07_Jignesh Bhandi

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.

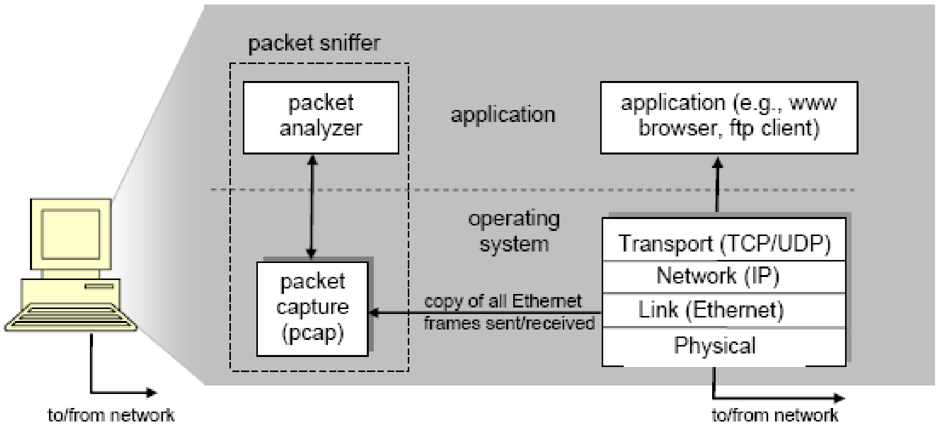


Figure 1: Packet sniffer structure

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.

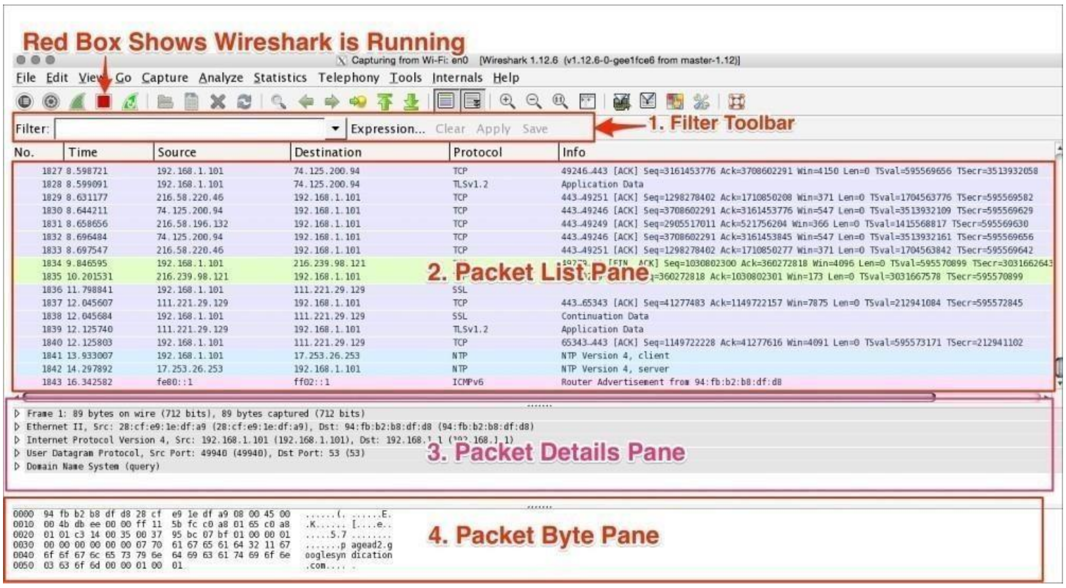


Figure2: Wireshark Windowpanes

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).

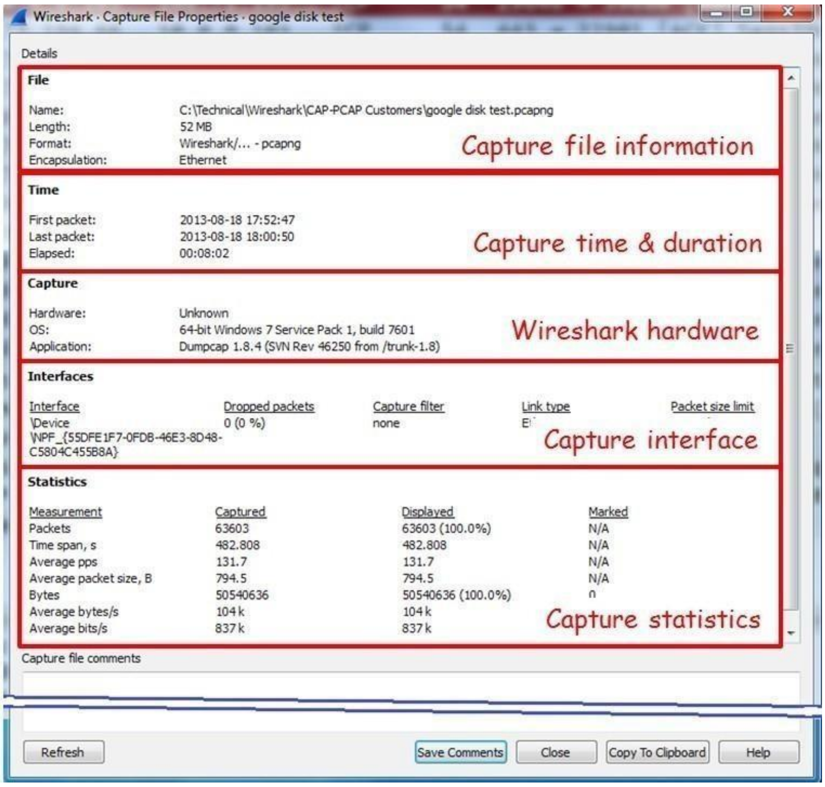


Figure3:Capture File Properties

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 Graph**s

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).

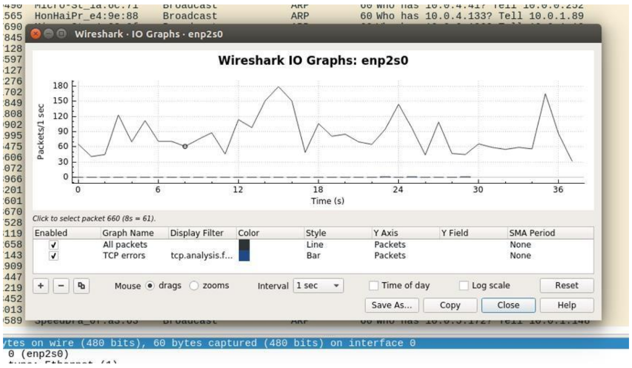
- **Flow Graphing**

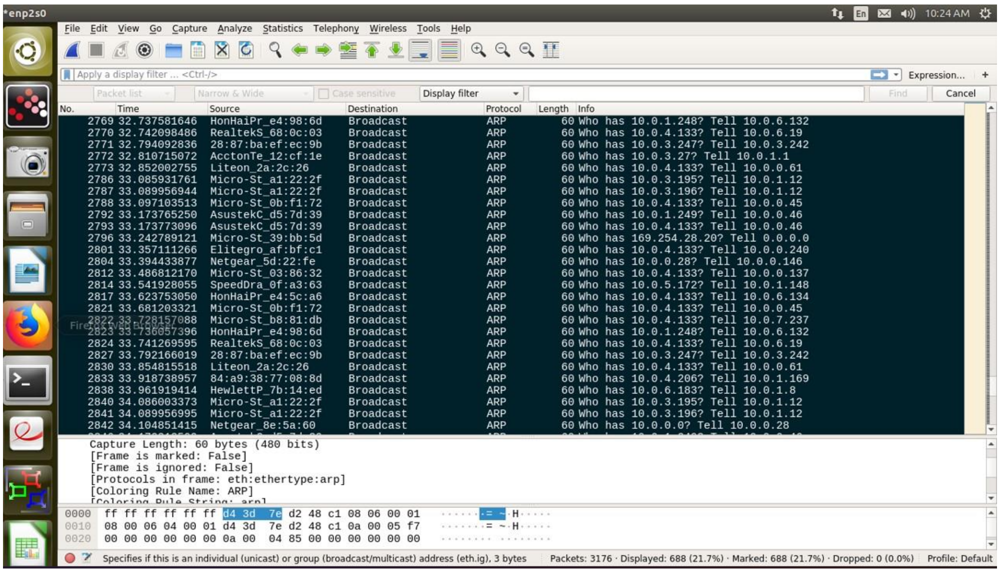
The flow graphing feature is very useful for visualizing connections and showing the flow of data over time. Basically, a flow graph contains a column based view of a connection between hosts and organizes the traffic which can be interpreted visually.It provides a quick and easy to use way of checking 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.

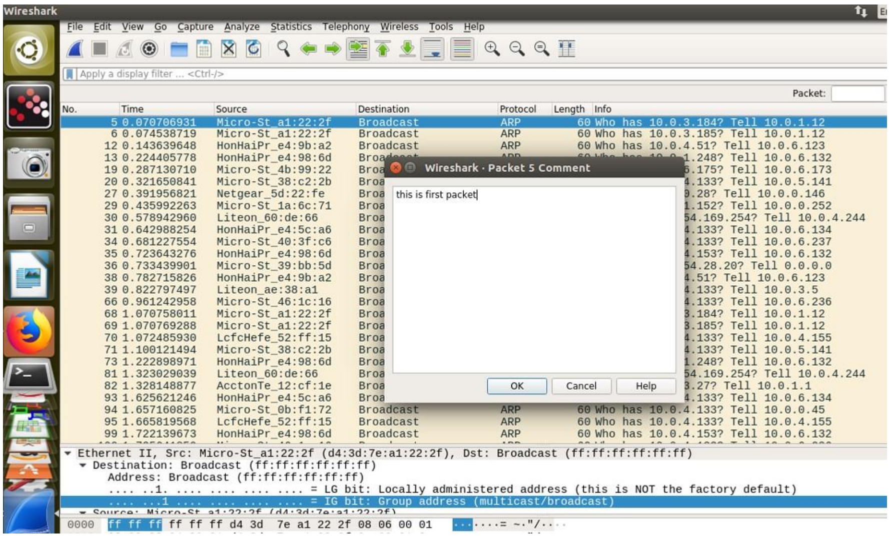
Steps:

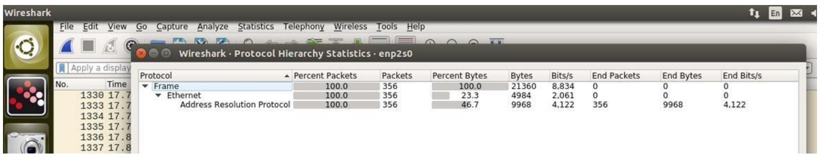
1. Start wireshark and select the interface to begin packet capture.
2. A packet capture summary window appears which shows the packets being captured.
3. Stop the packet capturing process after few seconds.
4. Inspect the packet capture window and explore the various protocols.
5. Explore the different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

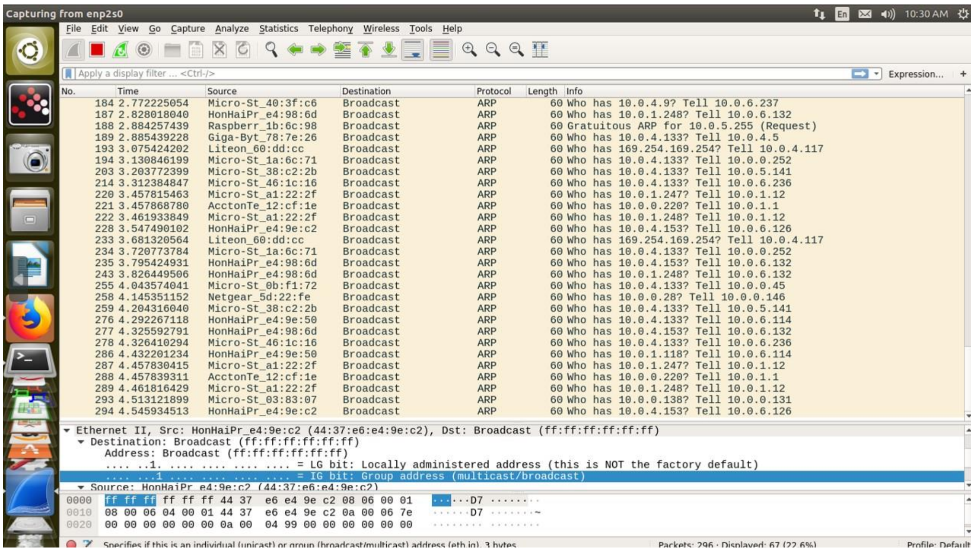
Screenshot of O/P:

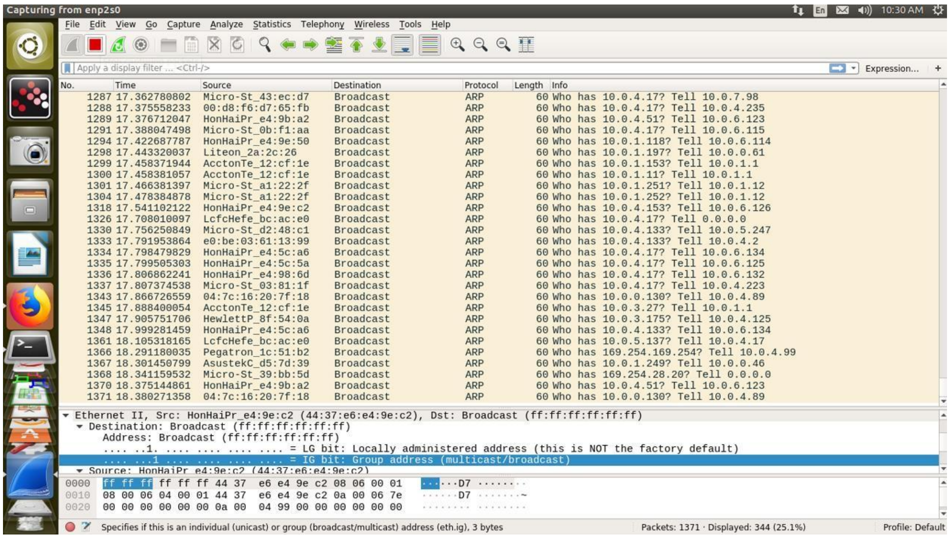


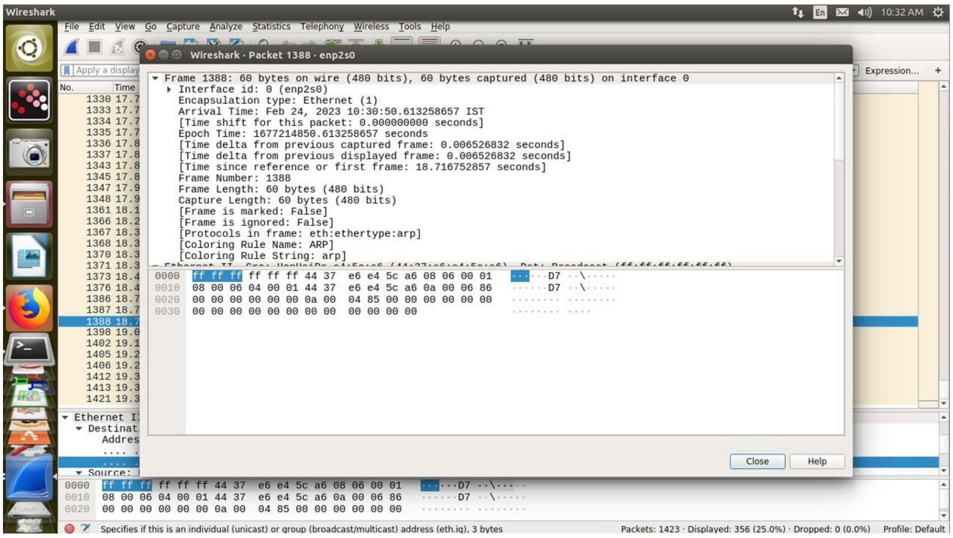


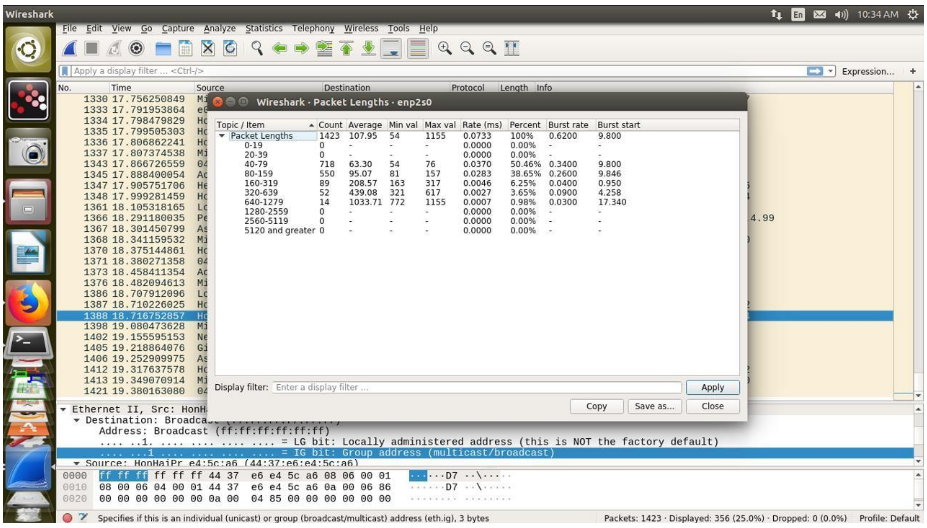


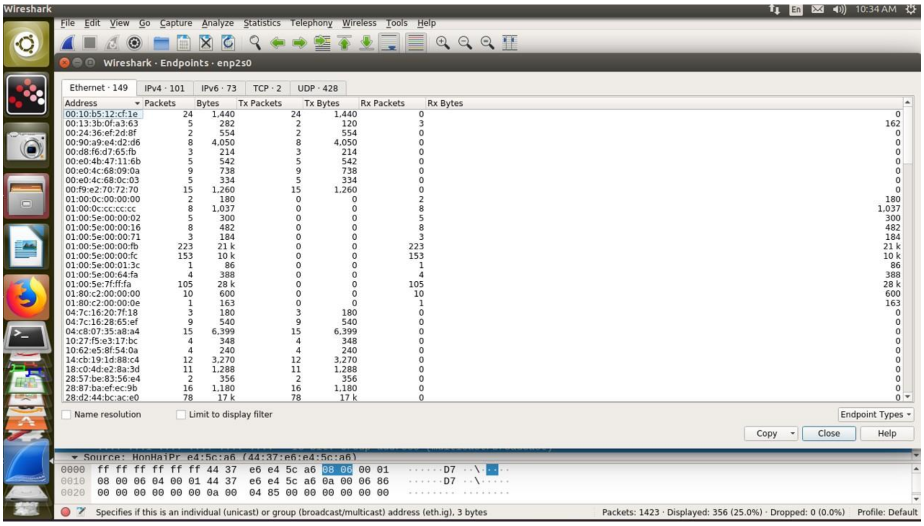


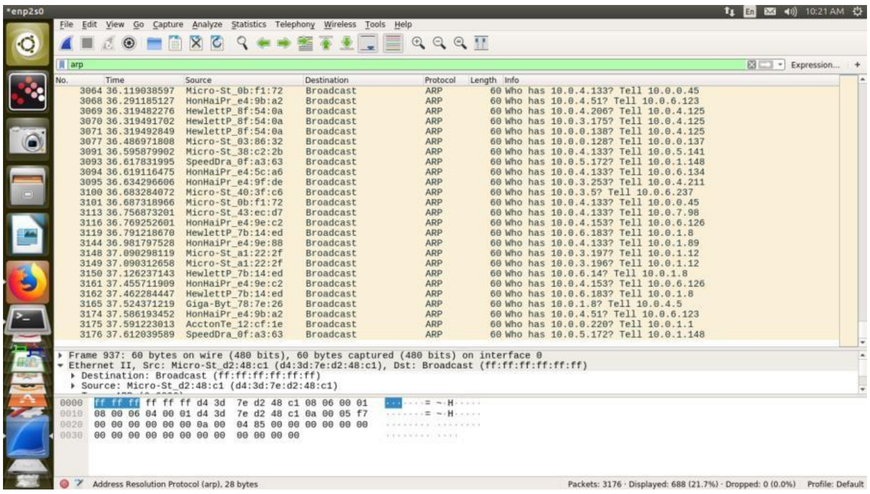

Capturing from enp2s0
Ethernet II, Src: HonHaiPr_e4:9e:c2 (44:37:e6:e4:9e:c2), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Specifies if this is an individual (unicast) or group (broadcast/multicast) address (eth.ig), 3 bytes
Packets: 296 · Displayed: 67 (22.6%)

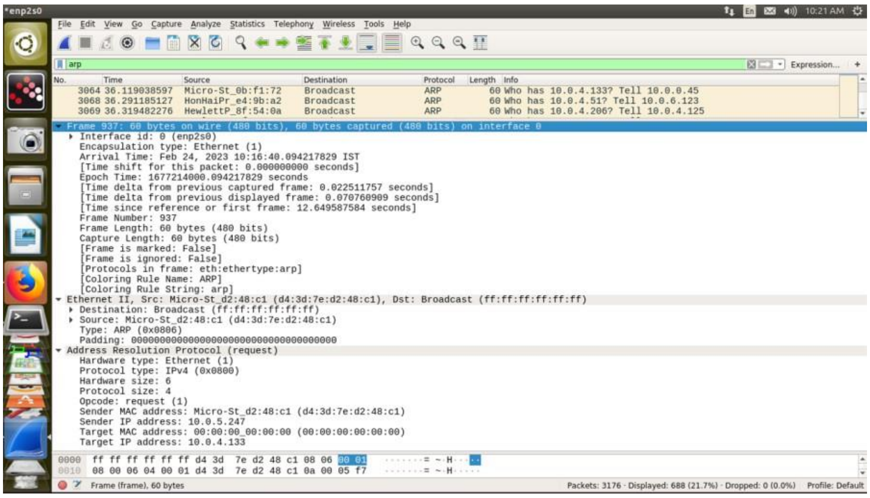

Capturing from enp2s0
Ethernet II, Src: HonHaiPr_e4:9e:c2 (44:37:e6:e4:9e:c2), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Specifies if this is an individual (unicast) or group (broadcast/multicast) address (eth.ig), 3 bytes
Packets: 1371 · Displayed: 344 (25.1%)
Profile: Default

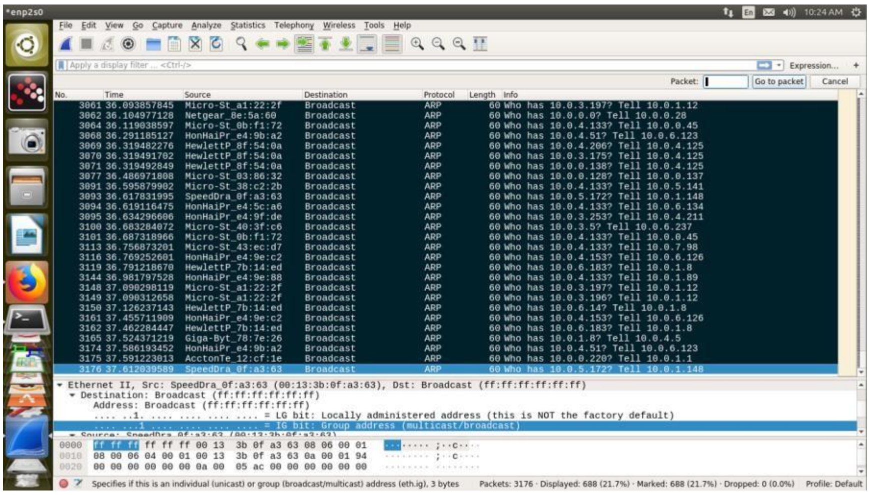

Wireshark · Packet 1388 · enp2s0
Frame 1388: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Interface id: 0 (enp2s0)
Encapsulation type: Ethernet (1)
Arrival Time: Feb 24, 2023 10:30:50.613258657 IST
Frame Number: 1388
Frame Length: 60 bytes (480 bits)
Capture Length: 60 bytes (480 bits)
[Protocols in frame: eth:ethertype:arp]
[Coloring Rule Name: ARP]
[Coloring Rule String: arp]
Packets: 1423 · Displayed: 356 (25.0%) · Dropped: 0 (0.0%)
Profile: Default


Wireshark · Packet Lengths · enp2s0


Wireshark · Endpoints · enp2s0




File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
arp
3064 36.119038597 Micro-St_0b:f1:72 Broadcast ARP 60 Who has 10.0.4.133? Tell 10.0.0.45
3068 36.291185127 HonHaiPr_e4:9b:a2 Broadcast ARP 60 Who has 10.0.4.51? Tell 10.0.6.123
3069 36.319482276 HewlettP_8f:54:0a Broadcast ARP 60 Who has 10.0.4.206? Tell 10.0.4.125
Frame 937: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Interface id: 0 (enp2s0)
Encapsulation type: Ethernet (1)
Arrival Time: Feb 24, 2023 10:16:40.094217829 IST
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1677214000.094217829 seconds
[Time delta from previous captured frame: 0.022511757 seconds]
[Time delta from previous displayed frame: 0.070760909 seconds]
[Time since reference or first frame: 12.649587584 seconds]
Frame Number: 937
Frame Length: 60 bytes (480 bits)
Capture Length: 60 bytes (480 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: Micro-St_d2:48:c1 (d4:3d:7e:d2:48:c1), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Destination: Broadcast (ff:ff:ff:ff:ff:ff)
Source: Micro-St_d2:48:c1 (d4:3d:7e:d2:48:c1)
Type: ARP (0x0806)
Address Resolution Protocol (request)
Hardware type: Ethernet (1)
Protocol type: IPv4 (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (1)
Sender MAC address: Micro-St_d2:48:c1 (d4:3d:7e:d2:48:c1)
Sender IP address: 10.0.5.247
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
Target IP address: 10.0.4.133
Frame (frame), 60 bytes
Packets: 3176 · Displayed: 688 (21.7%) · Dropped: 0 (0.0%)
Profile: Default


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Packet:
Go to packet
Cancel
Ethernet II, Src: SpeedDra_0f:a3:63 (00:13:3b:0f:a3:63), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Destination: Broadcast (ff:ff:ff:ff:ff:ff)
Address: Broadcast (ff:ff:ff:ff:ff:ff)
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

Conclusion: