

Name : Aditya Shashikant Raul  
MIS : 142203016  
Branch : Computer Engineering  
Division : 2

=====

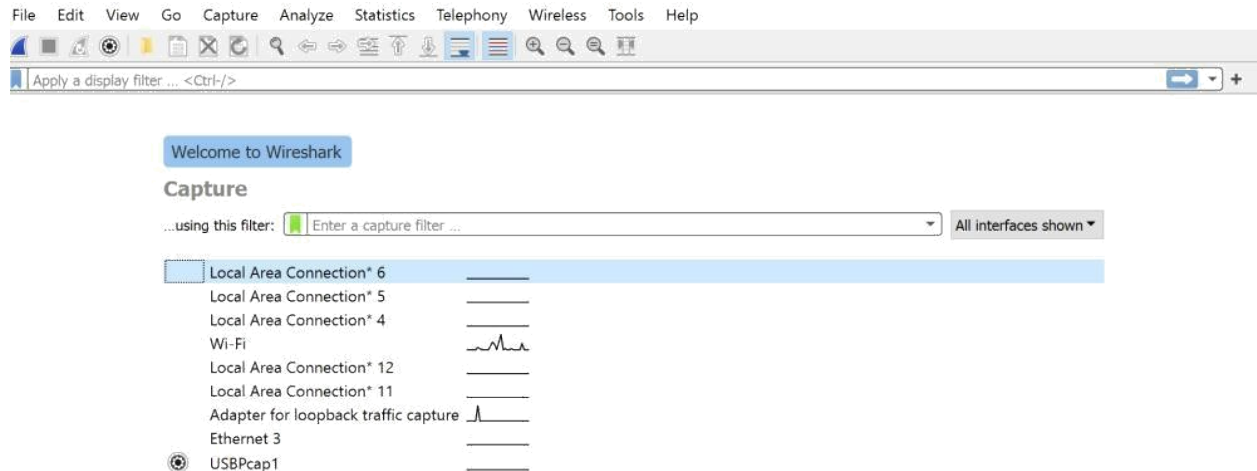
## Analyze TCP, UDP and IPV4 Header using Wireshark

### TCP Analysis using Wireshark:

TCP or Transmission Control Protocol is one of the most important protocols or standards for enabling communication possible amongst devices present over a particular network. It has algorithms that solve complex errors arising in packet communications, i.e. corrupted packets, invalid packets, duplicates, etc. Since it is used with IP(Internet Protocol), many times it is also referred to as TCP/IP. In order to start a communication, the TCP first establishes a connection using the three-way-handshake. TCP's efficiency over other protocols lies in its error detecting and correction attribute. Not only this, it organizes packets and segments larger data into a number of packets without disrupting the integrity of the data.

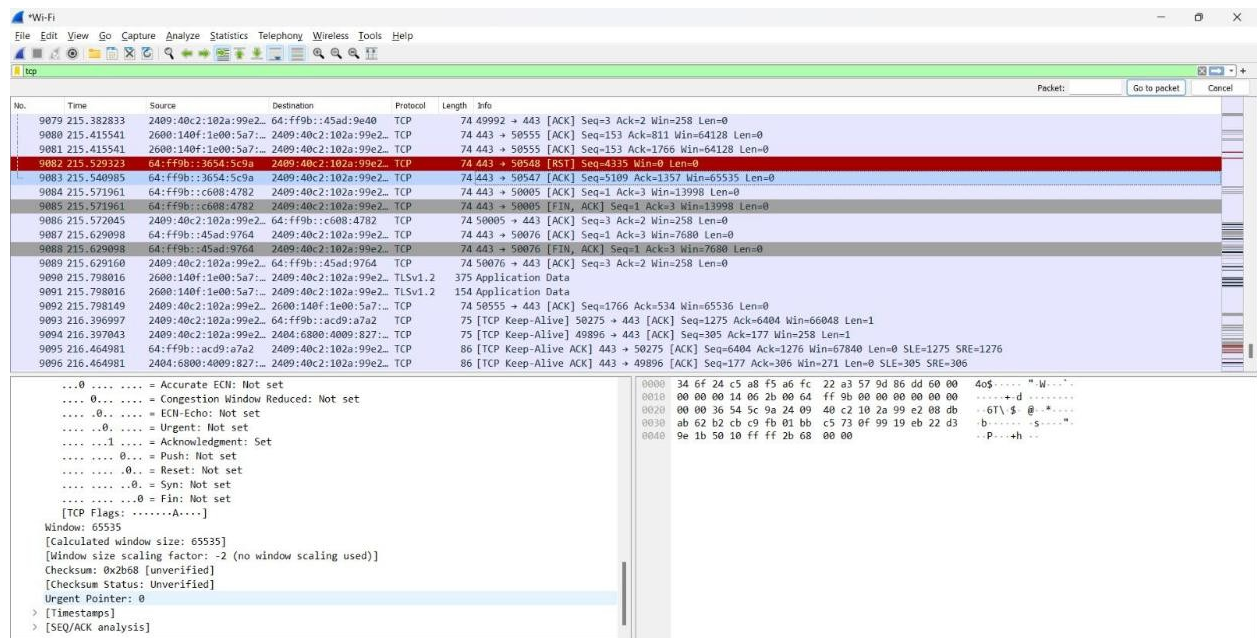
So now we are a bit familiar with TCP, let's look at how we can analyze TCP using Wireshark, which is the most widely used protocol analyzer in the world. In order to analyze TCP, you first need to launch Wireshark and follow the steps given below:

- From the menu bar, select capture -> options -> interfaces.
- In the interfaces, choose a particular Ethernet adapter and note down its IP, and click the start button of the selected adapter.
- Now we shall be capturing packets. Browse to a particular web address to generate traffic to capture packets from the communication for e.g. [geeksforgeeks.org](http://geeksforgeeks.org) and return to Wireshark and stop the capture by selecting stop from the capture menu. You can have a look at it in the image below.

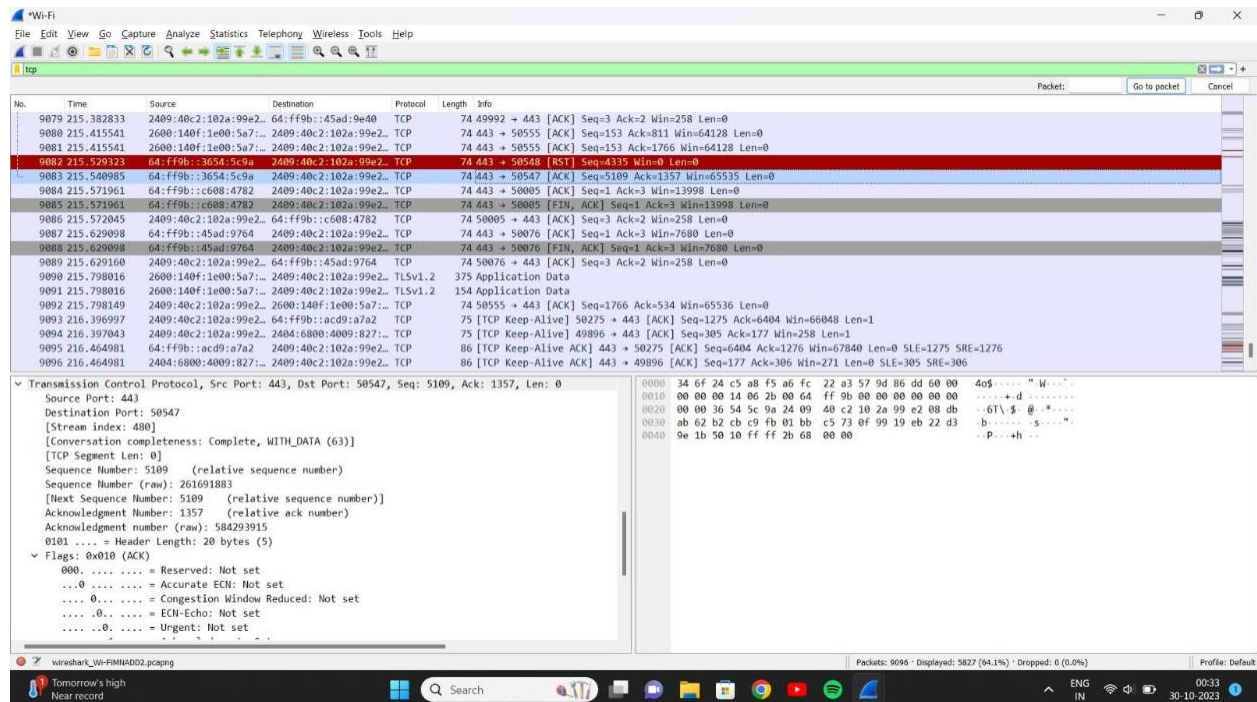


Now we have the captured packets and you will be having the captured packet list on the screen. Since we are concerned here with only TCP packets as we are doing TCP analysis, we shall be filtering out TCP packets from the packet pool. You can apply a filter in any of the following ways:

In the display filter bar on the screen, enter TCP and apply the filter.



From analyzing the menu in the menu bar select display filters or from capture select capture filters and then TCP only and ok.



Here you will have the list of TCP packets. The first three packets of this list are part of the three-way handshake mechanism of TCP to establish a connection. Let's get a basic knowledge of this mechanism which happens in the following 3 steps:

A synchronization packet (SYN) is sent by your local host IP to the server it desires to connect to.

The server reciprocates by sending an acknowledgment packet (ACK) to the local host signaling that it has received the SYN request of the host IP to connect and also sends a synchronization packet (SYN) to the local host to confirm the connection. So this one is basically an SYN+ACK packet.

The host answers this request by sending the ACK on receiving the SYN of the server.

Source port: This is the port of your host network used for communication.

Destination port: This is the port of the destination server.

TCP segment length: It represents the data length in the selected packet.

Sequence number: It is a method used by Wireshark to give particular indexing to each packet for tracking packets with ease. This indexing starts from 0.

Next sequence number: It is the sum of the sequence number and the segment length of the current packet.

Acknowledgment number: It contains the byte length of data received.

Header length: It is the length of the TCP header and can vary from 20 to 60.

We can observe three connection establishment steps in the first three packets of the TCP list where each of the packet types i.e. ACK, SYN, SYN-ACK are listed on their respective sides. Now to examine a packet closely we shall select a packet and in the expert view in the packet detail section just below the packet list we shall be having the TCP parameters as you can see in the below diagram.

The image shows a Wireshark packet capture analysis. The top pane displays a list of network packets. The selected packet (No. 9082) is a TCP segment from source 2409:40c2:102a:99e2::64 to destination 64:ff9b::45ad:9e40. The packet details pane shows the Transmission Control Protocol (TCP) parameters for this segment.

**Packet List:**

No.	Time	Source	Destination	Protocol	Length	Info
9079	215.382833	2409:40c2:102a:99e2::64	64:ff9b::45ad:9e40	TCP	74	49992 → 443 [ACK] Seq=3 Ack=2 Win=258 Len=0
9080	215.415541	2600:140f:1e00:5a7::2409:40c2:102a:99e2::	64:ff9b::45ad:9e40	TCP	74	443 → 50555 [ACK] Seq=153 Ack=811 Win=64128 Len=0
9081	215.415541	2600:140f:1e00:5a7::2409:40c2:102a:99e2::	64:ff9b::45ad:9e40	TCP	74	443 → 50555 [ACK] Seq=153 Ack=1766 Win=64128 Len=0
9082	215.52322	64:ff9b::3654:5c9a	2409:40c2:102a:99e2::	TCP	74	443 → 50548 [RST] Seq=4335 Win=0 Len=0
9083	215.540995	64:ff9b::3654:5c9a	2409:40c2:102a:99e2::	TCP	74	443 → 50547 [ACK] Seq=5109 Ack=1357 Win=65535 Len=0
9084	215.571961	64:ff9b::c608:4782	2409:40c2:102a:99e2::	TCP	74	443 → 50005 [ACK] Seq=1 Ack=3 Win=13998 Len=0
9085	215.571961	64:ff9b::c608:4782	2409:40c2:102a:99e2::	TCP	74	443 → 50005 [FIN, ACK] Seq=1 Ack=3 Win=13998 Len=0
9086	215.572045	2409:40c2:102a:99e2::	64:ff9b::c608:4782	TCP	74	50005 → 443 [ACK] Seq=3 Ack=2 Win=258 Len=0
9087	215.629098	64:ff9b::45ad:9764	2409:40c2:102a:99e2::	TCP	74	443 → 50076 [ACK] Seq=1 Ack=3 Win=7680 Len=0
9088	215.629098	64:ff9b::45ad:9764	2409:40c2:102a:99e2::	TCP	74	443 → 50076 [FIN, ACK] Seq=1 Ack=3 Win=7680 Len=0
9089	215.629160	2409:40c2:102a:99e2::	64:ff9b::45ad:9764	TCP	74	50076 → 443 [ACK] Seq=3 Ack=2 Win=258 Len=0
9090	215.798016	2600:140f:1e00:5a7::2409:40c2:102a:99e2::	64:ff9b::45ad:9764	TLSv1.2	375	Application Data
9091	215.798016	2600:140f:1e00:5a7::2409:40c2:102a:99e2::	64:ff9b::45ad:9764	TLSv1.2	354	Application Data
9092	215.798149	2409:40c2:102a:99e2::	2600:140f:1e00:5a7::	TCP	74	50555 → 443 [ACK] Seq=1766 Ack=534 Win=65536 Len=0
9093	216.396997	2409:40c2:102a:99e2::	64:ff9b::acd9:a7a2	TCP	75	[TCP Keep-Alive] 50275 → 443 [ACK] Seq=1275 Ack=6404 Win=66048 Len=1
9094	216.397043	2409:40c2:102a:99e2::	2404:6800:4009:827::	TCP	75	[TCP Keep-Alive] 49896 → 443 [ACK] Seq=305 Ack=177 Win=258 Len=1
9095	216.464981	64:ff9b::acd9:a7a2	2409:40c2:102a:99e2::	TCP	86	[TCP Keep-Alive ACK] 443 → 50275 [ACK] Seq=6404 Ack=1276 Win=67840 Len=0 SLE=1275 SRE=1276
9096	216.464981	2404:6800:4009:827::	2409:40c2:102a:99e2::	TCP	86	[TCP Keep-Alive ACK] 443 → 49896 [ACK] Seq=177 Ack=305 Win=271 Len=0 SLE=305 SRE=306

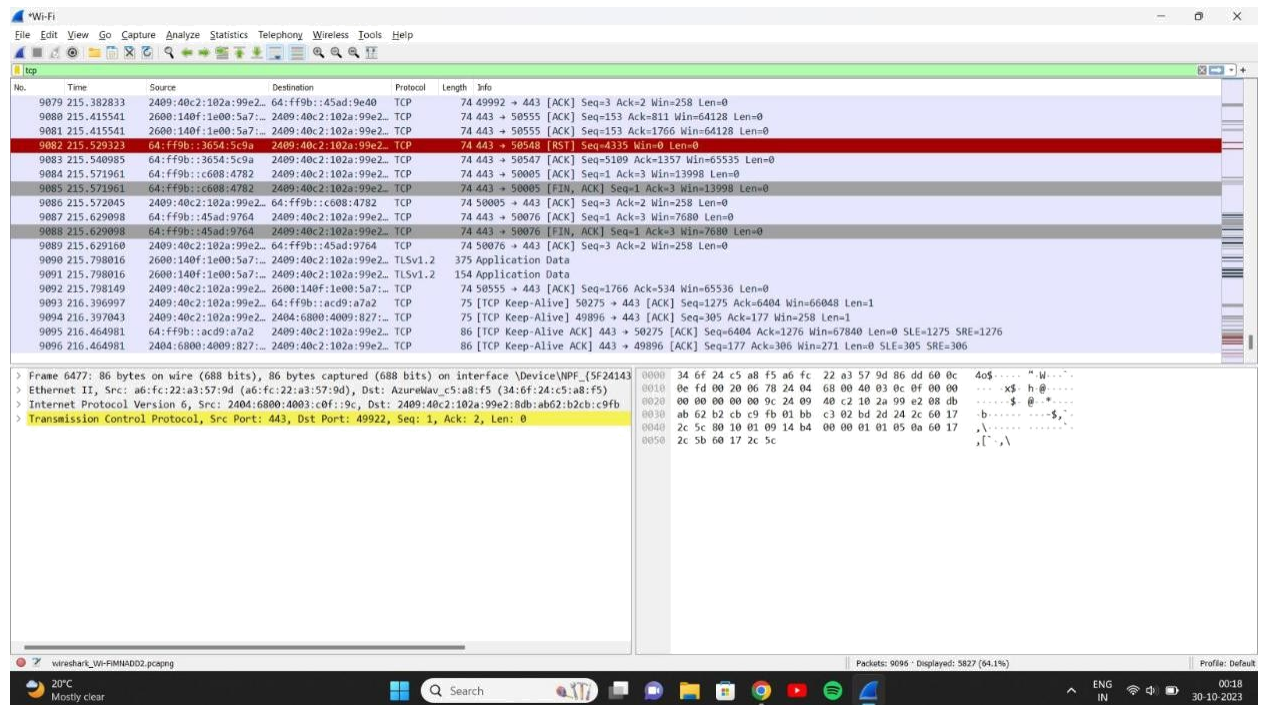
**Packet Details (Selected Packet 9082):**

- Transmission Control Protocol, Src Port: 443, Dst Port: 50548, Seq: 5109, Ack: 1357, Len: 0
- Destination Port: 50548
- [Stream index: 480]
- [Conversation completeness: Complete, WITH\_DATA (63)]
- [TCP Segment Len: 0]
- Sequence Number: 5109 (relative sequence number)
- Sequence Number (raw): 261091883
- [Next Sequence Number: 5109 (relative sequence number)]
- Acknowledgment Number: 1357 (relative ack number)
- Acknowledgment number (raw): 584293915
- 0101 .... = Header Length: 20 bytes (5)
- Flags: 0x010 (ACK)
- 000. .... = Reserved: Not set
- ...0 .... = Accurate ECN: Not set
- ....0 .... = Congestion Window Reduced: Not set
- ...0... = ECH-Echo: Not set
- ....0. .... = Urgent: Not set

**Packet Bytes:**

```
0000 34 6f 24 c5 a8 f5 a6 fc 22 a3 57 9d 86 dd 60 00 40$ - - - - - "W...":
0010 00 00 00 14 06 2b 00 64 ff 9b 00 00 00 00 00 00 - - - - - + d - - - - -
0020 00 00 36 54 5c 9a 24 09 40 c2 10 2a 99 e2 08 db -6T\ $ - @ - - - - -
0030 ab 62 b2 cb c9 fb 01 bb c5 73 0f 99 19 eb 22 d3 -b- - - - - s- - - - -
0040 9e 1b 50 10 ff ff ff 2b 68 00 00 -P- - - - - h - - - - -
```



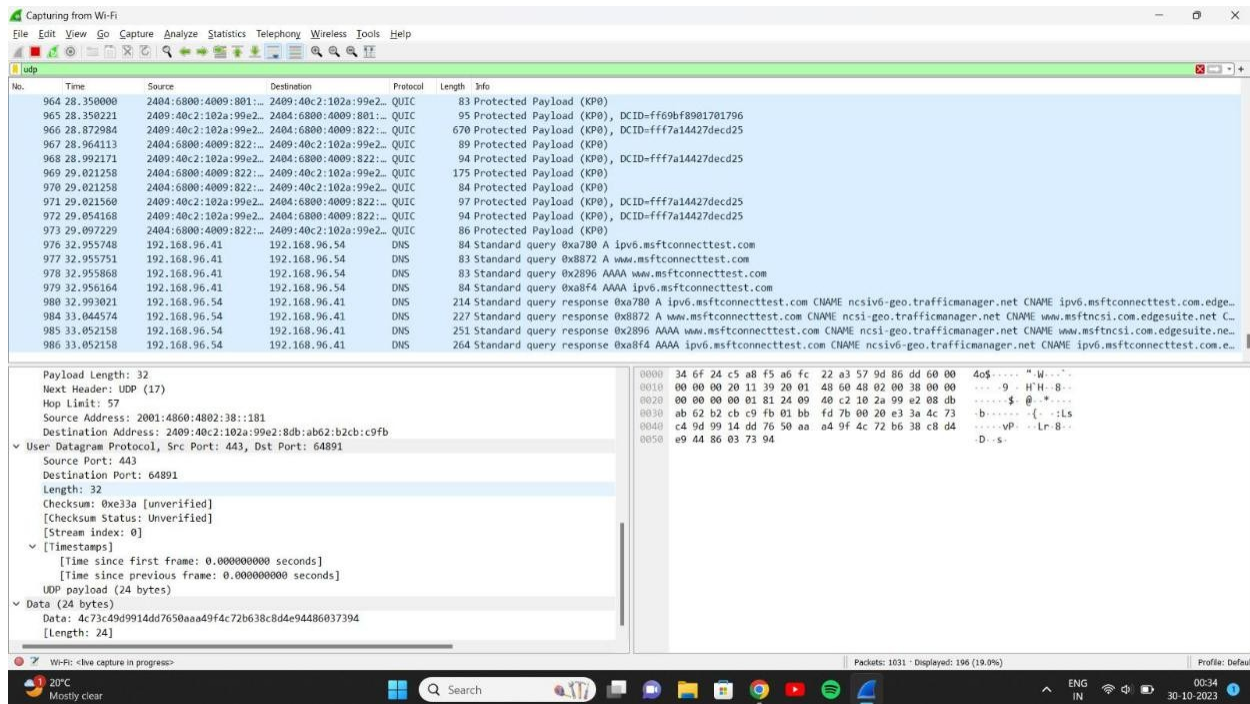


## UDP Analysis using Wireshark:

The User Datagram Protocol, or UDP, is a communication protocol used across the Internet for especially time-sensitive transmissions such as [video playback](#) or [DNS](#) lookups. It speeds up communications by not formally establishing a connection before data is transferred. This allows data to be transferred very quickly, but it can also cause [packets](#) to become lost in transit — and create opportunities for exploitation in the form of [DDoS attacks](#).

Like all [networking protocols](#), UDP is a standardized method for transferring data between two computers in a network. Compared to other protocols, UDP accomplishes this process in a simple fashion: it sends packets (units of data transmission) directly to a target computer, without establishing a connection first, indicating the order of said packets, or checking whether they arrived as intended. (UDP packets are referred to as ‘datagrams’.)

We have to follow the same steps for udp analysis .The only difference is we are filtering the packets based on udp protocol.



## UDP Header –

UDP header is an 8-bytes fixed and simple header, while for TCP it may vary from 20 bytes to 60 bytes. The first 8 Bytes contains all necessary header information and the remaining part consists of data. UDP port number fields are each 16 bits long, therefore the range for port numbers is defined from 0 to 65535; port number 0 is reserved. Port numbers help to distinguish different user requests or processes.

**Source Port:** Source Port is a 2 Byte long field used to identify the port number of the source.

**Destination Port:** It is a 2 Byte long field, used to identify the port of the destined packet.

**Length:** Length is the length of UDP including the header and the data. It is a 16-bits field.

**Checksum:** Checksum is 2 Bytes long field. It is the 16-bit one's complement of the one's complement sum of the UDP header, the pseudo-header of information

from the IP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

The screenshot shows the Wireshark interface with a packet capture of UDP traffic. The packet list on the left shows 13 packets, all of which are DNS queries from 192.168.96.41 to 192.168.96.54. The packet details pane on the right shows the structure of the first packet (No. 1), which is a DNS query. The packet is 86 bytes long and contains a standard query for the domain 'www.msftconnecttest.com'. The packet bytes pane on the right shows the raw data of the packet, including the Ethernet II header, IP header, and UDP header.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	2001:4860:4802:38::...	2409:40c2:102a:99e2...	UDP	86	443 → 64891 Len=24
2	2.951364	192.168.96.41	192.168.96.54	DNS	83	Standard query 0x6dcdb A www.msftconnecttest.com
3	2.951364	192.168.96.41	192.168.96.54	DNS	84	Standard query 0x4941 A ipv6.msftconnecttest.com
4	2.951544	192.168.96.41	192.168.96.54	DNS	83	Standard query 0xe182 AAAA ipv6.msftconnecttest.com
5	2.956413	192.168.96.41	192.168.96.54	DNS	83	Standard query 0x152d AAAA www.msftconnecttest.com
6	2.991525	192.168.96.54	192.168.96.41	DNS	214	Standard query response 0x4941 A ipv6.msftconnecttest.com CNAME ncsiv6-geo.trafficmanager.net CNAME ipv6.msftconnecttest.com.edgesui...
7	3.032543	192.168.96.54	192.168.96.41	DNS	227	Standard query response 0x6dcdb A www.msftconnecttest.com CNAME ncsi-geo.trafficmanager.net CNAME www.msftncsi.com.edgesuite.net CNAM...
8	3.032543	192.168.96.54	192.168.96.41	DNS	264	Standard query response 0xe182 AAAA ipv6.msftconnecttest.com CNAME ncsiv6-geo.trafficmanager.net CNAME ipv6.msftconnecttest.com.edge...
10	3.053263	192.168.96.41	192.168.96.54	DNS	83	Standard query 0x152d AAAA www.msftconnecttest.com
11	3.077645	192.168.96.54	192.168.96.41	DNS	251	Standard query response 0x152d AAAA www.msftconnecttest.com CNAME ncsi-geo.trafficmanager.net CNAME www.msftncsi.com.edgesuite.net C...
12	3.077645	192.168.96.54	192.168.96.41	DNS	257	Standard query response 0x152d AAAA www.msftconnecttest.com CNAME ncsi-geo.trafficmanager.net CNAME www.msftncsi.com.edgesuite.net C...

Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface \Device\NPF\_{5F241432-C1F0-4B33-943E-D01F3403407B}

Section number: 1

Interface id: 0 (\Device\NPF\_{5F241432-C1F0-4B33-943E-D01F3403407B})

Encapsulation type: Ethernet (1)

Arrival Time: Oct 30, 2023 00:33:54.758110000 India Standard Time

[Time shift for this packet: 0.000000000 seconds]

Epoch Time: 1698066234.758110000 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: 86 bytes (688 bits)

Capture Length: 86 bytes (688 bits)

[Frame is marked: False]

[Frame is ignored: False]

[Protocols in frame: eth:ethertype:ipv6:udp:data]

[Coloring Rule Name: UDP]

[Coloring Rule String: udp]

Ethernet II, Src: a6:fc:22:a3:57:9d (a6:fc:22:a3:57:9d), Dst: AzureNav\_c5:a8:f5 (34:6f:24:c5:a8:f5)

The screenshot shows the Wireshark interface with a packet capture of UDP traffic. The packet list on the left shows 13 packets, all of which are DNS queries from 192.168.96.41 to 192.168.96.54. The packet details pane on the right shows the structure of the first packet (No. 1), which is a DNS query. The packet bytes pane on the right shows the raw data of the packet, including the Ethernet II header, IP header, and UDP header.

No.	Time	Source	Destination	Protocol	Length	Info
964	28.350000	2404:6800:4009:801::...	2409:40c2:102a:99e2...	QUIC	83	Protected Payload (KP0)
965	28.350221	2409:40c2:102a:99e2...	2404:6800:4009:801::...	QUIC	95	Protected Payload (KP0), DCID=fff6b9f01701796
966	28.872984	2409:40c2:102a:99e2...	2404:6800:4009:822::...	QUIC	670	Protected Payload (KP0), DCID=fff7a14427dec25
967	28.964113	2404:6800:4009:822::...	2409:40c2:102a:99e2...	QUIC	89	Protected Payload (KP0)
968	28.992171	2409:40c2:102a:99e2...	2404:6800:4009:822::...	QUIC	94	Protected Payload (KP0), DCID=fff7a14427dec25
969	29.021258	2404:6800:4009:822::...	2409:40c2:102a:99e2...	QUIC	175	Protected Payload (KP0)
970	29.021258	2404:6800:4009:822::...	2409:40c2:102a:99e2...	QUIC	84	Protected Payload (KP0)
971	29.021560	2409:40c2:102a:99e2...	2404:6800:4009:822::...	QUIC	97	Protected Payload (KP0), DCID=fff7a14427dec25
972	29.054168	2409:40c2:102a:99e2...	2404:6800:4009:822::...	QUIC	94	Protected Payload (KP0), DCID=fff7a14427dec25
973	29.097229	2404:6800:4009:822::...	2409:40c2:102a:99e2...	QUIC	86	Protected Payload (KP0)
976	32.955748	192.168.96.41	192.168.96.54	DNS	84	Standard query 0xa780 A ipv6.msftconnecttest.com
977	32.955751	192.168.96.41	192.168.96.54	DNS	83	Standard query 0x8872 A www.msftconnecttest.com
978	32.955868	192.168.96.41	192.168.96.54	DNS	83	Standard query 0x2896 AAAA www.msftconnecttest.com
979	32.956164	192.168.96.41	192.168.96.54	DNS	84	Standard query 0xa8f4 AAAA ipv6.msftconnecttest.com
980	32.993021	192.168.96.54	192.168.96.41	DNS	214	Standard query response 0xa780 A ipv6.msftconnecttest.com CNAME ncsiv6-geo.trafficmanager.net CNAME ipv6.msftconnecttest.com.edgesui...
984	33.044574	192.168.96.54	192.168.96.41	DNS	227	Standard query response 0x8872 A www.msftconnecttest.com CNAME ncsi-geo.trafficmanager.net CNAME www.msftncsi.com.edgesuite.net C...
985	33.052158	192.168.96.54	192.168.96.41	DNS	251	Standard query response 0x2896 AAAA www.msftconnecttest.com CNAME ncsi-geo.trafficmanager.net CNAME www.msftncsi.com.edgesuite.net...
986	33.052158	192.168.96.54	192.168.96.41	DNS	264	Standard query response 0xa8f4 AAAA ipv6.msftconnecttest.com CNAME ncsiv6-geo.trafficmanager.net CNAME ipv6.msftconnecttest.com.e...

Payload Length: 32

Next Header: UDP (17)

Hop Limit: 57

Source Address: 2001:4860:4802:38:181

Destination Address: 2409:40c2:102a:99e2:8db:a62:b2bc:c9fb

User Datagram Protocol, Src Port: 443, Dst Port: 64891

Source Port: 443

Destination Port: 64891

Length: 32

Checksum: 0xe33a [unverified]

[Checksum Status: Unverified]

[Stream index: 0]

[Timestamps]

[Time since first frame: 0.000000000 seconds]

[Time since previous frame: 0.000000000 seconds]

UDP payload (24 bytes)

Data (24 bytes)

Data: 4c73c49d9914dd7650aaa49f4c72b638c8d4e94486037394

[Length: 24]



## **IPv4 Analysis using Wireshark:**

IP stands for Internet Protocol and v4 stands for Version Four (IPv4). IPv4 was the primary version brought into action for production within the ARPANET in 1983. IP version four addresses are 32-bit integers which will be expressed in decimal notation.

Example- 192.0.2.126 could be an IPv4 address.

### **Parts of IPv4**

#### **Network part:**

The network part indicates the distinctive variety that's appointed to the network. The network part conjointly identifies the category of the network that's assigned.

#### **Host Part:**

The host part uniquely identifies the machine on your network. This part of the IPv4 address is assigned to every host.

For each host on the network, the network part is the same, however, the host half must vary.

#### **Subnet number:**

This is the nonobligatory part of IPv4. Local networks that have massive numbers of hosts are divided into subnets and subnet numbers are appointed to that.

## **Characteristics of IPv4**

- IPv4 could be a 32-Bit IP Address.
- IPv4 could be a numeric address, and its bits are separated by a dot.
- The number of header fields is twelve and the length of the header field is twenty.
- It has Unicast, broadcast, and multicast style of addresses.
- IPv4 supports VLSM (Virtual Length Subnet Mask).
- IPv4 uses the Post Address Resolution Protocol to map to the MAC address.
- RIP may be a routing protocol supported by the routed daemon.
- Networks ought to be designed either manually or with DHCP.
- Packet fragmentation permits from routers and causing host.

Wi-Fi network traffic capture in Wireshark. The packet list shows various DNS and TCP segments. The selected packet (No. 1773) is a TCP segment (Seq=52619, Ack=30, Win=65536) from 192.168.96.54 to 192.168.96.41. The packet details pane shows the Internet Protocol Version 4 and Transmission Control Protocol fields. The packet bytes pane displays the raw data in hexadecimal and ASCII.

Wi-Fi network traffic capture in Wireshark. The packet list shows various DNS and TCP segments. The selected packet (No. 1773) is a TCP segment (Seq=52619, Ack=30, Win=65536) from 192.168.96.54 to 192.168.96.41. The packet details pane shows the Internet Protocol Version 4 and Transmission Control Protocol fields. The packet bytes pane displays the raw data in hexadecimal and ASCII.