Aim : Analysis of Network Traces using Wireshark tool.

Description : Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed 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).

Here are some reasons people use Wireshark:

- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- QA engineers use it to verify network applications
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals

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.

Filtering in wireshark :

In wireshark, filters can be used to display and capture only the packets desired by the user. Filtering can be done based on

- Protocol
- IP addresses of source and destination
- Port Number
- Sequence number

How to open a pcap file :

.pcap files are data files created using the program and they contain the packet data of a network. These files are mainly used in analyzing the network characteristics of a certain data. Launch a .pcap file, or any other file on your PC, by double-clicking it. If your file associations are set up correctly, the application that's meant to open your .pcap file will open it.

How to trace an interface using wireshark :

The following methods can be used to start capturing packets with Wireshark:

You can double-click on an interface in the main window.

You can get an overview of the available interfaces using the "Capture Interfaces" dialog box (Capture → Options…). See Figure 4.1, "The "Capture Interfaces" dialog box on Microsoft Windows" or Figure 4.2, "The "Capture Interfaces" dialog box on Unix/Linux" for more information. You can start a capture from this dialog box using the Start button.

You can immediately start a capture using your current settings by selecting Capture → Start or by clicking the first toolbar button.

If you already know the name of the capture interface you can start Wireshark from the command line:

$ wireshark -i eth0 -k

This will start Wireshark capturing on interface eth0.

Filters

Wireshark has two filtering languages: capture filters and display filters. Capture filters are used for filtering when capturing packets "Filtering while capturing". Display filters are used for filtering which packets are displayed.

Display filters allow you to concentrate on the packets you are interested in while hiding the currently uninteresting ones. They allow you to only display packets based on:

Protocol

The presence of a field

The values of fields

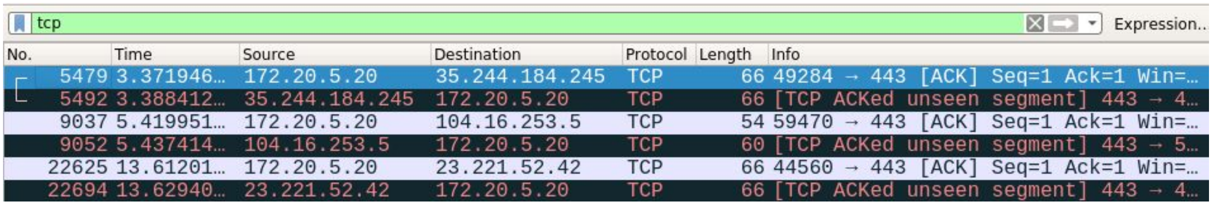
A comparison between fields

… and a lot more!

To only display packets containing a particular protocol, type the protocol name in the display filter toolbar of the Wireshark window and press enter to apply the filter.

Based on protocol

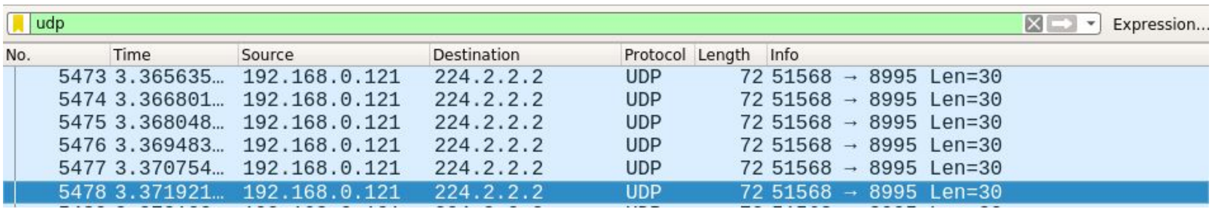
Tcp :



tcp

No.	Time	Source	Destination	Protocol	Length	Info
5479	3.371946…	172.20.5.20	35.244.184.245	TCP	66	49284 → 443 [ACK] Seq=1 Ack=1 Win=…
5492	3.388412…	35.244.184.245	172.20.5.20	TCP	66	[TCP ACKed unseen segment] 443 → 4…
9037	5.419951…	172.20.5.20	104.16.253.5	TCP	54	59470 → 443 [ACK] Seq=1 Ack=1 Win=…
9052	5.437414…	104.16.253.5	172.20.5.20	TCP	60	[TCP ACKed unseen segment] 443 → 5…
22625	13.61201…	172.20.5.20	23.221.52.42	TCP	66	44560 → 443 [ACK] Seq=1 Ack=1 Win=…
22694	13.62940…	23.221.52.42	172.20.5.20	TCP	66	[TCP ACKed unseen segment] 443 → 4…

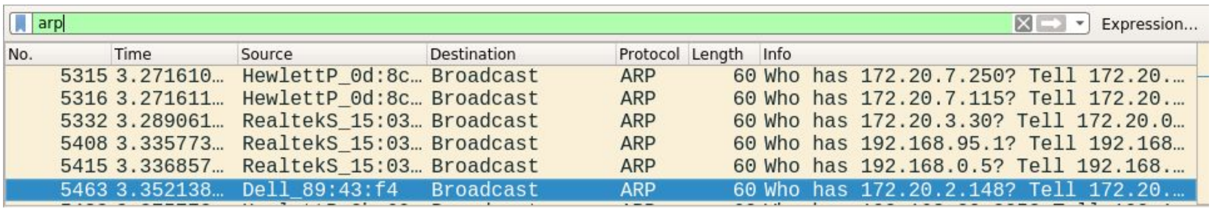
Udp :



udp

No.	Time	Source	Destination	Protocol	Length	Info
5473	3.365635…	192.168.0.121	224.2.2.2	UDP	72	51568 → 8995 Len=30
5474	3.366801…	192.168.0.121	224.2.2.2	UDP	72	51568 → 8995 Len=30
5475	3.368048…	192.168.0.121	224.2.2.2	UDP	72	51568 → 8995 Len=30
5476	3.369483…	192.168.0.121	224.2.2.2	UDP	72	51568 → 8995 Len=30
5477	3.370754…	192.168.0.121	224.2.2.2	UDP	72	51568 → 8995 Len=30
5478	3.371921…	192.168.0.121	224.2.2.2	UDP	72	51568 → 8995 Len=30

Arp: :

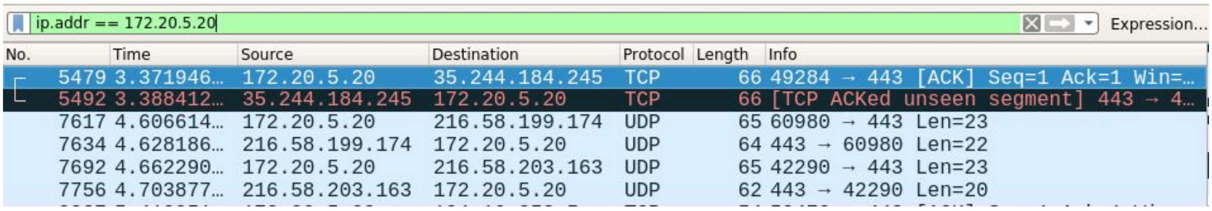


arp

No.	Time	Source	Destination	Protocol	Length	Info
5315	3.271610…	HewlettP_0d:8c…	Broadcast	ARP	60	Who has 172.20.7.250? Tell 172.20.…
5316	3.271611…	HewlettP_0d:8c…	Broadcast	ARP	60	Who has 172.20.7.115? Tell 172.20.…
5332	3.289061…	RealtekS_15:03…	Broadcast	ARP	60	Who has 172.20.3.30? Tell 172.20.0…
5408	3.335773…	RealtekS_15:03…	Broadcast	ARP	60	Who has 192.168.95.1? Tell 192.168…
5415	3.336857…	RealtekS_15:03…	Broadcast	ARP	60	Who has 192.168.0.5? Tell 192.168.…
5463	3.352138…	Dell_89:43:f4	Broadcast	ARP	60	Who has 172.20.2.148? Tell 172.20.…

Based on IP Address

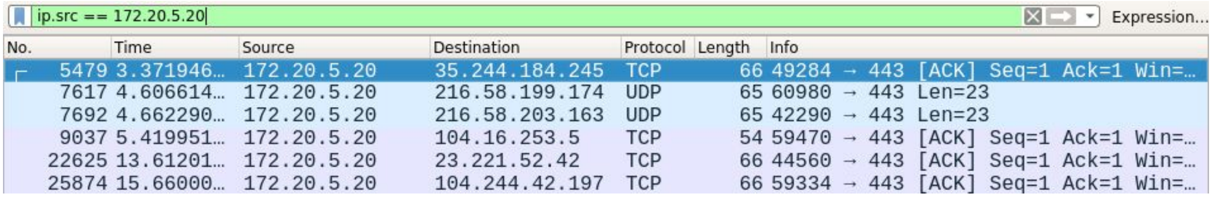
IP can be source or destination



ip.addr == 172.20.5.20

No.	Time	Source	Destination	Protocol	Length	Info
5479	3.371946…	172.20.5.20	35.244.184.245	TCP	66	49284 → 443 [ACK] Seq=1 Ack=1 Win=…
5492	3.388412…	35.244.184.245	172.20.5.20	TCP	66	[TCP ACKed unseen segment] 443 → 4…
7617	4.606614…	172.20.5.20	216.58.199.174	UDP	65	60980 → 443 Len=23
7634	4.628186…	216.58.199.174	172.20.5.20	UDP	64	443 → 60980 Len=22
7692	4.662290…	172.20.5.20	216.58.203.163	UDP	65	42290 → 443 Len=23
7756	4.703877…	216.58.203.163	172.20.5.20	UDP	62	443 → 42290 Len=20

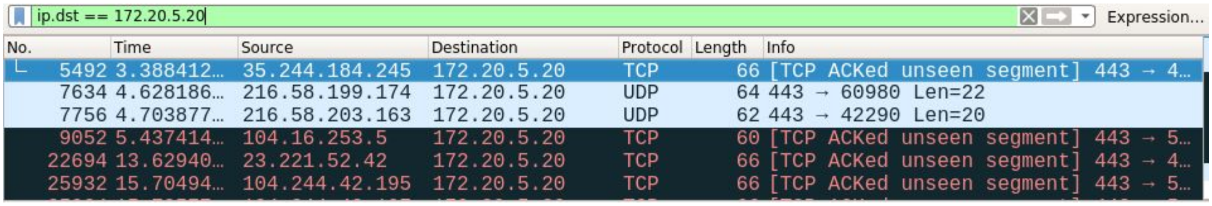
IP is source



ip.src == 172.20.5.20

No.	Time	Source	Destination	Protocol	Length	Info
5479	3.371946…	172.20.5.20	35.244.184.245	TCP	66	49284 → 443 [ACK] Seq=1 Ack=1 Win=…
7617	4.606614…	172.20.5.20	216.58.199.174	UDP	65	60980 → 443 Len=23
7692	4.662290…	172.20.5.20	216.58.203.163	UDP	65	42290 → 443 Len=23
9037	5.419951…	172.20.5.20	104.16.253.5	TCP	54	59470 → 443 [ACK] Seq=1 Ack=1 Win=…
22625	13.61201…	172.20.5.20	23.221.52.42	TCP	66	44560 → 443 [ACK] Seq=1 Ack=1 Win=…
25874	15.66000…	172.20.5.20	104.244.42.197	TCP	66	59334 → 443 [ACK] Seq=1 Ack=1 Win=…

IP is destination



ip.dst == 172.20.5.20

No.	Time	Source	Destination	Protocol	Length	Info
5492	3.388412…	35.244.184.245	172.20.5.20	TCP	66	[TCP ACKed unseen segment] 443 → 4…
7634	4.628186…	216.58.199.174	172.20.5.20	UDP	64	443 → 60980 Len=22
7756	4.703877…	216.58.203.163	172.20.5.20	UDP	62	443 → 42290 Len=20
9052	5.437414…	104.16.253.5	172.20.5.20	TCP	60	[TCP ACKed unseen segment] 443 → 5…
22694	13.62940…	23.221.52.42	172.20.5.20	TCP	66	[TCP ACKed unseen segment] 443 → 4…
25932	15.70494…	104.244.42.195	172.20.5.20	TCP	66	[TCP ACKed unseen segment] 443 → 5…

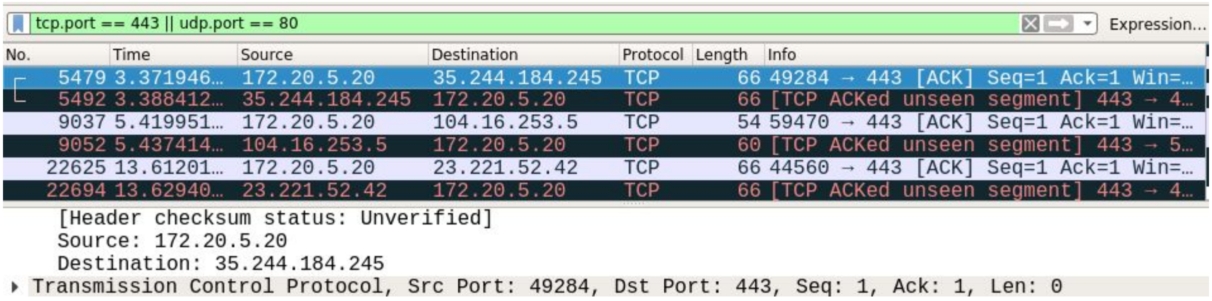
Display filters allow you to concentrate on the packets you are interested in while hiding the currently uninteresting ones. They allow you to only display packets based on:

Protocol
The presence of a field
The values of fields
A comparison between fields
… and a lot more!

To only display packets containing a particular protocol, type the protocol name in the display filter toolbar of the Wireshark window and press enter to apply the filter.

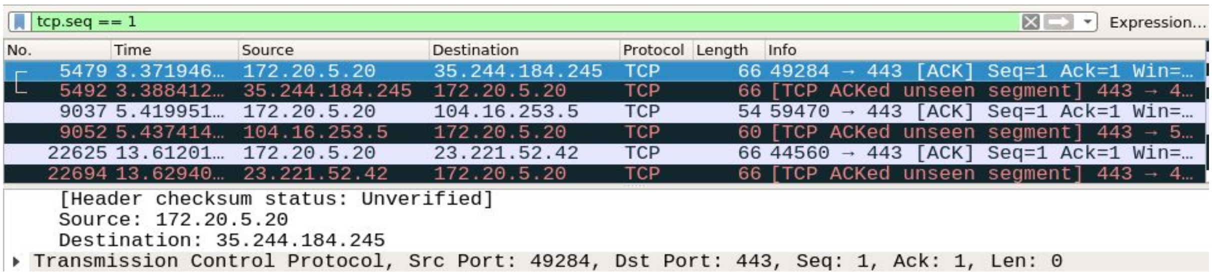
Based on Port number

Port 443



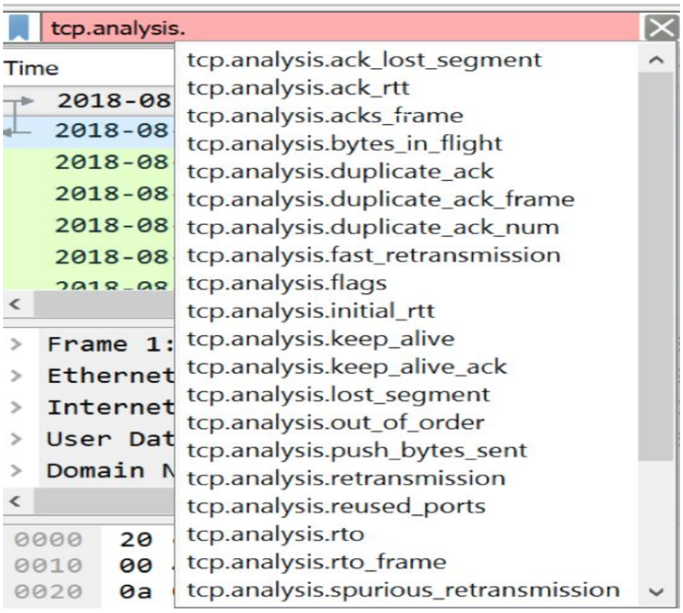
```
tcp.port == 443 || udp.port == 80                                                  Expression...
No.    Time         Source          Destination     Protocol Length Info
 5479 3.371946…  172.20.5.20     35.244.184.245  TCP       66 49284 → 443 [ACK] Seq=1 Ack=1 Win=…
 5492 3.388412…  35.244.184.245  172.20.5.20     TCP       66 [TCP ACKed unseen segment] 443 → 4…
 9037 5.419951…  172.20.5.20     104.16.253.5    TCP       54 59470 → 443 [ACK] Seq=1 Ack=1 Win=…
 9052 5.437414…  104.16.253.5    172.20.5.20     TCP       60 [TCP ACKed unseen segment] 443 → 5…
22625 13.61201…  172.20.5.20     23.221.52.42    TCP       66 44560 → 443 [ACK] Seq=1 Ack=1 Win=…
22694 13.62940…  23.221.52.42    172.20.5.20     TCP       66 [TCP ACKed unseen segment] 443 → 4…
      [Header checksum status: Unverified]
      Source: 172.20.5.20
      Destination: 35.244.184.245
 ▸ Transmission Control Protocol, Src Port: 49284, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
```

