SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

**Department of Computer Engineering**

**Class: S.Y. B.Tech.** **Semester: IV**

**Course Code: DJ19CEL405** **Course Name: Computer Networks Lab**

Name: Dhruvin Nitesh Chawda SAP ID: 60004210159

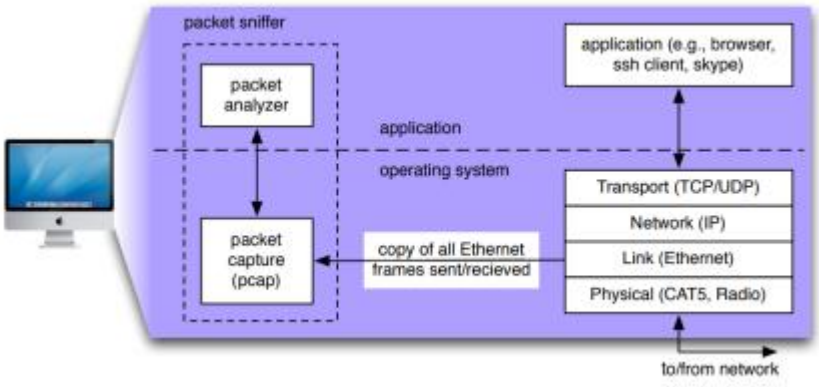
Date of Performance: 27/04/2023 Date of Submission:27/04/2023

# Experiment No: 9

## Aim: To study different networking devices and topologies.

## Theory:

- Packet sniffers are a basic tool for observing the messages on a network. As the name suggests, a packet sniffer captures ("sniffs") messages being sent/received from/by your 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.




- The figure above shows the structure of a packet sniffer. At the right are the protocols (in this case, Internet protocols) and applications (such as a web browser or ftp client) that normally run on your computer. The packet sniffer, shown within the dashed rectangle, 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. As you know, messages exchanged by higher layer protocols such as HTTP, FTP, TCP, UDP, DNS, or IP all are eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable. In the figure, 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 your computer.
- The existence of the packet capture box in this figure should give you cause to pause and think, particularly down two trains of thought. Firstly, it shows that any packet in a shared medium (Ethernet, Wi-Fi, etc) can be captured and examined without notification of the sender or receiver. You cannot rely on common link-layer protocols to protect your secrets or your privacy online. At a minimum, you should be using encryption

protocols (generally buried in the application layer, though sometimes found elsewhere) to protect all network traffic you generate or receive. Secondly, you have the ability to act as the "bad guy" and capture the network traffic of other people, examine it and exploit what you find.

- 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. 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".
- We will be using the Wireshark packet sniffer, allowing us to display the contents of messages being sent/received from/by protocols at different levels of the protocol stack. (Technically speaking, Wireshark is a packet analyzer that uses a packet capture library in your computer). Wireshark is a free network protocol analyzer that runs on Macintosh, Windows, and Linux/Unix computers. It's an ideal packet analyzer for our labs – it is stable, has a large user base and well-documented support that includes a userguide, man pages, and a detailed FAQ, rich functionality that includes the capability to analyze hundreds of protocols, and a well-designed user interface. It operates in computers using Ethernet, Token-Ring, FDDI, serial (PPP and SLIP), 802.11 wireless LANs, and ATM connections (if the OS on which it's running allows Wireshark to do so).

**Department of Computer Engineering**

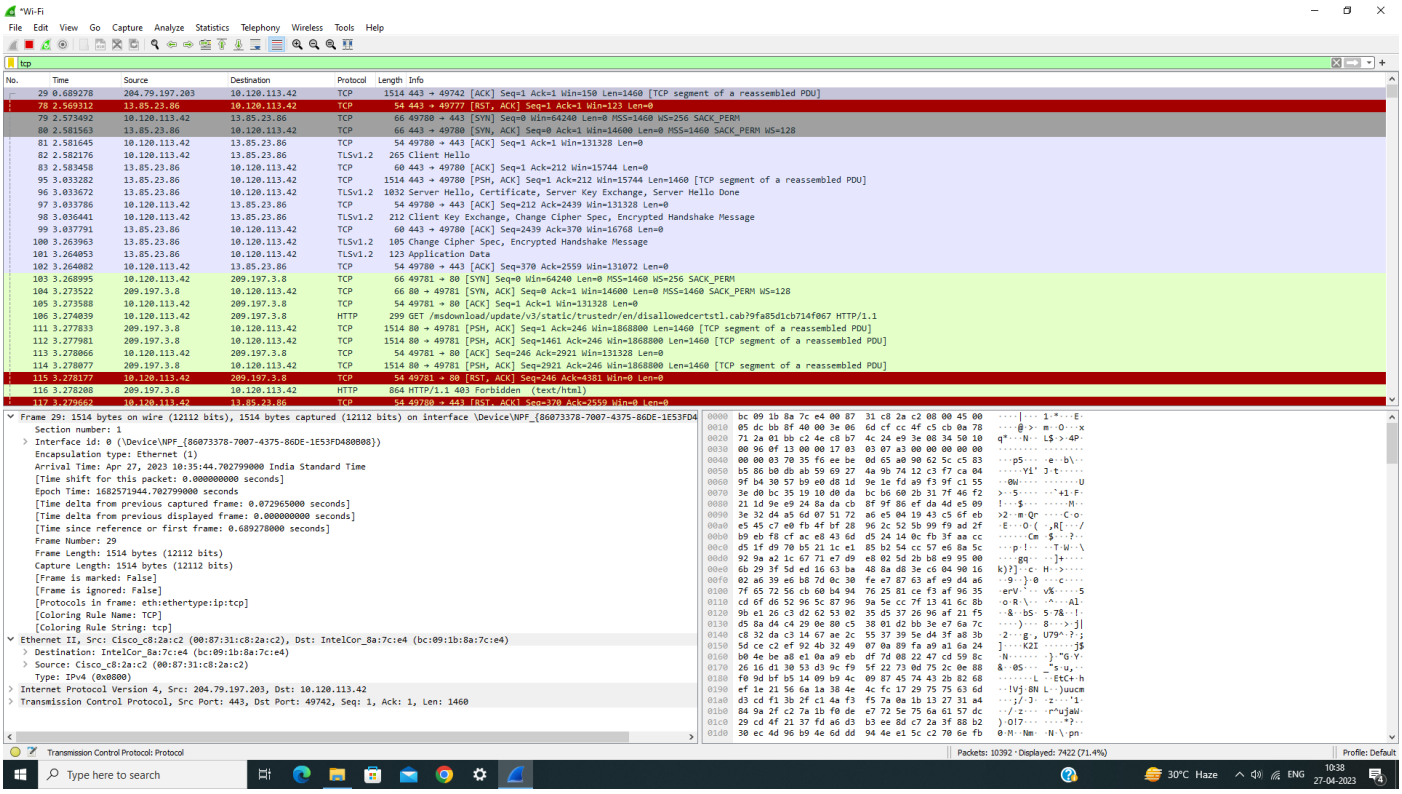
**Class: S.Y. B.Tech. Semester: IV**

**Course Code: DJ19CEL405 Course Name: Computer Networks Lab**

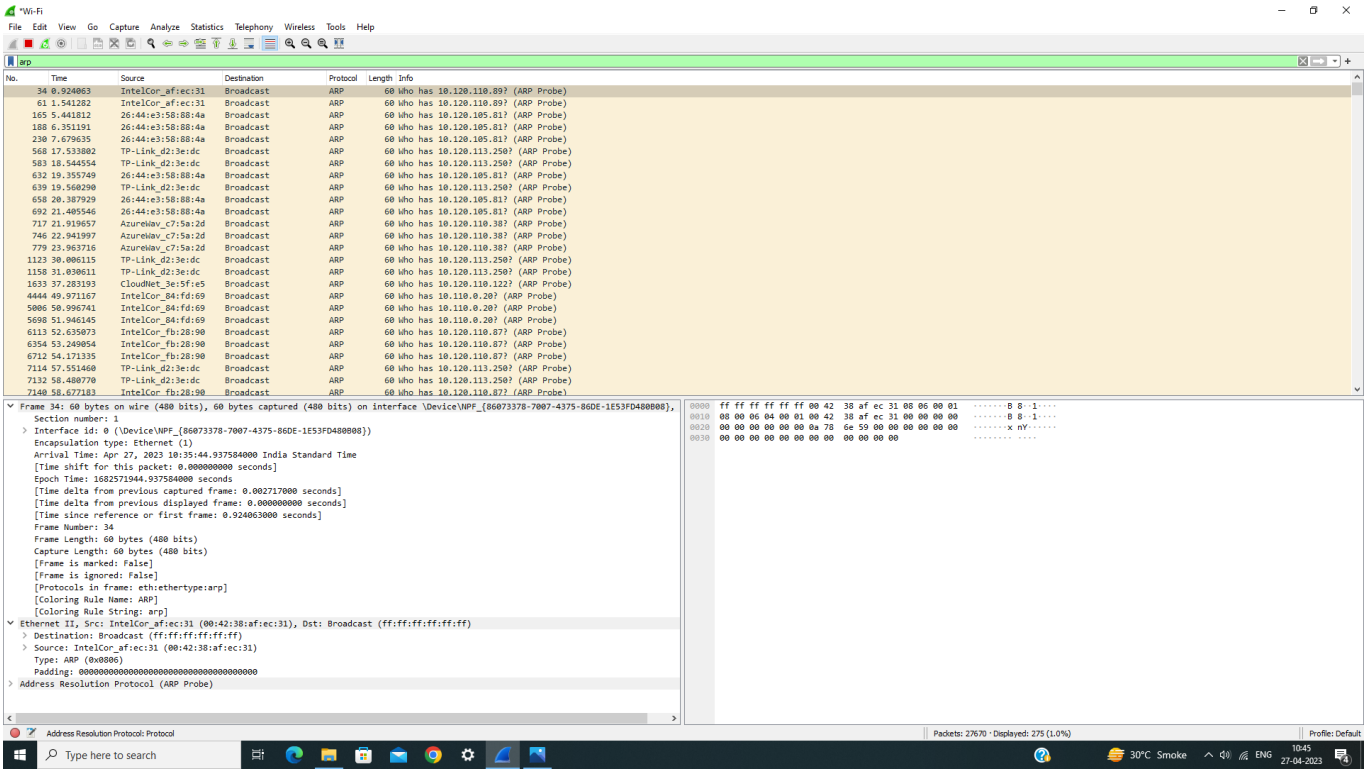
## Simulation :

## Filters in Wireshark :

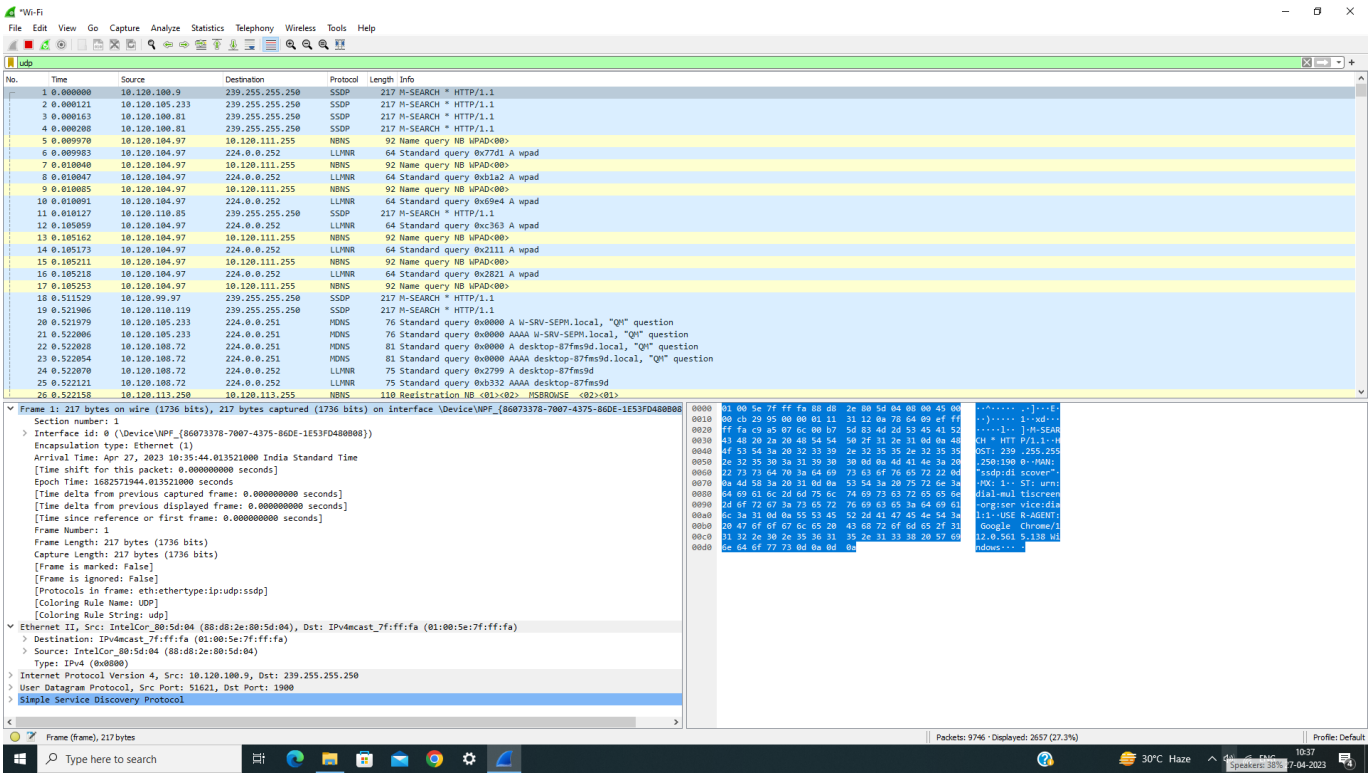
### 1) Tcp :



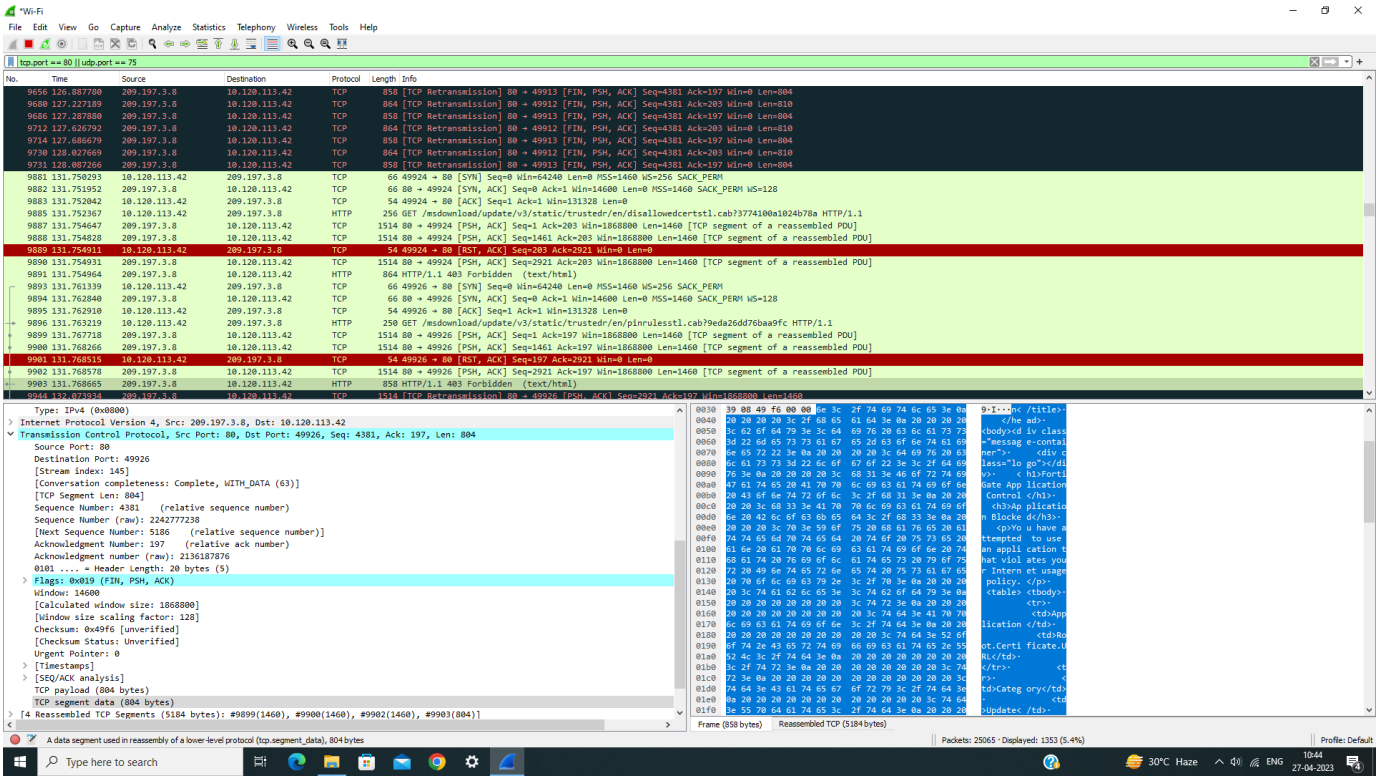
## 2) Arp


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
arp
No. Time Source Destination Protocol Length Info
34 0.924063 IntelCor_af:ec:31 Broadcast ARP 60 Who has 10.120.110.89? (ARP Probe)
61 1.541282 IntelCor_af:ec:31 Broadcast ARP 60 Who has 10.120.110.89? (ARP Probe)
165 5.441812 26:44:e3:58:88:4a Broadcast ARP 60 Who has 10.120.105.81? (ARP Probe)
188 6.351191 26:44:e3:58:88:4a Broadcast ARP 60 Who has 10.120.105.81? (ARP Probe)
230 7.679635 26:44:e3:58:88:4a Broadcast ARP 60 Who has 10.120.105.81? (ARP Probe)
568 17.533802 TP-Link_d2:3e:dc Broadcast ARP 60 Who has 10.120.113.250? (ARP Probe)
583 18.544554 TP-Link_d2:3e:dc Broadcast ARP 60 Who has 10.120.113.250? (ARP Probe)
632 19.355749 26:44:e3:58:88:4a Broadcast ARP 60 Who has 10.120.105.81? (ARP Probe)
639 19.560290 TP-Link_d2:3e:dc Broadcast ARP 60 Who has 10.120.113.250? (ARP Probe)
658 20.387929 26:44:e3:58:88:4a Broadcast ARP 60 Who has 10.120.105.81? (ARP Probe)
692 21.405546 26:44:e3:58:88:4a Broadcast ARP 60 Who has 10.120.105.81? (ARP Probe)
717 21.919657 AzureWav_c7:5a:2d Broadcast ARP 60 Who has 10.120.110.38? (ARP Probe)
746 22.941997 AzureWav_c7:5a:2d Broadcast ARP 60 Who has 10.120.110.38? (ARP Probe)
779 23.963716 AzureWav_c7:5a:2d Broadcast ARP 60 Who has 10.120.110.38? (ARP Probe)
1123 30.006115 TP-Link_d2:3e:dc Broadcast ARP 60 Who has 10.120.113.250? (ARP Probe)
1158 31.030611 TP-Link_d2:3e:dc Broadcast ARP 60 Who has 10.120.113.250? (ARP Probe)
1633 37.283193 CloudNet_3e:5f:e5 Broadcast ARP 60 Who has 10.120.110.122? (ARP Probe)
4444 49.971167 IntelCor_84:fd:69 Broadcast ARP 60 Who has 10.110.0.20? (ARP Probe)
5006 50.996741 IntelCor_84:fd:69 Broadcast ARP 60 Who has 10.110.0.20? (ARP Probe)
5698 51.946145 IntelCor_84:fd:69 Broadcast ARP 60 Who has 10.110.0.20? (ARP Probe)
6113 52.635073 IntelCor_fb:28:90 Broadcast ARP 60 Who has 10.120.110.87? (ARP Probe)
6354 53.249054 IntelCor_fb:28:90 Broadcast ARP 60 Who has 10.120.110.87? (ARP Probe)
6712 54.171335 IntelCor_fb:28:90 Broadcast ARP 60 Who has 10.120.110.87? (ARP Probe)
7114 57.551460 TP-Link_d2:3e:dc Broadcast ARP 60 Who has 10.120.113.250? (ARP Probe)
7132 58.480770 TP-Link_d2:3e:dc Broadcast ARP 60 Who has 10.120.113.250? (ARP Probe)
7140 58.677183 IntelCor_fb:28:90 Broadcast ARP 60 Who has 10.120.110.87? (ARP Probe)
Frame 34: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{86073378-7007-4375-86DE-1E53FD480B08}, id 0
Section number: 1
Interface id: 0 (\Device\NPF_{86073378-7007-4375-86DE-1E53FD480B08})
Encapsulation type: Ethernet (1)
Arrival Time: Apr 27, 2023 10:35:44.937584000 India Standard Time
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1682571944.937584000 seconds
[Time delta from previous captured frame: 0.002717000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.924063000 seconds]
Frame Number: 34
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: IntelCor_af:ec:31 (00:42:38:af:ec:31), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Destination: Broadcast (ff:ff:ff:ff:ff:ff)
Source: IntelCor_af:ec:31 (00:42:38:af:ec:31)
Type: ARP (0x0806)
Padding: 000000000000000000000000000000000000
Address Resolution Protocol (ARP Probe)
0000 ff ff ff ff ff ff 00 42 38 af ec 31 08 06 00 01
0010 08 00 06 04 00 01 00 42 38 af ec 31 00 00 00 00
0020 00 00 00 00 00 00 0a 78 6e 59 00 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00
Address Resolution Protocol: Protocol
Packets: 27670 · Displayed: 275 (1.0%)
Profile: Default
Type here to search
30°C Smoke
ENG
10:45
27-04-2023

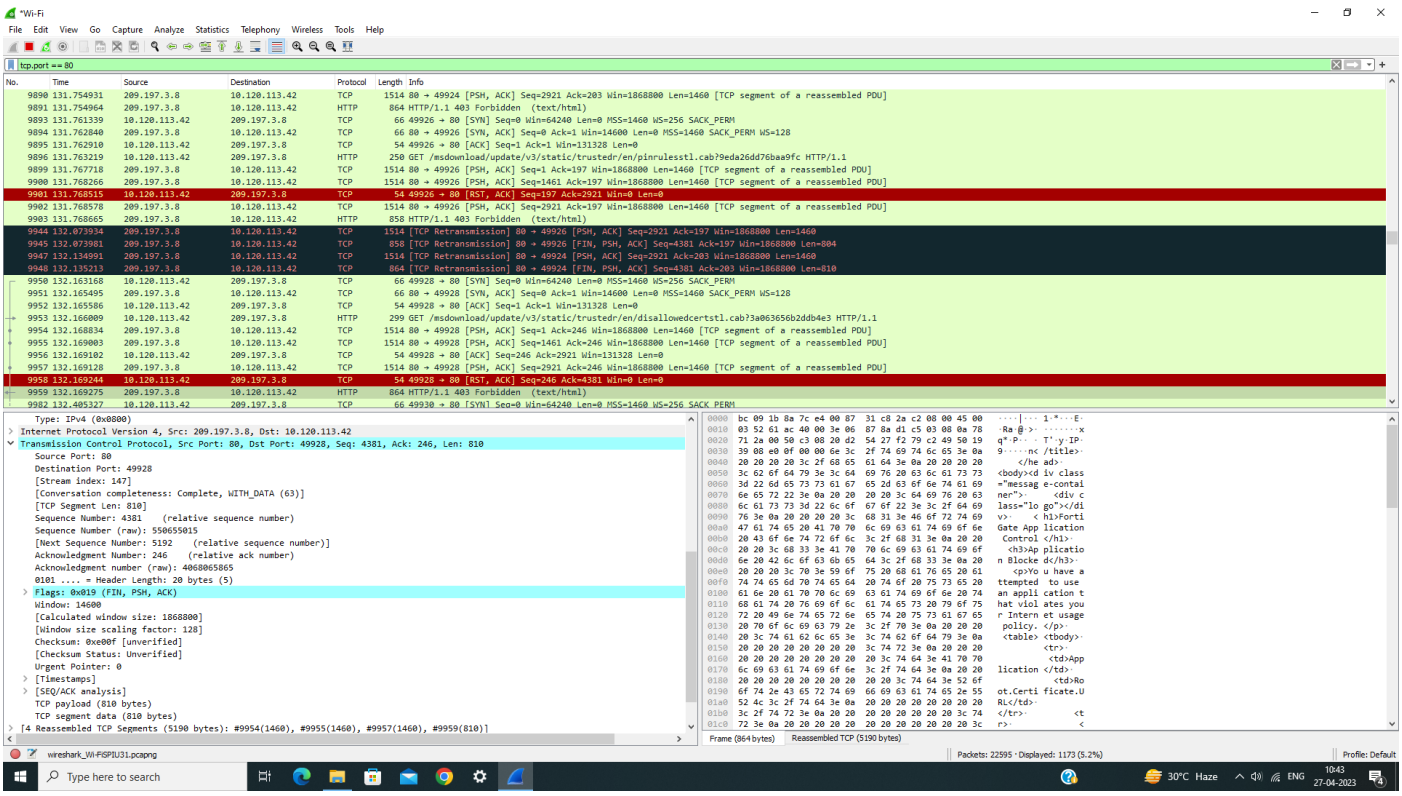
## 3) Udp


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
udp
No. Time Source Destination Protocol Length Info
1 0.000000 10.120.100.9 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1
2 0.000121 10.120.105.233 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1
3 0.000163 10.120.100.81 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1
4 0.000208 10.120.100.81 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1
5 0.009970 10.120.104.97 10.120.111.255 NBNS 92 Name query NB WPAD<00>
6 0.009983 10.120.104.97 224.0.0.252 LLMNR 64 Standard query 0x77d1 A wpad
7 0.010040 10.120.104.97 10.120.111.255 NBNS 92 Name query NB WPAD<00>
8 0.010047 10.120.104.97 224.0.0.252 LLMNR 64 Standard query 0xb1a2 A wpad
9 0.010085 10.120.104.97 10.120.111.255 NBNS 92 Name query NB WPAD<00>
10 0.010091 10.120.104.97 224.0.0.252 LLMNR 64 Standard query 0x69e4 A wpad
11 0.010127 10.120.110.85 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1
12 0.105059 10.120.104.97 224.0.0.252 LLMNR 64 Standard query 0xc363 A wpad
13 0.105162 10.120.104.97 10.120.111.255 NBNS 92 Name query NB WPAD<00>
14 0.105173 10.120.104.97 224.0.0.252 LLMNR 64 Standard query 0x2111 A wpad
15 0.105211 10.120.104.97 10.120.111.255 NBNS 92 Name query NB WPAD<00>
16 0.105218 10.120.104.97 224.0.0.252 LLMNR 64 Standard query 0x2821 A wpad
17 0.105253 10.120.104.97 10.120.111.255 NBNS 92 Name query NB WPAD<00>
18 0.511529 10.120.99.97 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1
19 0.521906 10.120.110.119 239.255.255.250 SSDP 217 M-SEARCH * HTTP/1.1
20 0.521979 10.120.105.233 224.0.0.251 MDNS 76 Standard query 0x0000 A W-SRV-SEPM.local, "QM" question
21 0.522006 10.120.105.233 224.0.0.251 MDNS 76 Standard query 0x0000 AAAA W-SRV-SEPM.local, "QM" question
22 0.522028 10.120.108.72 224.0.0.251 MDNS 81 Standard query 0x0000 A desktop-87fms9d.local, "QM" question
23 0.522054 10.120.108.72 224.0.0.251 MDNS 81 Standard query 0x0000 AAAA desktop-87fms9d.local, "QM" question
24 0.522070 10.120.108.72 224.0.0.252 LLMNR 75 Standard query 0x2799 A desktop-87fms9d
25 0.522121 10.120.108.72 224.0.0.252 LLMNR 75 Standard query 0xb332 AAAA desktop-87fms9d
26 0.522158 10.120.113.250 10.120.113.255 NBNS 110 Registration NB <01><02>__MSBROWSE__<02><01>
Frame 1: 217 bytes on wire (1736 bits), 217 bytes captured (1736 bits) on interface \Device\NPF_{86073378-7007-4375-86DE-1E53FD480B08}, id 0
Section number: 1
Interface id: 0 (\Device\NPF_{86073378-7007-4375-86DE-1E53FD480B08})
Encapsulation type: Ethernet (1)
Arrival Time: Apr 27, 2023 10:35:44.013521000 India Standard Time
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1682571944.013521000 seconds
[Time delta from previous captured frame: 0.000000000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1
Frame Length: 217 bytes (1736 bits)
Capture Length: 217 bytes (1736 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:ip:udp:ssdp]
[Coloring Rule Name: UDP]
[Coloring Rule String: udp]
Ethernet II, Src: IntelCor_80:5d:04 (88:d8:2e:80:5d:04), Dst: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
Destination: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
Source: IntelCor_80:5d:04 (88:d8:2e:80:5d:04)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 10.120.100.9, Dst: 239.255.255.250
User Datagram Protocol, Src Port: 51621, Dst Port: 1900
Simple Service Discovery Protocol
0000 01 00 5e 7f ff fa 88 d8 2e 80 5d 04 08 00 45 00
0010 00 cb 29 95 00 00 01 11 31 12 0a 78 64 09 ef ff
0020 ff fa c9 a5 07 6c 00 b7 5d 83 4d 2d 53 45 41 52
0030 43 48 20 2a 20 48 54 54 50 2f 31 2e 31 0d 0a 48
0040 4f 53 54 3a 20 32 33 39 2e 32 35 35 2e 32 35 35
0050 2e 32 35 30 3a 31 39 30 30 0d 0a 4d 41 4e 3a 20
0060 22 73 73 64 70 3a 64 69 73 63 6f 76 65 72 22 0d
0070 0a 4d 58 3a 20 31 0d 0a 53 54 3a 20 75 72 6e 3a
0080 64 69 61 6c 2d 6d 75 6c 74 69 73 63 72 65 65 6e
0090 2d 6f 72 67 3a 73 65 72 76 69 63 65 3a 64 69 61
00a0 6c 3a 31 0d 0a 55 53 45 52 2d 41 47 45 4e 54 3a
00b0 20 47 6f 6f 67 6c 65 20 43 68 72 6f 6d 65 2f 31
00c0 31 32 2e 30 2e 35 36 31 35 2e 31 33 38 20 57 69
00d0 6e 64 6f 77 73 0d 0a 0d 0a
Frame (frame), 217 bytes
Packets: 9746 · Displayed: 2657 (27.3%)
Profile: Default
Type here to search
30°C Haze
10:37
27-04-2023

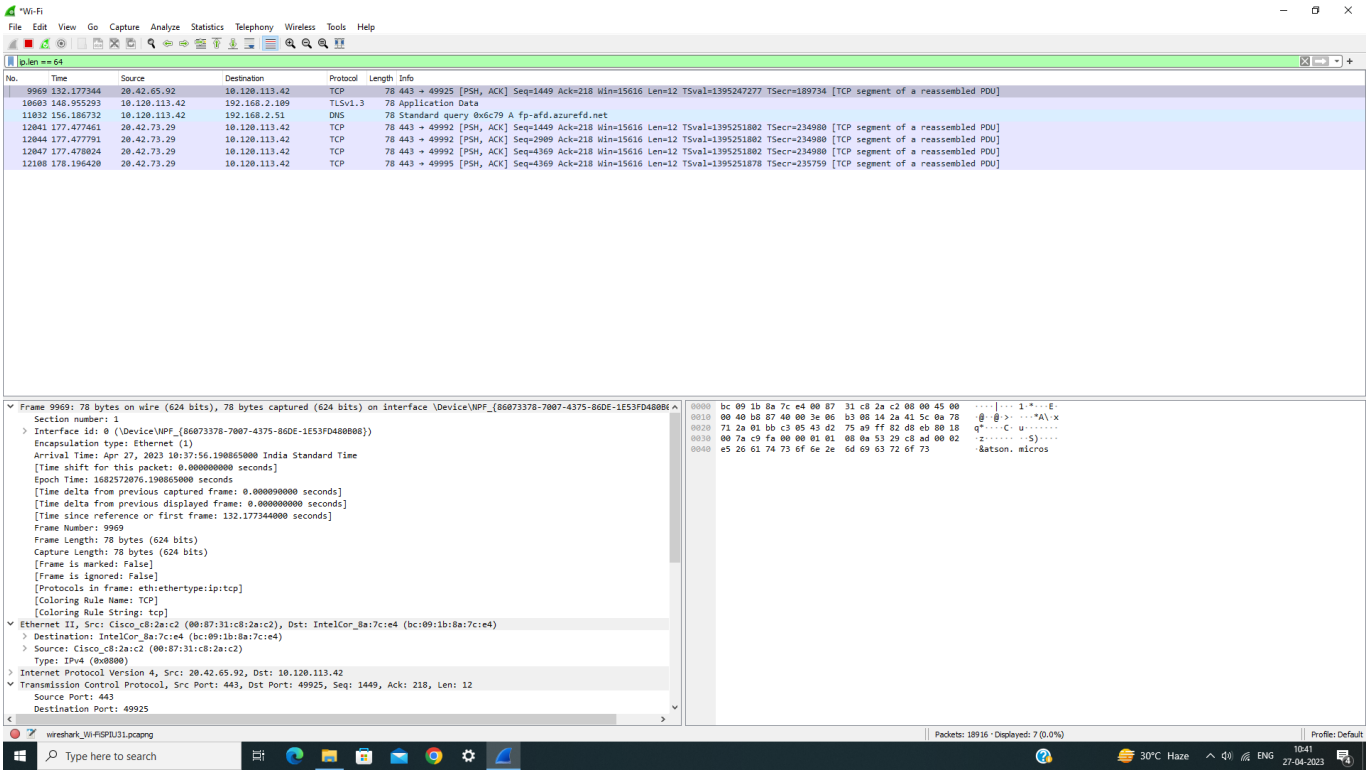
## 4) tcp.port == 80 || udp.port == 75



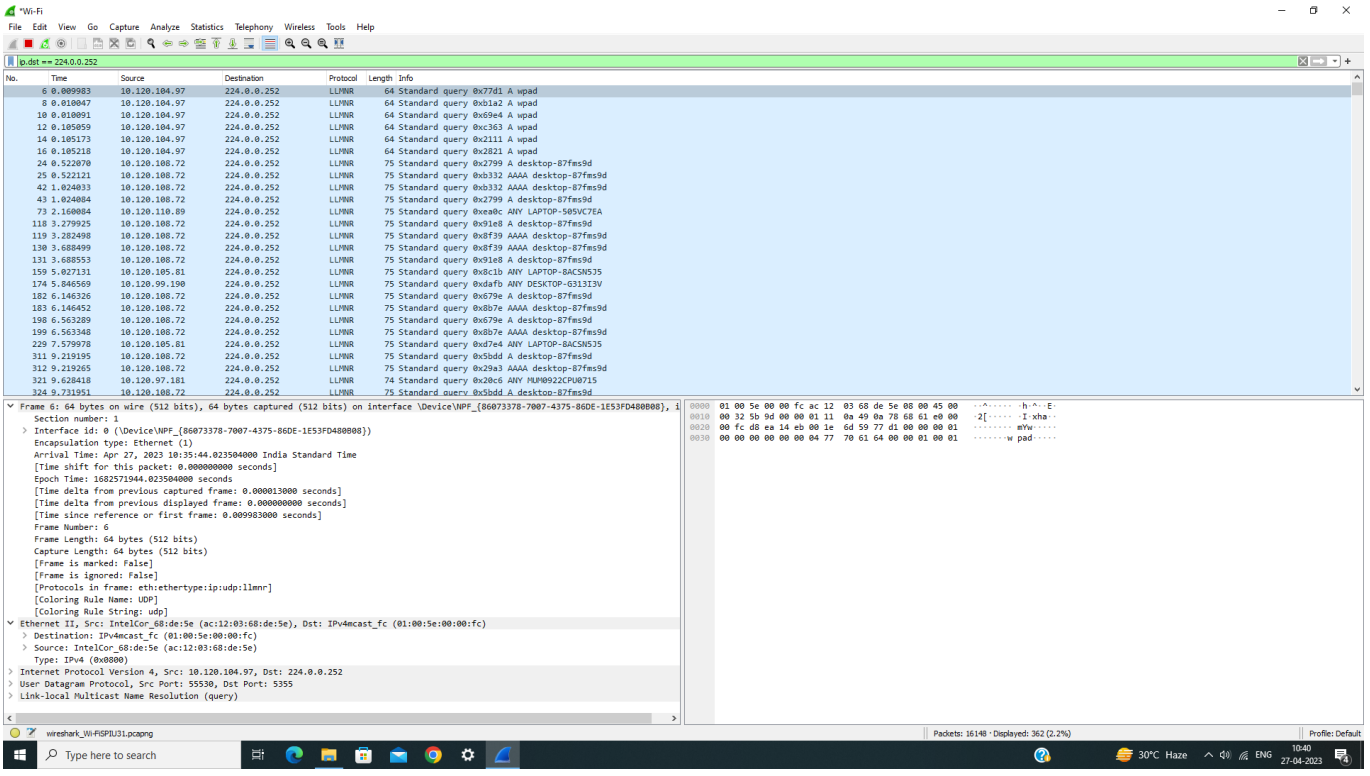

## 5) tcp.port == 80



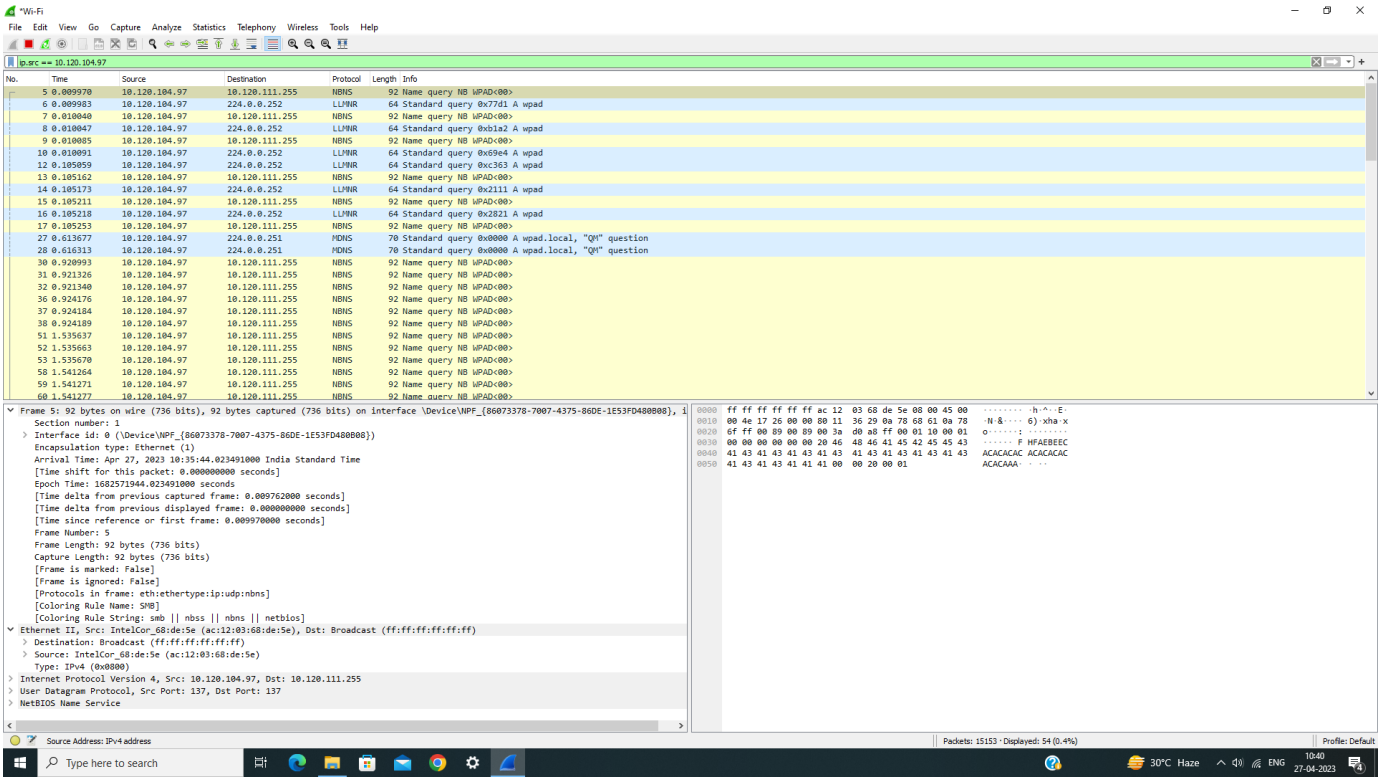
6) ip.len == 64



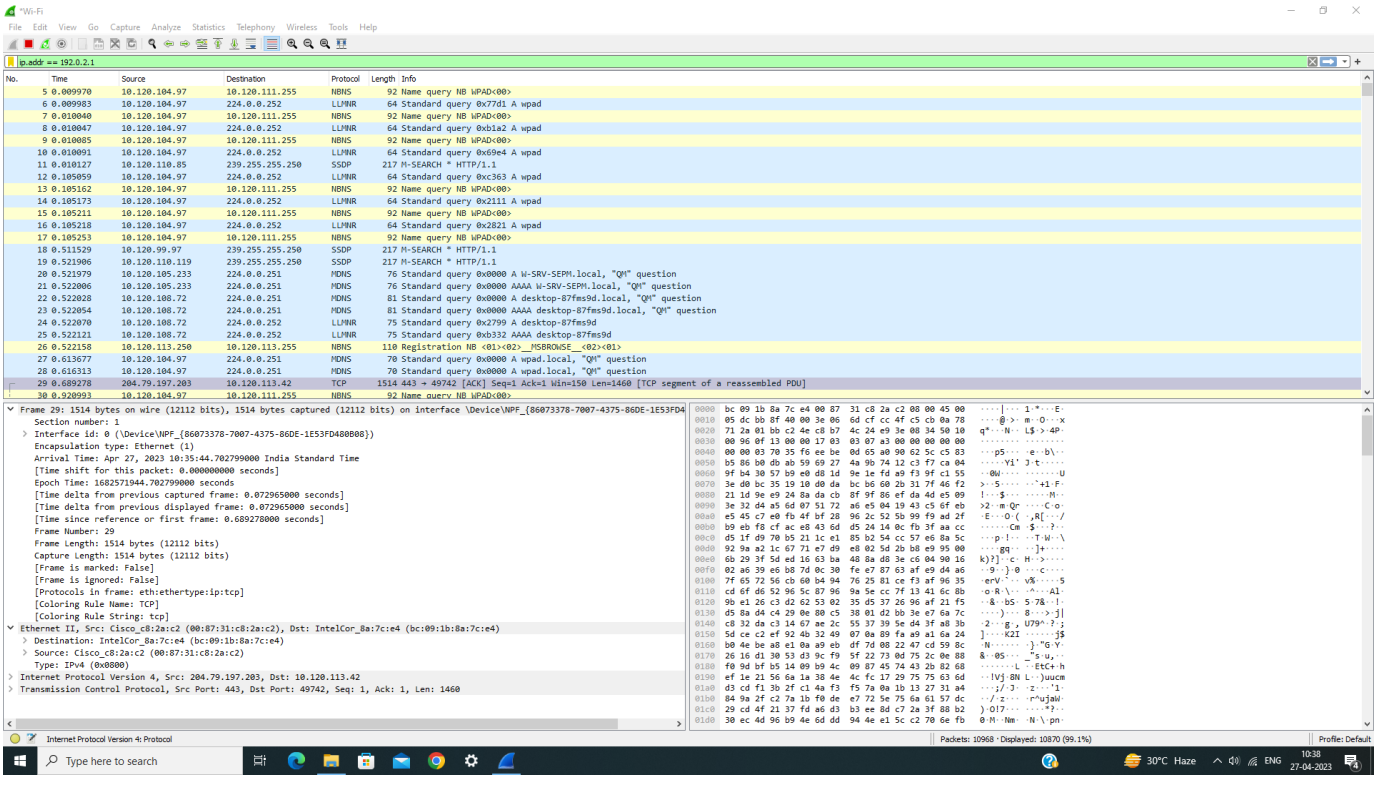

7) ip.dst == 224.0.0.252



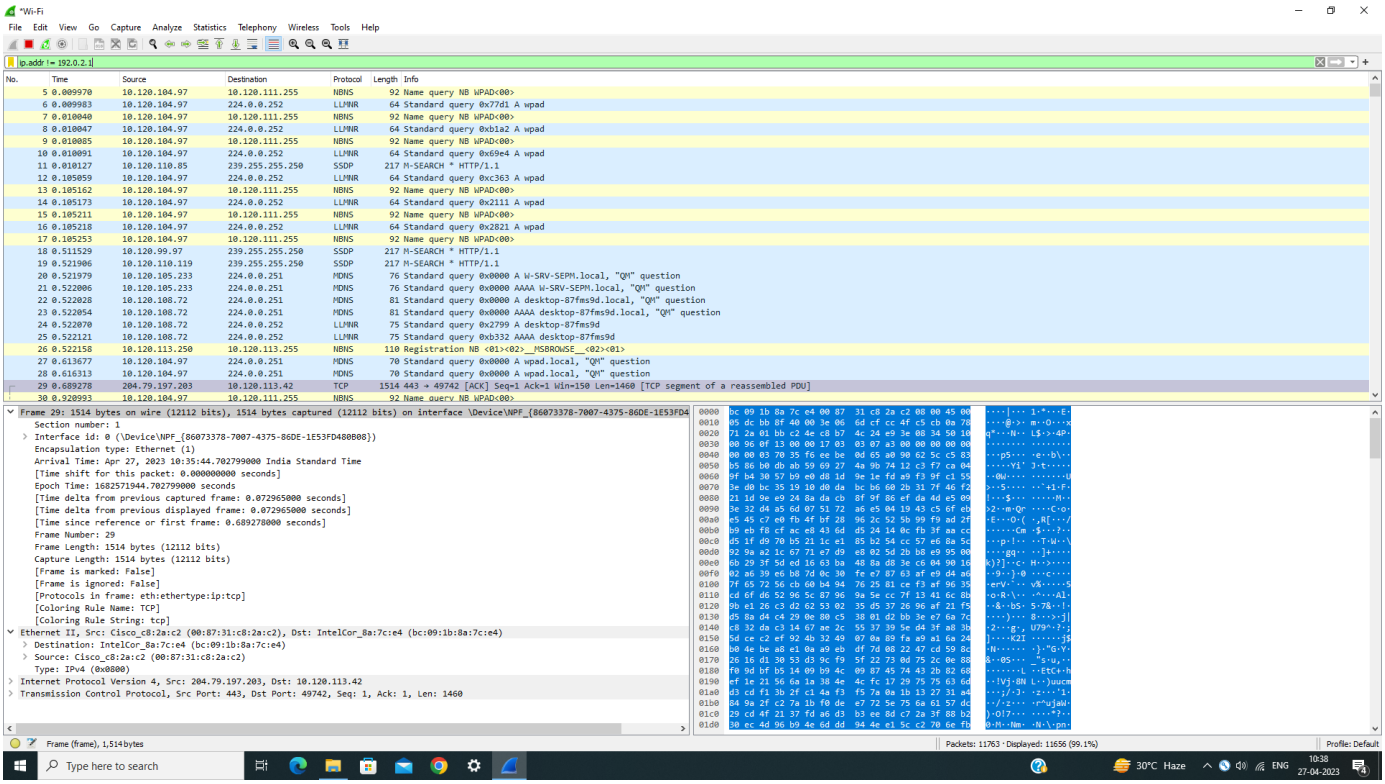
8) ip.src == 10.120.104.97



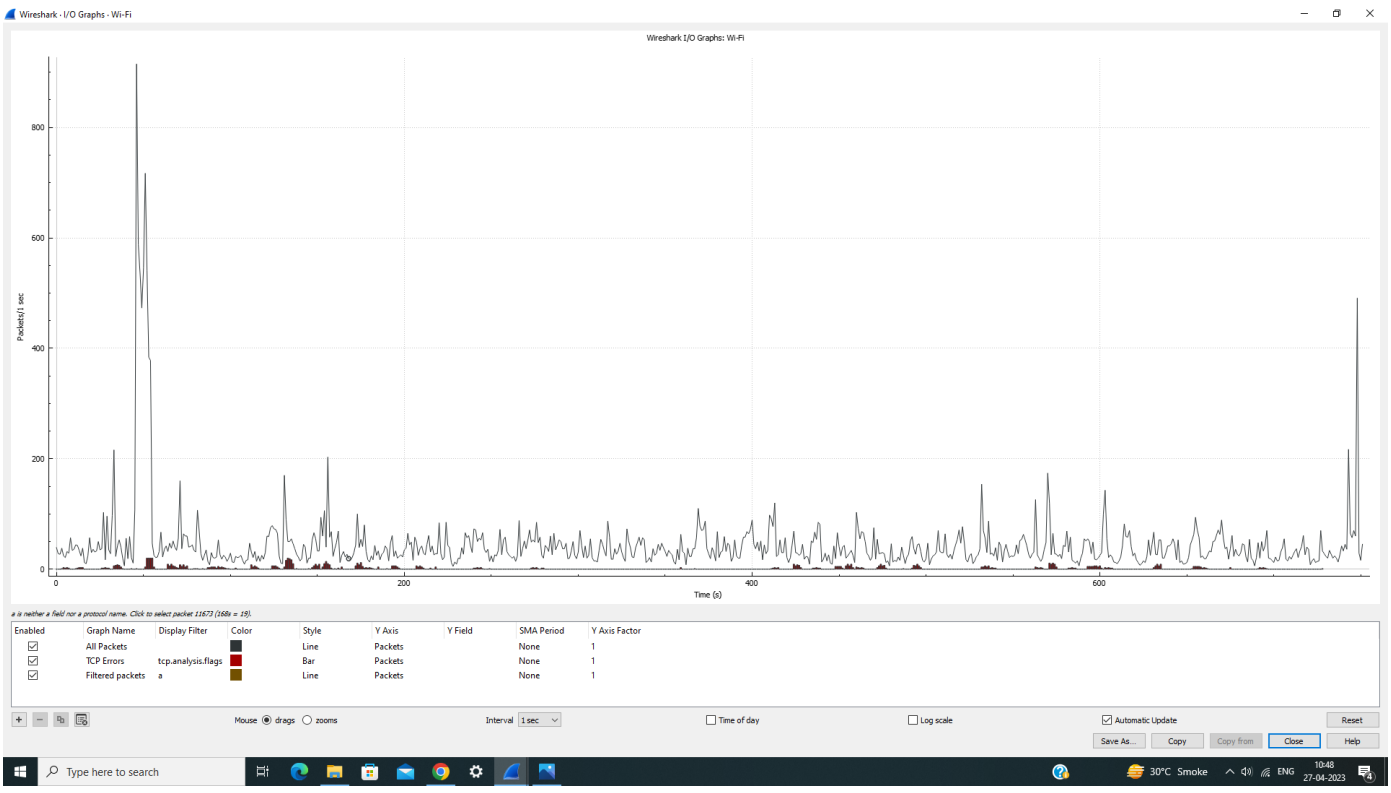

9) ip.addr == 192.0.2.1



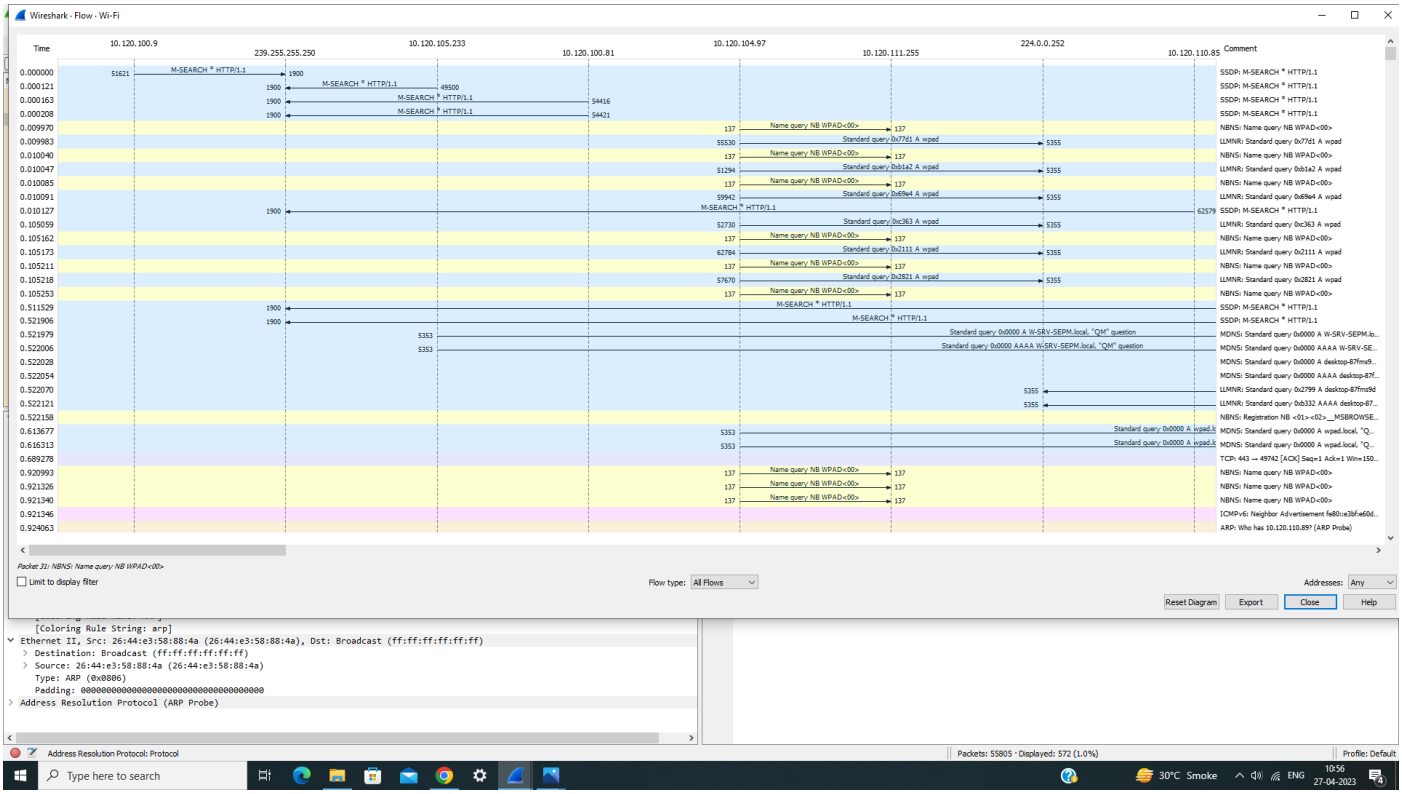
10) ip.addr != 192.0.2.1




# I/O graph :



## Flowchart:




## Conclusion:

Thus, we have simulated Packet Capturing in Wireshark