

**Name: Prathmesh Pattewar**

**Roll no:71**

**Prn : 12420193**

**Assignment No:6**

**Title:** Understanding protocol stack of Intranet.

**Problem Statement:** Analyze packet formats of Ethernet, IP, TCP and UDP captured through Wireshark for wired networks.

**Course Objective:** To learn transport layer and application layer protocols used in the Internet.

**Course Outcome:** Develop Client-Servers by the means of correct standards, protocols and technologies.

**Tools Required:** Wireshark.

**Theory:**

**1. Introduction**

Wireshark is a network packet analyzer. A network packet analyzer presents captured packet data in as much detail as possible. You could think of a network packet analyzer as a measuring device for examining what's happening inside a network cable, just like an electrician uses a voltmeter for examining what's happening inside an electric cable (but at a higher level, of course). In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, that has changed. Wireshark is available for free, is open source, and is one of the best packet analyzers available today.

**1.1 Features**

The following are some of the many features Wireshark provides:

- Available for *UNIX* and *Windows*.
- *Capture* live packet data from a network interface.
- *Open* files containing packet data captured with tcpdump/WinDump, Wireshark, and many other packet capture programs.
- *Import* packets from text files containing hex dumps of packet data.
- Display packets with *very detailed protocol information*.
- *Save* packet data captured.
- *Export* some or all packets in a number of capture file formats.
- *Filter packets* on many criteria.
- *Search* for packets on many criteria.
- *Colorize* packet display based on filters.
- Create various *statistics*.

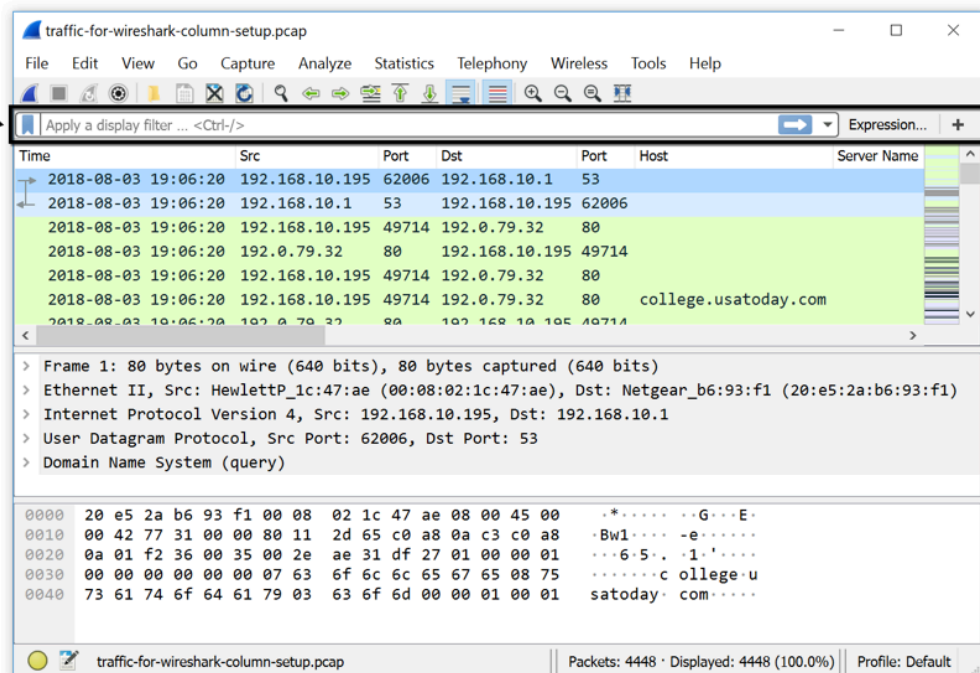
**2. Wireshark Filter**

**2.1 Display Filter**

Wireshark provides a display filter language that enables you to precisely control which packets are displayed. They can be used to check for the presence of a protocol or field, the value of a field, or even compare two fields to each other. These comparisons can be

combined with logical operators, like "and" and "or", and parentheses into complex expressions.

Display  
filter →



Wireshark's display filter uses Boolean expressions, so you can specify values and chain them together. The following expressions are commonly used:

- Equals: `==` or *eq*
- And: `&&` or *and*
- Or: `||` (double pipe) or *or*

### 2.1.1 Example

1. `eth.src == 00:11:22:33:44:55`: Source MAC address is 00:11:22:33:44:55
2. `ip.addr == 10.0.0.1`: Find all traffic that has IP of 10.0.0.1
3. `tcp.dstport != 80`: Destination tcp port is NOT 80
4. `ip.addr==10.10.10.1`
5. `ip.addr==192.168.1.10 && ip.addr==192.168.1.20`
6. `!(ip.addr==192.168.1.10 && ip.addr==192.168.1.20)`
7. `(ip.addr==192.168.1.10 && ip.addr==192.168.1.20) && (tcp.port==445 || tcp.port==139)`
8. `ip.src==10.10.10.0/24`
9. `eth.addr==00:1b:17:00:01:31`
10. `ip.addr==10.10.10.1 && tcp.port==80`
11. `tcp.port==80`
12. `tcp.port==80 || tcp.port==3389`
13. `tcp.dstport==80`
14. `eth.dst==ff:ff:ff:ff:ff:ff`
15. `ip.addr==255.255.255.255`  
`ip.host contains "imap"`

## 2.2 Capture Filters

Capture filters are used to decrease the size of captures by filtering out packets before they are added. **Capture filters** enable you to capture only traffic that you want to be captured,

eliminating an unwanted stream of packets. Capturing packets is a processor-intensive task, and packet analyzers use a good amount of primary memory while they are running. Packets are only sent to the capture engine if they meet a certain criterion (capture filter expressions). Capture filters do not facilitate advanced filtering options, as in display filters.

### 2.2.1 Examples

Capture only traffic to or from IP address 172.18.5.4:

- host 172.18.5.4

Capture traffic to or from a range of IP addresses:

- net 192.168.0.0/24

or

- net 192.168.0.0 mask 255.255.255.0

Capture traffic from a range of IP addresses:

- src net 192.168.0.0/24

or

- src net 192.168.0.0 mask 255.255.255.0

Capture traffic to a range of IP addresses:

- dst net 192.168.0.0/24

or

- dst net 192.168.0.0 mask 255.255.255.0

Capture only DNS (port 53) traffic:

- port 53

Capture non-HTTP and non-SMTP traffic on your server (both are equivalent):

- host www.example.com and not (port 80 or port 25)

host www.example.com and not port 80 and not port 25

Capture except all ARP and DNS traffic:

- port not 53 and not arp

Capture traffic within a range of ports

- (tcp[0:2] > 1500 and tcp[0:2] < 1550) or (tcp[2:2] > 1500 and tcp[2:2] < 1550)

or, with newer versions of libpcap (0.9.1 and later):

- tcp portrange 1501-1549

Capture only Ethernet type EAPOL:

- ether proto 0x888e

Reject ethernet frames towards the Link Layer Discovery Protocol Multicast group:

- not ether dst 01:80:c2:00:00:0e

Capture only IPv4 traffic - the shortest filter, but sometimes very useful to get rid of lower layer protocols like ARP and STP:

- ip

Capture only unicast traffic - useful to get rid of noise on the network if you only want to see traffic to and from your machine, not, for example, broadcast and multicast announcements:

- not broadcast and not multicast

Capture IPv6 "all nodes" (router and neighbor advertisement) traffic. Can be used to find rogue RAs:

- dst host ff02::1

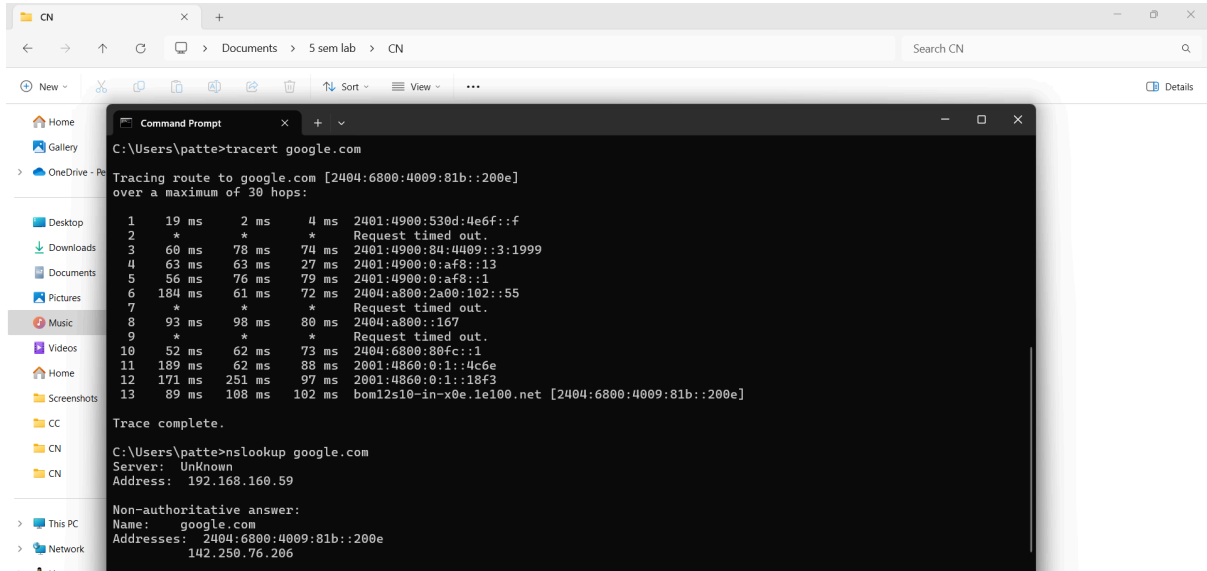
## Example:

### 1. Identify google.com IP Address

sudo apt-get install traceroute

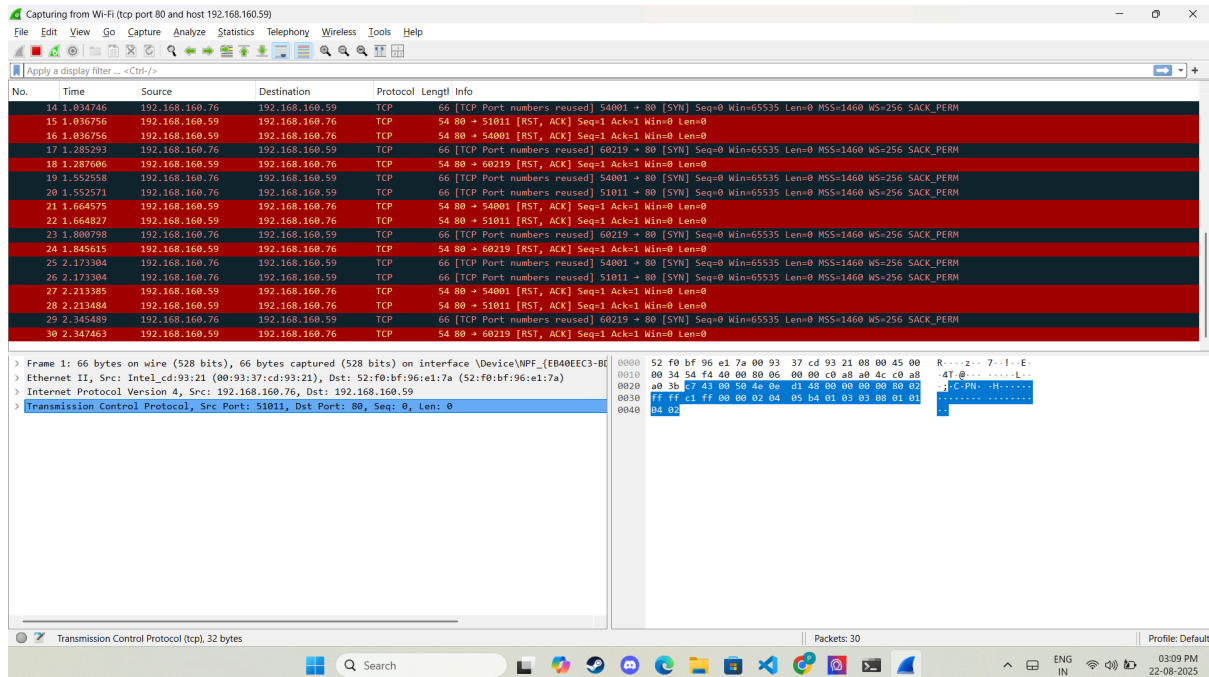
traceroute google.com

traceroute to google.com (192.168.160.59), 30 hops max, 60 byte packets



### 2. Capture all TCP traffic to/from google, during the time when you log in to your Google account.192.168.160.59

Capture filter: tcp and host



### 3. Capture all HTTP traffic to/from google, when you log in to your Facebook account

Capture filter: tcp port 80 and host 192.168.160.59

The screenshot shows Wireshark capturing traffic on Wi-Fi with the filter `tcp.port == 80 and ip.addr == 192.168.160.59`. The packet list shows 18 packets, including SYN, RST, and ACK packets. The packet details pane shows the selected packet (No. 18) as a Transmission Control Protocol, Src Port: 80, Dst Port: 51011, Seq: 1, Ack: 1, Len: 0. The packet bytes pane shows the raw data of the packet.

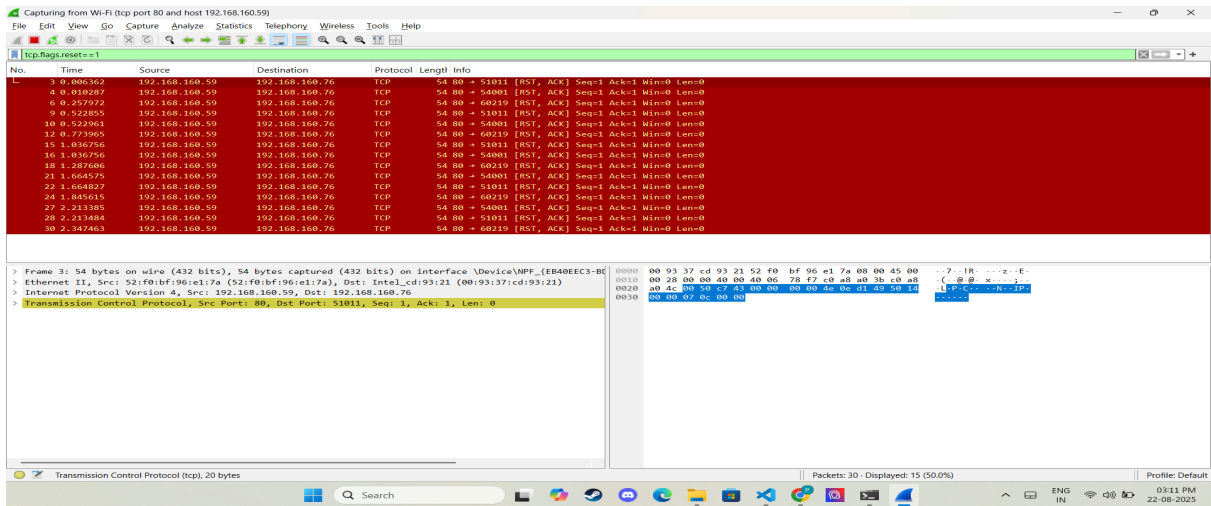
### 4. Write a DISPLAY filter expression to count all TCP packets (captured under item #1) that have the flags SYN, PSH, and RST set. Show the fraction of packets that had each flag set.

DISPLAY filter:

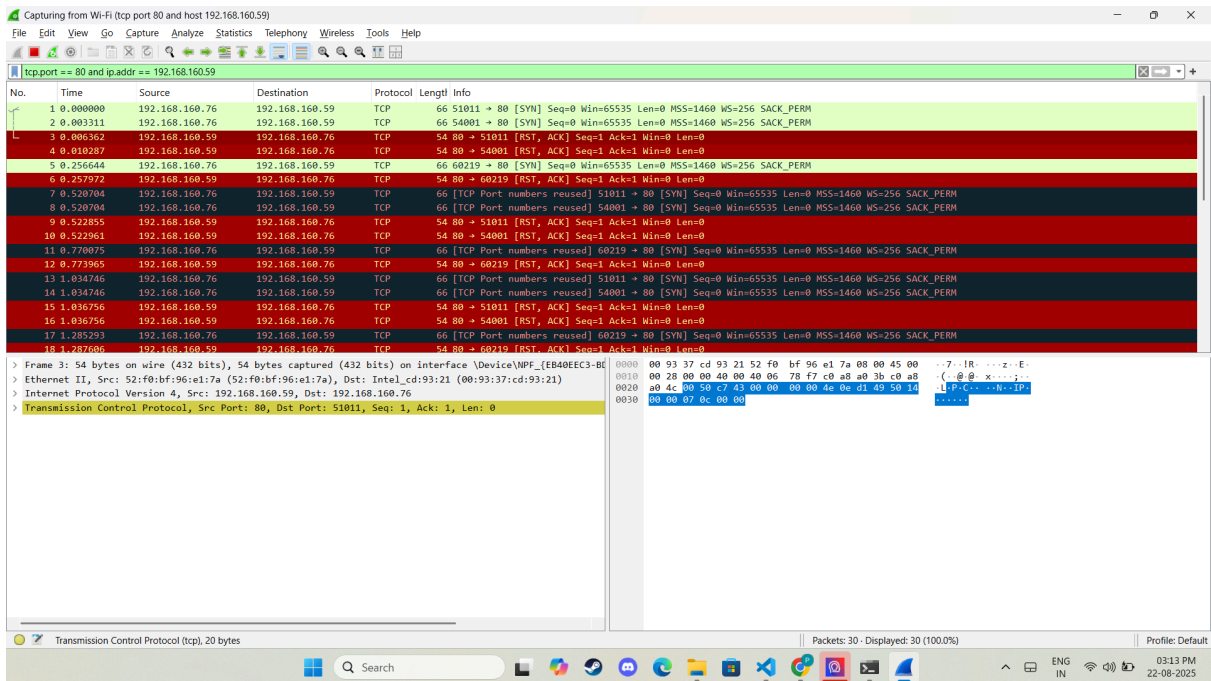
`tcp.flags.syn==1 && ip.dst==192.168.160.59`

`tcp.flags.reset==192.168.160.59`

The screenshot shows Wireshark capturing traffic on Wi-Fi with the filter `tcp.flags.syn==1 && ip.dst==192.168.160.59`. The packet list shows 29 packets, all of which are SYN packets. The packet details pane shows the selected packet (No. 29) as a Transmission Control Protocol, Src Port: 51011, Dst Port: 80, Seq: 0, Len: 0. The packet bytes pane shows the raw data of the packet.



## 5. Count how many TCP packets you received from/ sent to google, and how many of each were also HTTP packets.



## Conclusion:

Hence, we have Successfully Capture packets using Wireshark and wrote the exact packet capture filter expressions.