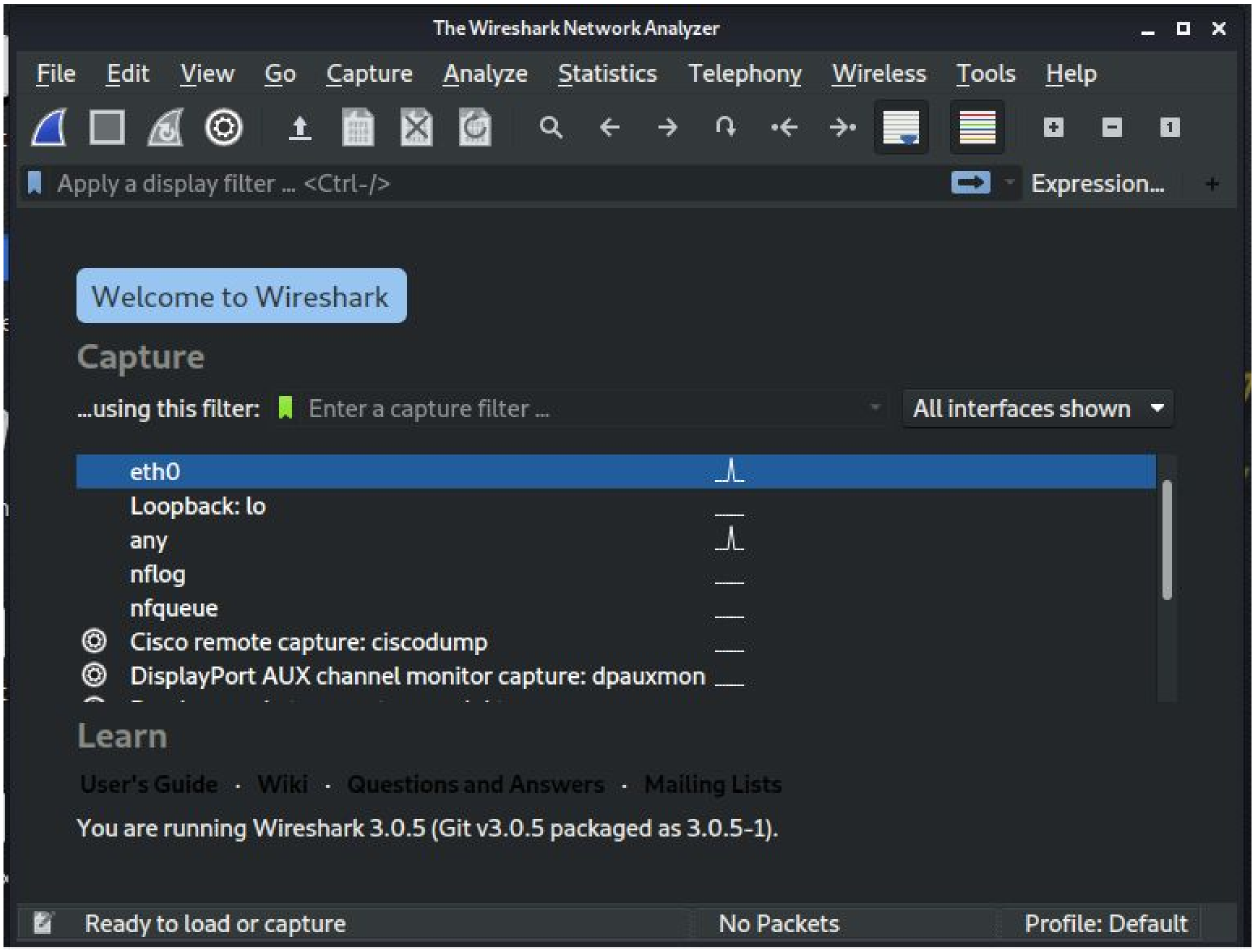
Wireshark for Noobs

By Anmol K Sachan

anmol221999@gmail.com
Linkedin: https://linkedin.com/in/anmolksachan/
Ig: https://instagram.com/the_guy_that_hacks

INDEX

1. Getting Started with Wireshark



eth0
Loopback: lo
any
nflog
nfqueue
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon

You are running Wireshark 3.0.5 (Git v3.0.5 packaged as 3.0.5-1).

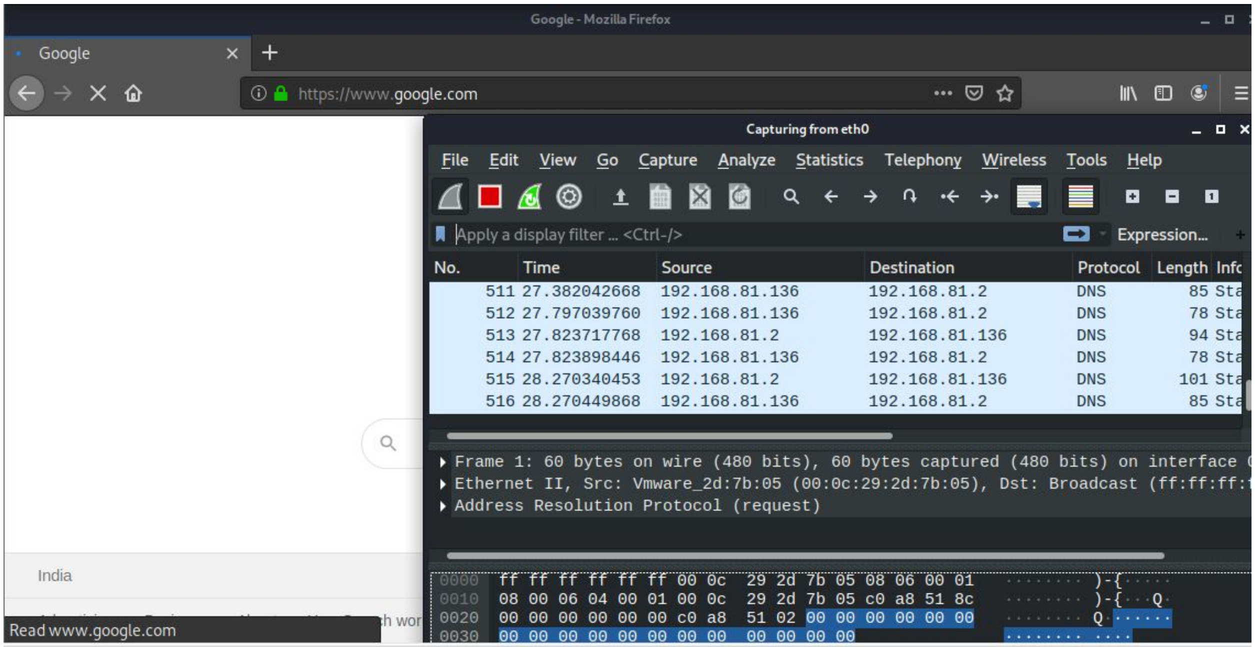
Ready to load or capture · No Packets · Profile: Default

Wireshark comes pre-installed in kali linux.

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Originally named Ethereal, the project was renamed Wireshark in May 2006 due to trademark issues.

The **GUI of wireshark** have

1. Title Bar
2. Main Menu
3. Main Toolbar
4. Filter Toolbar
5. Packet List
6. Intelligent Scrollbar
7. Packet Details
8. Packet Bytes
9. Status Bar



```
No.   Time         Source          Destination     Protocol Length Infc
  511 27.382042668 192.168.81.136  192.168.81.2    DNS          85 Sta
  512 27.797039760 192.168.81.136  192.168.81.2    DNS          78 Sta
  513 27.823717768 192.168.81.2    192.168.81.136  DNS          94 Sta
  514 27.823898446 192.168.81.136  192.168.81.2    DNS          78 Sta
  515 28.270340453 192.168.81.2    192.168.81.136  DNS         101 Sta
  516 28.270449868 192.168.81.136  192.168.81.2    DNS          85 Sta
```

```
▸ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface (
▸ Ethernet II, Src: Vmware_2d:7b:05 (00:0c:29:2d:7b:05), Dst: Broadcast (ff:ff:ff:f
▸ Address Resolution Protocol (request)
```

```
0000  ff ff ff ff ff ff 00 0c  29 2d 7b 05 08 06 00 01   ········ )-{·····
0010  08 00 06 04 00 01 00 0c  29 2d 7b 05 c0 a8 51 8c   ········ )-{···Q·
0020  00 00 00 00 00 00 c0 a8  51 02 00 00 00 00 00 00   ········ Q·······
0030  00 00 00 00 00 00 00 00  00 00 00 00               ········ ····
```

In the above simply clicking on eth0 interface starts capturing packets, while **sniffing** we can **analyse** and can apply **filters** to see the exact requirement.

Define the **four layers of the TCP/IP** reference model.

the TCP layer handles the message to be transmitted. This message is usually broken down into small units. These small units are known as packets. Further, these packets are transmitted over the network.

These packets are received by the corresponding TCP layer in the receiver and reassembled into the original message.

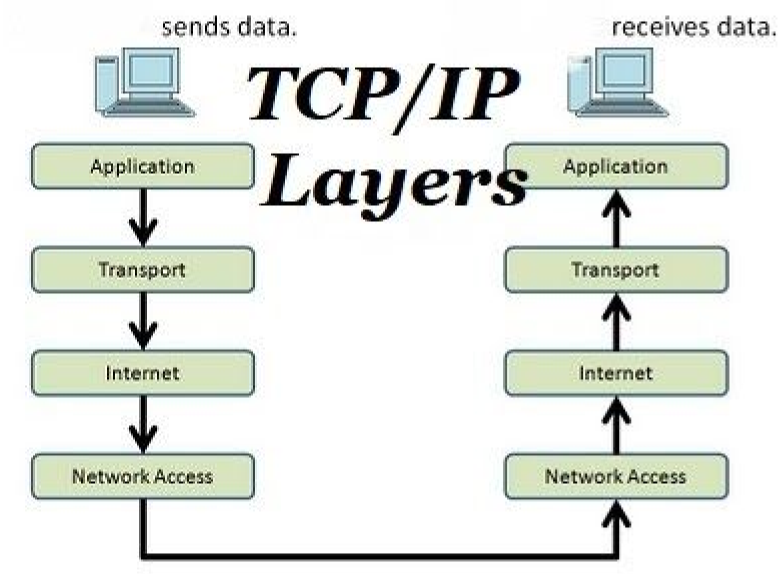
TCP/IP Model have 4 layers, those are:

Application Layer

Transport Layer

Internet Layer

Network Layer



Application layer:

The first layer is the application layer. This layer provides the applications a standardized data exchange. The protocols for these layers are given below:

- Hypertext Transfer Protocol (HTTP)
- File Transfer Protocol (FTP)
- Post Office Protocol 3 (POP3)
- Simple Mail Transfer Protocol (SMTP)
- Simple Network Management Protocol (SNMP)

  This layered work with all these protocols.

Transport layer:

The transport layer is the second layer of the TCP/IP model. The basic work of the transport layer is to maintain end-to-end communications. The protocols for these layers are given below:

- TCP
- User Datagram Protocol (UDP)

  These two protocols are used for the transport layer in TCP/IP.

Network layer:

The third layer of TCP IP is a network layer. It is also known as the internet layer. The network layer deals with packets. The following are protocols uses in this layer.

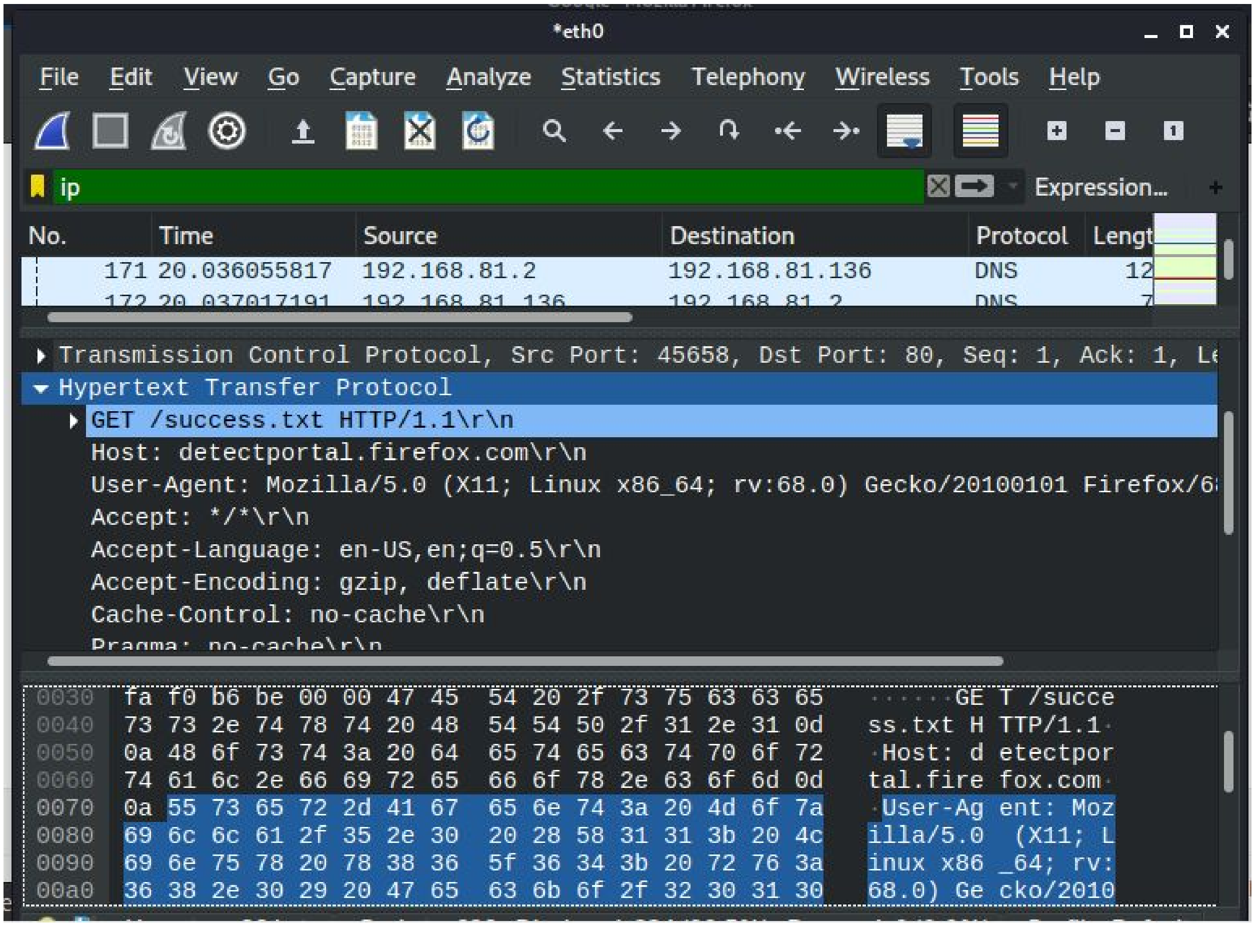
- IP
- Internet Control Message Protocol (ICMP)

Physical Layer

The last layer is the physical layer. This layered work with the following protocols.

- Ethernet for LAN(local area networks)
- Address Resolution Protocol (ARP)

Examine **packet header data** with Wireshark



Filter: ip

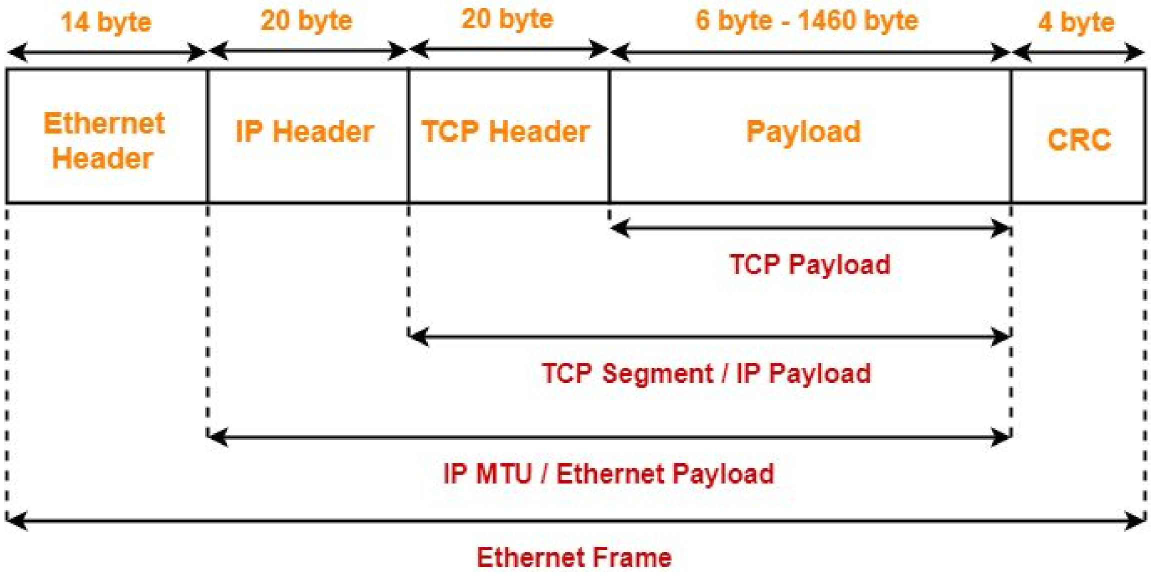
```
No.   Time            Source           Destination      Protocol  Lengt
  171 20.036055817    192.168.81.2     192.168.81.136   DNS         12
  172 20.037017191    192.168.81.136   192.168.81.2     DNS          7
```

```
▸ Transmission Control Protocol, Src Port: 45658, Dst Port: 80, Seq: 1, Ack: 1, Le
▾ Hypertext Transfer Protocol
  ▸ GET /success.txt HTTP/1.1\r\n
    Host: detectportal.firefox.com\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/6
    Accept: */*\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Cache-Control: no-cache\r\n
    Pragma: no-cache\r\n
```

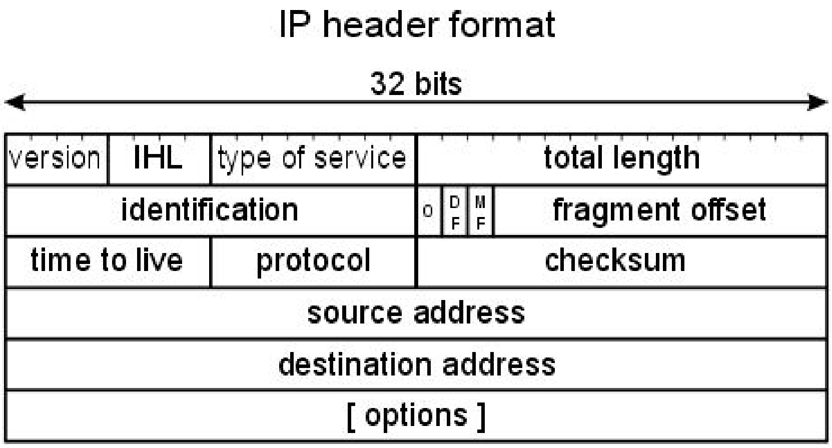
```
0030  fa f0 b6 be 00 00 47 45  54 20 2f 73 75 63 63 65   ······GE T /succe
0040  73 73 2e 74 78 74 20 48  54 54 50 2f 31 2e 31 0d   ss.txt H TTP/1.1·
0050  0a 48 6f 73 74 3a 20 64  65 74 65 63 74 70 6f 72   ·Host: d etectpor
0060  74 61 6c 2e 66 69 72 65  66 6f 78 2e 63 6f 6d 0d   tal.fire fox.com·
0070  0a 55 73 65 72 2d 41 67  65 6e 74 3a 20 4d 6f 7a   ·User-Ag ent: Moz
0080  69 6c 6c 61 2f 35 2e 30  20 28 58 31 31 3b 20 4c   illa/5.0  (X11; L
0090  69 6e 75 78 20 78 38 36  5f 36 34 3b 20 72 76 3a   inux x86 _64; rv:
00a0  36 38 2e 30 29 20 47 65  63 6b 6f 2f 32 30 31 30   68.0) Ge cko/2010
```

Pic. Headers of data packets shown above

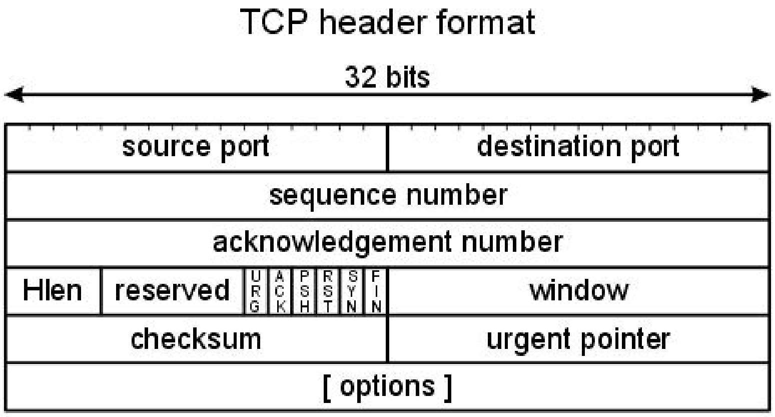
Define the **header fields** of **Ethernet frame**, **Internet Protocol** (IP), **Transport Control Protocol** (TCP), and **User Datagram Protocol** (UDP) packets / different types of packet headers, including the header fields and their values



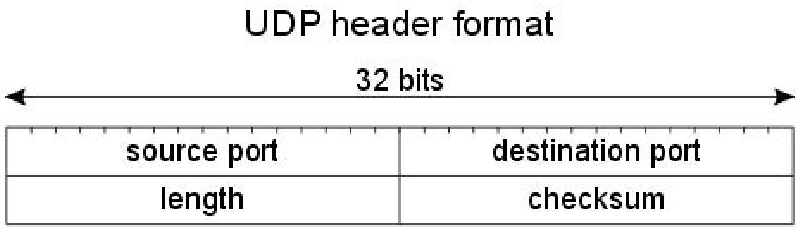
Ethernet Frame



IP Protocol

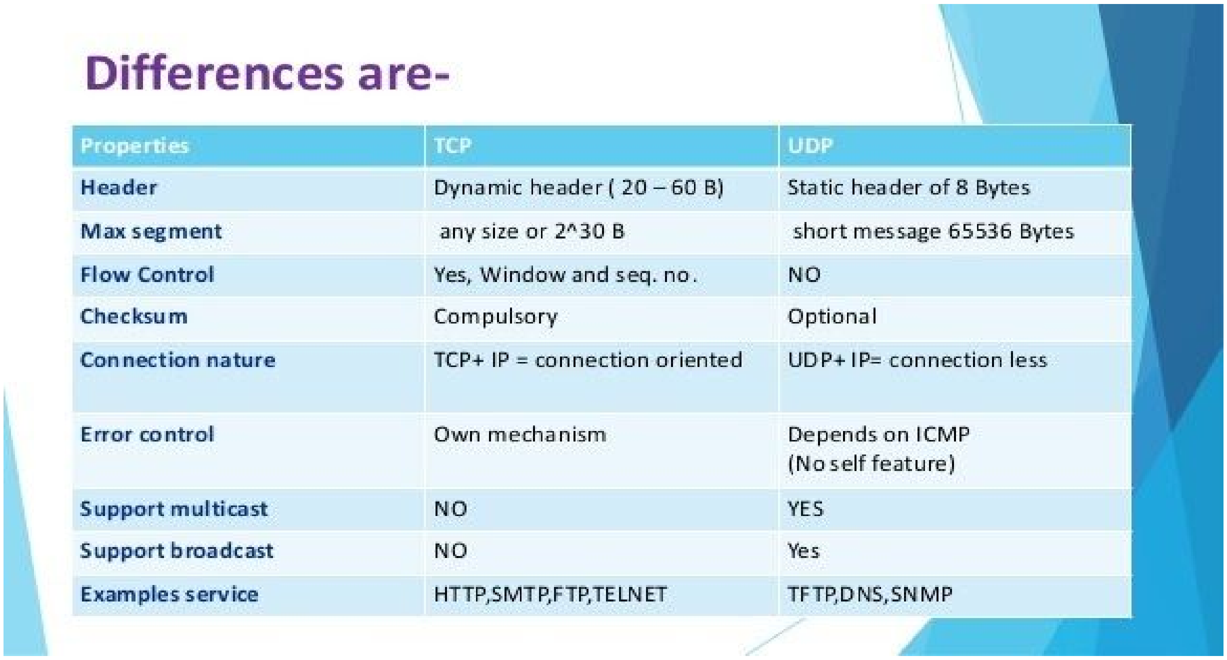


TCP Header



UDP Header

Compare and contrast **TCP** and **UDP**.

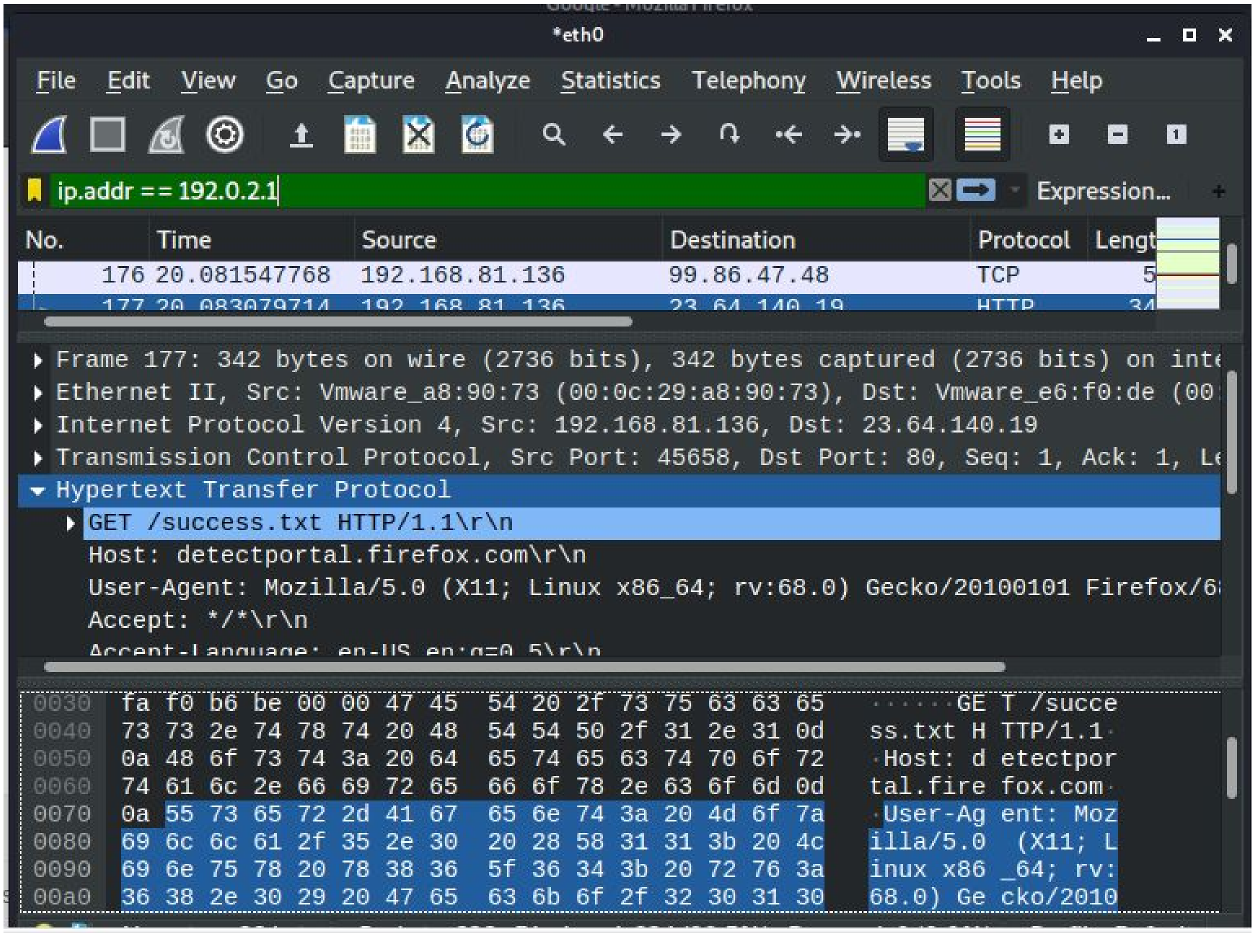


Differences are-

Properties	TCP	UDP
Header	Dynamic header (20 – 60 B)	Static header of 8 Bytes
Max segment	any size or 2^30 B	short message 65536 Bytes
Flow Control	Yes, Window and seq. no.	NO
Checksum	Compulsory	Optional
Connection nature	TCP+ IP = connection oriented	UDP+ IP= connection less
Error control	Own mechanism	Depends on ICMP (No self feature)
Support multicast	NO	YES
Support broadcast	NO	Yes
Examples service	HTTP,SMTP,FTP,TELNET	TFTP,DNS,SNMP

2. Start Sniffing: Perform a Live Capture of Network Traffic/Web Traffic

2.1 Filter Packets with the Filter Bar during capture and explain all possible filters used by you.



```
ip.addr == 192.0.2.1
```

```
No.       Time            Source            Destination      Protocol  Lengt
      176 20.081547768    192.168.81.136    99.86.47.48      TCP          5
      177 20.083079714    192.168.81.136    23.64.140.19     HTTP        34
```

```
▸ Frame 177: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on inte
▸ Ethernet II, Src: Vmware_a8:90:73 (00:0c:29:a8:90:73), Dst: Vmware_e6:f0:de (00
▸ Internet Protocol Version 4, Src: 192.168.81.136, Dst: 23.64.140.19
▸ Transmission Control Protocol, Src Port: 45658, Dst Port: 80, Seq: 1, Ack: 1, Le
▾ Hypertext Transfer Protocol
  ▸ GET /success.txt HTTP/1.1\r\n
    Host: detectportal.firefox.com\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/6
    Accept: */*\r\n
    Accept-Language: en-US,en;q=0.5\r\n
```

```
0030  fa f0 b6 be 00 00 47 45  54 20 2f 73 75 63 63 65   ······GE T /succe
0040  73 73 2e 74 78 74 20 48  54 54 50 2f 31 2e 31 0d   ss.txt H TTP/1.1·
0050  0a 48 6f 73 74 3a 20 64  65 74 65 63 74 70 6f 72   ·Host: d etectpor
0060  74 61 6c 2e 66 69 72 65  66 6f 78 2e 63 6f 6d 0d   tal.fire fox.com·
0070  0a 55 73 65 72 2d 41 67  65 6e 74 3a 20 4d 6f 7a   ·User-Ag ent: Moz
0080  69 6c 6c 61 2f 35 2e 30  20 28 58 31 31 3b 20 4c   illa/5.0  (X11; L
0090  69 6e 75 78 20 78 38 36  5f 36 34 3b 20 72 76 3a   inux x86 _64; rv:
00a0  36 38 2e 30 29 20 47 65  63 6b 6f 2f 32 30 31 30   68.0) Ge cko/2010
```