Based on Sequence Number



```
tcp.seq == 1                                                                       Expression...
No.    Time         Source          Destination     Protocol Length Info
 5479 3.371946…  172.20.5.20     35.244.184.245  TCP       66 49284 → 443 [ACK] Seq=1 Ack=1 Win=…
 5492 3.388412…  35.244.184.245  172.20.5.20     TCP       66 [TCP ACKed unseen segment] 443 → 4…
 9037 5.419951…  172.20.5.20     104.16.253.5    TCP       54 59470 → 443 [ACK] Seq=1 Ack=1 Win=…
 9052 5.437414…  104.16.253.5    172.20.5.20     TCP       60 [TCP ACKed unseen segment] 443 → 5…
22625 13.61201…  172.20.5.20     23.221.52.42    TCP       66 44560 → 443 [ACK] Seq=1 Ack=1 Win=…
22694 13.62940…  23.221.52.42    172.20.5.20     TCP       66 [TCP ACKed unseen segment] 443 → 4…
      [Header checksum status: Unverified]
      Source: 172.20.5.20
      Destination: 35.244.184.245
 ▸ Transmission Control Protocol, Src Port: 49284, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
```

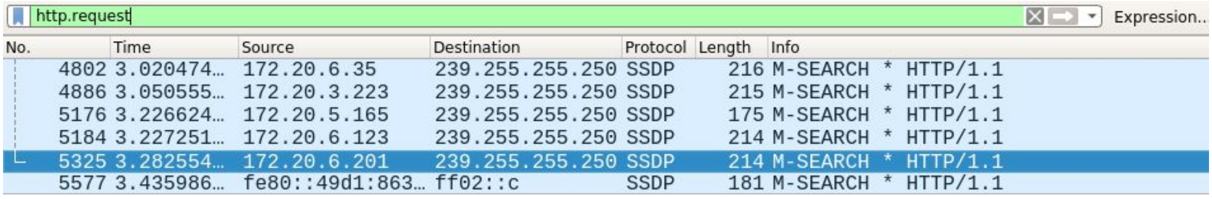
Tcp.analysis filter



```
tcp.analysis.
```

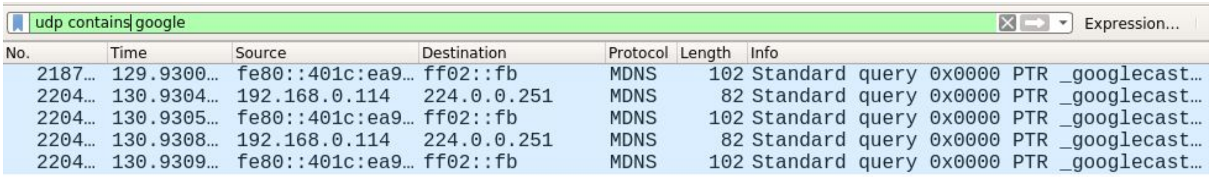
Time
2018-08
2018-08
2018-08
2018-08
2018-08
2018-08
Frame 1:
Ethernet
Internet
User Dat
Domain N
0000 20
0010 00
0020 0a
tcp.analysis.ack_lost_segment
tcp.analysis.ack_rtt
tcp.analysis.acks_frame
tcp.analysis.bytes_in_flight
tcp.analysis.duplicate_ack
tcp.analysis.duplicate_ack_frame
tcp.analysis.duplicate_ack_num
tcp.analysis.fast_retransmission
tcp.analysis.flags
tcp.analysis.initial_rtt
tcp.analysis.keep_alive
tcp.analysis.keep_alive_ack
tcp.analysis.lost_segment
tcp.analysis.out_of_order
tcp.analysis.push_bytes_sent
tcp.analysis.retransmission
tcp.analysis.reused_ports
tcp.analysis.rto
tcp.analysis.rto_frame
tcp.analysis.spurious_retransmission

Http.request



```
http.request                                                                       Expression...
No.    Time         Source          Destination     Protocol Length Info
 4802 3.020474…  172.20.6.35     239.255.255.250 SSDP     216 M-SEARCH * HTTP/1.1
 4886 3.050555…  172.20.3.223    239.255.255.250 SSDP     215 M-SEARCH * HTTP/1.1
 5176 3.226624…  172.20.5.165    239.255.255.250 SSDP     175 M-SEARCH * HTTP/1.1
 5184 3.227251…  172.20.6.123    239.255.255.250 SSDP     214 M-SEARCH * HTTP/1.1
 5325 3.282554…  172.20.6.201    239.255.255.250 SSDP     214 M-SEARCH * HTTP/1.1
 5577 3.435986…  fe80::49d1:863… ff02::c         SSDP     181 M-SEARCH * HTTP/1.1
```

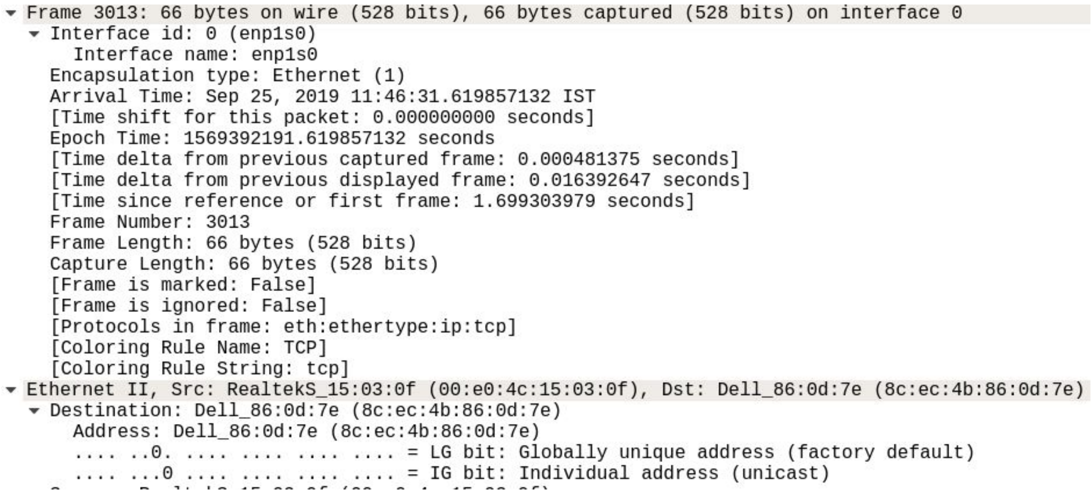
Udp contains google



udp contains google

```
No.     Time       Source              Destination    Protocol Length Info
  2187… 129.9300…  fe80::401c:ea9…     ff02::fb       MDNS        102 Standard query 0x0000 PTR _googlecast…
  2204… 130.9304…  192.168.0.114       224.0.0.251    MDNS         82 Standard query 0x0000 PTR _googlecast…
  2204… 130.9305…  fe80::401c:ea9…     ff02::fb       MDNS        102 Standard query 0x0000 PTR _googlecast…
  2204… 130.9308…  192.168.0.114       224.0.0.251    MDNS         82 Standard query 0x0000 PTR _googlecast…
  2204… 130.9309…  fe80::401c:ea9…     ff02::fb       MDNS        102 Standard query 0x0000 PTR _googlecast…
```

In this tool, we can also find the frame format and value of fields.



```
▾ Frame 3013: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
  ▾ Interface id: 0 (enp1s0)
      Interface name: enp1s0
    Encapsulation type: Ethernet (1)
    Arrival Time: Sep 25, 2019 11:46:31.619857132 IST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1569392191.619857132 seconds
    [Time delta from previous captured frame: 0.000481375 seconds]
    [Time delta from previous displayed frame: 0.016392647 seconds]
    [Time since reference or first frame: 1.699303979 seconds]
    Frame Number: 3013
    Frame Length: 66 bytes (528 bits)
    Capture Length: 66 bytes (528 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp]
    [Coloring Rule Name: TCP]
    [Coloring Rule String: tcp]
▾ Ethernet II, Src: RealtekS_15:03:0f (00:e0:4c:15:03:0f), Dst: Dell_86:0d:7e (8c:ec:4b:86:0d:7e)
  ▾ Destination: Dell_86:0d:7e (8c:ec:4b:86:0d:7e)
      Address: Dell_86:0d:7e (8c:ec:4b:86:0d:7e)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
```

Conclusion:

In this experiment, we do the analysis of the network traces. Using wireshark tool, we can check if the network has any problems or not. We can come to know about the different frame formats of protocols and their field values.