

**Don Bosco Institute of Technology**  
**Department of Electronics and Telecommunications**  
**Engineering**

**Class: Sem VI**

**Subject Code: ECL 602**

**Subject Name: Computer Communication Network Laboratory**

**Experiment No. 3**

**Aim:** To perform live packet capture and network traffic analysis using network protocol analyzer using Wireshark.

**Learning Objectives:**

- To make the students aware of packet capturing tool.
- To enable the students to analyze the captured packets.

**Learning Outcomes:**

After successful completion of the experiment students will be able to:

- Use wireshark as a network protocol analyzer .
- Analyze the captured packets using statistical tools.

**Theory:**

A better way to understand network protocols is to observe how they actually work. A basic tool for observing the messages exchanged between executing protocol entities is the **packet sniffer**, which is an essential part of **network protocol analyzer**. WireShark is a free and open-source network protocol analyzer that runs on various operating systems including Linux, Unix, Mac, and Windows.

As the name suggests, a packet sniffer captures (“sniffs”) messages being sent/received from/by a computer; it will also typically store and/or display the contents of the various protocol fields in these captured messages. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer receives a copy of packets that are sent/received from/by application and protocols executing on your machine.

Figure 1 shows the structure of a packet sniffer. At the right of Figure 1 are the protocols (in this case, Internet protocols) and applications (such as a web browser or ftp client) that normally run on a computer. The packet sniffer, shown within the dashed rectangle in Figure 1 is an addition to the usual software in your computer, and consists of two parts. The packet capture library receives a copy of every link-layer frame that is sent from or received by your

computer. The message exchange by higher layer protocols such as HTTP, FTP,TCP, UDDP, DNS, or IP all are eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable. In Figure 1, the assumed physical media is an Ethernet, and so all upper layer protocols are eventually encapsulated within an Ethernet frame. Capturing all link-layer frames thus gives you all messages sent/received from/by all protocols and applications executing in the computer.

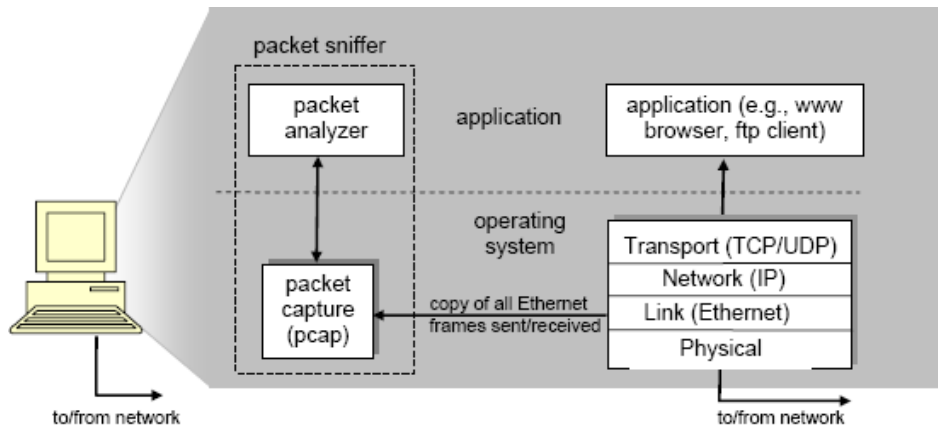


Figure 1: Packet sniffer structure

The second component of a packet sniffer is the packet analyzer, which displays the contents of all fields within a protocol message. In order to do so, the packet analyzer must “understand” the structure of all messages exchanged by protocols. For example, suppose we are interested in displaying the various fields in messages exchanged by the HTTP protocol in Figure 1. The packet analyzer understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. Finally, it understands the TCP segment structure, so it can extract the HTTP message contained in the TCP segment. Finally, it understands the HTTP protocol and so, for example, knows that the first bytes of an HTTP message will contain the string “GET,” “POST,” or “HEAD.”

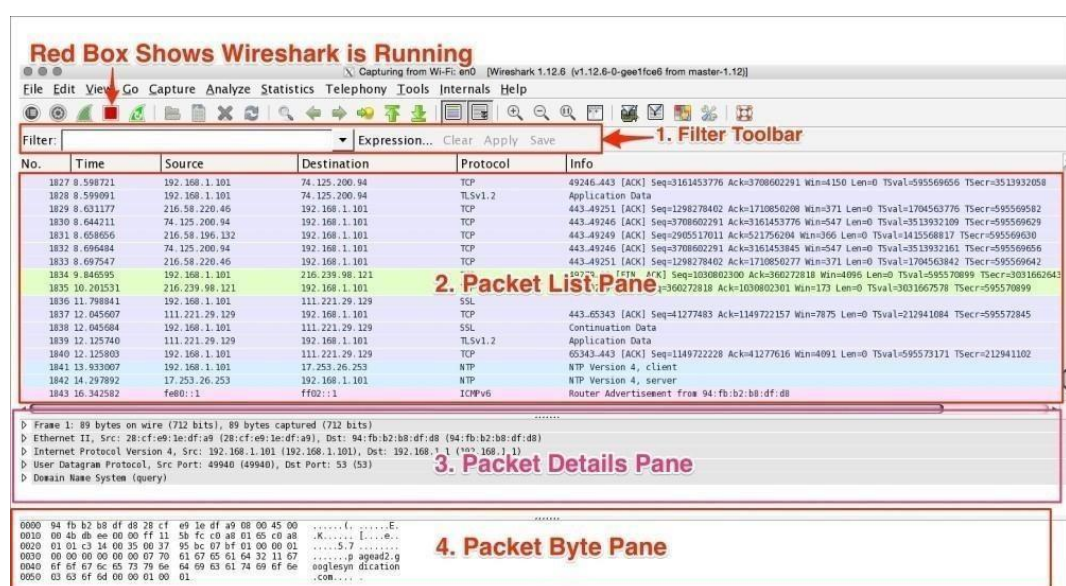


Figure2: Wireshark Windowpanes

## Packet List

The packet list pane, located at the top of the window, shows all packets found in the active capture file. Each packet has its own row and corresponding number assigned to it, along with each of these data points.

- **Time:** The timestamp of when the packet was captured is displayed in this column. The default format is the number of seconds.
- **Source:** This column contains the address (IP or other) where the packet originated.
- **Destination:** This column contains the address that the packet is being sent to.
- **Protocol:** The packet's protocol name, such as TCP, can be found in this column.
- **Length:** The packet length, in bytes, is displayed in this column.
- **Info:** Additional details about the packet are presented here. The contents of this column can vary greatly depending on packet contents.

## Packet Details

The details pane, found in the middle, presents the protocols and protocol fields of the selected packet in a collapsible format. In addition to expanding each selection, individual Wireshark filters based on specific details and follow streams of data based on protocol type via the details context menu can be applied, by right-clicking the mouse on the desired item in this pane.

## Packet Contents (Bytes)

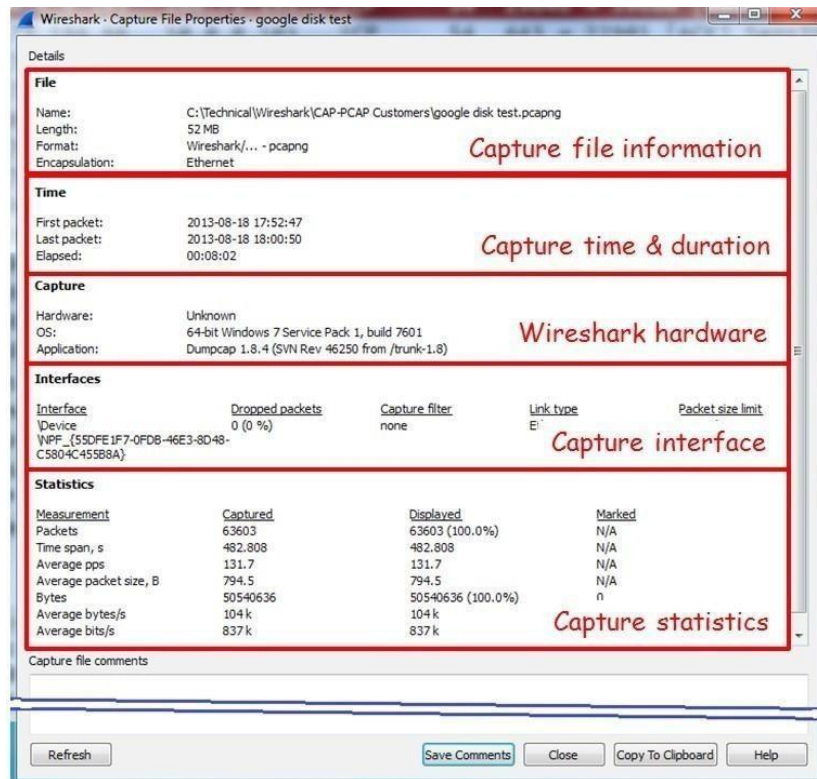
At the bottom is the packet bytes pane, which displays the raw data of the selected packet in a hexadecimal view. This contains 16 hexadecimal bytes and 16 ASCII bytes alongside the data offset. Selecting a specific portion of this data automatically highlights its corresponding section in the packet details pane and vice versa.

## Filter Toolbar

When capturing over a long time period, numerous packets are captured and it is required to investigate a selected portion of the packets (for example for particular protocol). Hence filters can be applied during the packet capture (such that only packets that meet the specified criteria are captured - called capture filters) or after the capture (such that analysis is only performed on packets that meet the specified criteria - called display filters). Display filters are used mainly to view certain types of packets. They make analyzing the data easier. One place you can enter a display filter is just above the top (packet list) section. You can either type in the filter and press apply or create the filter using the Expression command.

## Capture File Properties

This menu simply gives a summary of the filtered data properties and the capture statistics (average packets or bytes per second).



**Figure3:Capture File Properties**

The Capture File Properties window (displayed in the screenshot) has the following:

- File: Provides file data, such as filename and path, length, and so on.
- Time: Start time, end time, and duration of capture.
- Capture: Hardware information for the PC that Wireshark is installed on.
- Interfaces: Interface information—the interface registry identifier on the left, if capture filter is turned on, interface type and packet size limit.
- Statistics: General capture statistics, including captured and displayed packets.

#### • Follow TCP Stream

One of Wireshark's most satisfying analysis features is its ability to reassemble TCP streams into an easily readable format. Rather than viewing data being sent from client to server in a bunch of small chunks, the Follow TCP Stream feature sorts the data to make it easier to view. This comes in handy when viewing plaintext application layer protocols such as HTTP, FTP, and so on.

The text displayed in this window is in two colors. The red text is used to signify traffic from the source to the destination, and the blue text is used to identify traffic in the opposite direction, from the destination to the source. The color relates to which side initiated the communication. Given this TCP stream, you can clearly see a great majority of the

communication between these two hosts. This communication begins with an initial GET request for the web root director ( / ) and a response from the server that the request was successful in the form of an HTTP/1.1 200 OK . A similar pattern is repeated throughout the stream as individual files are requested by the client and the server responds with them.

In addition to viewing the raw data in this window, you can also search within the text, save it as a file, print it, or choose to view the data in ASCII, EBCDIC, hex, or C array format. These options can be found at the bottom of the Follow TCP Stream window.

## · **Graphing**

Graphs are the bread and butter of analysis, and one of the best ways to get an overview of a data set. Wireshark includes a few different graphing features to assist in understanding capture data, the first of which is its IO graphing capabilities.

### □ **Viewing IO Graphs**

Wireshark's IO Graphs window allows to graph the throughput of data on a network. These graphs can be used to find spikes and lulls in data throughput, discover performance lags in individual protocols, and to compare simultaneous data streams. The IO Graphs window shows a graphical view of the flow of data over the course of the capture file. The overall traffic seen in a capture file which is usually measured in rate per second in bytes or packets (which you can always change if you prefer bits/bytes per second). In default the x-axis is the tick interval per second, and y-axis is the packets per tick (per second).

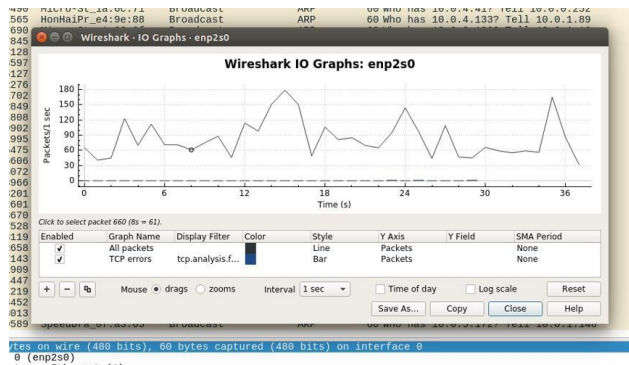
### □ **Flow Graphing**

The flow graphing feature is very useful for visualizing connections and showing the flow of data over time. Basically, a flow graph contains a column based view of a connection between hosts and organizes the traffic which can be interpreted visually. It provides a quick and easy to use way of checking connections between a client and a server. It can show where there might be issues with a TCP connection, such as timeouts, re-transmitted frames, or dropped connections.