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
```

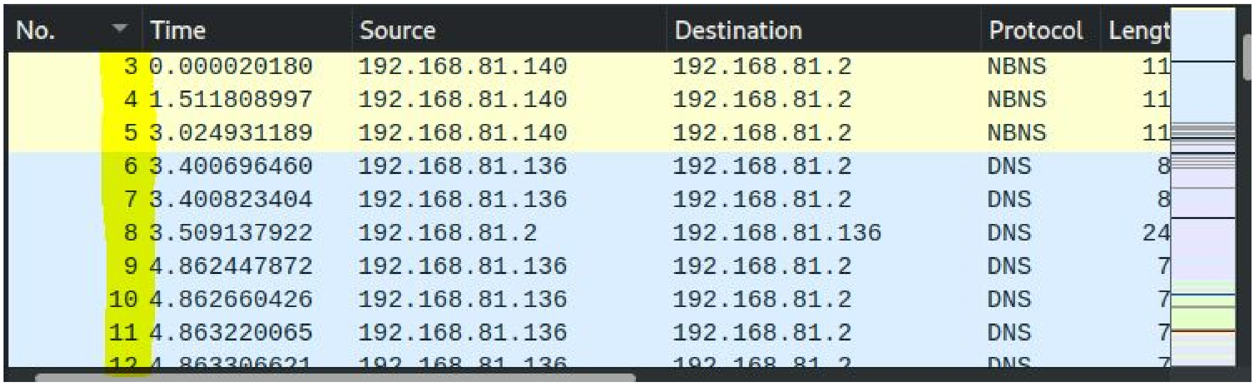
Capture except all ARP and DNS traffic:

```
port not 53 and not arp
```

To capture vlan traffic

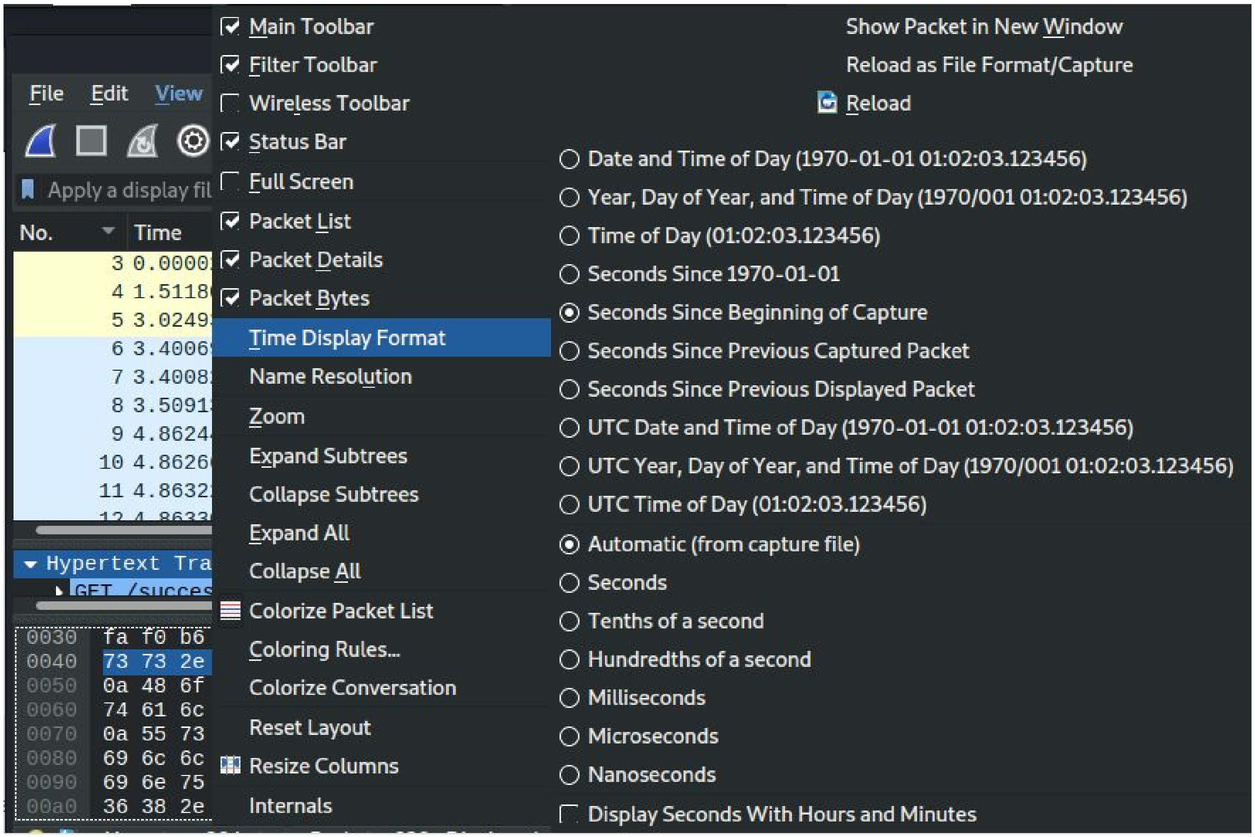
```
vlan
```

3. View Packet Summaries with the Packet List Window

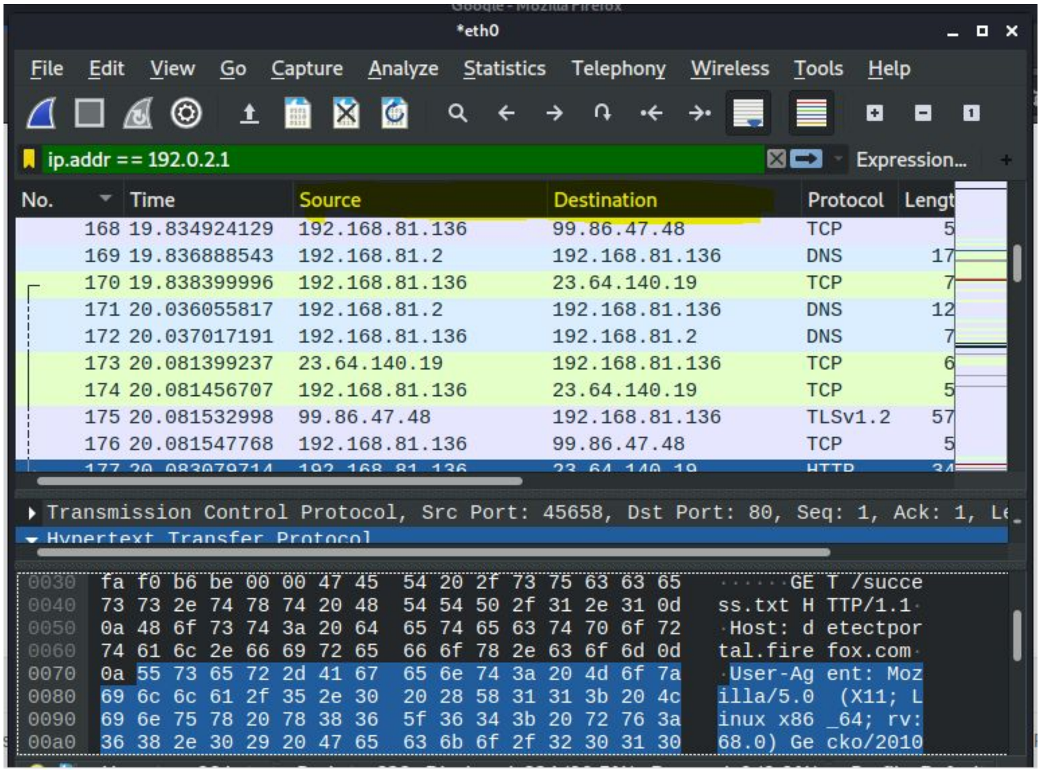


No.	Time	Source	Destination	Protocol	Lengt
3	0.000020180	192.168.81.140	192.168.81.2	NBNS	11
4	1.511808997	192.168.81.140	192.168.81.2	NBNS	11
5	3.024931189	192.168.81.140	192.168.81.2	NBNS	11
6	3.400696460	192.168.81.136	192.168.81.2	DNS	8
7	3.400823404	192.168.81.136	192.168.81.2	DNS	8
8	3.509137922	192.168.81.2	192.168.81.136	DNS	24
9	4.862447872	192.168.81.136	192.168.81.2	DNS	7
10	4.862660426	192.168.81.136	192.168.81.2	DNS	7
11	4.863220065	192.168.81.136	192.168.81.2	DNS	7
12	4.863306621	192.168.81.136	192.168.81.2	DNS	7

Packet number (No.): Numbers each packet starts with 1 for the first packet.


File
Edit
View
Apply a display fil
☑ Main Toolbar
☑ Filter Toolbar
☐ Wireless Toolbar
☑ Status Bar
☐ Full Screen
☑ Packet List
☑ Packet Details
☑ Packet Bytes
Time Display Format
Name Resolution
Zoom
Expand Subtrees
Collapse Subtrees
Expand All
Collapse All
Colorize Packet List
Coloring Rules...
Colorize Conversation
Reset Layout
Resize Columns
Internals
Show Packet in New Window
Reload as File Format/Capture
Reload
○ Date and Time of Day (1970-01-01 01:02:03.123456)
○ Year, Day of Year, and Time of Day (1970/001 01:02:03.123456)
○ Time of Day (01:02:03.123456)
○ Seconds Since 1970-01-01
● Seconds Since Beginning of Capture
○ Seconds Since Previous Captured Packet
○ Seconds Since Previous Displayed Packet
○ UTC Date and Time of Day (1970-01-01 01:02:03.123456)
○ UTC Year, Day of Year, and Time of Day (1970/001 01:02:03.123456)
○ UTC Time of Day (01:02:03.123456)
● Automatic (from capture file)
○ Seconds
○ Tenths of a second
○ Hundredths of a second
○ Milliseconds
○ Microseconds
○ Nanoseconds
☐ Display Seconds With Hours and Minutes

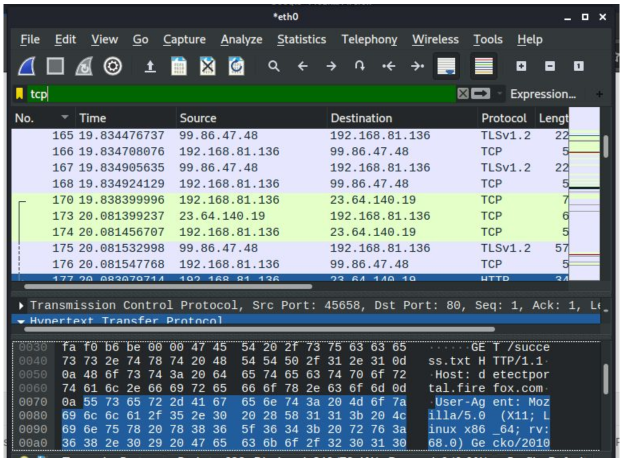
Timestamp (Time): Default is the number of seconds since the beginning of the capture



```
ip.addr == 192.0.2.1
```

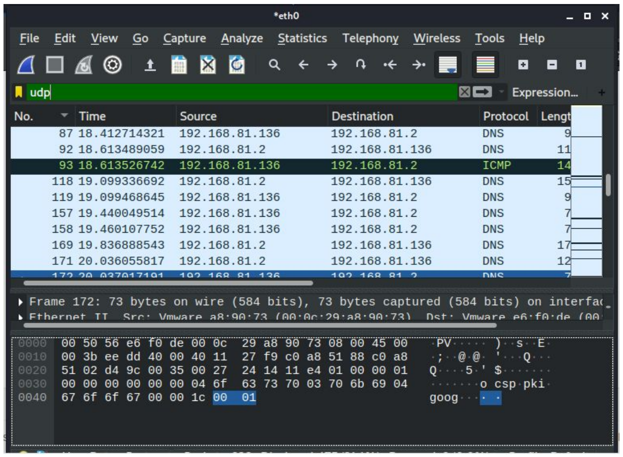
```
No.   Time          Source           Destination      Protocol  Lengt
  168 19.834924129  192.168.81.136   99.86.47.48      TCP          5
  169 19.836888543  192.168.81.2     192.168.81.136   DNS         17
  170 19.838399996  192.168.81.136   23.64.140.19     TCP          7
  171 20.036055817  192.168.81.2     192.168.81.136   DNS         12
  172 20.037017191  192.168.81.136   192.168.81.2     DNS          7
  173 20.081399237  23.64.140.19     192.168.81.136   TCP          6
  174 20.081456707  192.168.81.136   23.64.140.19     TCP          5
  175 20.081532998  99.86.47.48      192.168.81.136   TLSv1.2     57
  176 20.081547768  192.168.81.136   99.86.47.48      TCP          5
  177 20.083079714  192.168.81.136   23.64.140.19     HTTP        34

▸ Transmission Control Protocol, Src Port: 45658, Dst Port: 80, Seq: 1, Ack: 1, Le
▾ Hypertext Transfer Protocol

0030  fa f0 b6 be 00 00 47 45  54 20 2f 73 75 63 63 65   ······GE T /succe
0040  73 73 2e 74 78 74 20 48  54 54 50 2f 31 2e 31 0d   ss.txt H TTP/1.1·
0050  0a 48 6f 73 74 3a 20 64  65 74 65 63 74 70 6f 72   ·Host: d etectpor
0060  74 61 6c 2e 66 69 72 65  66 6f 78 2e 63 6f 6d 0d   tal.fire fox.com·
0070  0a 55 73 65 72 2d 41 67  65 6e 74 3a 20 4d 6f 7a   ·User-Ag ent: Moz
0080  69 6c 6c 61 2f 35 2e 30  20 28 58 31 31 3b 20 4c   illa/5.0  (X11; L
0090  69 6e 75 78 20 78 38 36  5f 36 34 3b 20 72 76 3a   inux x86 _64; rv:
00a0  36 38 2e 30 29 20 47 65  63 6b 6f 2f 32 30 31 30   68.0) Ge cko/2010
```

IP Addresses (Source, Destination): The source and destination address of the packet.



```
tcp
```

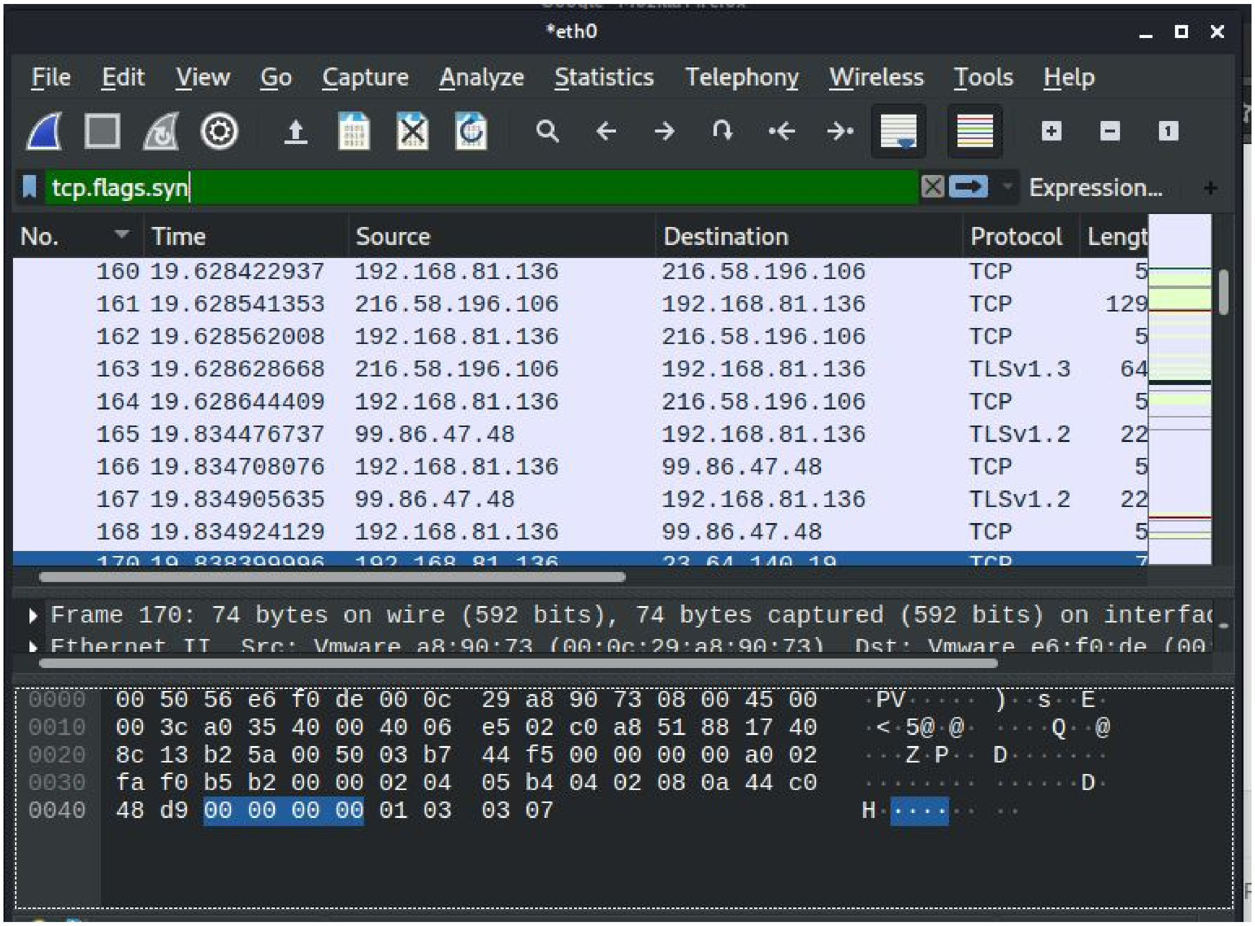
```
No.   Time          Source           Destination      Protocol  Lengt
  165 19.834476737  99.86.47.48      192.168.81.136   TLSv1.2     22
  166 19.834708076  192.168.81.136   99.86.47.48      TCP          5
  167 19.834905635  99.86.47.48      192.168.81.136   TLSv1.2     22
  168 19.834924129  192.168.81.136   99.86.47.48      TCP          5
  170 19.838399996  192.168.81.136   23.64.140.19     TCP          7
  173 20.081399237  23.64.140.19     192.168.81.136   TCP          6
  174 20.081456707  192.168.81.136   23.64.140.19     TCP          5
  175 20.081532998  99.86.47.48      192.168.81.136   TLSv1.2     57
  176 20.081547768  192.168.81.136   99.86.47.48      TCP          5
  177 20.083079714  192.168.81.136   23.64.140.19     HTTP        34

▸ Transmission Control Protocol, Src Port: 45658, Dst Port: 80, Seq: 1, Ack: 1, Le
▾ Hypertext Transfer Protocol

0030  fa f0 b6 be 00 00 47 45  54 20 2f 73 75 63 63 65   ······GE T /succe
0040  73 73 2e 74 78 74 20 48  54 54 50 2f 31 2e 31 0d   ss.txt H TTP/1.1·
0050  0a 48 6f 73 74 3a 20 64  65 74 65 63 74 70 6f 72   ·Host: d etectpor
0060  74 61 6c 2e 66 69 72 65  66 6f 78 2e 63 6f 6d 0d   tal.fire fox.com·
0070  0a 55 73 65 72 2d 41 67  65 6e 74 3a 20 4d 6f 7a   ·User-Ag ent: Moz
0080  69 6c 6c 61 2f 35 2e 30  20 28 58 31 31 3b 20 4c   illa/5.0  (X11; L
0090  69 6e 75 78 20 78 38 36  5f 36 34 3b 20 72 76 3a   inux x86 _64; rv:
00a0  36 38 2e 30 29 20 47 65  63 6b 6f 2f 32 30 31 30   68.0) Ge cko/2010
```



```
udp
```

```
No.   Time          Source           Destination      Protocol  Lengt
   87 18.412714321  192.168.81.136   192.168.81.2     DNS          9
   92 18.613489059  192.168.81.2     192.168.81.136   DNS         11
   93 18.613526742  192.168.81.136   192.168.81.2     ICMP        14
  118 19.099336692  192.168.81.2     192.168.81.136   DNS         15
  119 19.099468645  192.168.81.136   192.168.81.2     DNS          9
  157 19.440049514  192.168.81.136   192.168.81.2     DNS          7
  158 19.460107752  192.168.81.136   192.168.81.2     DNS          7
  169 19.836888543  192.168.81.2     192.168.81.136   DNS         17
  171 20.036055817  192.168.81.2     192.168.81.136   DNS         12
  172 20.037017191  192.168.81.136   192.168.81.2     DNS          7

▸ Frame 172: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interfac
▸ Ethernet II, Src: Vmware_a8:90:73 (00:0c:29:a8:90:73), Dst: Vmware_e6:f0:de (00

0000  00 50 56 e6 f0 de 00 0c  29 a8 90 73 08 00 45 00   ·PV·····  )··s··E·
0010  00 3b ee dd 40 00 40 11  27 f9 c0 a8 51 88 c0 a8   ·;··@·@·  '···Q···
0020  51 02 d4 9c 00 35 00 27  24 14 11 e4 01 00 00 01   Q····5·' $·······
0030  00 00 00 00 00 00 04 6f  63 73 70 03 70 6b 69 04   ·······o csp·pki·
0040  67 6f 6f 67 00 00 1c 00  01                        goog···· ·
```

Protocols (Protocol): The packet protocol (TCP, UDP, NBNS, etc.).



tcp.flags.syn

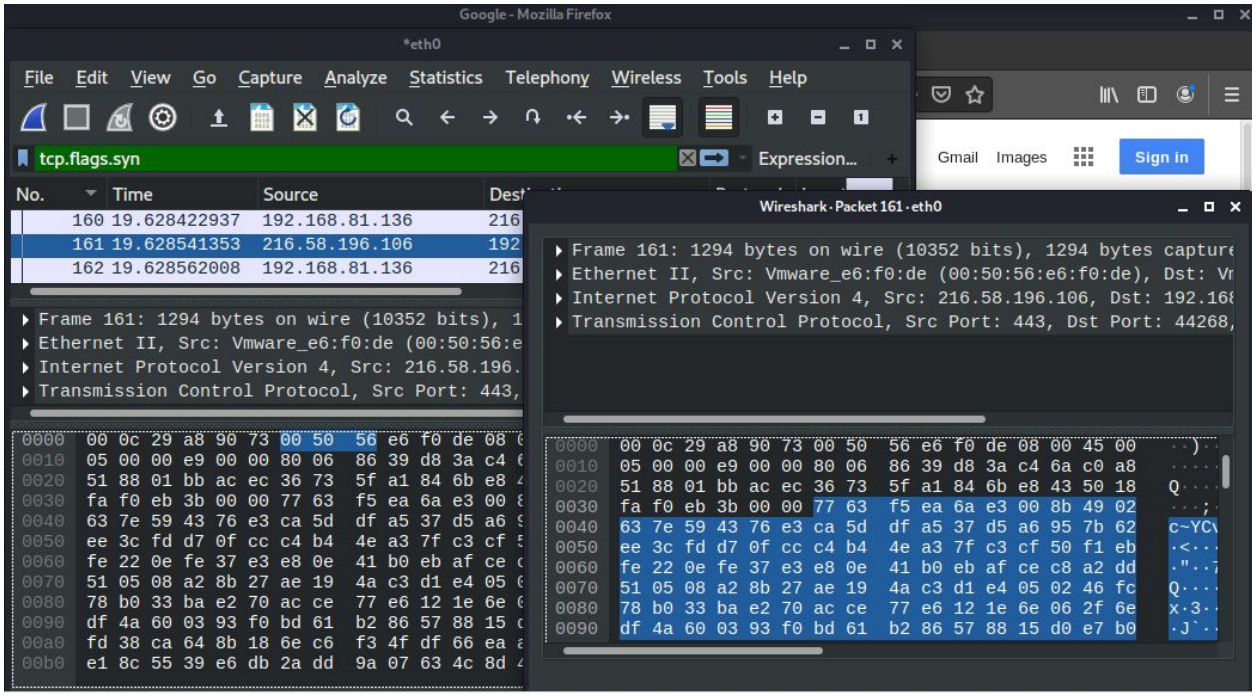
```
No.     Time          Source           Destination      Protocol  Lengt
    160 19.628422937  192.168.81.136   216.58.196.106   TCP          5
    161 19.628541353  216.58.196.106   192.168.81.136   TCP        129
    162 19.628562008  192.168.81.136   216.58.196.106   TCP          5
    163 19.628628668  216.58.196.106   192.168.81.136   TLSv1.3     64
    164 19.628644409  192.168.81.136   216.58.196.106   TCP          5
    165 19.834476737  99.86.47.48      192.168.81.136   TLSv1.2     22
    166 19.834708076  192.168.81.136   99.86.47.48      TCP          5
    167 19.834905635  99.86.47.48      192.168.81.136   TLSv1.2     22
    168 19.834924129  192.168.81.136   99.86.47.48      TCP          5
    170 19.838399996  192.168.81.136   23.64.140.19     TCP          7
```

```
▸ Frame 170: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interfac
▸ Ethernet II, Src: Vmware_a8:90:73 (00:0c:29:a8:90:73), Dst: Vmware_e6:f0:de (00:
```

```
0000  00 50 56 e6 f0 de 00 0c  29 a8 90 73 08 00 45 00   ·PV·····)··s··E·
0010  00 3c a0 35 40 00 40 06  e5 02 c0 a8 51 88 17 40   ·<·5@·@·····Q··@
0020  8c 13 b2 5a 00 50 03 b7  44 f5 00 00 00 00 a0 02   ···Z·P··D·······
0030  fa f0 b5 b2 00 00 02 04  05 b4 04 02 08 0a 44 c0   ··············D·
0040  48 d9 00 00 00 00 01 03  03 07                     H·········
```

Additional Protocol Information (info): Example: for a TCP packet, this field states if it is a SYN, ACK, or FIN packet.

4. Study Packet Details with the Packet Details Window



tcp.flags.syn

```
No.    Time          Source           Dest
   160 19.628422937  192.168.81.136   216
   161 19.628541353  216.58.196.106   192
   162 19.628562008  192.168.81.136   216
```

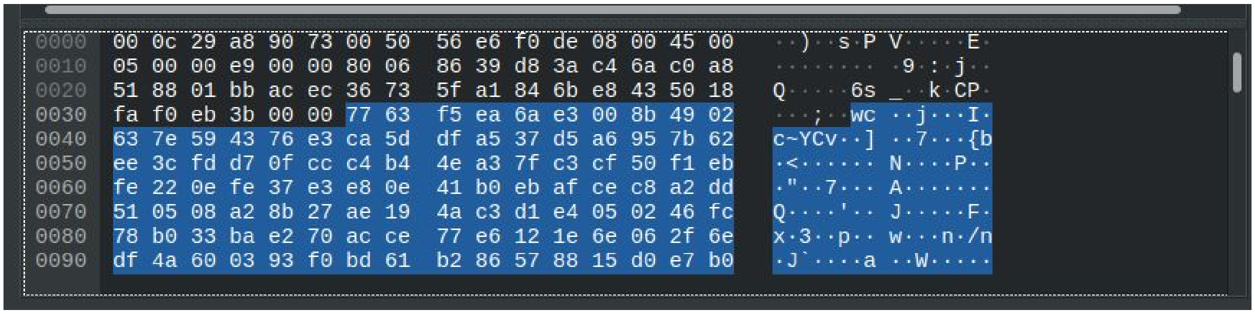
```
▸ Frame 161: 1294 bytes on wire (10352 bits), 1
▸ Ethernet II, Src: Vmware_e6:f0:de (00:50:56:e
▸ Internet Protocol Version 4, Src: 216.58.196.
▸ Transmission Control Protocol, Src Port: 443,
```

```
0000  00 0c 29 a8 90 73 00 50  56 e6 f0 de 08 0
0010  05 00 00 e9 00 00 80 06  86 39 d8 3a c4 6
0020  51 88 01 bb ac ec 36 73  5f a1 84 6b e8 4
0030  fa f0 eb 3b 00 00 77 63  f5 ea 6a e3 00 8
0040  63 7e 59 43 76 e3 ca 5d  df a5 37 d5 a6 9
0050  ee 3c fd d7 0f cc c4 b4  4e a3 7f c3 cf 5
0060  fe 22 0e fe 37 e3 e8 0e  41 b0 eb af ce c
0070  51 05 08 a2 8b 27 ae 19  4a c3 d1 e4 05 0
0080  78 b0 33 ba e2 70 ac ce  77 e6 12 1e 6e 0
0090  df 4a 60 03 93 f0 bd 61  b2 86 57 88 15 d
00a0  fd 38 ca 64 8b 18 6e c6  f3 4f df 66 ea a
00b0  e1 8c 55 39 e6 db 2a dd  9a 07 63 4c 8d 4
```

```
▸ Frame 161: 1294 bytes on wire (10352 bits), 1294 bytes capture
▸ Ethernet II, Src: Vmware_e6:f0:de (00:50:56:e6:f0:de), Dst: Vr
▸ Internet Protocol Version 4, Src: 216.58.196.106, Dst: 192.168
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 44268,
```

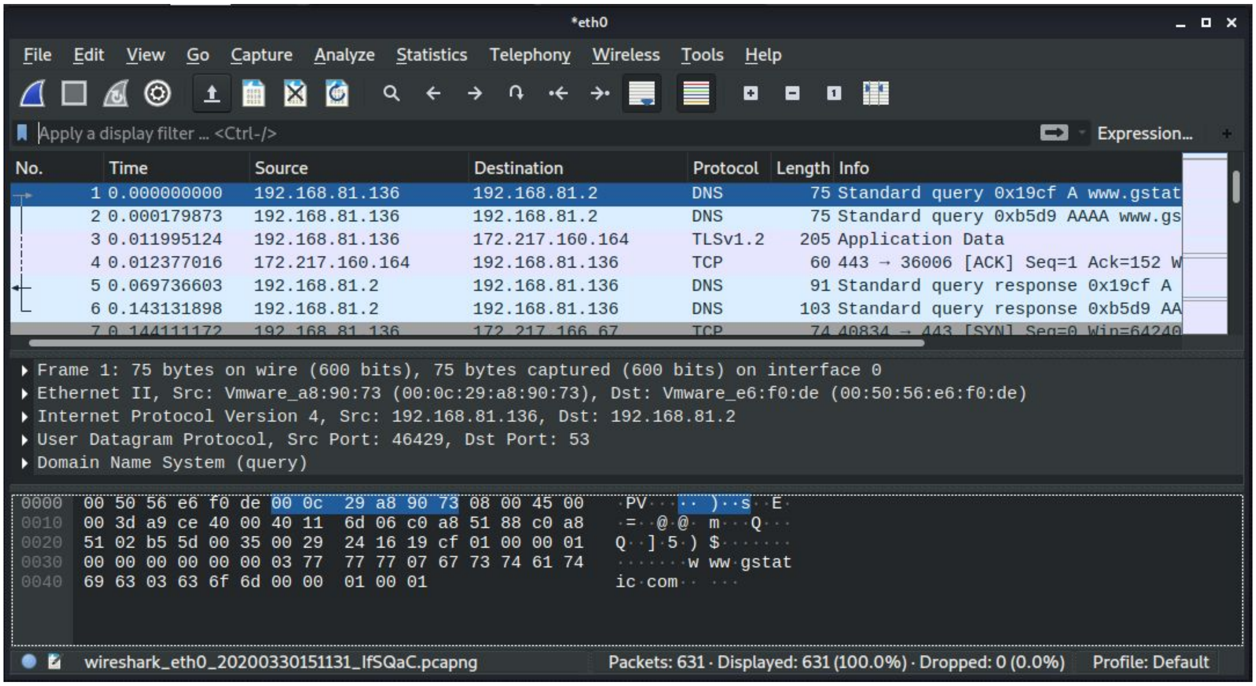
```
0000  00 0c 29 a8 90 73 00 50  56 e6 f0 de 08 00 45 00   ··)·
0010  05 00 00 e9 00 00 80 06  86 39 d8 3a c4 6a c0 a8   ·····
0020  51 88 01 bb ac ec 36 73  5f a1 84 6b e8 43 50 18   Q····
0030  fa f0 eb 3b 00 00 77 63  f5 ea 6a e3 00 8b 49 02   ···;·
0040  63 7e 59 43 76 e3 ca 5d  df a5 37 d5 a6 95 7b 62   c~YCv
0050  ee 3c fd d7 0f cc c4 b4  4e a3 7f c3 cf 50 f1 eb   ·<···
0060  fe 22 0e fe 37 e3 e8 0e  41 b0 eb af ce c8 a2 dd   ·"··7
0070  51 05 08 a2 8b 27 ae 19  4a c3 d1 e4 05 02 46 fc   Q····
0080  78 b0 33 ba e2 70 ac ce  77 e6 12 1e 6e 06 2f 6e   x·3··
0090  df 4a 60 03 93 f0 bd 61  b2 86 57 88 15 d0 e7 b0   ·J`··
```

5. View Packet Data with the Individual Packet Bytes Window



```
0000  00 0c 29 a8 90 73 00 50  56 e6 f0 de 08 00 45 00   ··)··s·P V·····E·
0010  05 00 00 e9 00 00 80 06  86 39 d8 3a c4 6a c0 a8   ········ ·9·:·j··
0020  51 88 01 bb ac ec 36 73  5f a1 84 6b e8 43 50 18   Q·····6s _··k·CP·
0030  fa f0 eb 3b 00 00 77 63  f5 ea 6a e3 00 8b 49 02   ···;··wc ··j···I·
0040  63 7e 59 43 76 e3 ca 5d  df a5 37 d5 a6 95 7b 62   c~YCv··] ··7···{b
0050  ee 3c fd d7 0f cc c4 b4  4e a3 7f c3 cf 50 f1 eb   ·<······ N····P··
0060  fe 22 0e fe 37 e3 e8 0e  41 b0 eb af ce c8 a2 dd   ·"··7··· A·······
0070  51 05 08 a2 8b 27 ae 19  4a c3 d1 e4 05 02 46 fc   Q····'·· J·····F·
0080  78 b0 33 ba e2 70 ac ce  77 e6 12 1e 6e 06 2f 6e   x·3··p·· w···n·/n
0090  df 4a 60 03 93 f0 bd 61  b2 86 57 88 15 d0 e7 b0   ·J`····a ··W·····
```

6. Simply Browsing the Internet


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.   Time          Source            Destination       Protocol  Length Info
    1 0.000000000   192.168.81.136    192.168.81.2      DNS           75 Standard query 0x19cf A www.gstat
    2 0.000179873   192.168.81.136    192.168.81.2      DNS           75 Standard query 0xb5d9 AAAA www.gs
    3 0.011995124   192.168.81.136    172.217.160.164   TLSv1.2      205 Application Data
    4 0.012377016   172.217.160.164   192.168.81.136    TCP           60 443 → 36006 [ACK] Seq=1 Ack=152 W
    5 0.069736603   192.168.81.2      192.168.81.136    DNS           91 Standard query response 0x19cf A
    6 0.143131898   192.168.81.2      192.168.81.136    DNS          103 Standard query response 0xb5d9 AA
    7 0.144111172   192.168.81.136    172.217.166.67    TCP           74 40834 → 443 [SYN] Seq=0 Win=64240
```

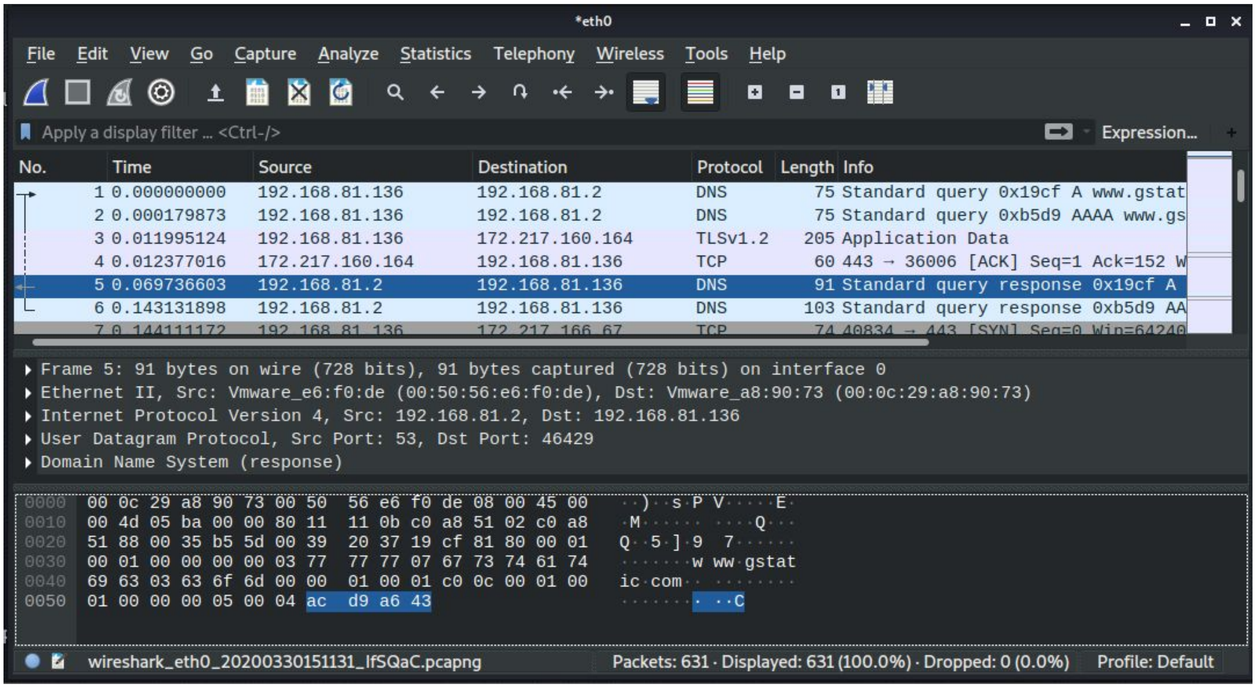
```
▸ Frame 1: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface 0
▸ Ethernet II, Src: Vmware_a8:90:73 (00:0c:29:a8:90:73), Dst: Vmware_e6:f0:de (00:50:56:e6:f0:de)
▸ Internet Protocol Version 4, Src: 192.168.81.136, Dst: 192.168.81.2
▸ User Datagram Protocol, Src Port: 46429, Dst Port: 53
▸ Domain Name System (query)
```

```
0000  00 50 56 e6 f0 de 00 0c  29 a8 90 73 08 00 45 00   ·PV···· ·)··s··E·
0010  00 3d a9 ce 40 00 40 11  6d 06 c0 a8 51 88 c0 a8   ·=··@·@· m···Q···
0020  51 02 b5 5d 00 35 00 29  24 16 19 cf 01 00 00 01   Q··]·5·) $·······
0030  00 00 00 00 00 00 03 77  77 77 07 67 73 74 61 74   ·······w ww·gstat
0040  69 63 03 63 6f 6d 00 00  01 00 01                  ic·com·· ···
```

```
wireshark_eth0_20200330151131_IfSQaC.pcapng    Packets: 631 · Displayed: 631 (100.0%) · Dropped: 0 (0.0%)    Profile: Default
```

Data after browsing internet

7. Viewing the Packet Header Data


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.   Time          Source            Destination       Protocol  Length Info
    1 0.000000000   192.168.81.136    192.168.81.2      DNS           75 Standard query 0x19cf A www.gstat
    2 0.000179873   192.168.81.136    192.168.81.2      DNS           75 Standard query 0xb5d9 AAAA www.gs
    3 0.011995124   192.168.81.136    172.217.160.164   TLSv1.2      205 Application Data
    4 0.012377016   172.217.160.164   192.168.81.136    TCP           60 443 → 36006 [ACK] Seq=1 Ack=152 W
    5 0.069736603   192.168.81.2      192.168.81.136    DNS           91 Standard query response 0x19cf A
    6 0.143131898   192.168.81.2      192.168.81.136    DNS          103 Standard query response 0xb5d9 AA
    7 0.144111172   192.168.81.136    172.217.166.67    TCP           74 40834 → 443 [SYN] Seq=0 Win=64240
```

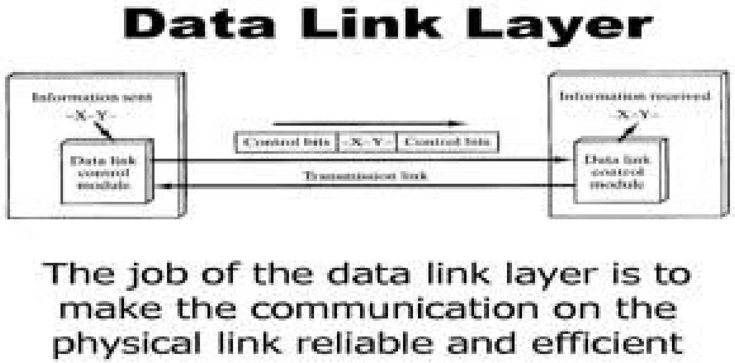
```
▸ Frame 5: 91 bytes on wire (728 bits), 91 bytes captured (728 bits) on interface 0
▸ Ethernet II, Src: Vmware_e6:f0:de (00:50:56:e6:f0:de), Dst: Vmware_a8:90:73 (00:0c:29:a8:90:73)
▸ Internet Protocol Version 4, Src: 192.168.81.2, Dst: 192.168.81.136
▸ User Datagram Protocol, Src Port: 53, Dst Port: 46429
▸ Domain Name System (response)
```

```
0000  00 0c 29 a8 90 73 00 50  56 e6 f0 de 08 00 45 00   ··)··s·P V·····E·
0010  00 4d 05 ba 00 00 80 11  11 0b c0 a8 51 02 c0 a8   ·M······ ····Q···
0020  51 88 00 35 b5 5d 00 39  20 37 19 cf 81 80 00 01   Q··5·]·9  7······
0030  00 01 00 00 00 00 03 77  77 77 07 67 73 74 61 74   ·······w ww·gstat
0040  69 63 03 63 6f 6d 00 00  01 00 01 c0 0c 00 01 00   ic·com·· ········
0050  01 00 00 00 05 00 04 ac  d9 a6 43                  ········ ··C
```

```
wireshark_eth0_20200330151131_IfSQaC.pcapng    Packets: 631 · Displayed: 631 (100.0%) · Dropped: 0 (0.0%)    Profile: Default
```

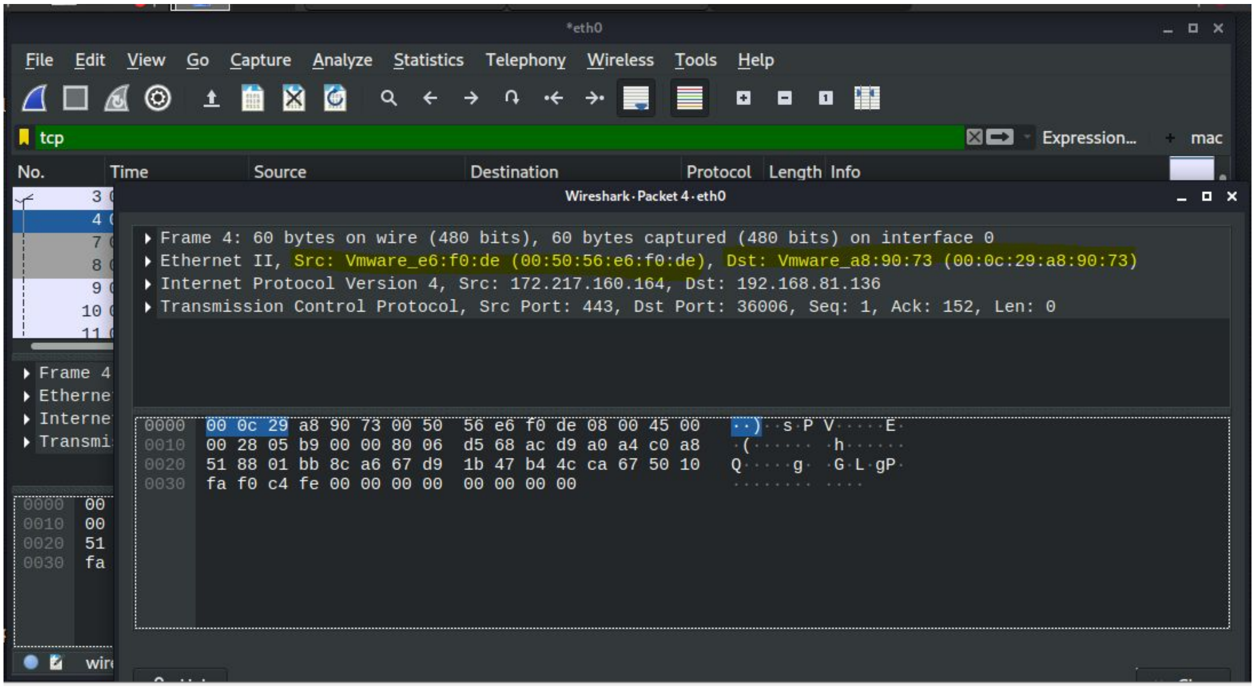
7.1 Capture Packets with Wireshark

7.2. Explore the Network Interface Layer / Data Link Layer



The job of the data link layer is to make the communication on the physical link reliable and efficient

7.2.2. View Ethernet Frame Data Captured with Wireshark



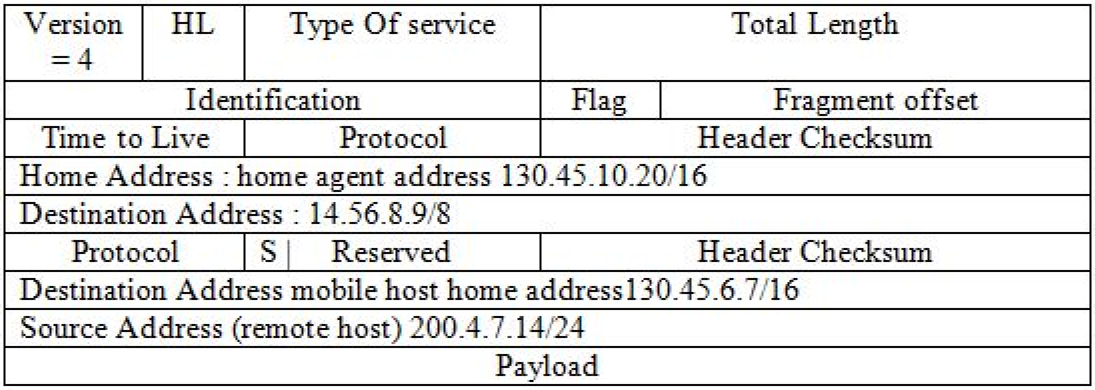
*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp
Expression...
+ mac
No. Time Source Destination Protocol Length Info
Wireshark · Packet 4 · eth0

```
▸ Frame 4: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▸ Ethernet II, Src: Vmware_e6:f0:de (00:50:56:e6:f0:de), Dst: Vmware_a8:90:73 (00:0c:29:a8:90:73)
▸ Internet Protocol Version 4, Src: 172.217.160.164, Dst: 192.168.81.136
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 36006, Seq: 1, Ack: 152, Len: 0
```

```
0000  00 0c 29 a8 90 73 00 50  56 e6 f0 de 08 00 45 00   ··)··s·P V·····E·
0010  00 28 05 b9 00 00 80 06  d5 68 ac d9 a0 a4 c0 a8   ·(······ ·h······
0020  51 88 01 bb 8c a6 67 d9  1b 47 b4 4c ca 67 50 10   Q·····g· ·G·L·gP·
0030  fa f0 c4 fe 00 00 00 00  00 00 00 00               ········ ····
```

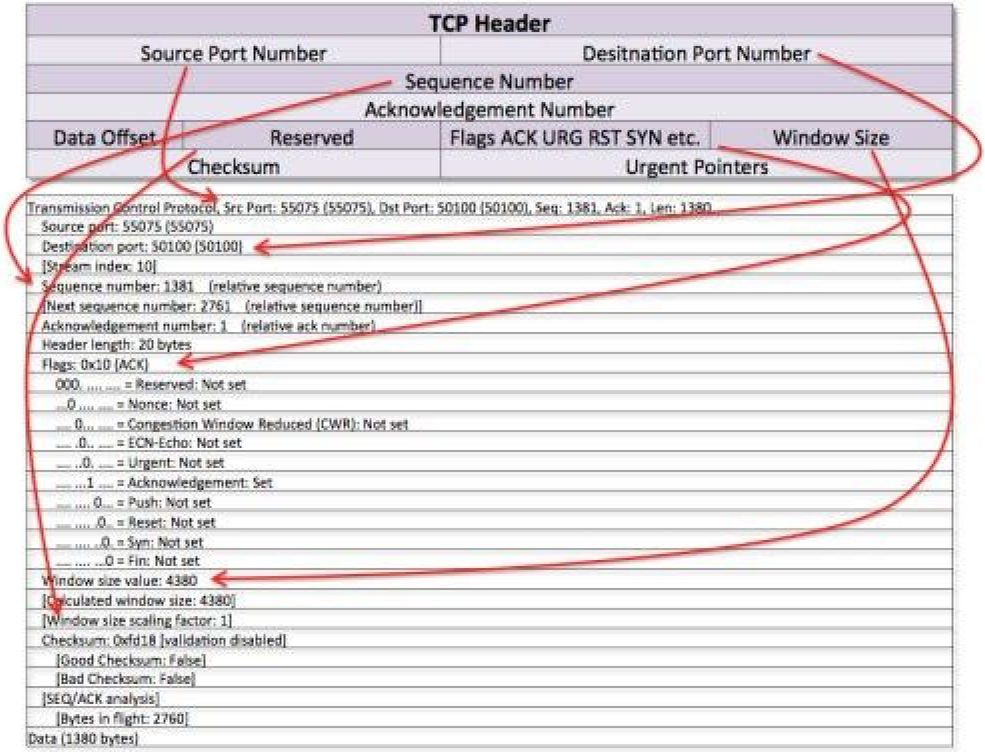
8.1 Exploring the Internet Layer

8.1.1. IPv4 Header: Pictured Below



<table>
<tr><td>Version = 4</td><td>HL</td><td colspan="2">Type Of service</td><td colspan="2">Total Length</td></tr>
<tr><td colspan="4">Identification</td><td>Flag</td><td>Fragment offset</td></tr>
<tr><td colspan="2">Time to Live</td><td colspan="2">Protocol</td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="6">Home Address : home agent address 130.45.10.20/16</td></tr>
<tr><td colspan="6">Destination Address : 14.56.8.9/8</td></tr>
<tr><td colspan="2">Protocol</td><td>S</td><td>Reserved</td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="6">Destination Address mobile host home address130.45.6.7/16</td></tr>
<tr><td colspan="6">Source Address (remote host) 200.4.7.14/24</td></tr>
<tr><td colspan="6">Payload</td></tr>
</table>

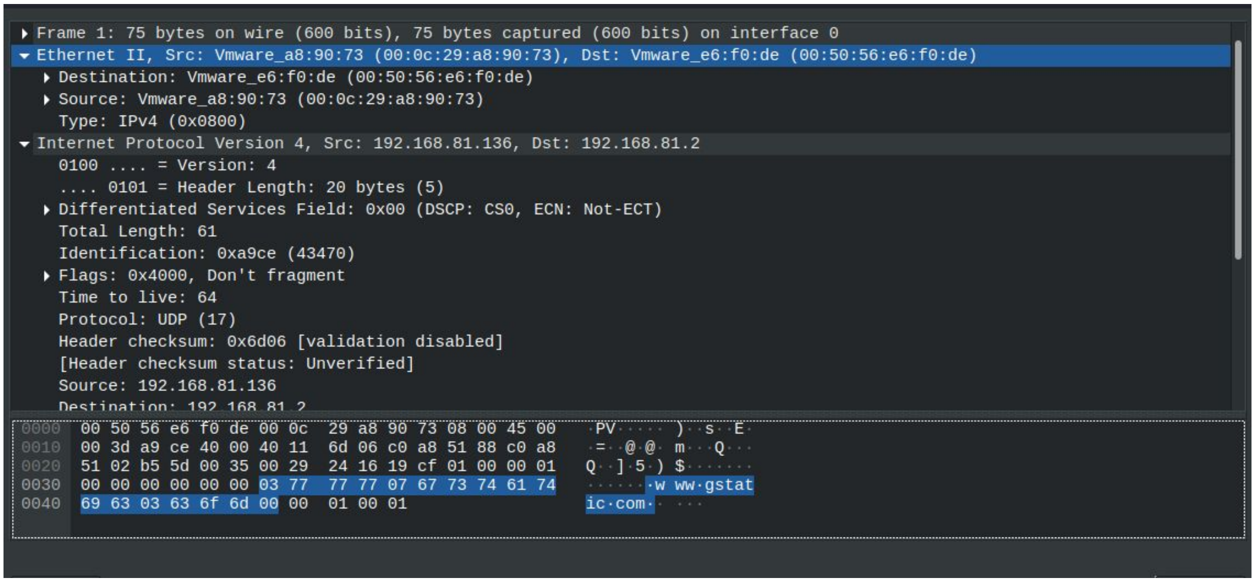
8.1.2. View IP Header Data for a TCP Packet Captured with Wireshark



<table>
<tr><th colspan="4">TCP Header</th></tr>
<tr><td colspan="2">Source Port Number</td><td colspan="2">Desitnation Port Number</td></tr>
<tr><td colspan="4">Sequence Number</td></tr>
<tr><td colspan="4">Acknowledgement Number</td></tr>
<tr><td>Data Offset</td><td>Reserved</td><td>Flags ACK URG RST SYN etc.</td><td>Window Size</td></tr>
<tr><td colspan="2">Checksum</td><td colspan="2">Urgent Pointers</td></tr>
</table>

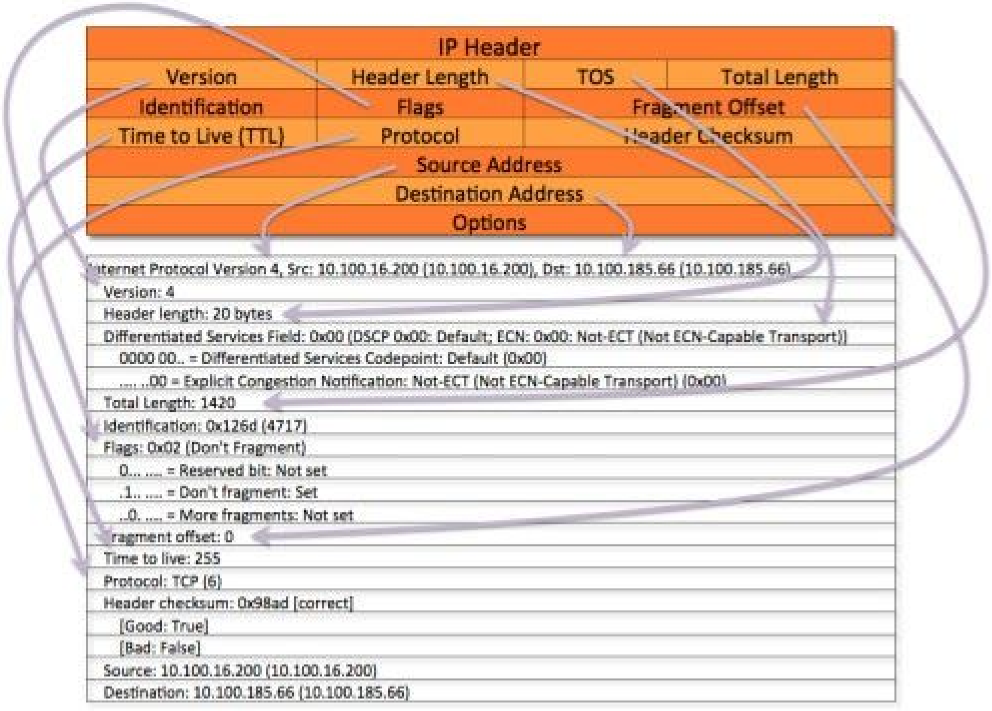
```
Transmission Control Protocol, Src Port: 55075 (55075), Dst Port: 50100 (50100), Seq: 1381, Ack: 1, Len: 1380
   Source port: 55075 (55075)
   Destination port: 50100 (50100)
   [Stream index: 10]
   Sequence number: 1381    (relative sequence number)
   [Next sequence number: 2761    (relative sequence number)]
   Acknowledgement number: 1    (relative ack number)
   Header length: 20 bytes
   Flags: 0x10 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgement: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
   Window size value: 4380
   [Calculated window size: 4380]
   [Window size scaling factor: 1]
   Checksum: 0xfd18 [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
   [SEQ/ACK analysis]
      [Bytes in flight: 2760]
Data (1380 bytes)
```

8.1.3 View IP Header Data for a UDP Packet



```
▸ Frame 1: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface 0
▾ Ethernet II, Src: Vmware_a8:90:73 (00:0c:29:a8:90:73), Dst: Vmware_e6:f0:de (00:50:56:e6:f0:de)
  ▸ Destination: Vmware_e6:f0:de (00:50:56:e6:f0:de)
  ▸ Source: Vmware_a8:90:73 (00:0c:29:a8:90:73)
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 192.168.81.136, Dst: 192.168.81.2
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 61
    Identification: 0xa9ce (43470)
  ▸ Flags: 0x4000, Don't fragment
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x6d06 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.81.136
    Destination: 192.168.81.2

0000  00 50 56 e6 f0 de 00 0c  29 a8 90 73 08 00 45 00   ·PV·····)··s··E·
0010  00 3d a9 ce 40 00 40 11  6d 06 c0 a8 51 88 c0 a8   ·=··@·@·m···Q···
0020  51 02 b5 5d 00 35 00 29  24 16 19 cf 01 00 00 01   Q··]·5·)$·······
0030  00 00 00 00 00 00 03 77  77 77 07 67 73 74 61 74   ·······w ww·gstat
0040  69 63 03 63 6f 6d 00 00  01 00 01                  ic·com·· ···
```

8.1.4. View IP Header Data for an ARP Packet

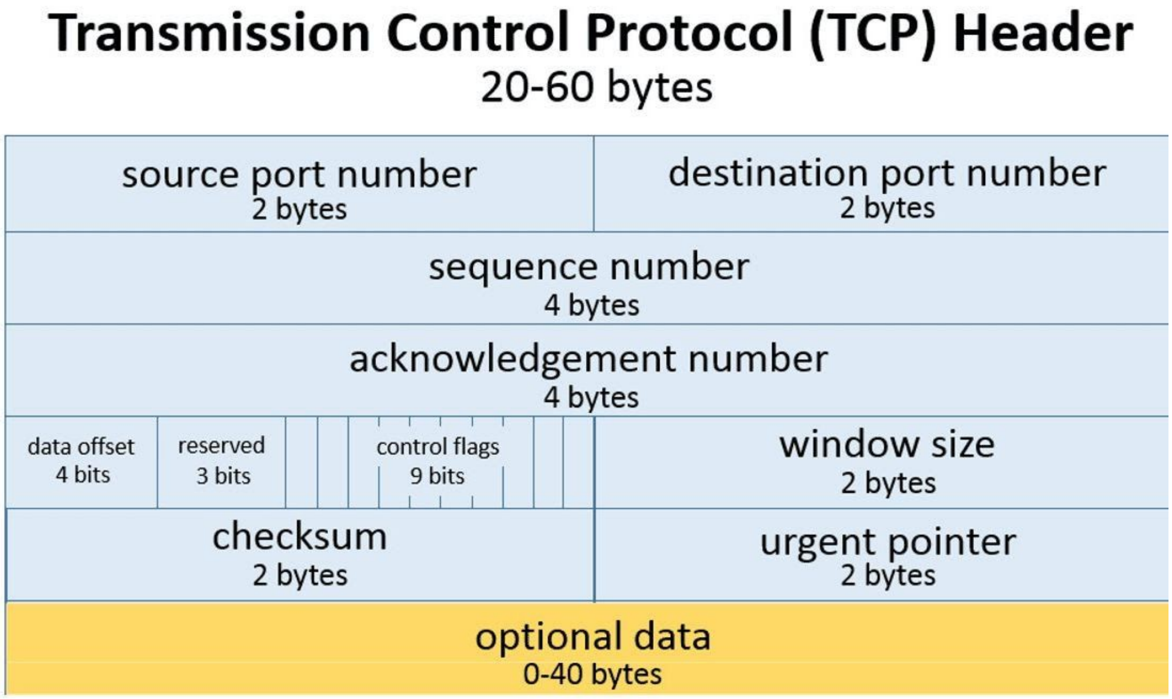


IP Header			
Version	Header Length	TOS	Total Length
Identification	Flags	Fragment Offset	
Time to Live (TTL)	Protocol	Header Checksum	
Source Address			
Destination Address			
Options			

```
Internet Protocol Version 4, Src: 10.100.16.200 (10.100.16.200), Dst: 10.100.185.66 (10.100.185.66)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    0000 00.. = Differentiated Services Codepoint: Default (0x00)
    .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport) (0x00)
  Total Length: 1420
  Identification: 0x126d (4717)
  Flags: 0x02 (Don't Fragment)
    0... .... = Reserved bit: Not set
    .1.. .... = Don't fragment: Set
    ..0. .... = More fragments: Not set
  Fragment offset: 0
  Time to live: 255
  Protocol: TCP (6)
  Header checksum: 0x98ad [correct]
    [Good: True]
    [Bad: False]
  Source: 10.100.16.200 (10.100.16.200)
  Destination: 10.100.185.66 (10.100.185.66)
```

9 Exploring the Transport Layer

9.1.1. TCP Header: Pictured Below



Transmission Control Protocol (TCP) Header
20-60 bytes

source port number
2 bytes

destination port number
2 bytes

sequence number
4 bytes

acknowledgement number
4 bytes

data offset
4 bits

reserved
3 bits

control flags
9 bits

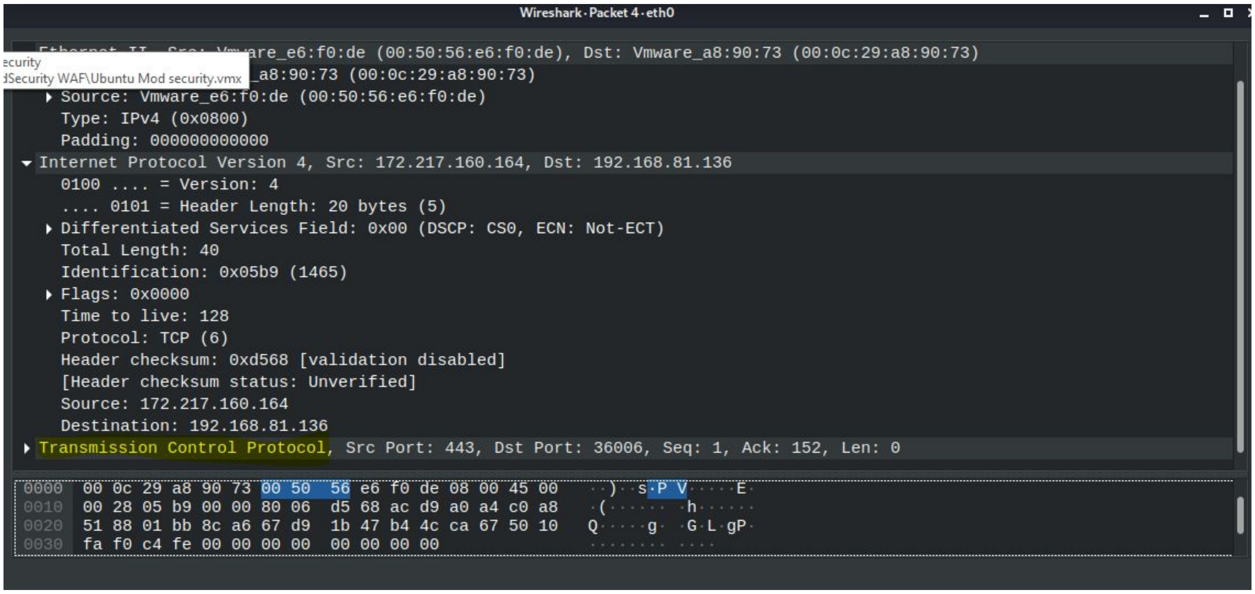
window size
2 bytes

checksum
2 bytes

urgent pointer
2 bytes

optional data
0-40 bytes

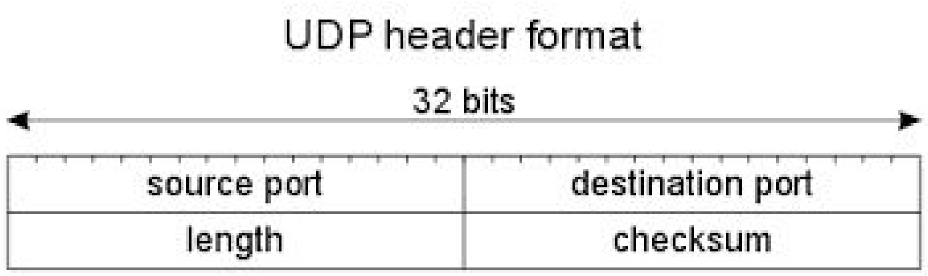
9.1.2 View TCP Header Data for a TCP Packet Captured with Wireshark



Wireshark · Packet 4 · eth0
ecurity
dSecurity WAF\Ubuntu Mod security.vmx

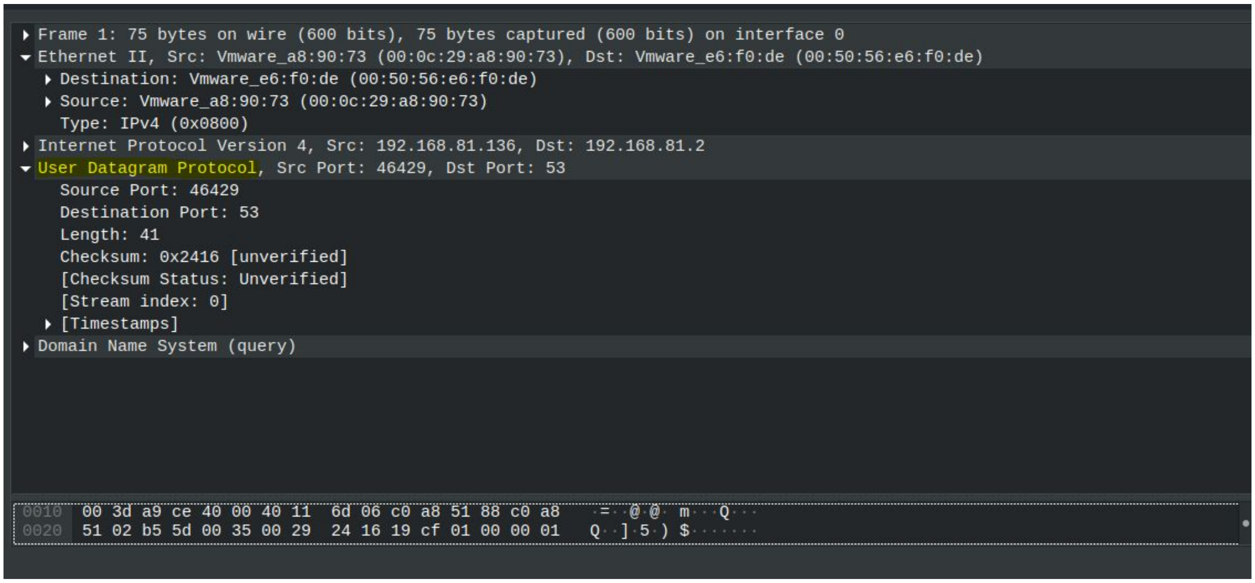
```
  Ethernet II, Src: Vmware_e6:f0:de (00:50:56:e6:f0:de), Dst: Vmware_a8:90:73 (00:0c:29:a8:90:73)
                  _a8:90:73 (00:0c:29:a8:90:73)
  ▸ Source: Vmware_e6:f0:de (00:50:56:e6:f0:de)
    Type: IPv4 (0x0800)
    Padding: 000000000000
▾ Internet Protocol Version 4, Src: 172.217.160.164, Dst: 192.168.81.136
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x05b9 (1465)
  ▸ Flags: 0x0000
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0xd568 [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.217.160.164
    Destination: 192.168.81.136
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 36006, Seq: 1, Ack: 152, Len: 0

0000  00 0c 29 a8 90 73 00 50  56 e6 f0 de 08 00 45 00   ··)··s·P V·····E·
0010  00 28 05 b9 00 00 80 06  d5 68 ac d9 a0 a4 c0 a8   ·(······ ·h······
0020  51 88 01 bb 8c a6 67 d9  1b 47 b4 4c ca 67 50 10   Q·····g· ·G·L·gP·
0030  fa f0 c4 fe 00 00 00 00  00 00 00 00               ········ ····
```

9.1.3 UDP Header: Pictured Below



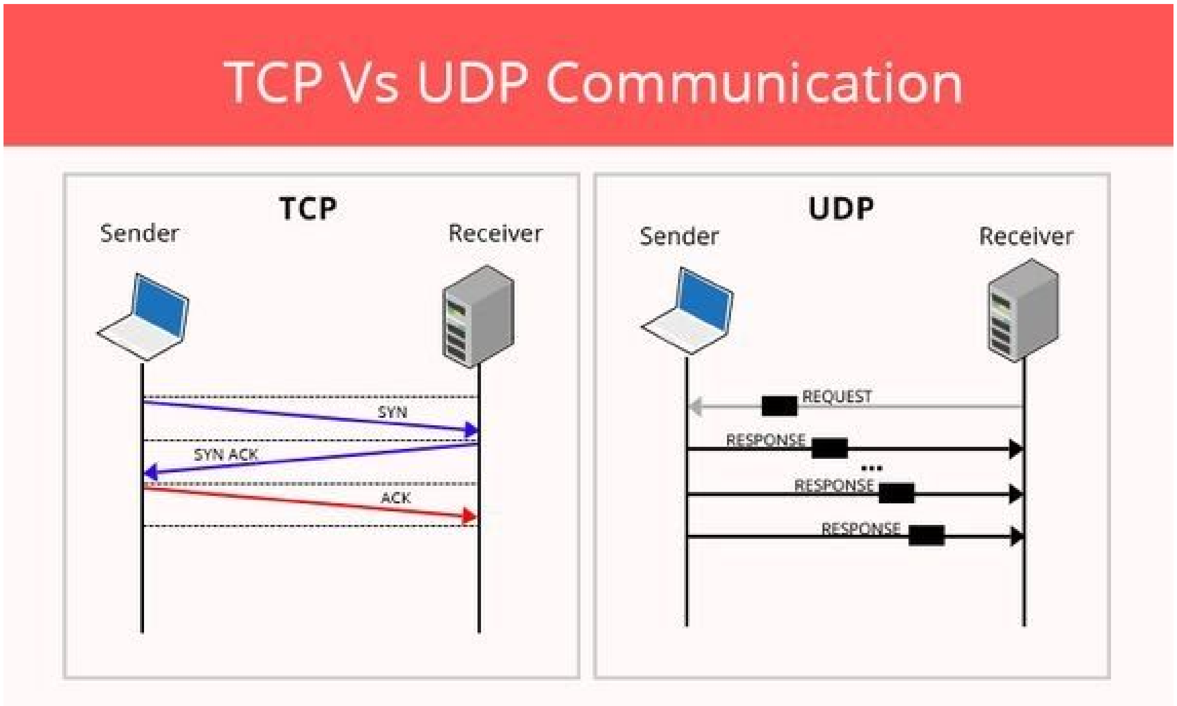
source port	destination port
length	checksum

9.1.4 View UDP Header Data for a UDP Packet Captured with Wireshark



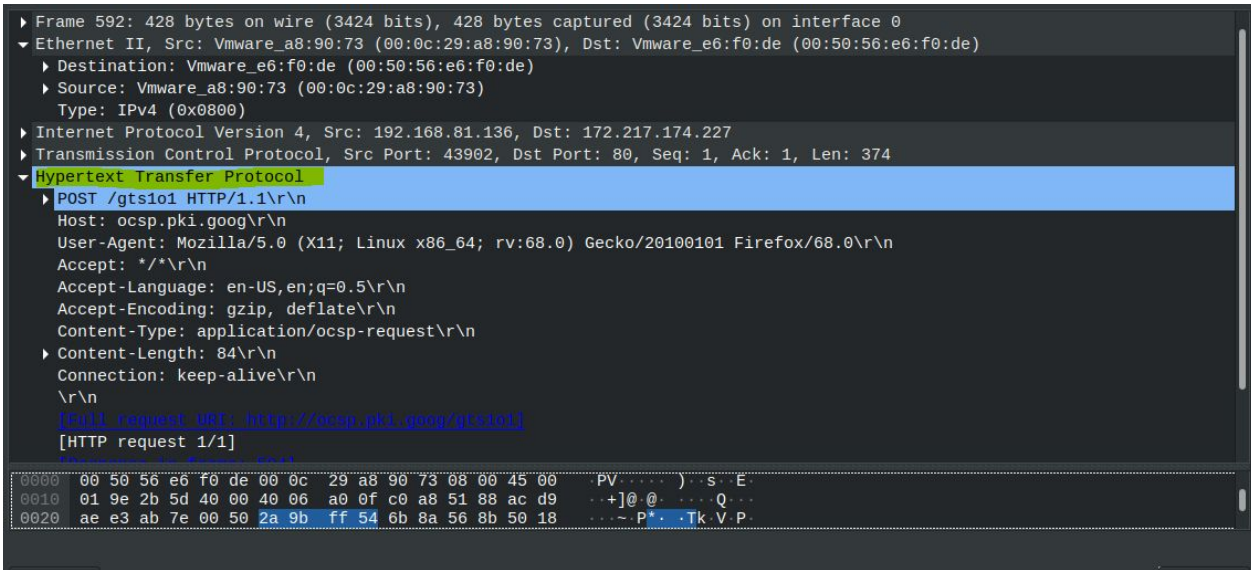
```
▸ Frame 1: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface 0
▾ Ethernet II, Src: Vmware_a8:90:73 (00:0c:29:a8:90:73), Dst: Vmware_e6:f0:de (00:50:56:e6:f0:de)
  ▸ Destination: Vmware_e6:f0:de (00:50:56:e6:f0:de)
  ▸ Source: Vmware_a8:90:73 (00:0c:29:a8:90:73)
    Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 192.168.81.136, Dst: 192.168.81.2
▾ User Datagram Protocol, Src Port: 46429, Dst Port: 53
    Source Port: 46429
    Destination Port: 53
    Length: 41
    Checksum: 0x2416 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  ▸ [Timestamps]
▸ Domain Name System (query)

0010  00 3d a9 ce 40 00 40 11  6d 06 c0 a8 51 88 c0 a8   ·=··@·@· m···Q···
0020  51 02 b5 5d 00 35 00 29  24 16 19 cf 01 00 00 01   Q··]·5·) $·······
```

9.1.5 Compare and Contrast IP, TCP, and UDP



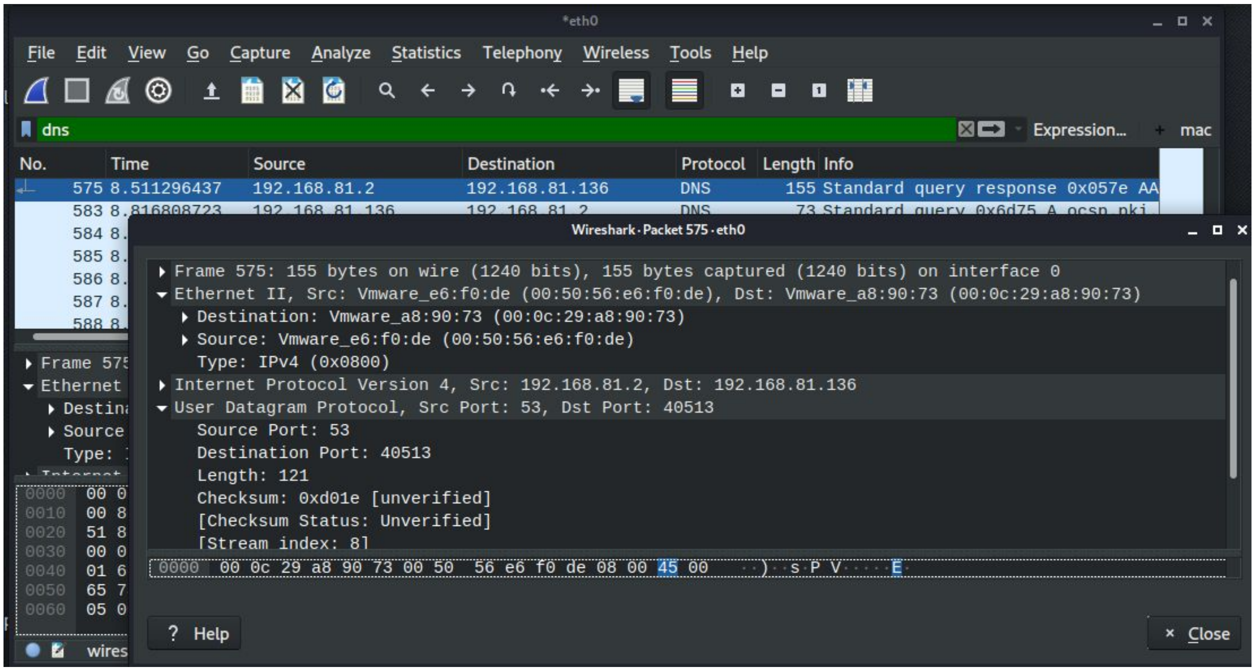
10. Explore the Application Layer

10.1.1 Analyze an HTTP Packet



```
▸ Frame 592: 428 bytes on wire (3424 bits), 428 bytes captured (3424 bits) on interface 0
▾ Ethernet II, Src: Vmware_a8:90:73 (00:0c:29:a8:90:73), Dst: Vmware_e6:f0:de (00:50:56:e6:f0:de)
  ▸ Destination: Vmware_e6:f0:de (00:50:56:e6:f0:de)
  ▸ Source: Vmware_a8:90:73 (00:0c:29:a8:90:73)
    Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 192.168.81.136, Dst: 172.217.174.227
▸ Transmission Control Protocol, Src Port: 43902, Dst Port: 80, Seq: 1, Ack: 1, Len: 374
▾ Hypertext Transfer Protocol
  ▸ POST /gts1o1 HTTP/1.1\r\n
    Host: ocsp.pki.goog\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0\r\n
    Accept: */*\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Content-Type: application/ocsp-request\r\n
  ▸ Content-Length: 84\r\n
    Connection: keep-alive\r\n
    \r\n
    [Full request URI: http://ocsp.pki.goog/gts1o1]
    [HTTP request 1/1]

0000  00 50 56 e6 f0 de 00 0c  29 a8 90 73 08 00 45 00   ·PV·····)··s··E·
0010  01 9e 2b 5d 40 00 40 06  a0 0f c0 a8 51 88 ac d9   ··+]@·@·····Q···
0020  ae e3 ab 7e 00 50 2a 9b  ff 54 6b 8a 56 8b 50 18   ···~·P*···Tk·V·P·
```

10.1.2 Analyze a DNS Packet



*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression…
+ mac

```
dns

No.   Time         Source          Destination     Protocol  Length Info
  575 8.511296437  192.168.81.2    192.168.81.136  DNS          155 Standard query response 0x057e AA
  583 8.816808723  192.168.81.136  192.168.81.2    DNS           73 Standard query 0x6d75 A ocsp.pki
  584 8.
  585 8.
  586 8.
  587 8.
  588 8.
```

Wireshark · Packet 575 · eth0

```
▸ Frame 575: 155 bytes on wire (1240 bits), 155 bytes captured (1240 bits) on interface 0
▾ Ethernet II, Src: Vmware_e6:f0:de (00:50:56:e6:f0:de), Dst: Vmware_a8:90:73 (00:0c:29:a8:90:73)
  ▸ Destination: Vmware_a8:90:73 (00:0c:29:a8:90:73)
  ▸ Source: Vmware_e6:f0:de (00:50:56:e6:f0:de)
    Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 192.168.81.2, Dst: 192.168.81.136
▾ User Datagram Protocol, Src Port: 53, Dst Port: 40513
    Source Port: 53
    Destination Port: 40513
    Length: 121
    Checksum: 0xd01e [unverified]
    [Checksum Status: Unverified]
    [Stream index: 8]

0000  00 0c 29 a8 90 73 00 50  56 e6 f0 de 08 00 45 00   ··)··s·P V·····E·
```

11. Common Questions in mind

Que. 1. Does Wireshark capture all the traffic on the Internet? If so, explain why. If not, which traffic does it capture?
Ans. In all likelihood, it will only see traffic your machine is participating in, or which is broadcast to all machines.

The reason for this is that for years, most LANs have been built based on switched Ethernet technology, as opposed to hub-based Ethernet or bus-based networking. In those older technologies, every machine on the LAN saw all traffic, purely because they were all electrically connected to each other. With switched Ethernet, the switch makes decisions about which packets to send to which ports. This makes the network faster and slightly more secure.

(Switched Ethernet isn't a very good security measure, because it's easy to defeat with ARP poisoning.)

Now, maybe it is possible you are still on a hub-based Ethernet, or similar. That can only be the case with 100 Mbit/s and slower networks. Part of the Gigabit Ethernet spec is a requirement for switches. You won't find a GigE hub.

I should also note that wireless networking effectively behaves like LANs of old: every machine connected to a given Wi-Fi network can see all traffic, purely due to the nature of radio communication.

If you are on a wired LAN with managed switches and you have administrative access to those switches, you will probably find a feature you can enable in them called port mirroring. That feature exists specifically to restore the older pre-switched LAN behavior: it designates one port as special, directing copies of all traffic to it, even packets not aimed at MAC addresses connected to that port.

Que. 2. Write Wireshark filters to: View UDP traffic when scan is performed.
Ans. simply type UDP and hit enter, and you will be able to see all the udp packets that were captured.

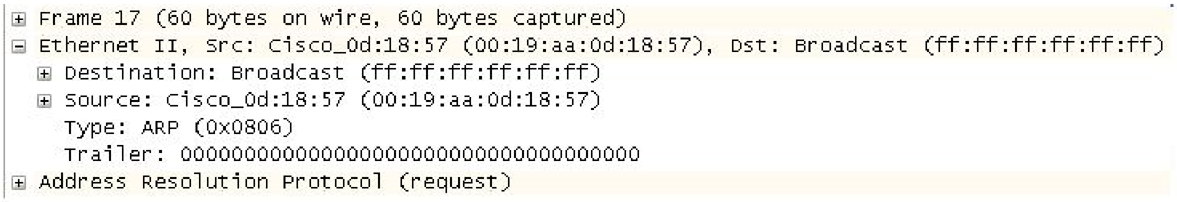
Que. 3. View ICMP traffic from any address.
Ans. To analyze ICMP Echo Request traffic:

1. Observe the traffic captured in the top Wireshark packet list pane. Look for traffic with ICMP listed as the protocol. To view only ICMP traffic, type **icmp** (lower case) in the Filter box and press **Enter.**
2. Select the first ICMP packet, labeled **Echo (ping) request**.
3. Observe the packet details in the middle Wireshark packet details pane. Notice that it is an Ethernet II / Internet Protocol Version 4 / Internet Control Message Protocol frame.

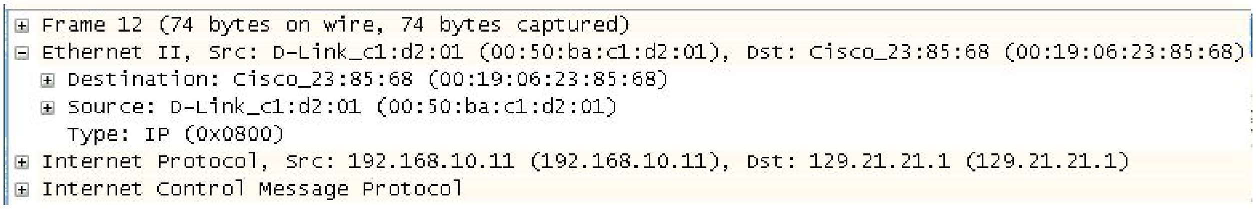
4. Expand Internet Control Message Protocol to view ICMP details.
5. Observe the Type. Notice that the type is 8 (Echo (ping) request).
6. Select Data in the middle Wireshark packet details pane to highlight the data portion of the frame.
7. Observe the packet contents in the bottom Wireshark packet bytes pane. Notice that Windows sends an alphabet sequence during ping requests.

Que. 4. Why do ARP packets not have IP headers?
Ans. While there are IP or protocol addresses used in this message, it does not actually have an IP header. The IP addresses seen are simply part of the ARP header. This means that ARP messages are not routable and that routers will not pass ARP traffic on to another network. Consequently, the MAC address of a node not on the source node's LAN cannot be determined.

It also means that the Ethertype in an Ethernet frame carrying an ARP message is different than in standard data traffic. This difference is shown below



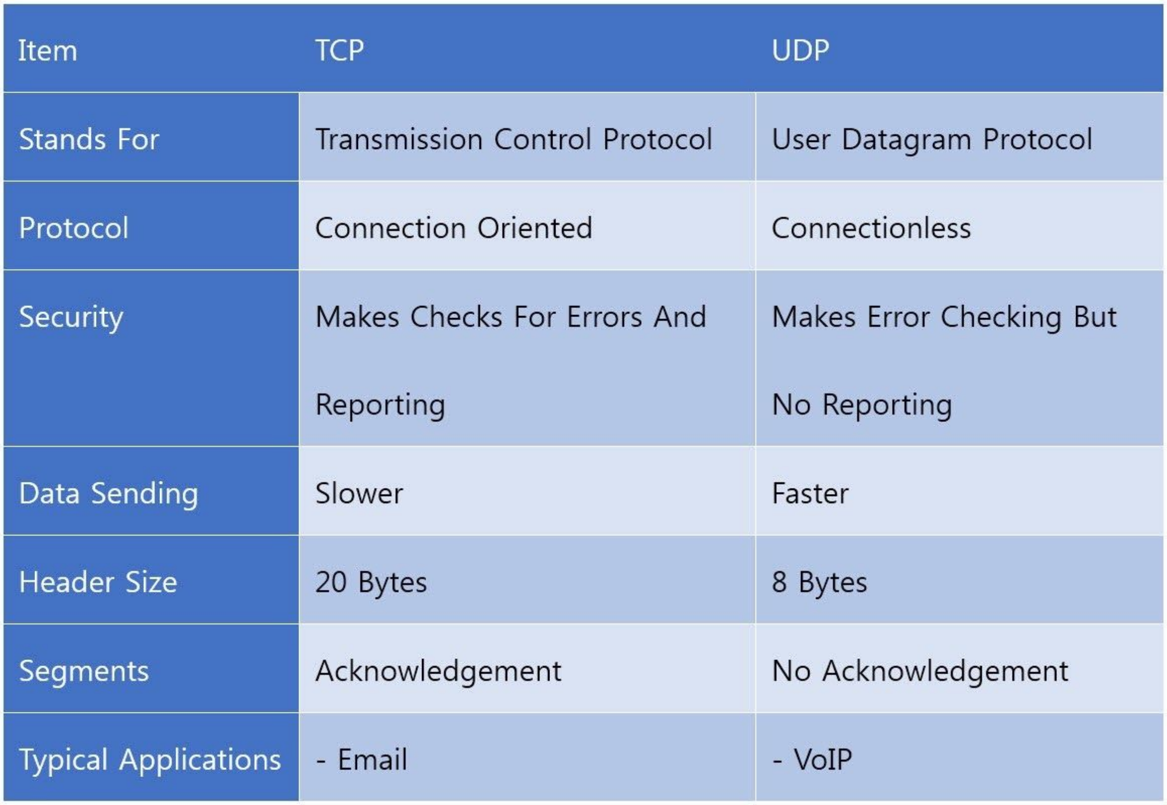
```
⊞ Frame 17 (60 bytes on wire, 60 bytes captured)
⊟ Ethernet II, Src: Cisco_0d:18:57 (00:19:aa:0d:18:57), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ⊞ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ⊞ Source: Cisco_0d:18:57 (00:19:aa:0d:18:57)
    Type: ARP (0x0806)
    Trailer: 000000000000000000000000000000000000
⊞ Address Resolution Protocol (request)
```



```
⊞ Frame 12 (74 bytes on wire, 74 bytes captured)
⊟ Ethernet II, Src: D-Link_c1:d2:01 (00:50:ba:c1:d2:01), Dst: Cisco_23:85:68 (00:19:06:23:85:68)
  ⊞ Destination: Cisco_23:85:68 (00:19:06:23:85:68)
  ⊞ Source: D-Link_c1:d2:01 (00:50:ba:c1:d2:01)
    Type: IP (0x0800)
⊞ Internet Protocol, Src: 192.168.10.11 (192.168.10.11), Dst: 129.21.21.1 (129.21.21.1)
⊞ Internet Control Message Protocol
```

Que. 5. Compare and contrast UDP and TCP headers.

Ans.



Item	TCP	UDP
Stands For	Transmission Control Protocol	User Datagram Protocol
Protocol	Connection Oriented	Connectionless
Security	Makes Checks For Errors And Reporting	Makes Error Checking But No Reporting
Data Sending	Slower	Faster
Header Size	20 Bytes	8 Bytes
Segments	Acknowledgement	No Acknowledgement
Typical Applications	- Email	- VoIP

Que. 6. Do ICMP packets specify a port? Look online and explain why or why not.
Ans. **ICMP** is a protocol that is designed specifically for diagnostic purposes and **ping** is nothing but an ICMP echo request and echo reply that's why there is no concept of **port** numbers in **ICMP**. **Port** numbers are transport-layer addresses used by some transport protocols.