Experiment No. 5

Aim: Use Wire shark to understand the operation of TCP/IP layers:

- Ethernet Layer: Frame header, Frame size etc.
- Data Link Layer: MAC address, ARP (IP and MAC address binding)
- Network Layer: IP Packet (header, fragmentation), ICMP (Query and Echo)
- Transport Layer: TCP Ports, TCP handshake segments etc.
- Application Layer: DHCP, FTP, HTTP header formats

Resource required: Computer or Laptop, Wireshark Installed, Access to a local network where you can capture packets

Theory:

Wireshark : it's a secret agent used to spy, understanding the things working behind used by hacker and new learner used to checks how internet works

It's a detective tool for computer networks- to check what is happening in the network troubleshoot issues or to check how the network protocols work. whether data is traveling from any ethernet, Bluetooth, wifi or any networks

Networking protocols:

Networking protocols are formal rules and conventions that govern how data is transmitted and received across networks. They ensure that devices can communicate effectively, securely, and reliably.

🔗 1. Communication Protocols

These define how data is transferred between devices.

Protocol	Description
TCP (Transmission Control Protocol)	Connection-oriented; ensures reliable data transfer.

Protocol	Description
UDP (User Datagram Protocol)	Connectionless; faster but less reliable than TCP.
IP (Internet Protocol)	Routes packets between devices across networks (IP addresses).
ICMP (Internet Control Message Protocol)	Sends error messages (e.g., "host unreachable", "ping").

🌐 2. Internet & Web Protocols

Protocol	Description
HTTP (Hypertext Transfer Protocol)	Used for web communication (insecure).
HTTPS (HTTP Secure)	Secure version of HTTP using SSL/TLS.
FTP (File Transfer Protocol)	Transfers files between systems (can be insecure).
SFTP (SSH File Transfer Protocol)	Secure file transfer over SSH.
DNS (Domain Name System)	Resolves domain names to IP addresses.

📡 3. Network Management Protocols

Protocol	Description
DHCP (Dynamic Host Configuration Protocol)	Automatically assigns IP addresses to devices.
SNMP (Simple Network Management Protocol)	Monitors and manages network devices.
NTP (Network Time Protocol)	Synchronizes time across devices.

🔒 4. Security Protocols

Protocol	Description
SSL/TLS (Secure Sockets Layer / Transport Layer Security)	Encrypts data over a network (used in HTTPS, email, etc.).
SSH (Secure Shell)	Secure remote login and command execution.

🧭 5. Routing Protocols

Protocol	Description
RIP (Routing Information Protocol)	Basic distance-vector routing protocol.
OSPF (Open Shortest Path First)	Link-state routing; more efficient than RIP.
BGP (Border Gateway Protocol)	Core routing protocol of the internet.

🧰 Common Protocol Stack Example (TCP/IP Model):

Layer	Example Protocols
Application	HTTP, FTP, DNS, SMTP
Transport	TCP, UDP
Internet	IP, ICMP
Link	Ethernet, Wi-Fi

Practical: Using Wireshark to Understand TCP/IP Layers

Preparation:

- Install Wireshark on your PC/laptop.
- Connect to an active network (LAN or Wi-Fi).
- Make sure you have permission to capture network traffic on this network.

The Ethernet layer is part of the Data Link Layer (Layer 2) of the OSI model. It defines how data is formatted into frames, how devices are addressed (using MAC addresses), and how error detection is performed for communication within a Local Area Network (LAN).

Ethernet is the most widely used LAN technology today.

◇ Functions of Ethernet Layer

1. Framing
    - Breaks network-layer packets into manageable units called Ethernet frames.
    - Adds headers (source MAC, destination MAC) and trailers (CRC error checking).
2. Addressing
    - Uses MAC (Media Access Control) addresses to identify devices on a LAN.
    - Example: 00:1A:2B:3C:4D:5E.
3. Error Detection
    - Uses CRC (Cyclic Redundancy Check) in the frame trailer to detect errors.
4. Access Control
    - Uses CSMA/CD (Carrier Sense Multiple Access with Collision Detection) in traditional Ethernet to handle multiple devices on the same medium.
    - Modern Ethernet (switch-based) reduces collisions.

☑ How to Use Wireshark to Analyze the Ethernet Layer (Layer 2)

📌 Step 1: Capture Traffic

1. Open Wireshark.
2. Start capturing on the appropriate network interface (e.g., Wi-Fi or Ethernet).

3. Perform some network activity (like opening a website).
4. Stop the capture after collecting some packets.

📌 Step 2: Filter for Ethernet Packets

Ethernet is the base layer for most packets, so you don't need a specific filter to see them. But to narrow down:

- Use a filter like eth or just click on any packet.

📌 Step 3: Examine the Ethernet Frame

Click on a packet (e.g., an HTTP or TCP packet), and expand the first section labeled:

Frame X: <number of bytes on wire>, <number of bytes captured>

This tells you:

- Frame size on the wire (in bytes): The actual size as transmitted.
- Captured size: May be smaller if the capture was truncated.

Then expand the section:

Ethernet II

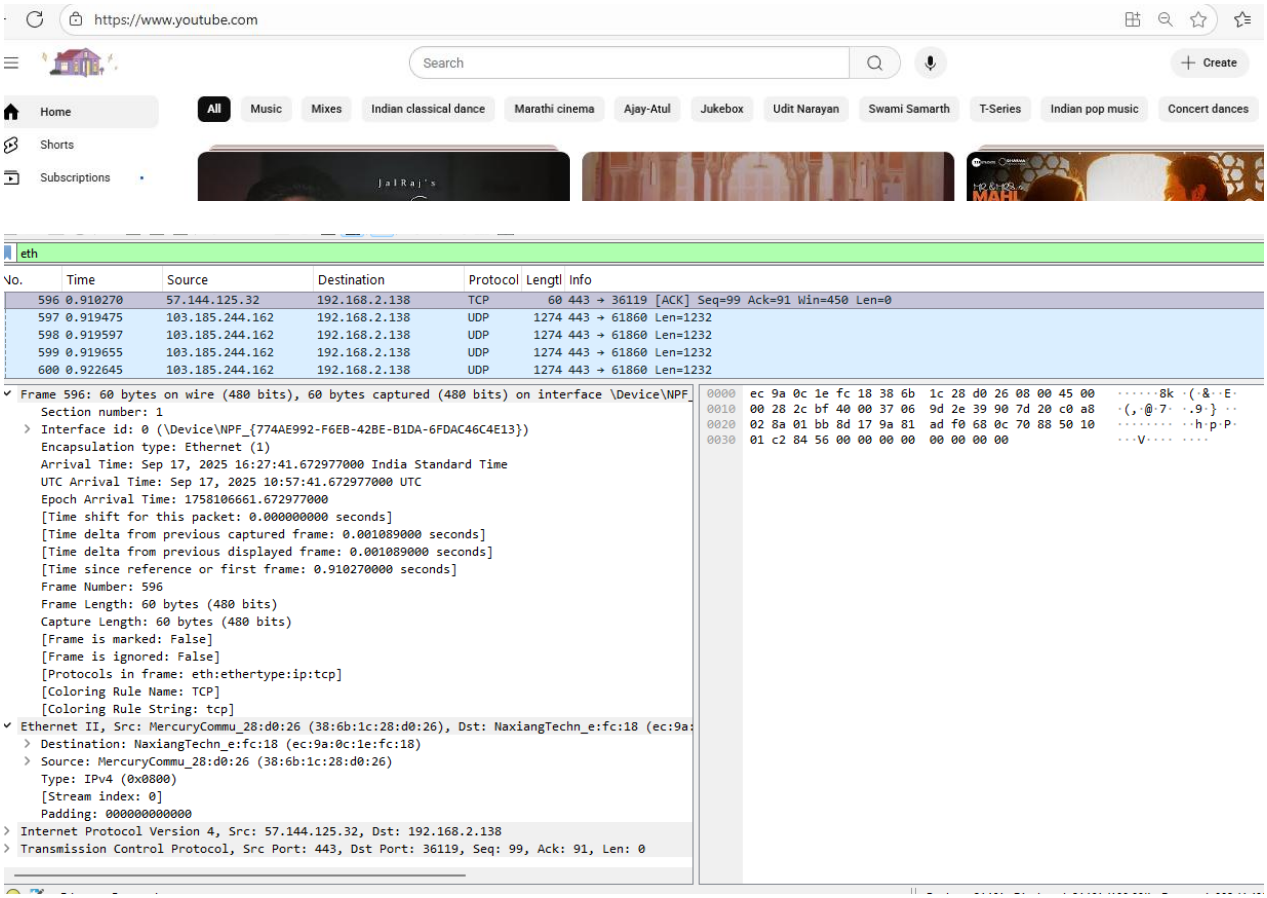
You'll see:

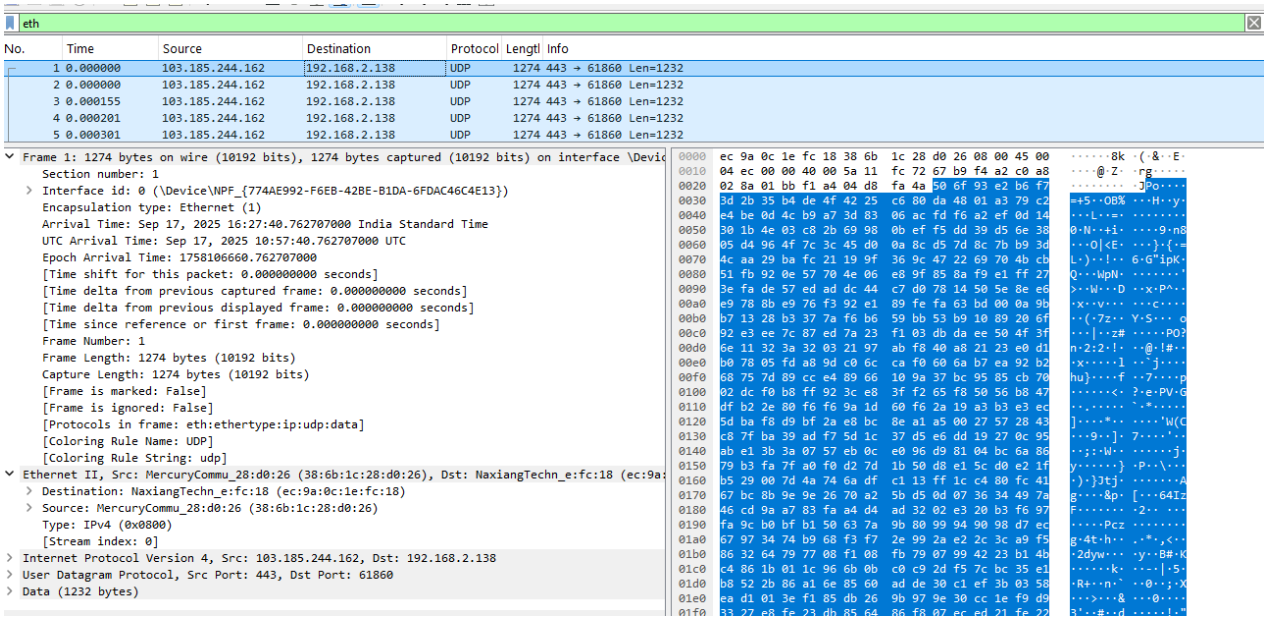
Field	Description
Destination MAC address	MAC address of the device the frame is being sent to
Source MAC address	MAC address of the sender
Type	Ethernet Type (e.g., 0x0800 for IPv4)

📌 Step 4: Important Ethernet Fields

Field	Purpose
Preamble	(Not shown in Wireshark) 7-byte sync pattern + 1-byte Start Frame Delimiter (SFD)
Destination MAC	Identifies the receiver

Field	Purpose
Source MAC	Identifies the sender
Type / Length	Indicates which protocol is encapsulated (IP, ARP, etc.)
Payload	The actual data (e.g., IP packet)
FCS (Frame Check Sequence)	Error detection (not usually captured by Wireshark)





🔍 Step-by-Step Wireshark Analysis for Data Link Layer

📶 1. Start a Packet Capture

1. Open Wireshark.
2. Choose the correct network interface (e.g., Ethernet, Wi-Fi).
3. Click Start Capture (the shark fin icon).
4. Perform some network activity (e.g., open a website or ping a local IP) to generate packets.

🧱 2. Data Link Layer – MAC Addresses

- The Data Link Layer is Layer 2 of the OSI model and is responsible for MAC addressing and frame delivery on a local network.

Wireshark is a packet analyzer that lets you capture and inspect frames traveling on a network.
At the Data Link Layer (Layer 2), you'll mostly be looking at Ethernet frames (on wired LANs) or 802.11 frames (on Wi-Fi).

In the Packet Details Pane, you'll see multiple expandable layers:

- Frame → Overall capture info (arrival time, interface).
- Ethernet II (Data Link Layer) → Contains MAC addresses and EtherType.

- Internet Protocol (IP) → Network layer info.
- TCP/UDP → Transport layer info.

🎯 Filter to See MAC Address Information

Use this filter:

eth

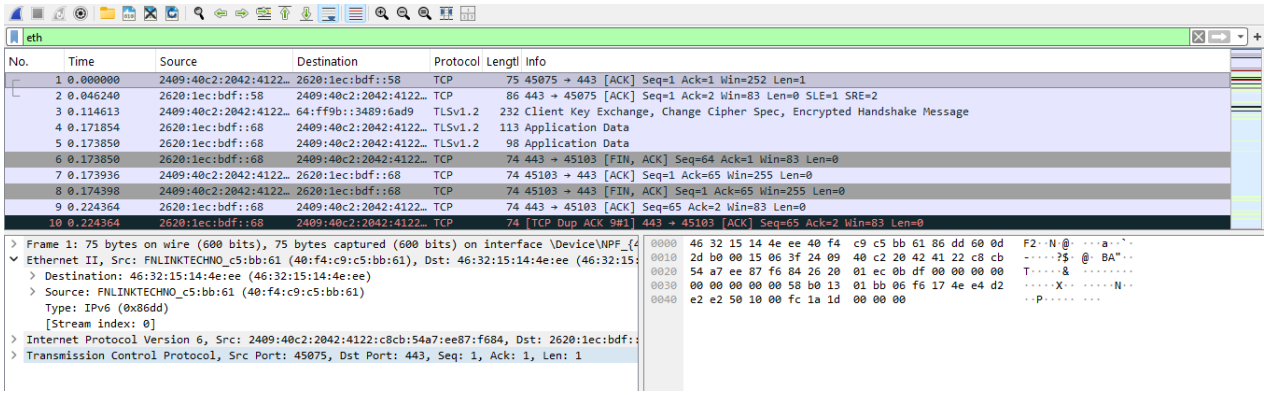
🔍 How to View MAC Addresses:

1. Click on a packet in the capture (e.g., an ARP or IP packet).
2. Expand the Ethernet II section.
3. You will see:
    - Source: MAC address of the sender.
    - Destination: MAC address of the receiver.
    - Example:
    - Ethernet II
    - Destination: 00:1a:2b:3c:4d:5e
    - Source: 5e:4d:3c:2b:1a:00
    - Type: IPv4 (0x0800)

```
C:\Users\janu>ping www.youtube.com

Pinging youtube-ui.l.google.com [142.250.70.110] with 32 bytes of data:
Reply from 142.250.70.110: bytes=32 time=14ms TTL=116
Reply from 142.250.70.110: bytes=32 time=17ms TTL=116
Reply from 142.250.70.110: bytes=32 time=17ms TTL=116
Reply from 142.250.70.110: bytes=32 time=14ms TTL=116

Ping statistics for 142.250.70.110:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 14ms, Maximum = 17ms, Average = 15ms
```



🔗 3. ARP – IP to MAC Binding

ARP (Address Resolution Protocol) maps an IP address to a MAC address within a local network.

🎯 Filter ARP Packets:

Use this filter:

arp

🧠 What You'll See:

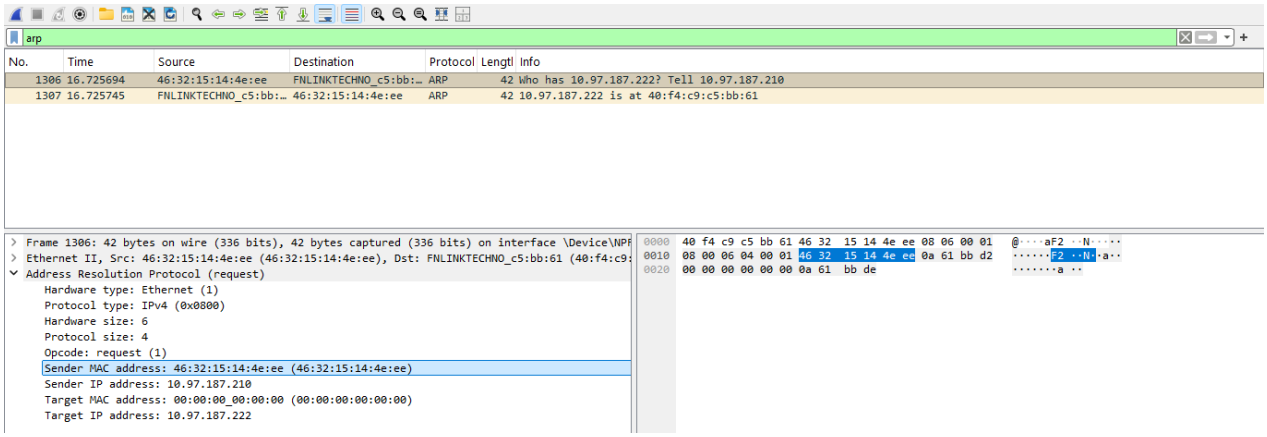
- Who has 192.168.1.1? Tell 192.168.1.2
    - This is a request from one host asking for the MAC address associated with an IP.
- 192.168.1.1 is at 00:1a:2b:3c:4d:5e
    - This is the reply showing the IP-to-MAC mapping.

🔍 Expand the ARP Section

In an ARP packet:

- Sender MAC address
- Sender IP address
- Target MAC address
- Target IP address

This allows you to clearly see the binding between IP and MAC addresses.



🌐 NETWORK LAYER (Layer 3)

At the Network Layer, the main components you're going to observe are:

- IP Packets (IPv4 or IPv6)
- Fragmentation
- ICMP (Internet Control Message Protocol) — for diagnostics like ping/traceroute

Let's go step-by-step:

🧱 1. IP Packets – Header and Fragmentation

🎯 Filter:

ip

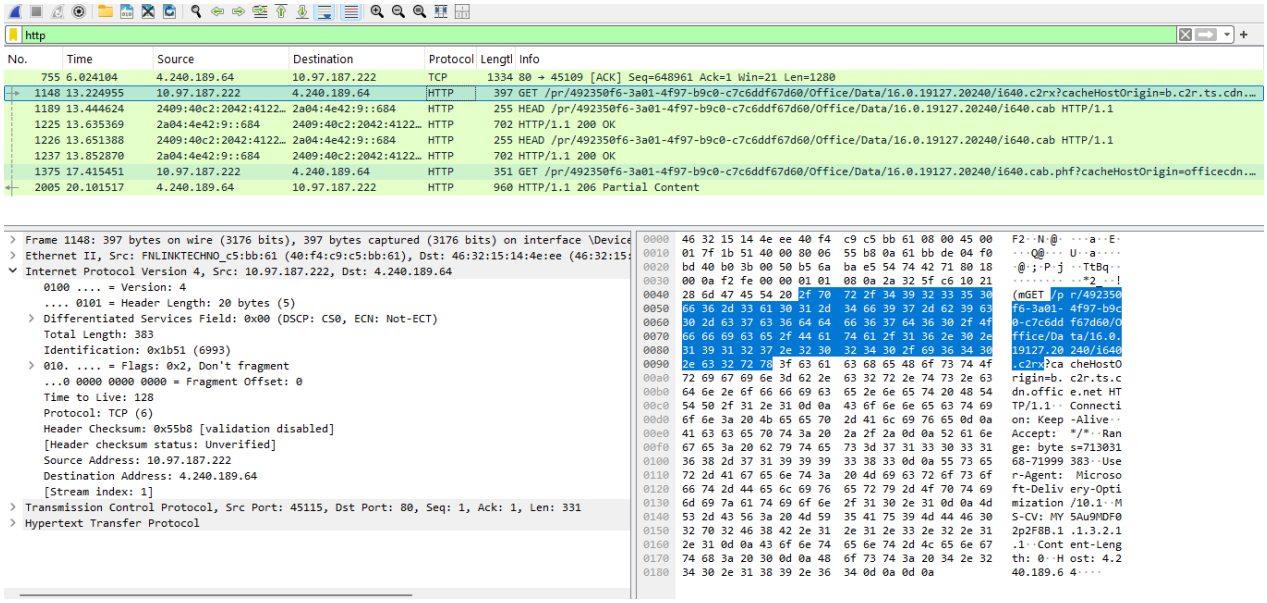
🔍 How to View IP Header:

1. Select an IP packet in your capture (e.g., HTTP, ICMP, or any IP-based traffic).
2. Expand the Internet Protocol Version 4 (IPv4) section.

You'll see fields like:

- Version (4 for IPv4)
- Header Length
- Total Length (size of the packet including header and data)
- Identification (used in fragmentation)

- Flags (DF – Don't Fragment, MF – More Fragments)
- Fragment Offset (position of this fragment in the original datagram)
- Time To Live (TTL)
- Protocol (e.g., 1 = ICMP, 6 = TCP, 17 = UDP)
- Source IP
- Destination IP



2. ICMP – Internet Control Message Protocol

ICMP is used for:

- Echo Request/Reply (ping)
- Destination unreachable
- Time exceeded (traceroute)

Filter ICMP Traffic:

icmp

How to Generate ICMP:

- Open Terminal/Command Prompt and type:

ping 8.8.8.8

This generates ICMP Echo Requests and receives Echo Replies.

🔍 View ICMP Fields in Wireshark:

1. Select an ICMP packet.
2. Expand Internet Control Message Protocol section.

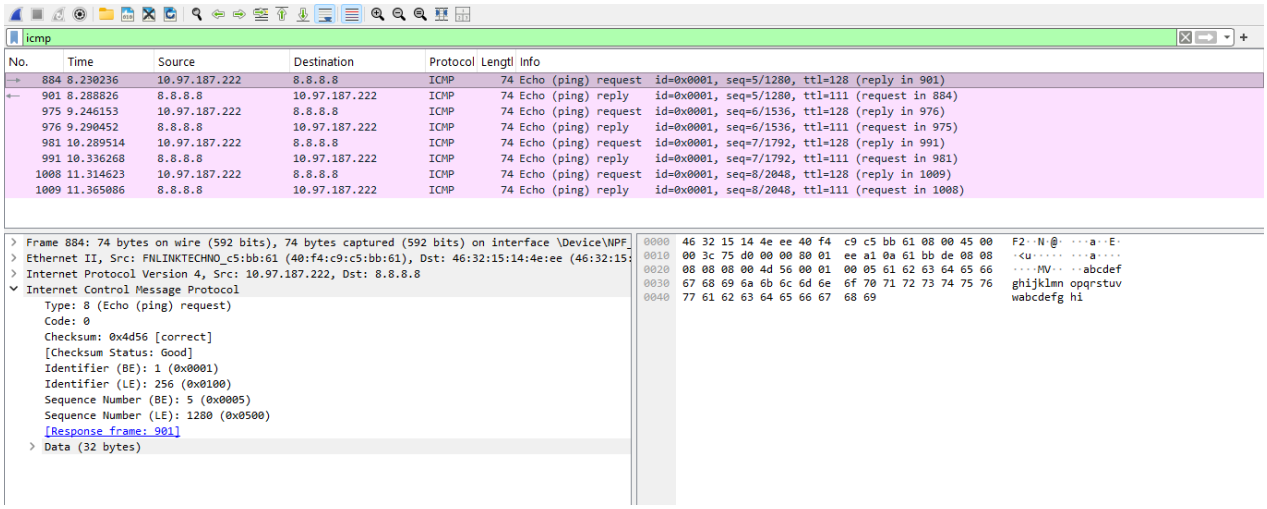
You'll see:

- Type:
  - 8 = Echo Request
  - 0 = Echo Reply
  - 3 = Destination Unreachable
  - 11 = Time Exceeded (TTL expired)
- Code (refines the message type)
- Checksum
- Identifier and Sequence Number (match requests to replies)

```
C:\Users\janu>ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=58ms TTL=111
Reply from 8.8.8.8: bytes=32 time=44ms TTL=111
Reply from 8.8.8.8: bytes=32 time=47ms TTL=111
Reply from 8.8.8.8: bytes=32 time=50ms TTL=111

Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 44ms, Maximum = 58ms, Average = 49ms
```



🔍 Goal:

Analyze TCP communication by capturing and interpreting:

- TCP Handshake (3-way handshake)
- TCP Source/Destination Ports
- TCP Segments (including flags, sequence numbers, etc.)

🧷 Step-by-Step Process

☑ Step 1: Start Wireshark and Begin Capture

1. Launch Wireshark.
2. Select your active network interface (usually Ethernet or Wi-Fi).
3. Click the blue shark fin button to start capturing.

🌐 Step 2: Generate TCP Traffic

You can open a web browser and go to a website (e.g., http://example.com) or use:

ping google.com

Or better:

telnet google.com 80

If Telnet is not installed:

curl http://example.com

This generates visible TCP traffic, including handshakes.

🔎 Step 3: Filter TCP Packets

In Wireshark's top filter bar, enter:

tcp

Press Enter. This filters only TCP traffic.

🔍 Step 4: Identify the TCP 3-Way Handshake

Look for 3 sequential packets between the same source and destination IP:

Example:

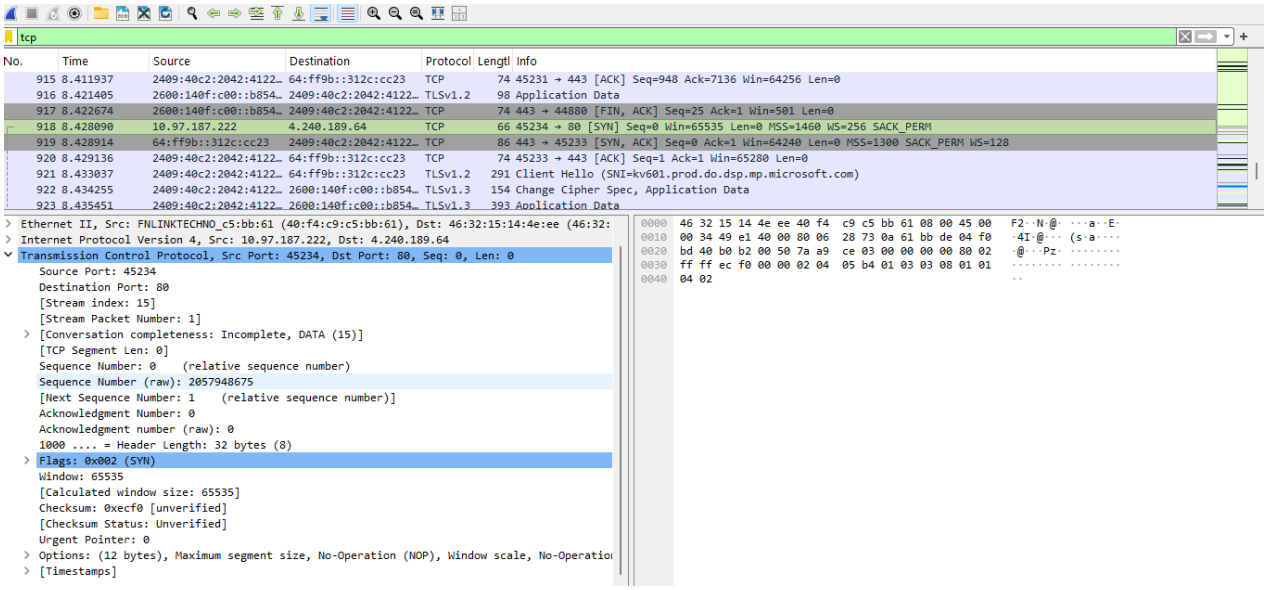
1. SYN — Initiates the connection
2. SYN, ACK — Acknowledges and responds
3. ACK — Final acknowledgment

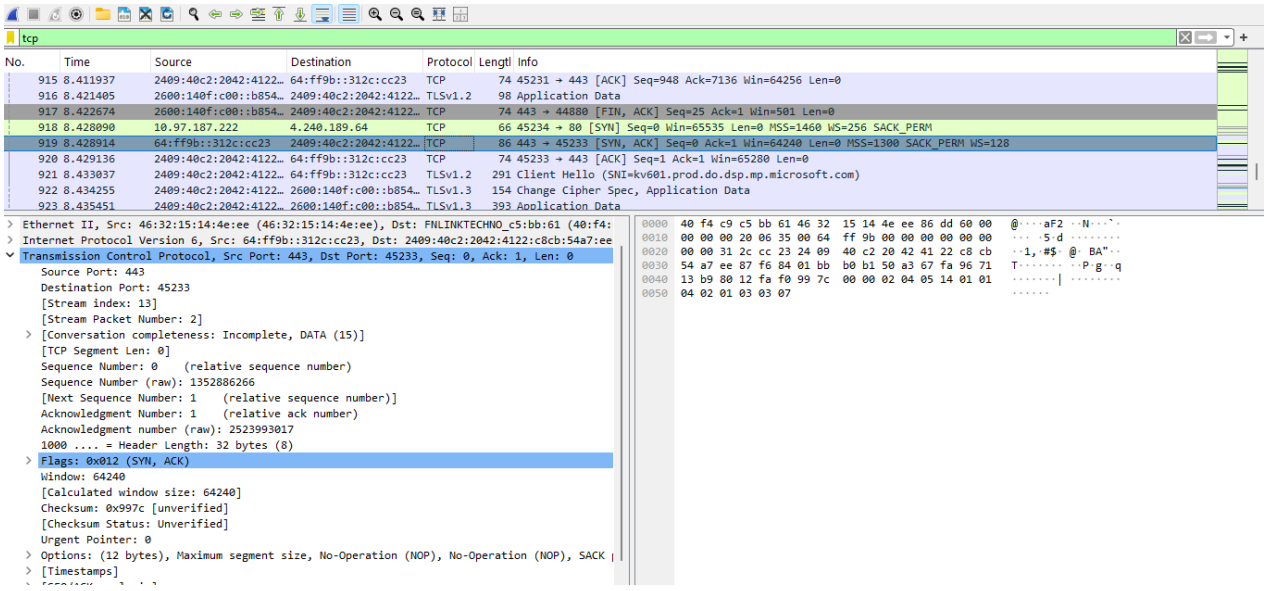
Each packet will show flags:

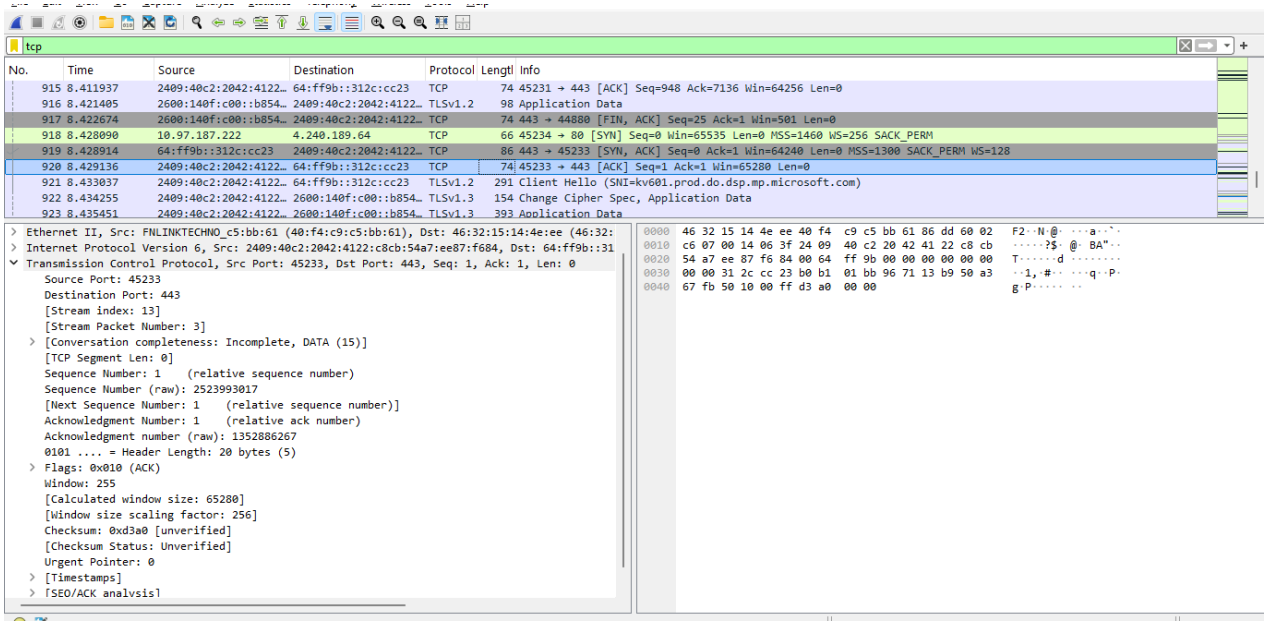
No	Source	Destination	Info
1	Your IP	Server IP	SYN
2	Server IP	Your IP	SYN, ACK
3	Your IP	Server IP	ACK

Click on each packet and expand:

- Transmission Control Protocol section


tcp
No. Time Source Destination Protocol Lengtl Info
915 8.411937 2409:40c2:2042:4122… 64:ff9b::312c:cc23 TCP 74 45231 → 443 [ACK] Seq=948 Ack=7136 Win=64256 Len=0
916 8.421405 2600:140f:c00::b854… 2409:40c2:2042:4122… TLSv1.2 98 Application Data
917 8.422674 2600:140f:c00::b854… 2409:40c2:2042:4122… TCP 74 443 → 44880 [FIN, ACK] Seq=25 Ack=1 Win=501 Len=0
918 8.428090 10.97.187.222 4.240.189.64 TCP 66 45234 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
919 8.428914 64:ff9b::312c:cc23 2409:40c2:2042:4122… TCP 86 443 → 45233 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1300 SACK_PERM WS=128
920 8.429136 2409:40c2:2042:4122… 64:ff9b::312c:cc23 TCP 74 45233 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
921 8.433037 2409:40c2:2042:4122… 64:ff9b::312c:cc23 TLSv1.2 291 Client Hello (SNI=kv601.prod.do.dsp.mp.microsoft.com)
922 8.434255 2409:40c2:2042:4122… 2600:140f:c00::b854… TLSv1.3 154 Change Cipher Spec, Application Data
923 8.435451 2409:40c2:2042:4122… 2600:140f:c00::b854… TLSv1.3 393 Application Data
Ethernet II, Src: FNLINKTECHNO_c5:bb:61 (40:f4:c9:c5:bb:61), Dst: 46:32:15:14:4e:ee (46:32:
Internet Protocol Version 4, Src: 10.97.187.222, Dst: 4.240.189.64
Transmission Control Protocol, Src Port: 45234, Dst Port: 80, Seq: 0, Len: 0
Source Port: 45234
Destination Port: 80
[Stream index: 15]
[Stream Packet Number: 1]
[Conversation completeness: Incomplete, DATA (15)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 2057948675
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
1000 .... = Header Length: 32 bytes (8)
Flags: 0x002 (SYN)
Window: 65535
[Calculated window size: 65535]
Checksum: 0xecf0 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operatio
[Timestamps]
0000 46 32 15 14 4e ee 40 f4 c9 c5 bb 61 08 00 45 00 F2··N·@· ···a··E·
0010 00 34 49 e1 40 00 80 06 28 73 0a 61 bb de 04 f0 ·4I·@··· (s·a····
0020 bd 40 b0 b2 00 50 7a a9 ce 03 00 00 00 00 80 02 ·@···Pz· ········
0030 ff ff ec f0 00 00 02 04 05 b4 01 03 03 08 01 01 ········ ········
0040 04 02 ··


tcp
No. Time Source Destination Protocol Lengtl Info
915 8.411937 2409:40c2:2042:4122… 64:ff9b::312c:cc23 TCP 74 45231 → 443 [ACK] Seq=948 Ack=7136 Win=64256 Len=0
916 8.421405 2600:140f:c00::b854… 2409:40c2:2042:4122… TLSv1.2 98 Application Data
917 8.422674 2600:140f:c00::b854… 2409:40c2:2042:4122… TCP 74 443 → 44880 [FIN, ACK] Seq=25 Ack=1 Win=501 Len=0
918 8.428090 10.97.187.222 4.240.189.64 TCP 66 45234 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
919 8.428914 64:ff9b::312c:cc23 2409:40c2:2042:4122… TCP 86 443 → 45233 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1300 SACK_PERM WS=128
920 8.429136 2409:40c2:2042:4122… 64:ff9b::312c:cc23 TCP 74 45233 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
921 8.433037 2409:40c2:2042:4122… 64:ff9b::312c:cc23 TLSv1.2 291 Client Hello (SNI=kv601.prod.do.dsp.mp.microsoft.com)
922 8.434255 2409:40c2:2042:4122… 2600:140f:c00::b854… TLSv1.3 154 Change Cipher Spec, Application Data
923 8.435451 2409:40c2:2042:4122… 2600:140f:c00::b854… TLSv1.3 393 Application Data
Ethernet II, Src: 46:32:15:14:4e:ee (46:32:15:14:4e:ee), Dst: FNLINKTECHNO_c5:bb:61 (40:f4:
Internet Protocol Version 6, Src: 64:ff9b::312c:cc23, Dst: 2409:40c2:2042:4122:c8cb:54a7:ee
Transmission Control Protocol, Src Port: 443, Dst Port: 45233, Seq: 0, Ack: 1, Len: 0
Source Port: 443
Destination Port: 45233
[Stream index: 13]
[Stream Packet Number: 2]
[Conversation completeness: Incomplete, DATA (15)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 1352886266
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 2523993017
1000 .... = Header Length: 32 bytes (8)
Flags: 0x012 (SYN, ACK)
Window: 64240
[Calculated window size: 64240]
Checksum: 0x997c [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK
[Timestamps]
0000 40 f4 c9 c5 bb 61 46 32 15 14 4e ee 86 dd 60 00 @····aF2 ··N···`·
0010 00 00 00 20 06 35 00 64 ff 9b 00 00 00 00 00 00 ··· ·5·d ········
0020 00 00 31 2c cc 23 24 09 40 c2 20 42 41 22 c8 cb ··1,·#$· @· BA"··
0030 54 a7 ee 87 f6 84 01 bb b0 b1 50 a3 67 fa 96 71 T····· ··P·g··q
0040 13 b9 80 12 fa f0 99 7c 00 00 02 04 05 14 01 01 ·····| ········
0050 04 02 01 03 03 07 ····


tcp
No. Time Source Destination Protocol Lengtl Info
915 8.411937 2409:40c2:2042:4122… 64:ff9b::312c:cc23 TCP 74 45231 → 443 [ACK] Seq=948 Ack=7136 Win=64256 Len=0
916 8.421405 2600:140f:c00::b854… 2409:40c2:2042:4122… TLSv1.2 98 Application Data
917 8.422674 2600:140f:c00::b854… 2409:40c2:2042:4122… TCP 74 443 → 44880 [FIN, ACK] Seq=25 Ack=1 Win=501 Len=0
918 8.428090 10.97.187.222 4.240.189.64 TCP 66 45234 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
919 8.428914 64:ff9b::312c:cc23 2409:40c2:2042:4122… TCP 86 443 → 45233 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1300 SACK_PERM WS=128
920 8.429136 2409:40c2:2042:4122… 64:ff9b::312c:cc23 TCP 74 45233 → 443 [ACK] Seq=1 Ack=1 Win=65280 Len=0
921 8.433037 2409:40c2:2042:4122… 64:ff9b::312c:cc23 TLSv1.2 291 Client Hello (SNI=kv601.prod.do.dsp.mp.microsoft.com)
922 8.434255 2409:40c2:2042:4122… 2600:140f:c00::b854… TLSv1.3 154 Change Cipher Spec, Application Data
923 8.435451 2409:40c2:2042:4122… 2600:140f:c00::b854… TLSv1.3 393 Application Data
Ethernet II, Src: FNLINKTECHNO_c5:bb:61 (40:f4:c9:c5:bb:61), Dst: 46:32:15:14:4e:ee (46:32:
Internet Protocol Version 6, Src: 2409:40c2:2042:4122:c8cb:54a7:ee87:f684, Dst: 64:ff9b::31
Transmission Control Protocol, Src Port: 45233, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
Source Port: 45233
Destination Port: 443
[Stream index: 13]
[Stream Packet Number: 3]
[Conversation completeness: Incomplete, DATA (15)]
[TCP Segment Len: 0]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 2523993017
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 1352886267
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window: 255
[Calculated window size: 65280]
[Window size scaling factor: 256]
Checksum: 0xd3a0 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
[Timestamps]
[SEQ/ACK analysis]
0000 46 32 15 14 4e ee 40 f4 c9 c5 bb 61 86 dd 60 02 F2··N·@· ···a··`·
0010 c6 07 00 14 06 3f 24 09 40 c2 20 42 41 22 c8 cb ·····?$· @· BA"··
0020 54 a7 ee 87 f6 84 00 64 ff 9b 00 00 00 00 00 00 T·····d ········
0030 00 00 31 2c cc 23 b0 b1 01 bb 96 71 13 b9 50 a3 ··1,·#·· ···q··P·
0040 67 fb 50 10 00 ff d3 a0 00 00 g·P····· ··

Objective

Understand how Application Layer protocols (DHCP, FTP, HTTP) behave using Wireshark, and how their headers and messages are structured within the TCP/IP stack.

TCP/IP Layers (Recap)

TCP/IP Layer	Example Protocols
Application	HTTP, FTP, DHCP
Transport	TCP, UDP
Network	IP
Data Link	Ethernet, Wi-Fi

We'll now focus on the Application Layer protocols.

Step-by-Step Analysis in Wireshark

1. DHCP (Dynamic Host Configuration Protocol)

Overview

- Used to automatically assign IP addresses to devices on a network.
- Operates over UDP ports 67 (server) and 68 (client).

How to Capture

1. Restart your computer's network connection (disable/enable Wi-Fi or Ethernet).
2. Start Wireshark before doing that.
3. Filter packets:
4. bootp

("bootp" is the filter for DHCP in Wireshark)

What to Look For

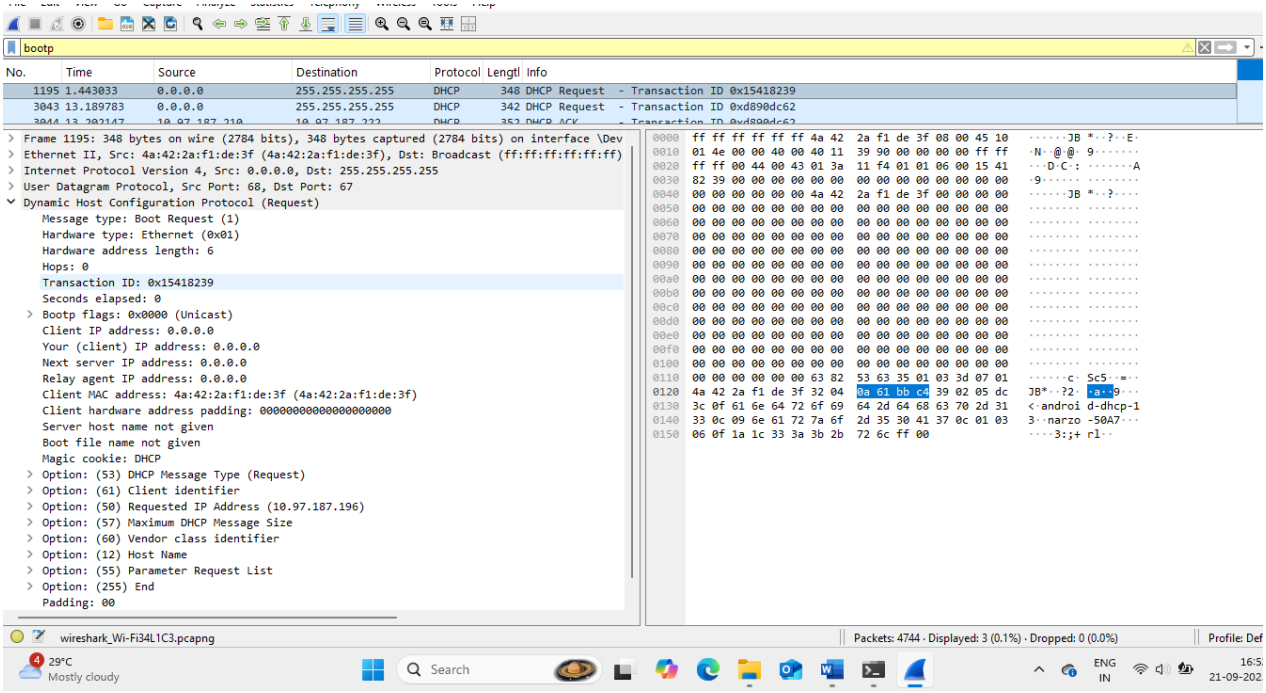
You'll typically see:

- DHCP Discover
- DHCP Offer
- DHCP Request
- DHCP ACK

🔍 Header Format (in Wireshark)

Expand the packet sections:

- Bootstrap Protocol (BOOTP)
    - Message type (e.g., Boot Request/Reply)
    - Client IP, Your IP, Server IP
    - MAC address
- DHCP Options
    - Option 53: DHCP Message Type (Discover, Offer, etc.)
    - Option 1: Subnet Mask
    - Option 3: Router (Default Gateway)
    - Option 6: DNS Server



2. 🗀 FTP (File Transfer Protocol)

📍 Overview

- Used to transfer files between client and server.
- Runs over TCP port 21 for commands, port 20 for data (active mode).

🎯 How to Capture

1. Start Wireshark.
2. Use an FTP client (like FileZilla) or command line:
3. ftp ftp.dlptest.com
4. Filter packets:
5. ftp

📊 What to Look For

You'll see:

- FTP command/control packets (e.g., USER, PASS, LIST, RETR)
- FTP data transfer packets (on another TCP stream)

🔍 Header Format (in Wireshark)

In the FTP section:

- Command: USER, PASS, LIST, etc.
- Response codes: 220, 331, 230, etc.

Expand the Transmission Control Protocol (TCP) section as well to see transport layer info.

3. 🌐 HTTP (HyperText Transfer Protocol)

📍 Overview

- Used for web communication.
- Uses TCP port 80 (HTTP) or 443 (HTTPS, not directly viewable unless decrypted).

🎯 How to Capture

1. Start Wireshark.
2. Open a browser and visit a non-HTTPS website like:
3. http://example.com
4. Filter packets:
5. http

📊 What to Look For

You'll see:

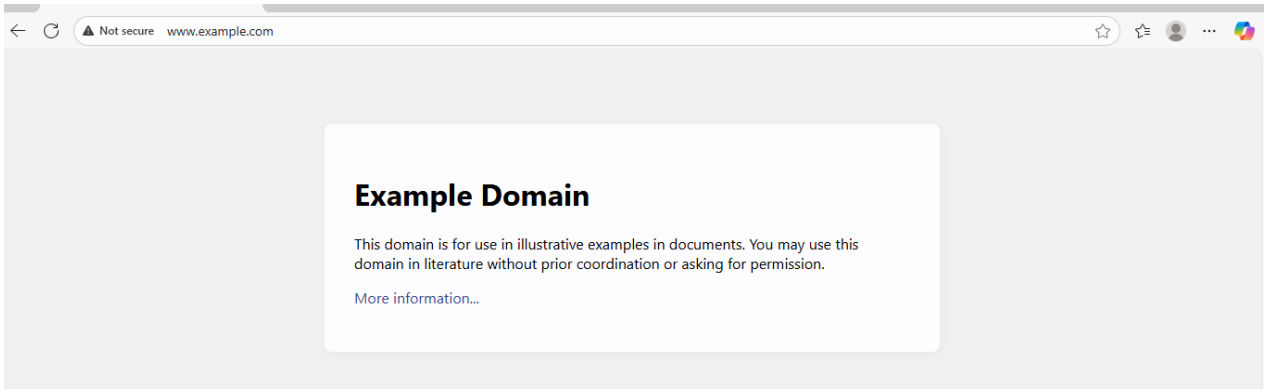
- HTTP GET or POST requests
- HTTP 200 OK, 301 Redirect, 404 Not Found responses

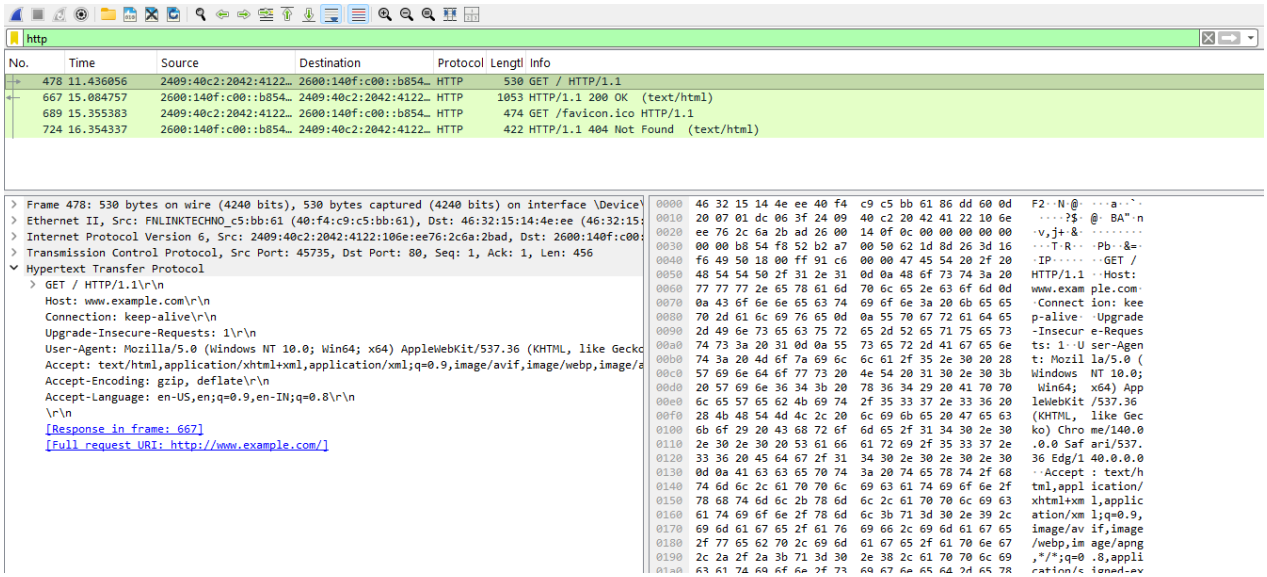
🔍 HTTP Header Format (in Wireshark)

Expand the Hypertext Transfer Protocol section:

- HTTP Request:
- GET /index.html HTTP/1.1
- Host: example.com

- User-Agent: Mozilla/5.0
- Accept: text/html
- HTTP Response:
- HTTP/1.1 200 OK
- Content-Type: text/html
- Content-Length: 1256





Conclusion:

This practical exercise using Wireshark provided hands-on experience in analyzing the operation of the TCP/IP protocol suite across all layers. By capturing and inspecting network packets, the following key insights were gained:

- Ethernet Layer: Understanding of frame structures including source and destination MAC addresses and frame size, which forms the foundation for network communication at the data link level.
- Data Link Layer: Observation of MAC addresses and the ARP protocol demonstrated how devices map IP addresses to MAC addresses for local network communication.
- Network Layer: Detailed analysis of IP packet headers revealed how routing, fragmentation, and protocol identification occur. The ICMP protocol was explored through ping requests and replies, illustrating network diagnostics.
- Transport Layer: The TCP protocol's port numbers, flags, and the three-way handshake mechanism were studied, emphasizing how reliable connections are established and maintained.
- Application Layer: Protocols such as DHCP, FTP, and HTTP were analyzed, showing how IP configuration, file transfers, and web communications work at the application level, including their header formats.

Overall, Wireshark proved to be an essential tool for visualizing and understanding the complex processes of TCP/IP networking. This practical reinforced theoretical concepts by allowing real-time observation and analysis of network traffic, enhancing comprehension of how data flows through network layers in everyday internet communication.