### **Steps:**

1. Start wireshark and select the interface to begin packet capture.
2. A packet capture summary window appears which shows the packets being captured.
3. Stop the packet capturing process after few seconds.
4. Inspect the packet capture window and explore the various protocols.
5. Explore the different features of wireshark.
6. Take screenshots for all the above features and interpret the outputs.

### **Screenshot of O/P:**



enp2s0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-F>

Packet list

No.	Time	Source	Destination	Protocol	Length	Info
2769	32.737581646	HonHaiPr_e4:98:0d	Broadcast	ARP	60	Who has 10.0.1.248? Tell 10.0.0.132
2770	32.742998486	RealtekS_68:0c:03	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.19
2771	32.744998486	RealtekS_68:0c:03	Broadcast	ARP	60	Who has 10.0.3.247? Tell 10.0.0.242
2772	32.810715072	AcctonTe_12:cf:1e	Broadcast	ARP	60	Who has 10.0.3.277? Tell 10.0.0.1
2773	32.852002755	Liteon_2a:2c:26	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.0.61
2780	33.085931701	Micro-St_a1:22:2f	Broadcast	ARP	60	Who has 10.0.3.195? Tell 10.0.0.112
2781	33.089956944	Micro-St_a1:22:2f	Broadcast	ARP	60	Who has 10.0.3.196? Tell 10.0.0.112
2788	33.097103513	Micro-St_0b:f1:72	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.45
2792	33.173765250	AsustekC_d5:7d:39	Broadcast	ARP	60	Who has 10.0.1.249? Tell 10.0.0.46
2793	33.173773696	AsustekC_d5:7d:39	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.46
2798	33.242789121	Micro-St_0b:f1:72	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.46
2801	33.357111266	Elitegro_a4:bf:c1	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.248
2804	33.394433877	Netgear_5d:22:fe	Broadcast	ARP	60	Who has 10.0.0.28? Tell 10.0.0.146
2812	33.466812170	Micro-St_03:86:32	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.157
2814	33.541928055	SpeedDra_0f:a3:63	Broadcast	ARP	60	Who has 10.0.5.172? Tell 10.0.0.148
2817	33.623753950	HonHaiPr_e4:5c:a6	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.134
2821	33.681203321	Micro-St_0b:f1:72	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.45
2822	33.68171088	Micro-St_0b:f1:72	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.7.237
2823	33.68171088	Micro-St_0b:f1:72	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.7.237
2824	33.712695955	RealtekS_68:0c:03	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.19
2827	33.792160619	28:87:ba:ef:ec:9b	Broadcast	ARP	60	Who has 10.0.3.247? Tell 10.0.0.242
2830	33.854319516	Liteon_2a:2c:26	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.0.61
2833	33.918738957	84:a9:38:77:08:8d	Broadcast	ARP	60	Who has 10.0.4.206? Tell 10.0.0.1.109
2838	33.961919414	HewlettP_7b:1d:ed	Broadcast	ARP	60	Who has 10.0.6.183? Tell 10.0.0.1.8
2840	34.086003373	Micro-St_a1:22:2f	Broadcast	ARP	60	Who has 10.0.3.195? Tell 10.0.0.112
2841	34.089956995	Micro-St_a1:22:2f	Broadcast	ARP	60	Who has 10.0.3.196? Tell 10.0.0.112
2842	34.104851415	Netgear_5d:22:fe	Broadcast	ARP	60	Who has 10.0.0.0? Tell 10.0.0.28

Capture Length: 60 bytes (480 bits)

[Frame is marked: False]

[Frame is ignored: False]

[Protocols in frame: eth:ethertype:arp]

[Coloring Rule Name: ARP]

0000 ff ff ff ff ff ff ff ff d2 48 c1 08 06 00 01 ... .. H ...

0010 08 00 06 04 00 01 d4 3d 7e d2 48 c1 0a 05 ff ... .. H ...

0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ... .. H ...

Specifies if this is an individual (unicast) or group (broadcast/multicast) address (eth.ig). 3 bytes

Packets: 3176 - Displayed: 688 (21.7%) - Marked: 688 (21.7%) - Dropped: 0 (0.0%) - Profile: Default

Wireshark

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-F>

Packet list

No.	Time	Source	Destination	Protocol	Length	Info
5	0.070706931	Micro-St_a1:22:2f	Broadcast	ARP	60	Who has 10.0.3.184? Tell 10.0.0.1.12
6	0.074538719	Micro-St_a1:22:2f	Broadcast	ARP	60	Who has 10.0.3.185? Tell 10.0.0.1.12
12	0.143639646	HonHaiPr_e4:98:0d	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.1.12
13	0.224405778	HonHaiPr_e4:98:0d	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.1.12
19	0.287130710	Micro-St_0b:f1:72	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.1.12
20	0.321658841	Micro-St_38:c2:2b	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.1.12
27	0.391956821	Netgear_5d:22:fe	Broadcast	ARP	60	Who has 10.0.0.28? Tell 10.0.0.1.12
29	0.435992263	Micro-St_1a:6c:71	Broadcast	ARP	60	Who has 10.0.0.252? Tell 10.0.0.1.12
30	0.578942960	Liteon_60:de:66	Broadcast	ARP	60	Who has 10.0.0.252? Tell 10.0.0.1.12
31	0.642988254	HonHaiPr_e4:5c:a6	Broadcast	ARP	60	Who has 10.0.0.134? Tell 10.0.0.1.12
34	0.681227554	Micro-St_40:3f:c6	Broadcast	ARP	60	Who has 10.0.0.237? Tell 10.0.0.1.12
35	0.723643276	HonHaiPr_e4:98:0d	Broadcast	ARP	60	Who has 10.0.0.132? Tell 10.0.0.1.12
36	0.733439901	Micro-St_39:bb:5d	Broadcast	ARP	60	Who has 10.0.0.0? Tell 10.0.0.1.12
38	0.782715826	HonHaiPr_e4:9b:a2	Broadcast	ARP	60	Who has 10.0.0.123? Tell 10.0.0.1.12
39	0.822797497	Liteon_ae:38:a1	Broadcast	ARP	60	Who has 10.0.0.3.5? Tell 10.0.0.1.12
66	0.961242958	Micro-St_46:1c:16	Broadcast	ARP	60	Who has 10.0.0.236? Tell 10.0.0.1.12
68	1.070758011	Micro-St_a1:22:2f	Broadcast	ARP	60	Who has 10.0.0.1.12? Tell 10.0.0.1.12
69	1.070769288	Micro-St_a1:22:2f	Broadcast	ARP	60	Who has 10.0.0.1.12? Tell 10.0.0.1.12
70	1.072489300	LcfcHeFe_52:f1:15	Broadcast	ARP	60	Who has 10.0.4.155? Tell 10.0.0.1.12
71	1.100121494	Micro-St_38:c2:2b	Broadcast	ARP	60	Who has 10.0.0.1.12? Tell 10.0.0.1.12
73	1.222898971	HonHaiPr_e4:98:0d	Broadcast	ARP	60	Who has 10.0.0.132? Tell 10.0.0.1.12
81	1.323629839	Liteon_60:de:66	Broadcast	ARP	60	Who has 10.0.0.254? Tell 10.0.0.1.12
82	1.328148877	AcctonTe_12:cf:1e	Broadcast	ARP	60	Who has 10.0.0.1.1? Tell 10.0.0.1.12
93	1.625621246	HonHaiPr_e4:5c:a6	Broadcast	ARP	60	Who has 10.0.0.134? Tell 10.0.0.1.12
94	1.657160825	Micro-St_0b:f1:72	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.45
95	1.665819568	LcfcHeFe_52:f1:15	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.45
99	1.722139673	HonHaiPr_e4:98:0d	Broadcast	ARP	60	Who has 10.0.4.133? Tell 10.0.0.132

Wireshark - Packet 5 Comment

this is first packet

OK Cancel Help

Ethernet II, Src: Micro-St\_a1:22:2f (d4:3d:7e:a1:22:2f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

Destination: Broadcast (ff:ff:ff:ff:ff:ff)

Address: Broadcast (ff:ff:ff:ff:ff:ff)

... .. LG bit: Locally administered address (this is NOT the factory default)

... .. LG bit: Group address (multicast/broadcast)

0000 ff ff ff ff ff ff ff ff d4 3d 7e a1 22 2f 08 06 00 01 ... .. H ...

Wireshark

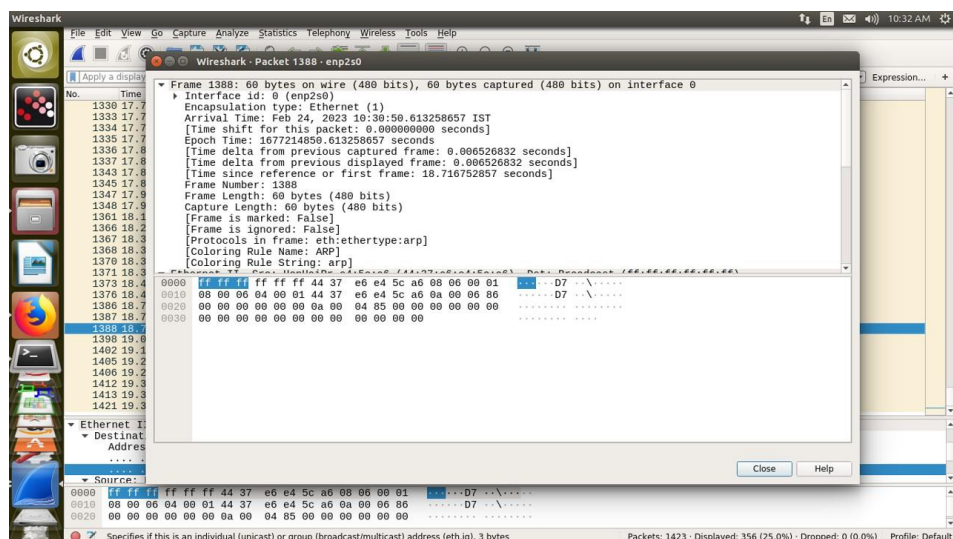
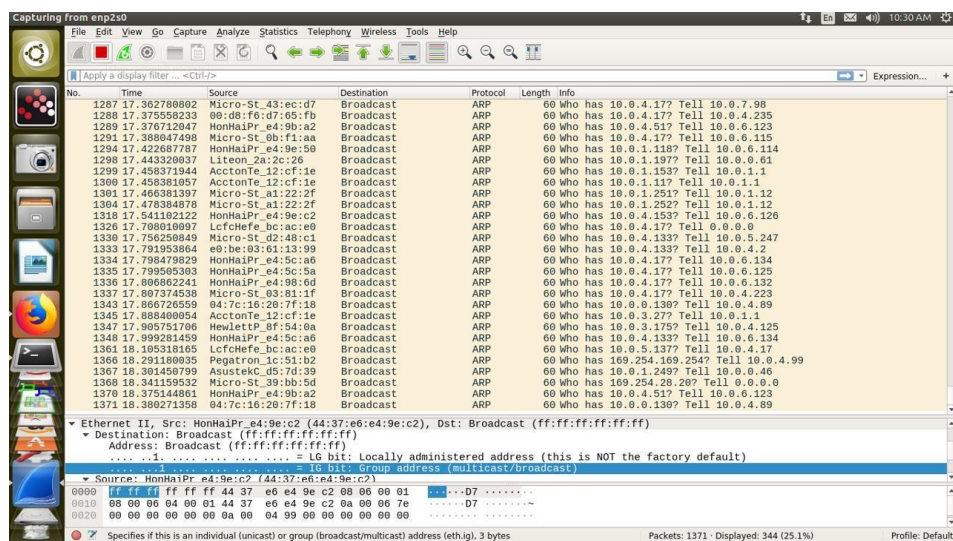
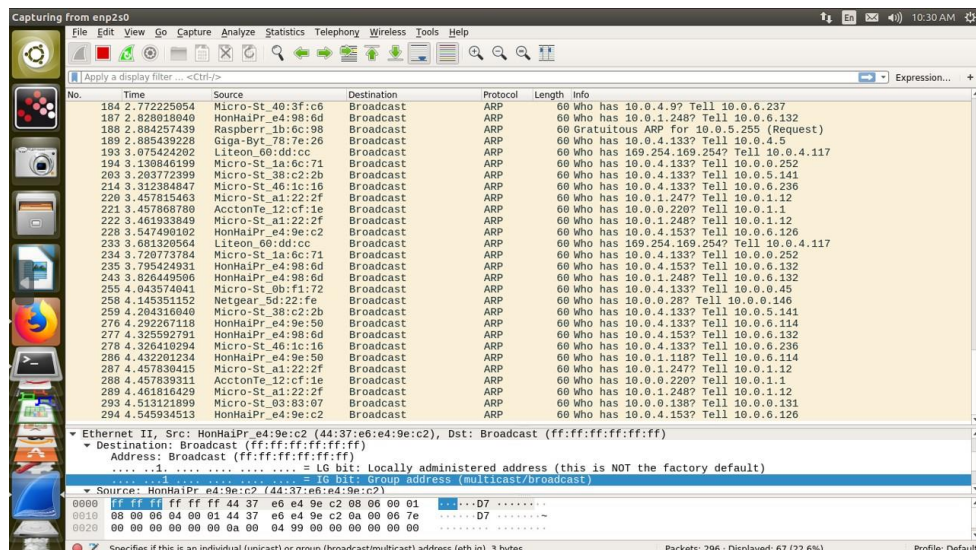
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

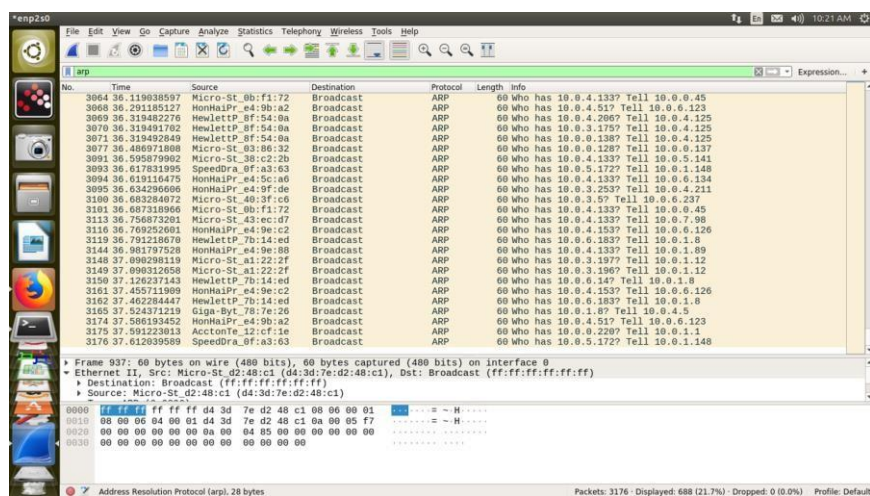
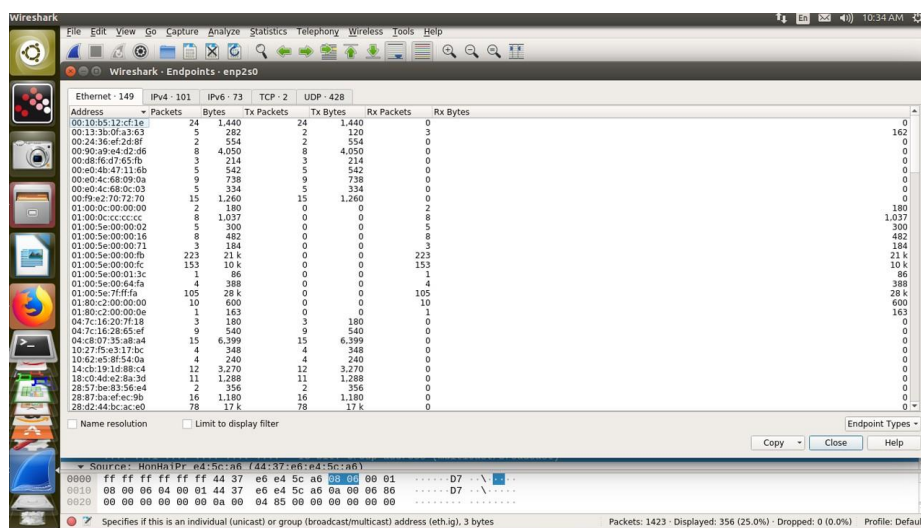
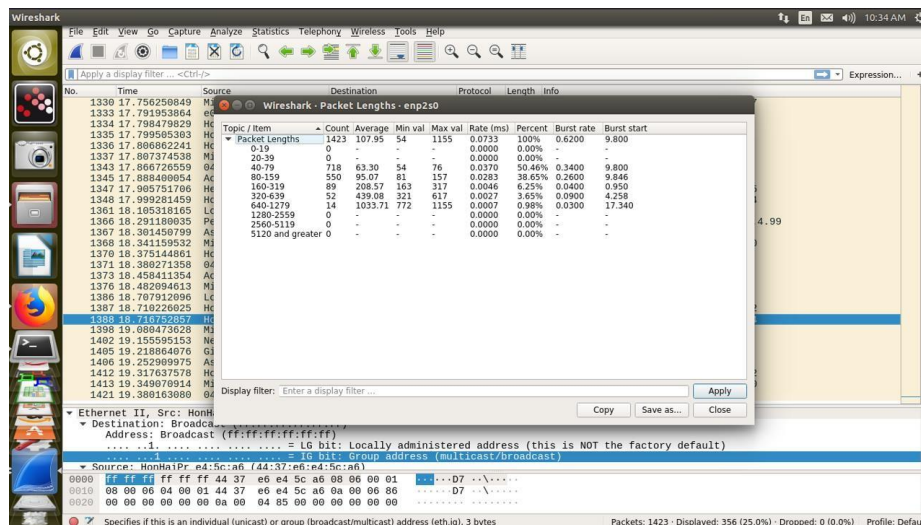
Apply a display filter ... <Ctrl-F>

Wireshark - Protocol Hierarchy Statistics: enp2s0

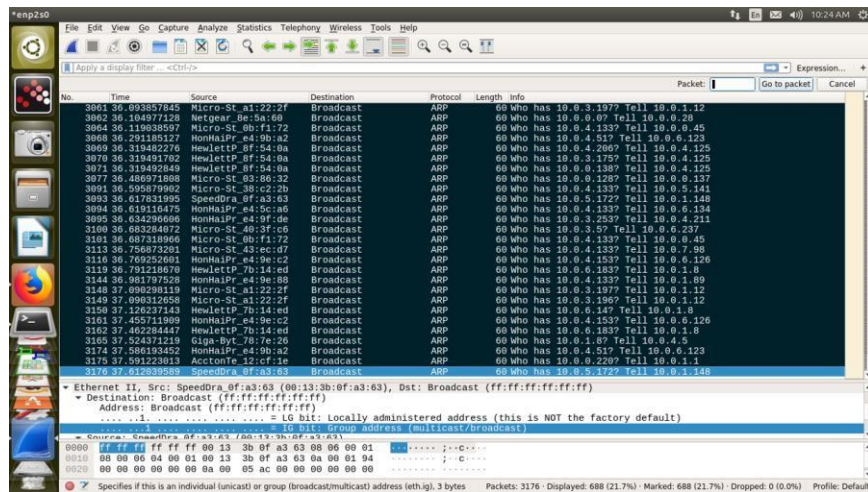
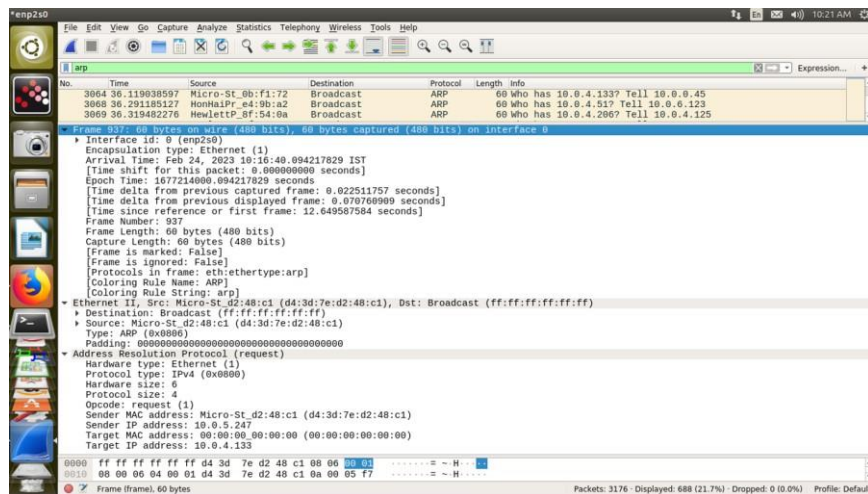
No.	Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
1330	Ethernet	100.0	356	100.0	21360	8.834	0	0	0
1331	Address Resolution Protocol	100.0	356	86.7	4984	2.063	0	9968	4.122











## Conclusion: