CS-356 Computer Networks Lab
Lab Session-1

Basic Networking Commands in Linux

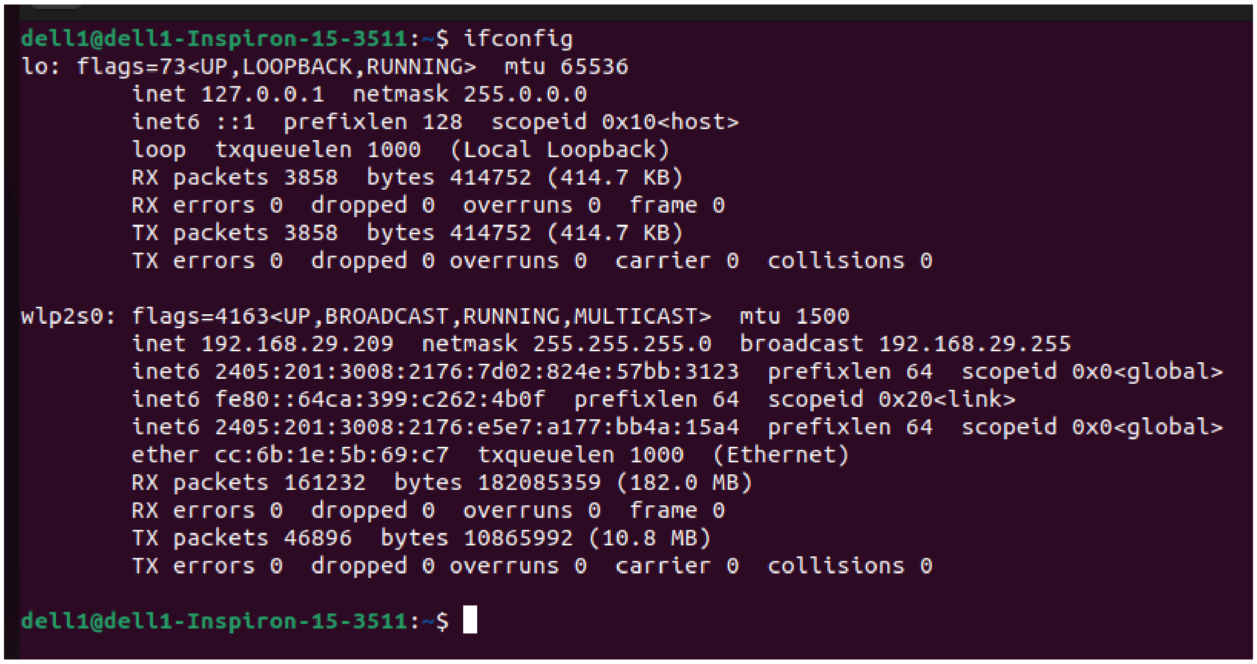
Following are some basic commands that users can use for simple network configuration tasks in Linux.

1. **ifconfig:** The ifconfig command, short for "interface configuration," is used to view and change the configuration of network interfaces on a Linux system. Although it is being replaced by the ip command in modern Linux distributions, it is still commonly used in some cases. Here are some examples of using ifconfig

   1.1. You can view the IP address, MAC address and MTU (Maximum Transmission Unit) with ifconfig command.

   Syntax:

   ```
   ifconfig
   ```

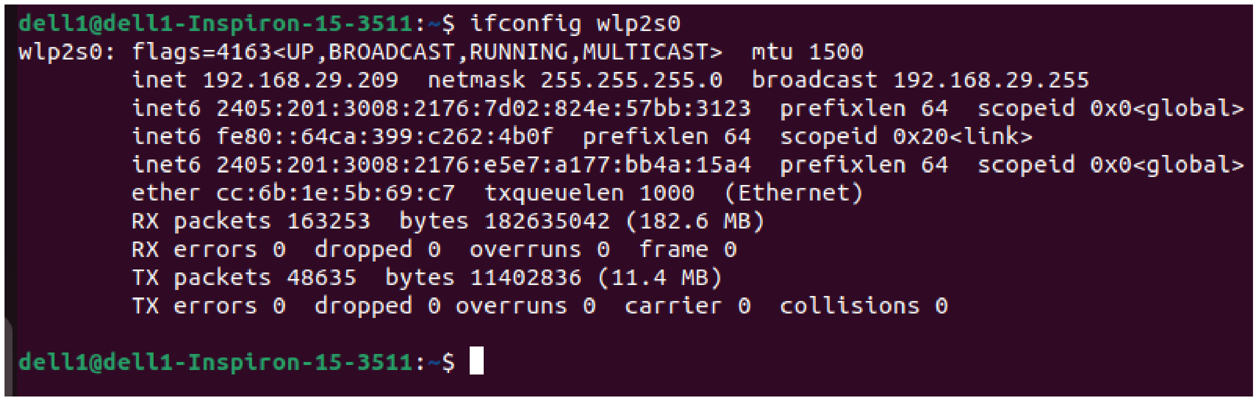


```
dell1@dell1-Inspiron-15-3511:~$ ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 3858  bytes 414752 (414.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3858  bytes 414752 (414.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.29.209  netmask 255.255.255.0  broadcast 192.168.29.255
        inet6 2405:201:3008:2176:7d02:824e:57bb:3123  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::64ca:399:c262:4b0f  prefixlen 64  scopeid 0x20<link>
        inet6 2405:201:3008:2176:e5e7:a177:bb4a:15a4  prefixlen 64  scopeid 0x0<global>
        ether cc:6b:1e:5b:69:c7  txqueuelen 1000  (Ethernet)
        RX packets 161232  bytes 182085359 (182.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 46896  bytes 10865992 (10.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

dell1@dell1-Inspiron-15-3511:~$ 
```

   1.2. To get the details of the specific interface, just write the interface name after the command.



```
dell1@dell1-Inspiron-15-3511:~$ ifconfig wlp2s0
wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.29.209  netmask 255.255.255.0  broadcast 192.168.29.255
        inet6 2405:201:3008:2176:7d02:824e:57bb:3123  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::64ca:399:c262:4b0f  prefixlen 64  scopeid 0x20<link>
        inet6 2405:201:3008:2176:e5e7:a177:bb4a:15a4  prefixlen 64  scopeid 0x0<global>
        ether cc:6b:1e:5b:69:c7  txqueuelen 1000  (Ethernet)
        RX packets 163253  bytes 182635042 (182.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 48635  bytes 11402836 (11.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

dell1@dell1-Inspiron-15-3511:~$ 
```

1.3. Other Options with ifconfig:

Display a shortlist of active interfaces: Using the -s option prints a shortlist of all interfaces, including those that are "down" (inactive)

```
ifconfig -s
```

Print a more detailed configuration for all interfaces: The -v option displays a more detailed configuration for all interfaces.

```
ifconfig -v
```

Assign an IP address to a network interface: To assign an IP address to a network interface, you can use the inet option followed by the IP address and netmask. For example, to assign the IP address 192.168.1.1 with a netmask 255.255.255.0 to the eth0 interface, you would run:

```
ifconfig eth0 inet 192.168.1.1 netmask 255.255.255.0
```

but these settings will be disabled after system reboot.

1.4. Enable or Disable specific interface:

To enable specific interface,

```
ifup eth0
```

To disable specific interface,

```
ifdown eth0
```

1.5. Setting MTU size

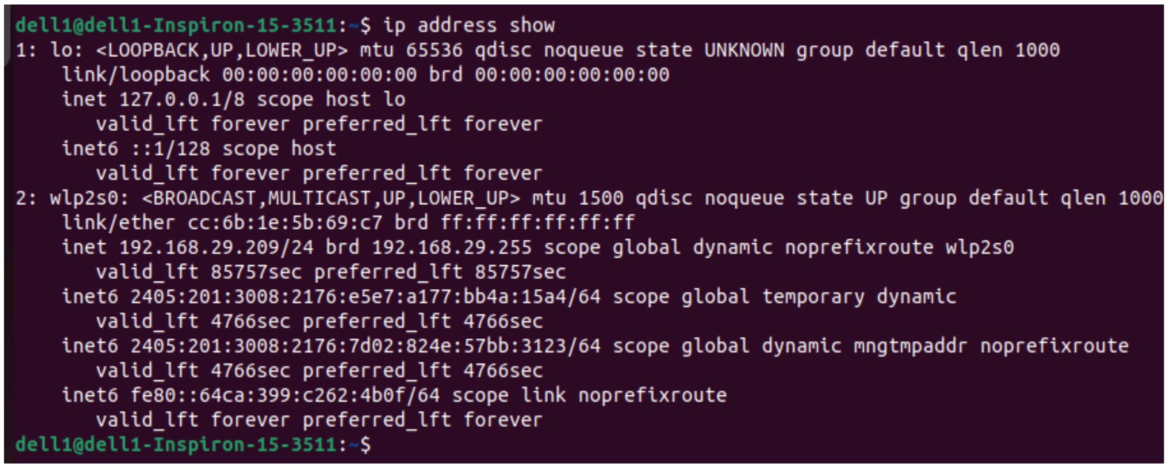
By default MTU (Maximum Transmission Unit) size is 1500, you can change size as per your wish.

```
ifconfig eth0 mtu xxxx
```

Replace xxxx with size.

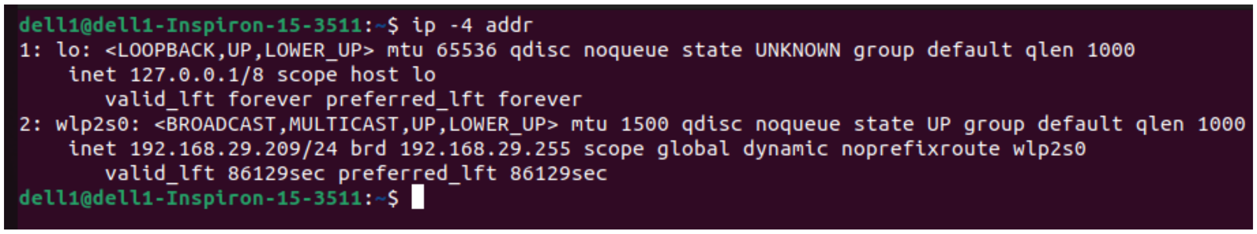
2. **ip command:** Linux IP command is the newer version of the ifconfig command. It is a handy tool for configuring the network interfaces for Linux administrators. It can be used to assign and remove addresses, take the interfaces up or down, and much more useful tasks.

2.1.Listing the IP addresses: `ip address show`

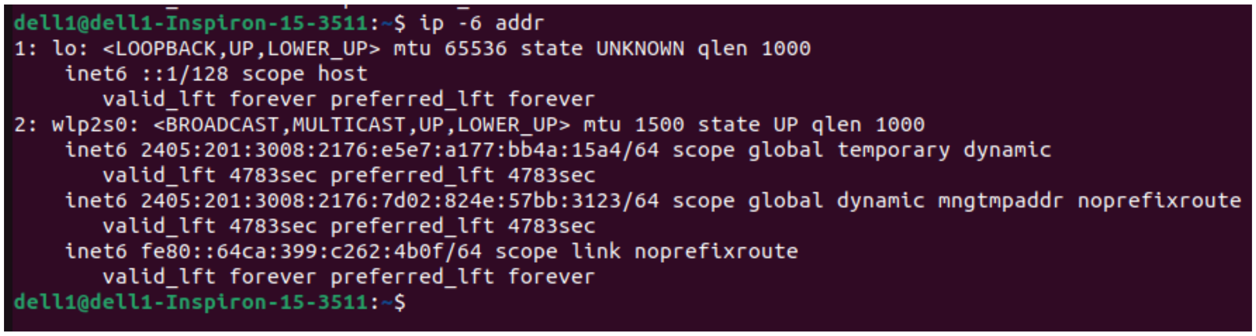


```
dell1@dell1-Inspiron-15-3511:~$ ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether cc:6b:1e:5b:69:c7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.29.209/24 brd 192.168.29.255 scope global dynamic noprefixroute wlp2s0
       valid_lft 85757sec preferred_lft 85757sec
    inet6 2405:201:3008:2176:e5e7:a177:bb4a:15a4/64 scope global temporary dynamic
       valid_lft 4766sec preferred_lft 4766sec
    inet6 2405:201:3008:2176:7d02:824e:57bb:3123/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 4766sec preferred_lft 4766sec
    inet6 fe80::64ca:399:c262:4b0f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
dell1@dell1-Inspiron-15-3511:~$
```

2.2. Display the Ipv4 and Ipv6 Addresses

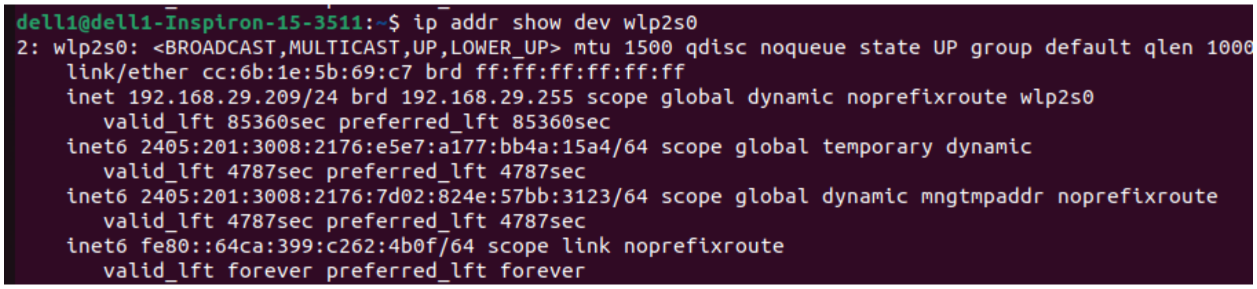


```
dell1@dell1-Inspiron-15-3511:~$ ip -4 addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    inet 192.168.29.209/24 brd 192.168.29.255 scope global dynamic noprefixroute wlp2s0
       valid_lft 86129sec preferred_lft 86129sec
dell1@dell1-Inspiron-15-3511:~$
```



```
dell1@dell1-Inspiron-15-3511:~$ ip -6 addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 state UNKNOWN qlen 1000
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 2405:201:3008:2176:e5e7:a177:bb4a:15a4/64 scope global temporary dynamic
       valid_lft 4783sec preferred_lft 4783sec
    inet6 2405:201:3008:2176:7d02:824e:57bb:3123/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 4783sec preferred_lft 4783sec
    inet6 fe80::64ca:399:c262:4b0f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
dell1@dell1-Inspiron-15-3511:~$
```

2.3. Display a Single Interface

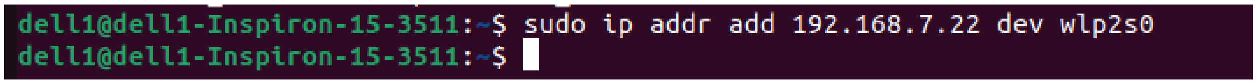


```
dell1@dell1-Inspiron-15-3511:~$ ip addr show dev wlp2s0
2: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether cc:6b:1e:5b:69:c7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.29.209/24 brd 192.168.29.255 scope global dynamic noprefixroute wlp2s0
       valid_lft 85360sec preferred_lft 85360sec
    inet6 2405:201:3008:2176:e5e7:a177:bb4a:15a4/64 scope global temporary dynamic
       valid_lft 4787sec preferred_lft 4787sec
    inet6 2405:201:3008:2176:7d02:824e:57bb:3123/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 4787sec preferred_lft 4787sec
    inet6 fe80::64ca:399:c262:4b0f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

2.4 Add an IP address to an Interface

The 'add' and 'dev' options are used to add an IP address to an interface. We have to specify the IP address and interface to be added. For example, add the IP address "192.168.7.22" to the 'wlp6s0' interface. To add this Ip address, execute the command as follows:

```
sudo ip addr add 192.168.7.22 dev wlp2s0
```



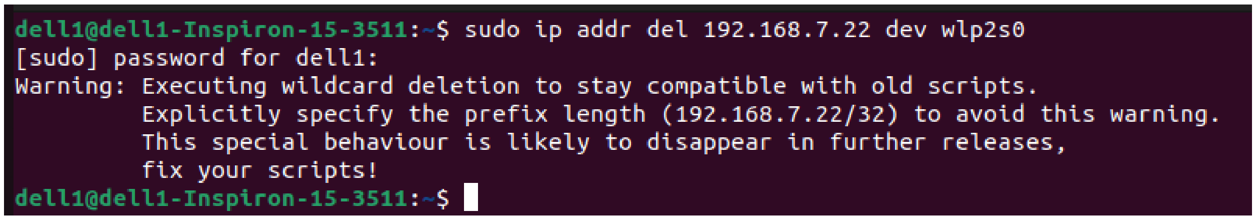
```
dell1@dell1-Inspiron-15-3511:~$ sudo ip addr add 192.168.7.22 dev wlp2s0
dell1@dell1-Inspiron-15-3511:~$
```

The above command will add the given IP address to the specified interface. If it is successfully added, it will not produce any output.

2.5. Deleting an IP Address

Deleting an IP address is the same as adding, except we have to replace the 'add' option with the 'del.' To delete an IP, execute the command as follows:

```
sudo ip addr del 192.168.7.22 dev wlp2s0
```

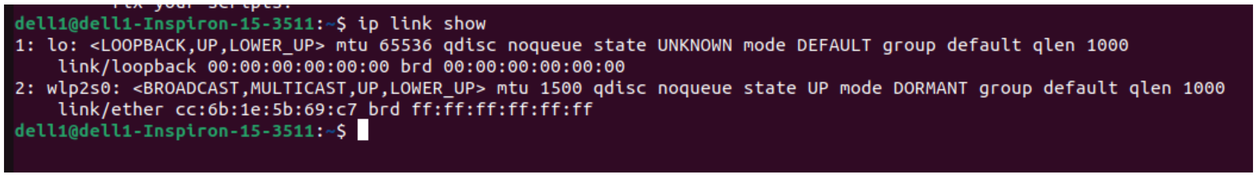


```
dell1@dell1-Inspiron-15-3511:~$ sudo ip addr del 192.168.7.22 dev wlp2s0
[sudo] password for dell1:
Warning: Executing wildcard deletion to stay compatible with old scripts.
         Explicitly specify the prefix length (192.168.7.22/32) to avoid this warning.
         This special behaviour is likely to disappear in further releases,
         fix your scripts!
dell1@dell1-Inspiron-15-3511:~$
```

2.6. IP with Network Interfaces

We can use the **link object** for working and inspecting the network interfaces. To display the installed interface on our system, execute the below command:

```
ip link show
```



```
dell1@dell1-Inspiron-15-3511:~$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DORMANT group default qlen 1000
    link/ether cc:6b:1e:5b:69:c7 brd ff:ff:ff:ff:ff:ff
dell1@dell1-Inspiron-15-3511:~$
```

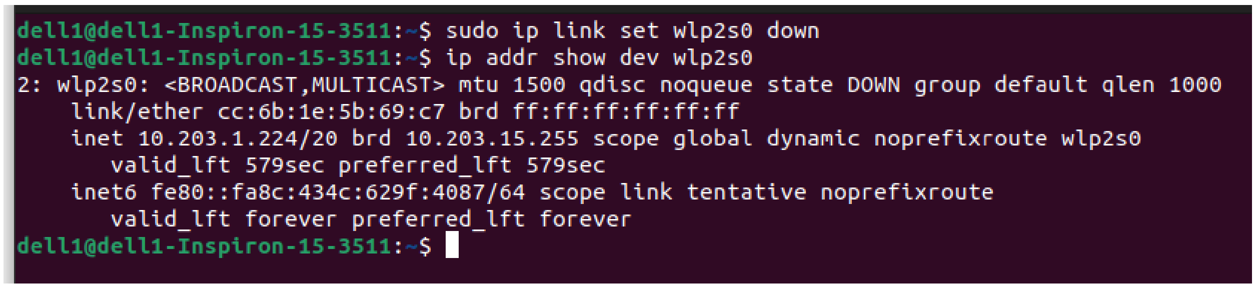
2.7. Start or Stop a Network Interface

The 'set' option with up and down arguments is used to start and stop a network interface. Consider the below command:

```
sudo ip link set wlp2s0 down
```

The above command will down the 'wlp2s0' interface. To display the status of the interface, execute the below command:

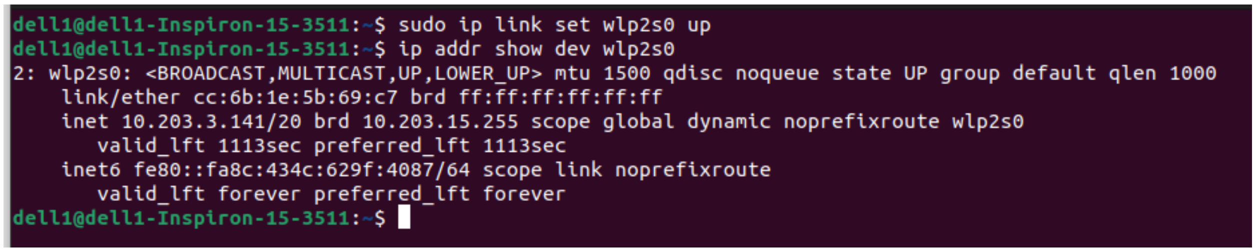
```
ip addr show dev wlp2s0
```



```
dell1@dell1-Inspiron-15-3511:~$ sudo ip link set wlp2s0 down
dell1@dell1-Inspiron-15-3511:~$ ip addr show dev wlp2s0
2: wlp2s0: <BROADCAST,MULTICAST> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether cc:6b:1e:5b:69:c7 brd ff:ff:ff:ff:ff:ff
    inet 10.203.1.224/20 brd 10.203.15.255 scope global dynamic noprefixroute wlp2s0
       valid_lft 579sec preferred_lft 579sec
    inet6 fe80::fa8c:434c:629f:4087/64 scope link tentative noprefixroute
       valid_lft forever preferred_lft forever
dell1@dell1-Inspiron-15-3511:~$
```

To start the interface, execute the below command:

```
sudo ip link set wlp2s0 up
```



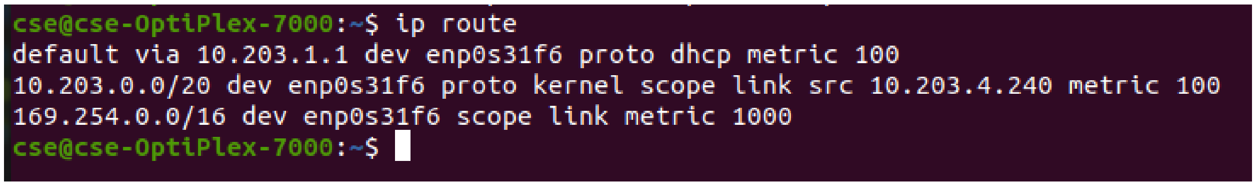
```
dell1@dell1-Inspiron-15-3511:~$ sudo ip link set wlp2s0 up
dell1@dell1-Inspiron-15-3511:~$ ip addr show dev wlp2s0
2: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether cc:6b:1e:5b:69:c7 brd ff:ff:ff:ff:ff:ff
    inet 10.203.3.141/20 brd 10.203.15.255 scope global dynamic noprefixroute wlp2s0
       valid_lft 1113sec preferred_lft 1113sec
    inet6 fe80::fa8c:434c:629f:4087/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
dell1@dell1-Inspiron-15-3511:~$
```

2.8 IP with Routes

The root object is used to inspect and manipulate routes. A route defines the forwarding process of network traffic and applied network interface. In the case of a shared network, the sending device can forward the packet directly. However, if the destination device is not directly connected, the sending device sends the packet to the default router. In this condition, the router deals with packets.

To display the defined routes of our system, execute the below command:

```
ip route
```



```
cse@cse-OptiPlex-7000:~$ ip route
default via 10.203.1.1 dev enp0s31f6 proto dhcp metric 100
10.203.0.0/20 dev enp0s31f6 proto kernel scope link src 10.203.4.240 metric 100
169.254.0.0/16 dev enp0s31f6 scope link metric 1000
cse@cse-OptiPlex-7000:~$
```

3. **traceroute command:**

A network diagnostic tool used to trace the route that packets take from the source to a specified destination. It helps you identify the network path and measure the transit delays of packets across a network. This can be useful for troubleshooting network connectivity issues, identifying bottlenecks, and understanding the network topology between your computer and a target host.
Utility:

1. **Network Path Analysis:** `traceroute` shows the route that packets take, listing all the intermediate routers (hops) between the source and the destination.
2. **Round-Trip Time (RTT) Measurement:** It displays the time taken for packets to travel from the source to each hop and back.
3. **Identifying Network Issues:** Helps in identifying network congestion, packet loss, or other issues by examining the delay at each hop.

Usage:

```
traceroute [options] destination
```

Example: `traceroute www.google.com`

Options:
`-n`: Display IP addresses instead of hostnames, which can speed up the process by avoiding DNS resolution delays.
`-q` (or `--queries`): Set the number of queries sent to each hop.
`-m` (or `--max-hop`): Set the maximum number of hops to reach the destination.

For example:

```
traceroute -n -q 3 -m 15 www.google.com
```

4. **tracepath:** The tracepath command is the same as the traceroute command, and it is used to find network delays. Besides, it does not need root privileges. By default, it comes pre-installed in Ubuntu. It traces the path to the destination and recognizes all hops in it. It identifies the point at which the network is weak if our network is not strong enough.

Syntax:

```
tracepath <destination>
```

5. ping: It is short for Packet Internet Groper. The ping command is one of the widely used commands for network troubleshooting. Basically, it inspects the network connectivity between two different nodes.

Syntax:

```
ping <destination>
```

6. netstat: It is short for network statistics. It gives statistical figures of many interfaces, which contain open sockets, connection information, and routing tables.

Syntax:

```
Netstat
```

7. ss: This command is the substitution for the netstat command. The ss command is more informative and much faster than netstat. The ss command's faster response is possible because it fetches every information from inside the kernel userspace.

Syntax:

```
ss
```

8. nslookup: The nslookup command is an older edition of the dig command. Also, it is utilized for DNS related problems.

Syntax:

```
nslookup <domainname>
```

9. dig: dig is short for Domain Information Groper. The dig command is an improvised edition of the nslookup command. It is utilized in DNS lookup to reserve the DNS name server. Also, it is used to balance DNS related problems. Mainly, it is used to authorize DNS mappings, host addresses, MX records, and every other DNS record for the best DNS topography understanding.

Syntax:

```
dig <domainname>
```

10. route: The route command shows and employs the routing table available for our system. Basically, a router is used to detect a better way to transfer the packets around a destination.

Syntax:

```
route
```

11. host: The host command shows the IP address for a hostname and the domain name for an IP address. Also, it is used to get DNS lookup for DNS related issues.

Syntax:

```
host -t <resourceName>
```

11. arp: The arp command is short for Address Resolution Protocol. This command is used to see and include content in the ARP table of the kernel.

Syntax:

```
arp
```

12. curl and wget: These commands are used to download files from CLI from the internet. curl must be specified with the "O" option to get the file, while wget is directly used.

curl Syntax:

```
curl -O <fileLink>
```

wget Syntax:

```
wget <fileLink>
```

13. mtr: The mtr command is a mix of the traceroute and ping commands. It regularly shows information related to the packets transferred using the ping time of all hops. Also, it is used to see network problems.

Syntax:

```
mtr <path>
```

14. whois: The whois command fetches every website related information. We can get every information of a website, such as an owner and the registration information.

Syntax:

```
    whois <websiteName>
```

15. ifplugstatus: The ifplugstatus command checks whether a cable is currently plugged into a network interface. It is not available in Ubuntu directly. We can install it with the help of the below command:

```
  sudo apt-get install ifplugd
```

Syntax:

```
    ifplugstatus
```

16. iftop: The iftop command is utilized in traffic monitoring.

17. tcpdump: The tcpdump command is widely used in network analysis with other commands of the Linux network. It analyses the traffic passing from the network interface and shows it. When balancing the network, this type of packet access will be crucial.

Syntax:

```
   tcpdump -i <network_device>
```

Introduction to Wireshark

1. **What is Wireshark?**
    - Open-source network protocol analyzer.
    - Used for troubleshooting, analysis, development, and education.
2. **Key Features:**
    - Packet Capture: Captures and displays packets on a network.
    - Live Analysis: Real-time monitoring of network activities.
    - Filtering: Powerful filters for isolating specific traffic.
3. **Supported Protocols:**
    - Analyzes a wide range of protocols (TCP, UDP, HTTP, DNS, etc.).
    - Provides detailed insights into packet content.
4. **Platform Compatibility:**
    - Available for Windows, macOS, and Linux.
    - Offers a consistent user experience across platforms.
5. **User Interface:**
    - Intuitive GUI for ease of use.
    - Tabbed interface for simultaneous analysis of multiple captures.
6. **Packet Details:**
    - Displays detailed information about each packet.

- Includes source/destination addresses, protocols, and payload.

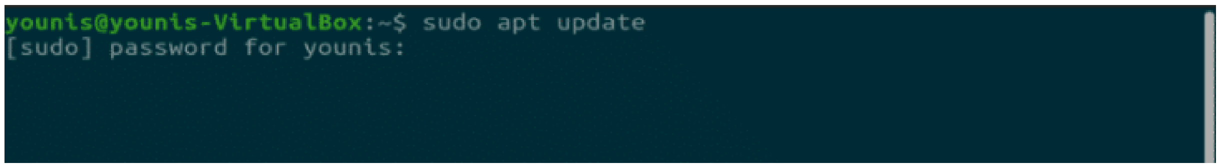
7. **Color-coded Traffic:**
    - Colors packets for quick identification (e.g., green for TCP, blue for UDP).
8. **Statistics and Reports:**
    - Generates detailed statistics on network traffic.
    - Provides graphical representations for easy interpretation.
11. **Customization:**
    - Customizable columns and display filters.
    - Tailor the interface to suit specific analysis needs.
12. **Security Analysis:**
    - Useful for detecting and analyzing security threats.
    - Identifies suspicious patterns and behaviors.
13. **Integration with Other Tools:**
    - Supports integration with third-party tools for extended functionality.
    - Enhances capabilities for specialized tasks.

Hands-On Wireshark

Steps to Install Wireshark

Step 1: Update APT

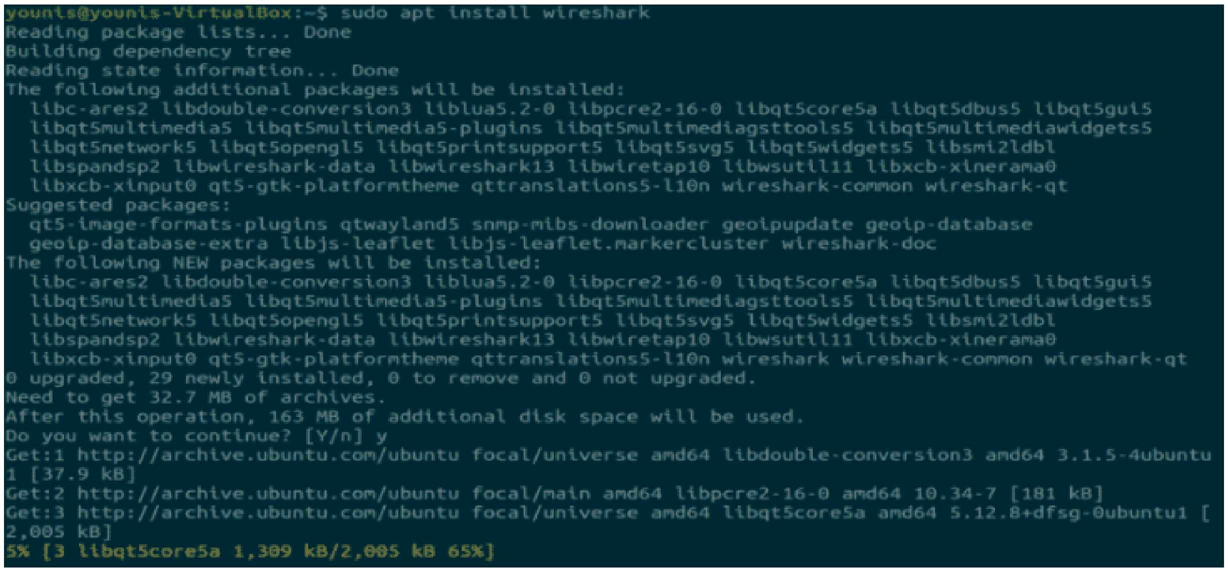
```
$ sudo apt update
```



```
younis@younis-VirtualBox:~$ sudo apt update
[sudo] password for younis:
```

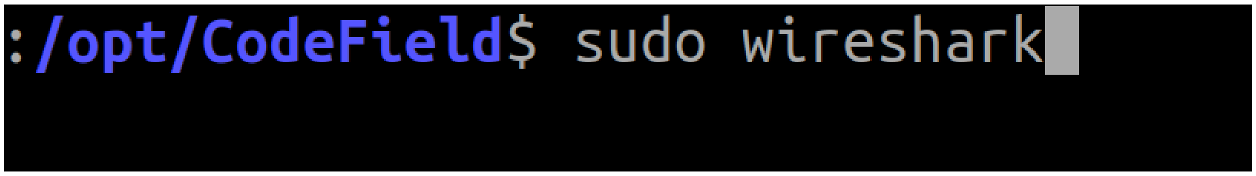
Step 2: Download and Install Wireshark

```
$ sudo apt install wireshark
```



```
younis@younis-VirtualBox:~$ sudo apt install wireshark
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libc-ares2 libdouble-conversion3 liblua5.2-0 libpcre2-16-0 libqt5core5a libqt5dbus5 libqt5gui5
  libqt5multimedia5 libqt5multimedia5-plugins libqt5multimediagsttools5 libqt5multimediawidgets5
  libqt5network5 libqt5opengl5 libqt5printsupport5 libqt5svg5 libqt5widgets5 libsmi2ldbl
  libspandsp2 libwireshark-data libwireshark13 libwiretap10 libwsutil11 libxcb-xinerama0
  libxcb-xinput0 qt5-gtk-platformtheme qttranslations5-l10n wireshark-common wireshark-qt
Suggested packages:
  qt5-image-formats-plugins qtwayland5 snmp-mibs-downloader geoipupdate geoip-database
  geoip-database-extra libjs-leaflet libjs-leaflet.markercluster wireshark-doc
The following NEW packages will be installed:
  libc-ares2 libdouble-conversion3 liblua5.2-0 libpcre2-16-0 libqt5core5a libqt5dbus5 libqt5gui5
  libqt5multimedia5 libqt5multimedia5-plugins libqt5multimediagsttools5 libqt5multimediawidgets5
  libqt5network5 libqt5opengl5 libqt5printsupport5 libqt5svg5 libqt5widgets5 libsmi2ldbl
  libspandsp2 libwireshark-data libwireshark13 libwiretap10 libwsutil11 libxcb-xinerama0
  libxcb-xinput0 qt5-gtk-platformtheme qttranslations5-l10n wireshark wireshark-common wireshark-qt
0 upgraded, 29 newly installed, 0 to remove and 0 not upgraded.
Need to get 32.7 MB of archives.
After this operation, 163 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 libdouble-conversion3 amd64 3.1.5-4ubuntu
1 [37.9 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal/main amd64 libpcre2-16-0 amd64 10.34-7 [181 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/universe amd64 libqt5core5a amd64 5.12.8+dfsg-0ubuntu1 [
2,005 kB]
5% [3 libqt5core5a 1,309 kB/2,005 kB 65%]
```

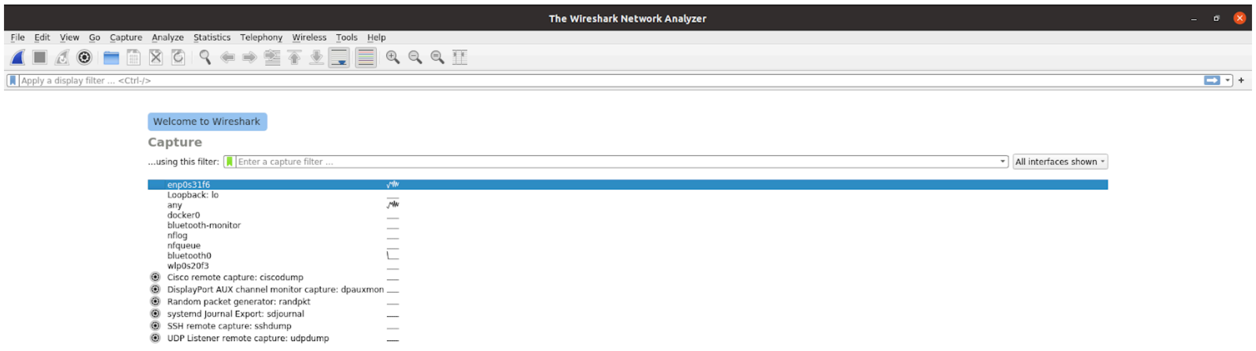
Step 3: Run Wireshark



```
:/opt/CodeField$ sudo wireshark
```

Interface and Working on Wireshark

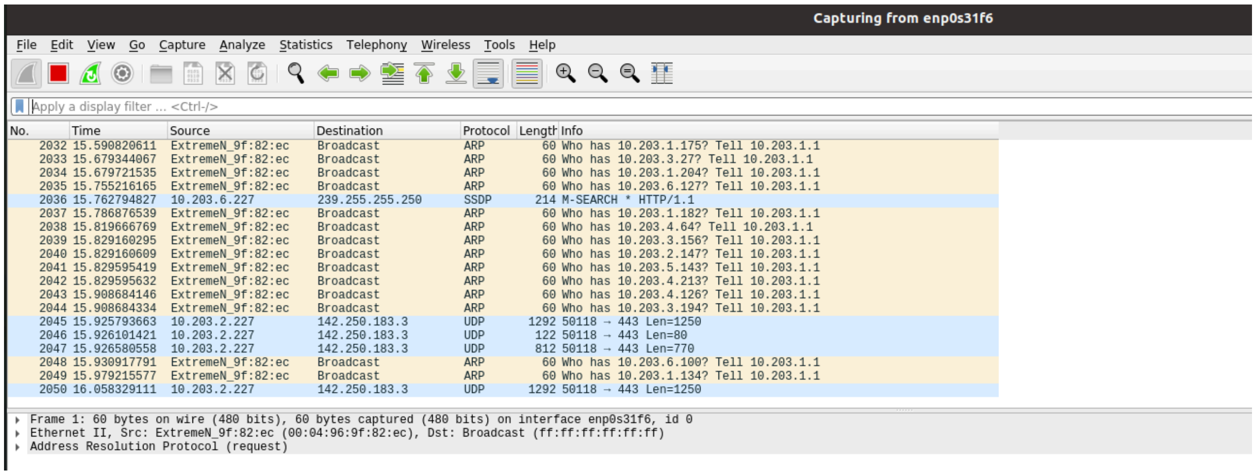
Choose the interface whose traffic you want to capture



The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown

```
enp0s31f6
Loopback: lo
any
docker0
bluetooth-monitor
nflog
nfqueue
bluetooth0
wlp0s20f3
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump
```

List of packets and their information

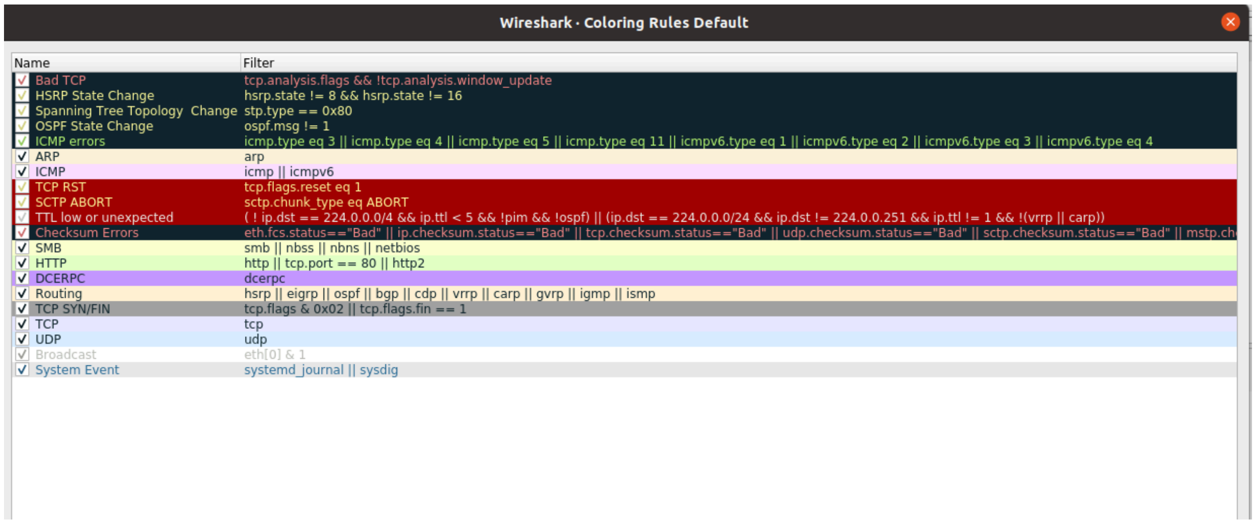

Capturing from enp0s31f6
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time         Source              Destination        Protocol Lengtl Info
 2032 15.590820611 ExtremeN_9f:82:ec   Broadcast          ARP        60 Who has 10.203.1.175? Tell 10.203.1.1
 2033 15.679344067 ExtremeN_9f:82:ec   Broadcast          ARP        60 Who has 10.203.3.27? Tell 10.203.1.1
 2034 15.679721535 ExtremeN_9f:82:ec   Broadcast          ARP        60 Who has 10.203.1.204? Tell 10.203.1.1
 2035 15.755216165 ExtremeN_9f:82:ec   Broadcast          ARP        60 Who has 10.203.6.127? Tell 10.203.1.1
 2036 15.762794827 10.203.6.227        239.255.255.250    SSDP      214 M-SEARCH * HTTP/1.1
 2037 15.786876539 ExtremeN_9f:82:ec   Broadcast          ARP        60 Who has 10.203.1.182? Tell 10.203.1.1
 2038 15.819666769 ExtremeN_9f:82:ec   Broadcast          ARP        60 Who has 10.203.4.64? Tell 10.203.1.1
 2039 15.829160295 ExtremeN_9f:82:ec   Broadcast          ARP        60 Who has 10.203.3.156? Tell 10.203.1.1
 2040 15.829160609 ExtremeN_9f:82:ec   Broadcast          ARP        60 Who has 10.203.2.147? Tell 10.203.1.1
 2041 15.829595419 ExtremeN_9f:82:ec   Broadcast          ARP        60 Who has 10.203.5.143? Tell 10.203.1.1
 2042 15.829595632 ExtremeN_9f:82:ec   Broadcast          ARP        60 Who has 10.203.4.213? Tell 10.203.1.1
 2043 15.908684146 ExtremeN_9f:82:ec   Broadcast          ARP        60 Who has 10.203.4.126? Tell 10.203.1.1
 2044 15.908684334 ExtremeN_9f:82:ec   Broadcast          ARP        60 Who has 10.203.3.194? Tell 10.203.1.1
 2045 15.925793663 10.203.2.227        142.250.183.3      UDP      1292 50118 → 443 Len=1250
 2046 15.926101421 10.203.2.227        142.250.183.3      UDP       122 50118 → 443 Len=80
 2047 15.926580558 10.203.2.227        142.250.183.3      UDP       812 50118 → 443 Len=770
 2048 15.930917791 ExtremeN_9f:82:ec   Broadcast          ARP        60 Who has 10.203.6.100? Tell 10.203.1.1
 2049 15.979215577 ExtremeN_9f:82:ec   Broadcast          ARP        60 Who has 10.203.1.134? Tell 10.203.1.1
 2050 16.058329111 10.203.2.227        142.250.183.3      UDP      1292 50118 → 443 Len=1250

▸ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface enp0s31f6, id 0
▸ Ethernet II, Src: ExtremeN_9f:82:ec (00:04:96:9f:82:ec), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▸ Address Resolution Protocol (request)
```

- Command Menu
- Listing of captured packets
- Details of selected packet header
- Packet contents in hexadecimal and ASCII

Color code to identify the types of packet



Name	Filter
✓ Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update
✓ HSRP State Change	hsrp.state != 8 && hsrp.state != 16
✓ Spanning Tree Topology Change	stp.type == 0x80
✓ OSPF State Change	ospf.msg != 1
✓ ICMP errors	icmp.type eq 3 || icmp.type eq 4 || icmp.type eq 5 || icmp.type eq 11 || icmpv6.type eq 1 || icmpv6.type eq 2 || icmpv6.type eq 3 || icmpv6.type eq 4
✓ ARP	arp
✓ ICMP	icmp || icmpv6
✓ TCP RST	tcp.flags.reset eq 1
✓ SCTP ABORT	sctp.chunk_type eq ABORT
✓ TTL low or unexpected	(! ip.dst == 224.0.0.0/4 && ip.ttl < 5 && !pim && !ospf) || (ip.dst == 224.0.0.0/24 && ip.dst != 224.0.0.251 && ip.ttl != 1 && !(vrrp || carp))
✓ Checksum Errors	eth.fcs.status=="Bad" || ip.checksum.status=="Bad" || tcp.checksum.status=="Bad" || udp.checksum.status=="Bad" || sctp.checksum.status=="Bad" || mstp.ch
✓ SMB	smb || nbss || nbns || netbios
✓ HTTP	http || tcp.port == 80 || http2
✓ DCERPC	dcerpc
✓ Routing	hsrp || eigrp || ospf || bgp || cdp || vrrp || carp || gvrp || igmp || ismp
✓ TCP SYN/FIN	tcp.flags & 0x02 || tcp.flags.fin == 1
✓ TCP	tcp
✓ UDP	udp
✓ Broadcast	eth[0] & 1
✓ System Event	systemd_journal || sysdig

Optimizing Wireshark Searches with Filter Strategies

1. **Protocol Filters:**
    - tcp: Filters for TCP packets.
    - udp: Filters for UDP packets.
    - icmp: Filters for ICMP packets.
2. **Address Filters:**
    - ip.addr == 192.168.1.1: Filters packets with a specific IP address.
    - ip.src == 10.0.0.1: Filters packets with a specific source IP address.
    - ip.dst == 172.16.0.1: Filters packets with a specific destination IP address.
3. **Port Filters:**

   - tcp.port == 80: Filters TCP packets on port 80 (HTTP).
   - udp.port == 53: Filters UDP packets on port 53 (DNS).
4. **Combining Filters:**
   - ip.addr == 192.168.1.1 && tcp.port == 80: Combines IP and port filters.
5. **Display Filters:**
   - http: Filters for HTTP packets.
   - dns: Filters for DNS packets.
   - ssl: Filters for SSL/TLS encrypted traffic.
6. **Time Filters:**
   - frame.time_relative > 5: Filters packets occurring more than 5 seconds after the start.
7. **Conversation Filters:**
   - ip.addr == 192.168.1.1 && ip.addr == 192.168.1.2: Filters packets between two specific IP addresses.
8. **Logical Operators:**
   - == (equal), != (not equal), > (greater than), < (less than), && (logical AND), || (logical OR).
9. **Display Specific Fields:**
   - http.request.method: Displays only HTTP request methods.
   - dns.qry.name: Displays only DNS query names.