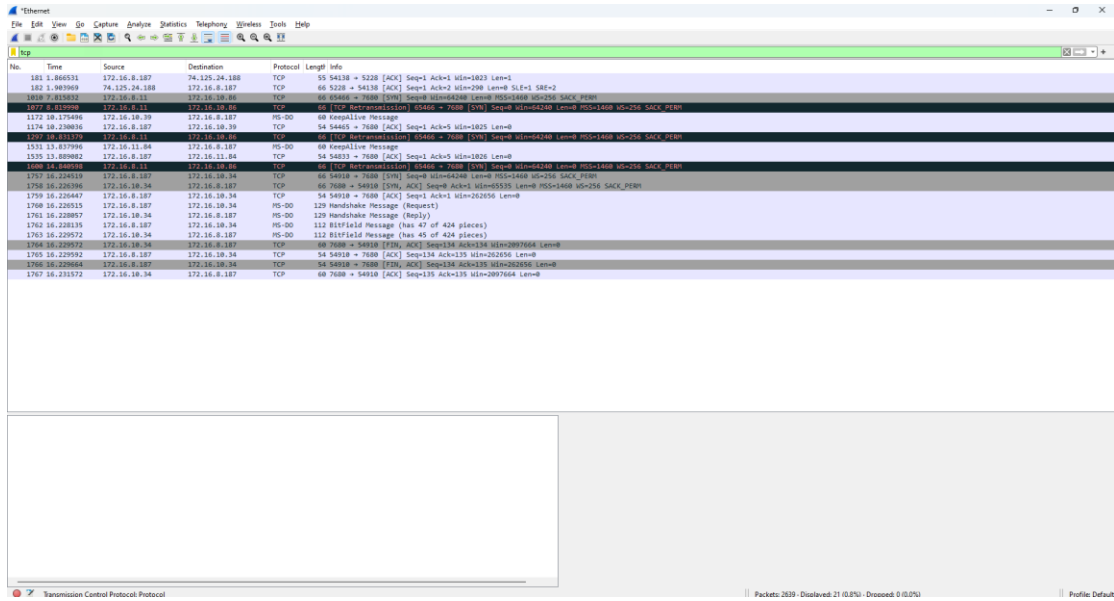
0.000	(0)	Echo (ping) request	(0)	ICMP: Echo (ping) request
2.183	(0)	Echo (ping) request	(0)	ICMP: Echo (ping) request
3.430	(25003)	Standard query A ww	(53)	DNS: Standard query A www.ynet.co.il
3.457	(25003)	Standard query resp	(53)	DNS: Standard query response CNAME ynet.co.il.d4p.net CNAME a39.
3.462	(28995)	Standard query A ww	(53)	DNS: Standard query A www.lenovo.com
3.624	(3866)	dzdaemon > http [SY	(80)	TCP: dzdaemon > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 W
3.728	(3866)	http > dzdaemon [SY	(80)	TCP: http > dzdaemon [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 M
3.728	(3866)	dzdaemon > http [AC	(80)	TCP: dzdaemon > http [ACK] Seq=1 Ack=1 Win=128480 Len=0
3.729	(3866)	GET / HTTP/1.1	(80)	HTTP: GET / HTTP/1.1
3.769	(3866)	http > dzdaemon [AC	(80)	TCP: http > dzdaemon [ACK] Seq=1 Ack=580 Win=6948 Len=0
3.771	(3866)	HTTP/1.0 301 Moved	(80)	HTTP: HTTP/1.0 301 Moved Permanently
3.772	(3866)	GET /home/0.7340.L-	(80)	HTTP: GET /home/0.7340.L=8.00.html HTTP/1.1
3.965	(3866)	[TCP Previous segme	(80)	TCP: [TCP Previous segment lost] [TCP segment of a reassembled PI
3.965	(3866)	[TCP Dup ACK 12#1]	(80)	TCP: [TCP Dup ACK 12#1] dzdaemon > http [ACK] Seq=1204 Ack=1
3.966	(3866)	[TCP segment of a r	(80)	TCP: [TCP segment of a reassembled PDU]
3.966	(3866)	[TCP Dup ACK 12#2]	(80)	TCP: [TCP Dup ACK 12#2] dzdaemon > http [ACK] Seq=1204 Ack=1
3.968	(3866)	[TCP segment of a r	(80)	TCP: [TCP segment of a reassembled PDU]
3.968	(3866)	[TCP Dup ACK 12#3]	(80)	TCP: [TCP Dup ACK 12#3] dzdaemon > http [ACK] Seq=1204 Ack=1
3.968	(28995)	Standard query resp	(53)	DNS: Standard query response CNAME www.lenovo.com.edgekey.net
3.990	(3866)	[TCP segment of a r	(80)	TCP: [TCP segment of a reassembled PDU]

Save As Close



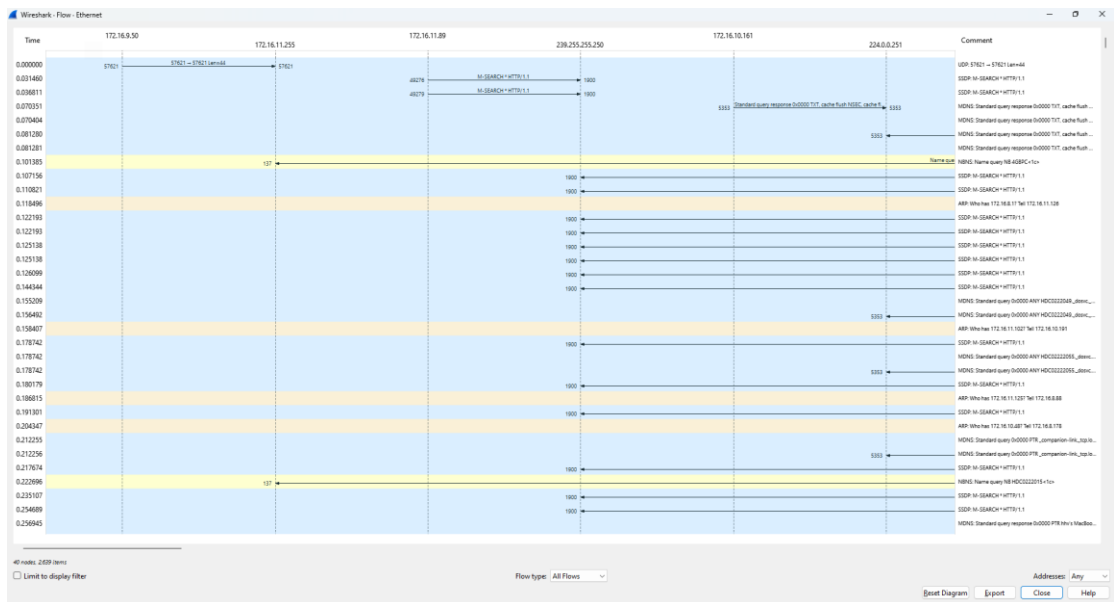
- Select Local Area Connection in Wireshark.
- Go to capture  option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search TCP packets in search bar.
- To see flow graph click Statistics  Flow graph.
- Save the packets.

## Output:




No.	Time	Source	Destination	Protocol	Length	Info
181	1.864531	172.16.8.187	74.125.24.188	TCP	55	54138 → 5228 [ACK] Seq=1 Ack=1 Win=1823 Len=1
182	1.907569	74.125.24.188	172.16.8.187	TCP	66	5228 → 54138 [ACK] Seq=1 Ack=2 Win=288 Len=0 SLEN=2
183	7.815812	172.16.8.11	172.16.18.86	TCP	66	8086 → 7680 [FIN] Seq=9 Win=4248 Len=0 RST=1408 Win=256 SACK_PERM
184	8.215527	172.16.8.11	172.16.18.86	TCP	66	8086 → 7680 [FIN] Seq=9 Win=4248 Len=0 RST=1408 Win=256 SACK_PERM
185	10.175496	172.16.18.39	172.16.8.187	HS-00	60	Keepalive Message
186	10.230836	172.16.8.187	172.16.18.39	TCP	54	54485 → 7680 [ACK] Seq=1 Ack=5 Win=1825 Len=0
187	10.230836	172.16.8.187	172.16.18.39	TCP	54	54485 → 7680 [ACK] Seq=1 Ack=5 Win=1825 Len=0
188	13.837996	172.16.11.84	172.16.8.187	HS-00	60	Keepalive Message
189	13.888062	172.16.8.187	172.16.11.84	TCP	54	54833 → 7680 [ACK] Seq=1 Ack=5 Win=1825 Len=0
190	14.141011	172.16.8.11	172.16.18.86	TCP	66	8086 → 7680 [FIN] Seq=9 Win=4248 Len=0 RST=1408 Win=256 SACK_PERM
191	17.224519	172.16.8.187	172.16.18.34	TCP	66	54918 → 7680 [FIN] Seq=9 Win=4248 Len=0 RST=1408 Win=256 SACK_PERM
192	18.228796	172.16.18.34	172.16.8.187	TCP	66	7680 → 54918 [FIN] Seq=9 Win=4248 Len=0 RST=1408 Win=256 SACK_PERM
193	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
194	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
195	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
196	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
197	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
198	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
199	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
200	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
201	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
202	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
203	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
204	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
205	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
206	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
207	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
208	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
209	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
210	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
211	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
212	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
213	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
214	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
215	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
216	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
217	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
218	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
219	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
220	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
221	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
222	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
223	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
224	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
225	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
226	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
227	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
228	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
229	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
230	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
231	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
232	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
233	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
234	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
235	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
236	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
237	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
238	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
239	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
240	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
241	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
242	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
243	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
244	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
245	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
246	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
247	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
248	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
249	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0
250	18.228796	172.16.8.187	172.16.18.34	TCP	54	54918 → 7680 [ACK] Seq=1 Ack=1 Win=26256 Len=0

## Flow Graph output

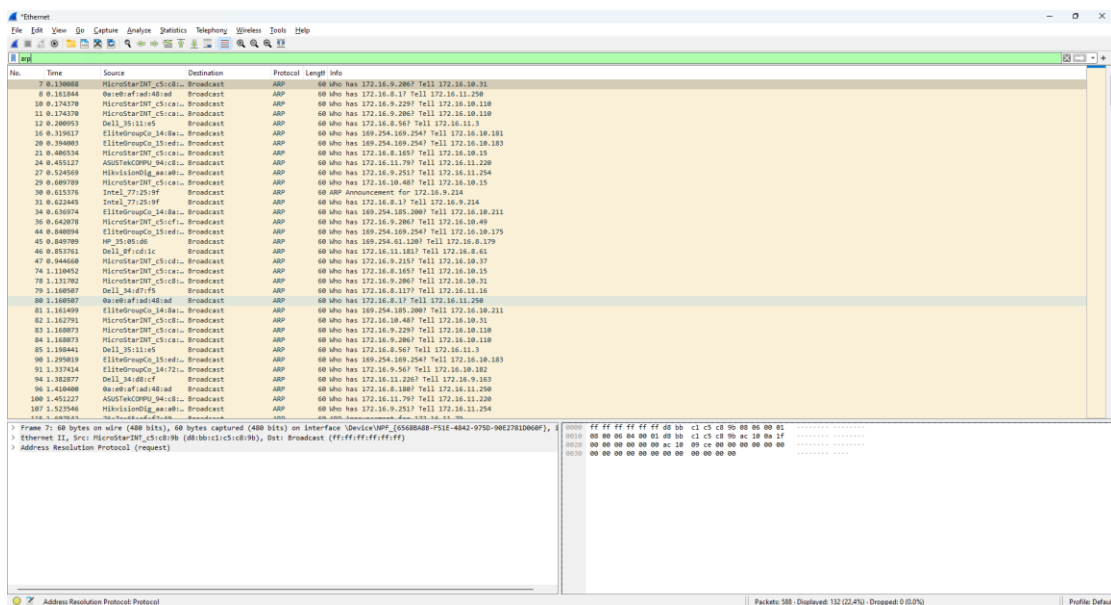


### 3. Create a Filter to display only ARP packets and inspect the packets.

#### Procedure


- Select Local Area Connection in Wireshark.
- Go to capture  Option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search ARP packets in search bar.
- Save the packets.

#### Output



### 4. Create a Filter to display only DNS packets and provide the flow graph.

#### Procedure

- Select Local Area Connection in Wireshark.
- Go to capture  Option
- Select stop capture automatically after 100 packets.

- Then click Start capture.
- Search DNS packets in search bar.
- To see flow graph click Statistics & Flow graph.
- Save the packets.

## Output

The screenshot displays the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Help, and Tools. The packet list pane shows four captured packets:

No.	Time	Source	Destination	Protocol	Length	Info
464	4.478572	172.16.8.187	172.16.8.1	DNS	78	Standard query 80d81 A undata.weather.com
469	4.538213	172.16.8.187	172.16.8.1	DNS	78	Standard query 80d81 A undata.weather.com
471	4.559889	172.16.8.1	172.16.8.187	DNS	165	Standard query response 80d81 A undata.weather.com CNAME weather.com.edgekey.net CNAME e12930.g.akamaiedge.net A 23.192.98.28
472	4.559889	172.16.8.1	172.16.8.187	DNS	165	Standard query response 80d81 A undata.weather.com CNAME weather.com.edgekey.net CNAME e12930.g.akamaiedge.net A 23.192.98.28

The packet details pane for the selected packet (No. 464) shows the following structure:

- Frame 464: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface \Device\NPF\_{05680A0B-F51E-4842-975D-98E27E1D800F}.
- Ethernet II, Src: HP\_383Ffa9 (7c:5b:38:3f:fa:9), Dst: Sophos\_cf8e45 (7c:5a:1c:cf:be:45)
- Internet Protocol Version 4, Src: 172.16.8.187, Dst: 172.16.8.1
- User Datagram Protocol, Src Port: 54683, Dst Port: 53
- Domain Name System (query)

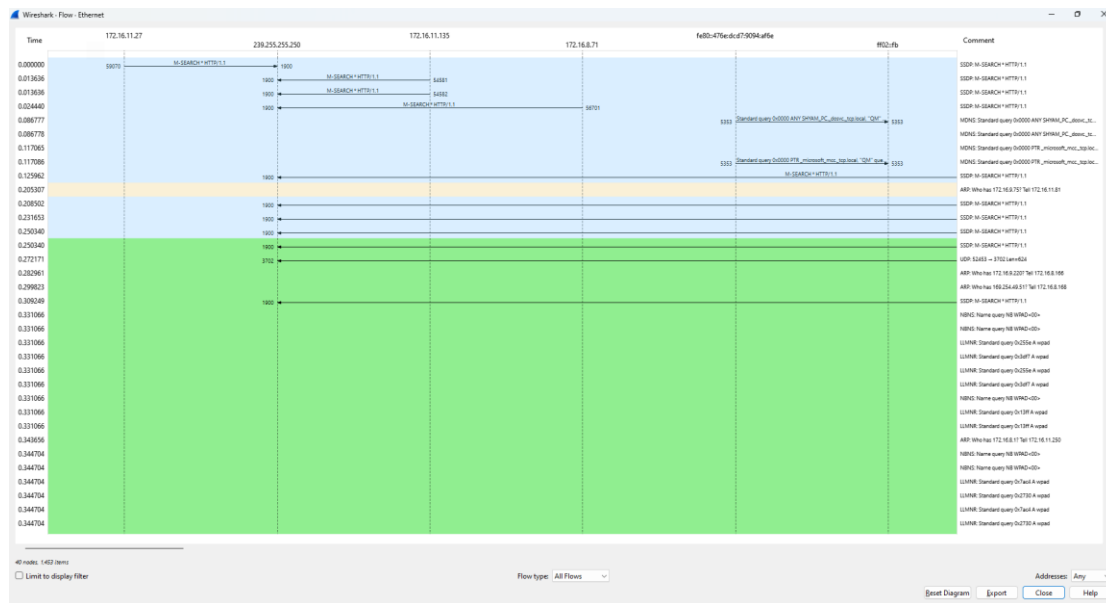
The packet bytes pane shows the raw data in hexadecimal and ASCII:

```

0000  7c 5a 1c cf be 45 7c 5b 38 3f fa 90 00 00 00 00 00  |2...[uXB?...E
0010  00 48 a7 03 00 00 00 11 00 00 ac 18 00 00 ac 18  |@.....
0020  00 03 05 20 00 35 00 2c 00 1a 8c 02 01 00 00 01  |...($,.....
0030  00 00 00 00 00 00 06 77 78 64 61 74 61 67 77 65  |.....vdata ve
0040  61 74 68 65 72 63 63 6f 6d 00 00 01 00 01  |other.co a....
  
```


The bottom status bar indicates: Packets: 1453 - Displayed: 4 (0.3%) - Dropped: 0 (0.0%) Profile: Default

## Flow Graph output

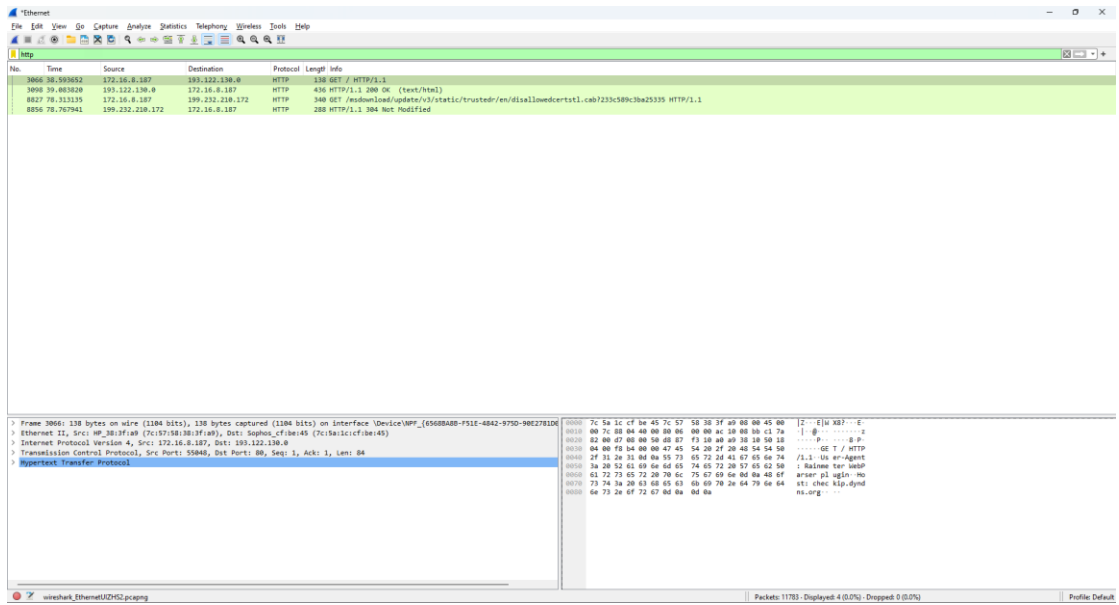


## 5.Create a Filter to display only HTTP packets and inspect the packets

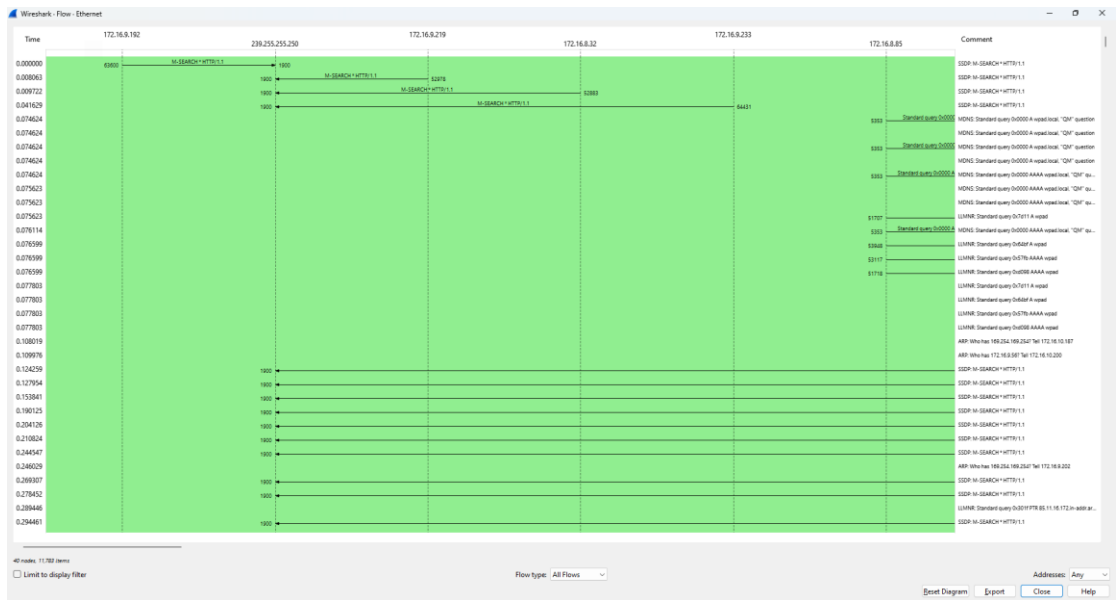
### Procedure

- Select Local Area Connection in Wireshark.
- Go to capture  option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search HTTP packets in the search bar.
- Save the packets.

### Output




## Flow Graph output



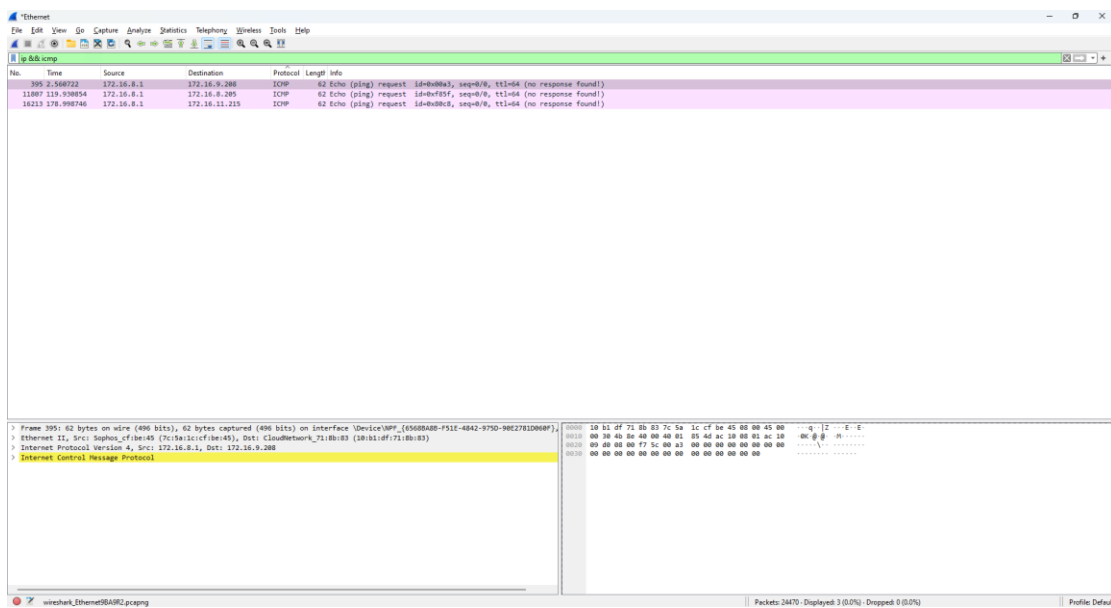


## 6.Create a Filter to display only IP/ICMP packets and inspect the packets.

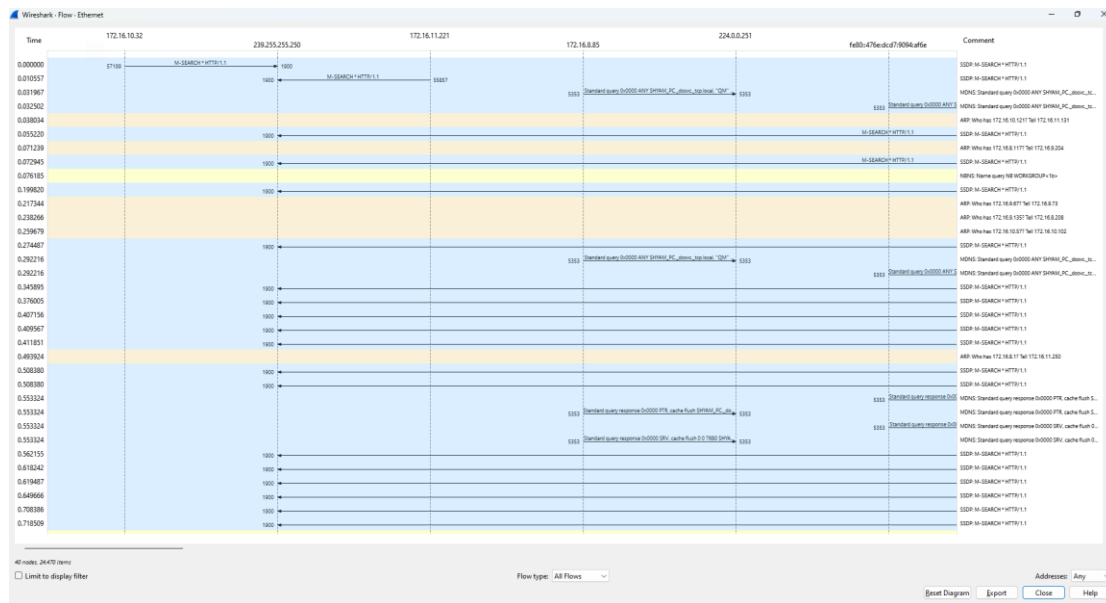
### Procedure

- Select Local Area Connection in Wireshark.
- Go to capture  option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search ICMP/IP packets in search bar.
- Save the packets

### Output




### Flow Graph output

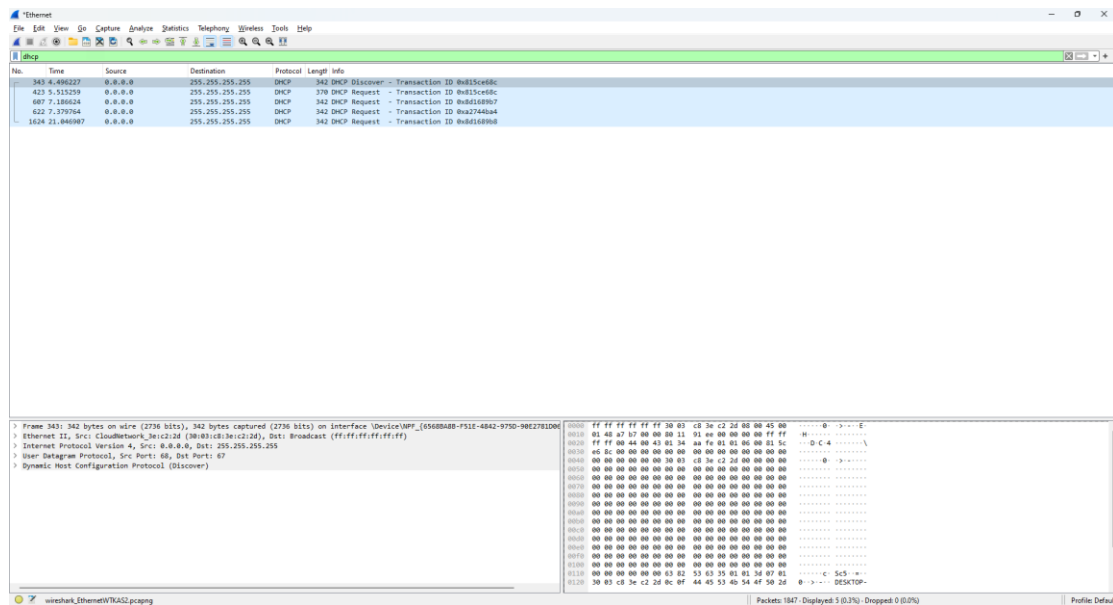


## 7.Create a Filter to display only DHCP packets and inspect the packets.

### Procedure

- Select Local Area Connection in Wireshark.
- Go to capture  option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search DHCP packets in search bar.
- Save the packets

### Output



## Flow graph output:

