Problem Statement 1: Familiarization of Network Environment, Understanding and using network utilities: ipconfig, netstat, ping, telnet, ftp, traceroute etc.

1. `ipconfig`

- **Purpose:** Displays network configuration details on Windows systems.
- **Usage:** `ipconfig` shows IP address, subnet mask, default gateway, and more.
- **Example:** `ipconfig /all` – shows detailed information about all network adapters.

2. `netstat`

- **Purpose:** Displays active TCP connections, listening ports, routing tables, and network statistics.
- **Usage:** Useful for checking open ports and diagnosing network issues.
- **Example:** `netstat -an` – shows all connections and listening ports in numeric form.

3. `ping`

- **Purpose:** Tests connectivity between your computer and another device (IP address or domain).
- **Usage:** Sends ICMP Echo Request packets and waits for a reply to measure response time and packet loss.
- **Example:** `ping google.com` – checks if Google's server is reachable.

4. `telnet`

- **Purpose:** Used to connect to remote computers using the Telnet protocol.
- **Usage:** Can be used to test connectivity to specific ports (like HTTP on port 80).
- **Example:** `telnet example.com 80` – attempts to connect to example.com via port 80.

5. `ftp`

- **Purpose:** Used for transferring files between computers using the File Transfer Protocol.
- **Usage:** Allows upload/download operations to and from an FTP server.
- **Example:** `ftp ftp.example.com` – connects to an FTP server for file operations.

6. `traceroute` (Linux/macOS) / `tracert` (Windows)

- **Purpose:** Traces the route packets take from your computer to a destination.
- **Usage:** Helps diagnose network latency and routing issues.
- **Example:** `traceroute google.com` – shows each hop a packet takes to reach Google.

Problem statement 2 - Familiarization with Transmission media and tools: Co-axial cable, UTP cable, Crimping tool, Connectors etc. Preparing the UTP cable for cross and direct connection using crimping tool.

1. Co-axial Cable

- **Use**: Used to transmit data, video, and voice signals.
- **Structure**: Central conductor, insulating layer, metal shielding, and outer jacket.
- **Example**: Often used in cable TV and older Ethernet networks (like 10Base2).
- **Advantage**: Good resistance to signal interference.

2. UTP Cable (Unshielded Twisted Pair)

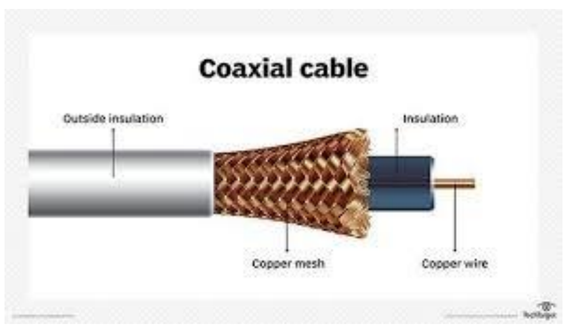
- **Use**: Most common cable for LAN networks, especially Ethernet.
- **Structure**: 4 pairs of twisted copper wires without shielding.
- **Example**: Cat5e, Cat6 cables used for internet connections.
- **Advantage**: Cheap, flexible, and easy to install.

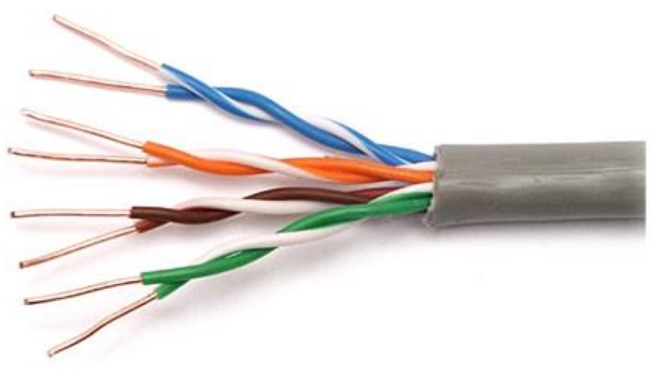
3. Crimping Tool

- **Use**: Used to attach connectors (like RJ-45) to the ends of UTP cables.
- **Function**: Binds the wires securely into the connector by compressing (crimping) it.
- **Tip**: Essential for custom-length network cable creation.

4. Connectors

- **RJ-45 Connector**:
  - **Use**: Used with UTP cables for Ethernet connections.
  - **Looks like**: A wider version of a telephone plug (RJ-11).
- **BNC Connector**:
  - **Use**: Used with coaxial cables.
  - **Application**: Often found in CCTV systems and legacy networks.





Problem Statement 3 - Installation and introduction of simulation tool.(Packet Tracer)

Step 1: Create a Cisco NetAcad Account

1. Go to **Cisco Networking Academy**
2. Click **"Sign up"** (top right).
3. Choose **"I'm a student"** or select "Self-paced" if you're learning on your own.
4. Create your account and verify your email.

Step 2: Enroll in a Free Course

Cisco only allows downloads after enrolling in at least one course.

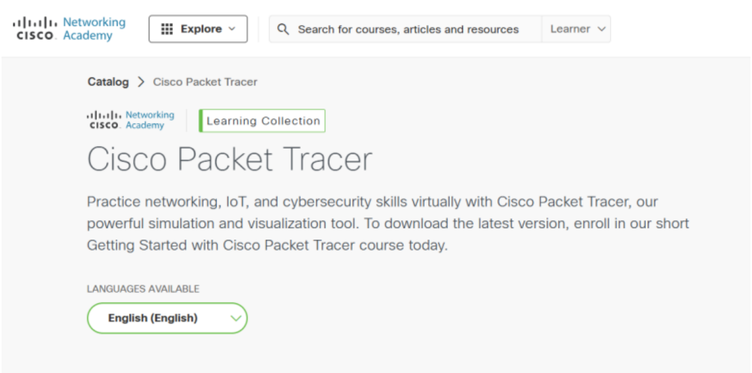
1. After login, go to the **Courses** section.
2. Enroll in this free course: **"Introduction to Packet Tracer"** (or search for it directly here)

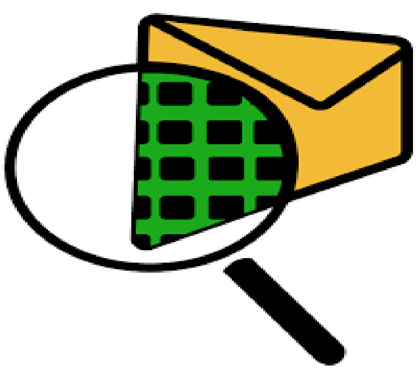
Step 3: Download Packet Tracer

1. Once enrolled, go to the **course page**.
2. Click on the **"Download Packet Tracer"** link.
3. Choose the correct version for your OS:
   - **Windows**
   - **macOS**
   - **Linux (Ubuntu)**

Step 4: Install It

- Run the downloaded file and follow on-screen installation instructions.
- Log in using your Cisco NetAcad credentials when launching Packet Tracer.





Problem Statement 4 - To configure a basic network topology consisting of routers, switches, and end devices such as PCs or laptops. Configure IP addresses and establish connectivity between devices. (Using packet Tracer)

Step 1: Open Cisco Packet Tracer

- Launch the application and open a new workspace.

Step 2: Add Devices

- From the bottom toolbar:
  - Click on **"End Devices"** (□ icon) → Drag **4-5 PCs** onto the workspace.
  - Click on **"Switches"** (□ icon) → Drag **one switch** (e.g., **2960**) to the center.

Step 3: Connect Devices

- Click the **"Connections"** lightning icon (⚡).
- Choose **Copper Straight-Through Cable** (used for PC to switch).
- Connect each PC to the switch:
  - Click on **PC1** → choose **FastEthernet0**.
  - Then click on **Switch** → choose **FastEthernet0/1**.
  - Repeat this for PC2 → FastEthernet0/2, and so on.

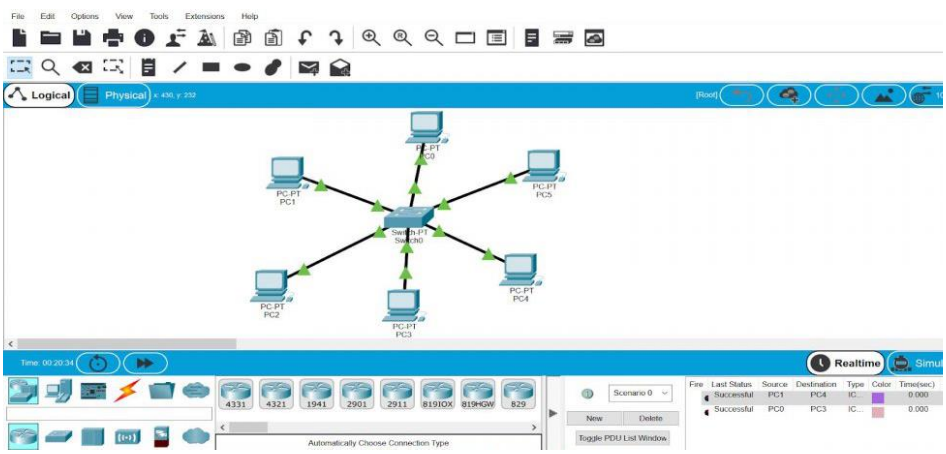
Step 4: Assign IP Addresses (Optional for Testing)

- Click on **each PC** → Go to **Desktop** tab → Click **IP Configuration**.
- Assign IPs like:
  - PC1: 192.168.1.1
  - PC2: 192.168.1.2
  - PC3: 192.168.1.3
  - Subnet Mask: 255.255.255.0 for all

Step 5: Test Connectivity

- On any PC, go to **Desktop** → **Command Prompt**.
- Use the `ping` command to test:

```
ping 192.168.1.2
```



Problem Statement 5 - To configure a DHCP server on a router or a dedicated DHCP server device. Assign IP addresses dynamically to devices on the network and verify successful address assignment.(Using packet Tracer)

1. Drag and Drop Devices

From the bottom bar:

- 1 **Router** (e.g., 2911)
- 1 **Switch** (e.g., 2960)
- 3 **PCs**
- Use **Copper Straight-Through Cables** to connect:
  - PCs to Switch ports (FastEthernet0/x)
  - Switch to Router (e.g., Router GigabitEthernet0/0)

2. Configure Router as DHCP Server

- Click **Router** → **CLI Tab**
- Type these commands:
- enable

```
configure terminal
interface gigabitEthernet0/0
ip address 192.168.1.1 255.255.255.0
no shutdown
exit
ip dhcp excluded-address 192.168.1.1 192.168.1.9
ip dhcp pool LAN
network 192.168.1.0 255.255.255.0
default-router 192.168.1.1
dns-server 8.8.8.8
exit
```

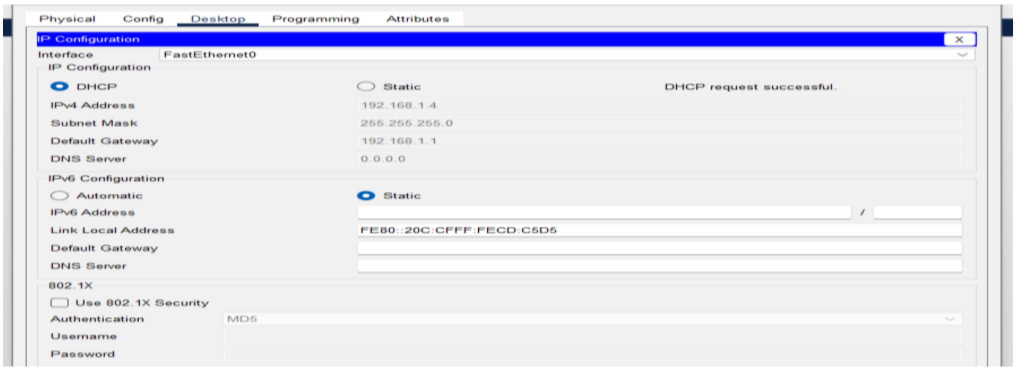
3. Set PCs to Receive IP Automatically

- Click **each PC** → Desktop → IP Configuration
- Select **DHCP**. (Each PC will get IPs like `192.168.1.10`, `.11`, etc.)

4. Test Connection

- On any PC → Desktop → Command Prompt:

```
ipconfig
ping 192.168.1.1
```



Problem Statement 6 - To configure a local DNS server to resolve domain names within a network. (Using packet Tracer)

Step 1: Select Devices

From the **bottom bar in Packet Tracer**, drag the following to the workspace:

- 1 **Router** → e.g., **Router-PT**
- 1 **Switch** → e.g., **Switch-PT**
- 2 **Servers**:
  - One for **DNS**
  - One as **Web Server** (for testing)
- 2 **PCs** → e.g., **PC0**, **PC1**

Step 2: Connect Devices with Cables

Use **Copper Straight-Through** cables:

- **PC0 → Switch** (FastEthernet0)
- **PC1 → Switch** (FastEthernet1)
- **DNS Server → Switch** (FastEthernet2)
- **Web Server → Switch** (FastEthernet3)
- **Switch → Router** (GigabitEthernet0/0 on Router)

Step 3: Assign IP Addresses

PC0

- IP: `192.168.1.10`
- Subnet: `255.255.255.0`
- Gateway: `192.168.1.1`
- **DNS: `192.168.1.2`**

PC1

- IP: `192.168.1.11`
- Subnet: `255.255.255.0`
- Gateway: `192.168.1.1`
- **DNS: `192.168.1.2`**

DNS Server

- IP: `192.168.1.2`
- Subnet: `255.255.255.0`
- Gateway: `192.168.1.1`

Web Server

- IP: `192.168.1.3`
- Subnet: `255.255.255.0`
- Gateway: `192.168.1.1`

Step 4: Configure the Router

1. Click the **Router → CLI Tab**
2. Enter commands:

enable
configure terminal
interface g0/0
ip address 192.168.1.1 255.255.255.0
no shutdown
exit

Step 5: Configure DNS Server

1. Click the **DNS Server → Services tab**
2. Click on **DNS**
3. Make sure **DNS Service: ON**
4. Add a DNS record:
    - **Name**: `www.mywebsite.com`
    - **Address**: `192.168.1.3` (Web Server IP)
    - Click **Add**

Step 6: Configure Web Server

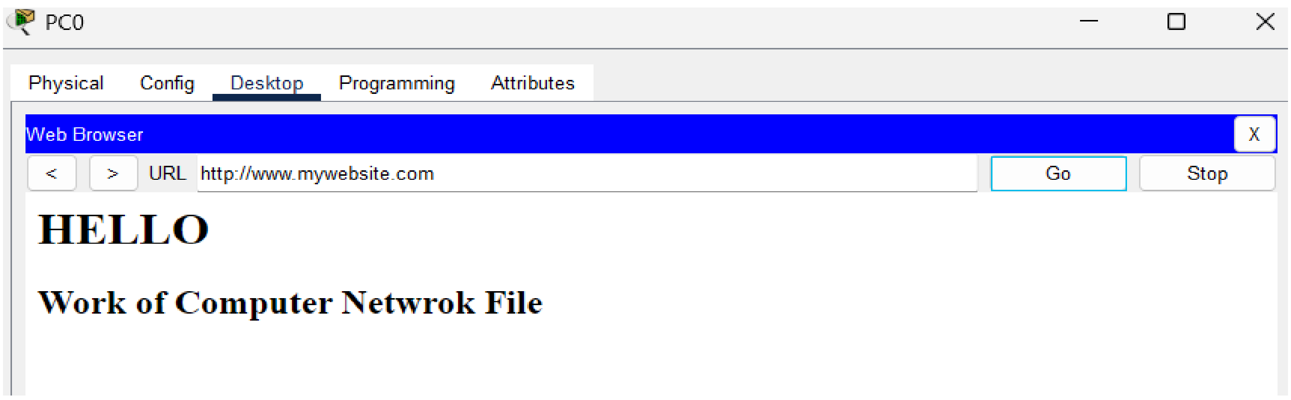
1. Click the **Web Server → Services tab**
2. Make sure **HTTP is ON**
3. (Optional) Edit the homepage in the HTTP settings

Step 7: Test DNS Resolution from PC

On **PC0** or **PC1**:

1. Go to **Desktop → Web Browser**
2. Type:

www.mywebsite.local



Problem Statement 7 - NAT (Network Address Translation): Set up NAT on a router to translate private IP addresses to public IP addresses for outbound internet connectivity. Test the translation and examine how NAT helps conserve IPv4 address space.(Using packet Tracer)

Step 1: Choose and Place Devices

- Open Cisco Packet Tracer.
- Drag and drop the following devices:
  - 1 Router (e.g., 2811)
  - 1 Switch (e.g., 2960)
  - 2 PCs (name them PC0 and PC1)
  - 1 Server (name it PublicServer)

Step 2: Connect Devices with Cables

- Use **Copper Straight-Through** cables to connect:
  - PC0 to Switch port Fa0/1
  - PC1 to Switch port Fa0/2
  - Router Fa0/0 to Switch port Fa0/24
  - Router Fa0/1 to PublicServer

Step 3: Assign IP Addresses

PC0

- IP Address: `192.168.1.10`
- Subnet Mask: `255.255.255.0`
- Default Gateway: `192.168.1.1`

PC1

- IP Address: `192.168.1.11`
- Subnet Mask: `255.255.255.0`
- Default Gateway: `192.168.1.1`

PublicServer

- IP Address: `200.0.0.2`
- Subnet Mask: `255.255.255.0`
- Default Gateway: `200.0.0.1`
- Go to **Services tab → HTTP → Turn ON**

Step 4: Configure Router Interfaces

- Click on Router → CLI, then type:

```
enable
configure terminal

interface fa0/0
ip address 192.168.1.1 255.255.255.0
no shutdown
exit
```

interface fa0/1
ip address 200.0.0.1 255.255.255.0
no shutdown
exit

Step 5: Configure NAT on the Router

- Still in CLI, type

access-list 1 permit 192.168.1.0 0.0.0.255

ip nat inside source list 1 interface fa0/1 overload

interface fa0/0

ip nat inside

exit

interface fa0/1

ip nat outside

exit

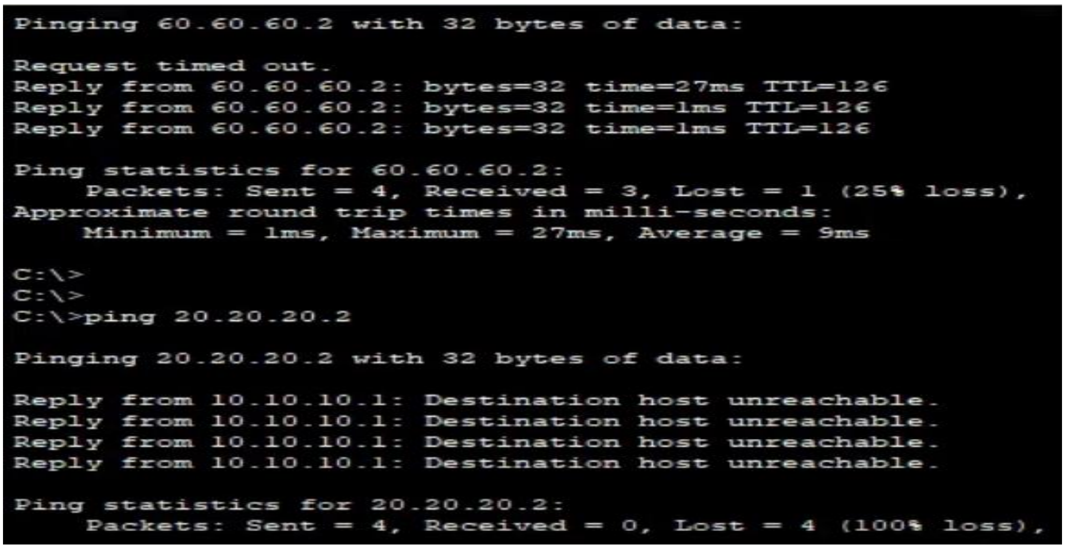
Step 6: Test NAT Functionality

From PC0 or PC1:

- Open Desktop → Command Prompt
- Ping the public server: ping 200.0.0.2 (You should receive replies.

Open Web Browser:

- Go to Desktop → Web Browser
- Enter: http://200.0.0.2 (You should see the server's web page.)



```
Pinging 60.60.60.2 with 32 bytes of data:

Request timed out.
Reply from 60.60.60.2: bytes=32 time=27ms TTL=126
Reply from 60.60.60.2: bytes=32 time=1ms TTL=126
Reply from 60.60.60.2: bytes=32 time=1ms TTL=126

Ping statistics for 60.60.60.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 27ms, Average = 9ms

C:\>
C:\>
C:\>ping 20.20.20.2

Pinging 20.20.20.2 with 32 bytes of data:

Reply from 10.10.10.1: Destination host unreachable.
Reply from 10.10.10.1: Destination host unreachable.
Reply from 10.10.10.1: Destination host unreachable.
Reply from 10.10.10.1: Destination host unreachable.

Ping statistics for 20.20.20.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Problem Statement 8 - Network Troubleshooting: Simulate network issues such as connectivity problems, incorrect configurations, or routing failures. Use Packet Tracer's simulation mode to diagnose and troubleshoot the network.

Step 1: Set Up a Simple Network

- Drag and drop:
  - 2 PCs (PC0 and PC1)
  - 1 Switch
  - 1 Router

Step 2: Connect the Devices

- Use **Copper Straight-Through** cables:
  - PC0 → Switch (e.g., Fa0/1)
  - PC1 → Switch (e.g., Fa0/2)
  - Switch → Router (Fa0/0)

Step 3: Configure Devices

On PC0:

- IP: `192.168.1.10`
- Subnet: `255.255.255.0`
- Gateway: `192.168.1.1`

On PC1:

- IP: `192.168.1.11`
- Subnet: `255.255.255.0`
- Gateway: `192.168.1.1`

On Router (CLI):

```
enable
configure terminal
interface fa0/0
ip address 192.168.1.1 255.255.255.0
no shutdown
exit
```

Step 4: Open Simulation Mode

- Go to the **bottom right** of Packet Tracer → Click **Simulation Mode**
- Use **Add Simple PDU Tool** (the envelope icon)
  - Click PC0 → then click PC1
- Press **Capture/Forward** to watch the packet movement

Step 5: Introduce Common Problems

Issue 1: Wrong IP on PC1

- Set PC1 IP to `192.168.2.11`
- Test ping again
- Simulation shows red X (failure)

- Fix: Change PC1 IP back to `192.168.1.11`

Issue 2: Wrong Gateway

- Set gateway on PC0 to `192.168.2.1`
- Ping PC1 → fails
- Fix: Set gateway back to `192.168.1.1`

Issue 3: Interface Shutdown on Router

- On Router:

interface fa0/0
shutdown

- Ping fails
- Fix: `no shutdown`

Issue 4: Cable Disconnected

- Delete cable between switch and router
- Packet will drop in simulation
- Fix: Reconnect cable correctly

Step 6: Use Simulation Details to Troubleshoot

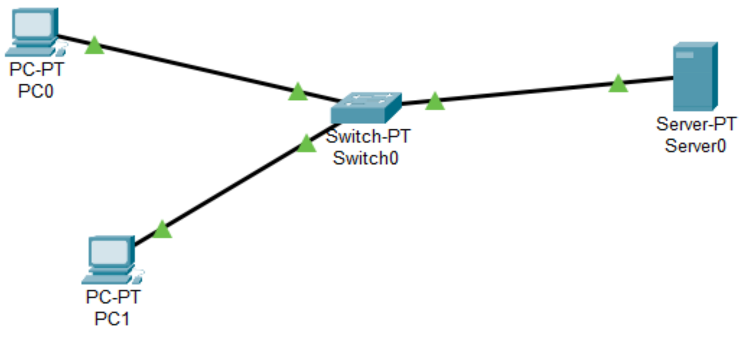
- Click the failed packet in Simulation Panel
- Read the **"Event List"** for reasons like:
    - "No ARP Reply"
    - "Destination unreachable"
    - "Request timed out"
- Use this info to determine the issue

Step 7: Fix Issues One-by-One

- Change configurations or reconnect cables as needed
- Run simulation again
- Observe when packets **successfully reach the destination (green arrow)**

Repeat with More Complex Issues

- Add a second router and test **routing issues**
- Remove default routes and test failure
- Try **incorrect subnet masks**



Problem Statement 9 - To monitor network traffic using Wire Shark

Step 1: Install Wireshark

- Visit https://www.wireshark.org/download.html
- Choose your OS (Windows, macOS, Linux) and install.
- On Windows, install **WinPcap** or **Npcap** when prompted (required for packet capture).

Step 2: Launch Wireshark

- Open Wireshark from your desktop or Start Menu.

Step 3: Select a Network Interface

- You'll see a list of network interfaces (like Wi-Fi, Ethernet).
- Look for the one actively transmitting data (it'll have fluctuating graphs).
- Double-click on it to start capturing packets.

Step 4: Start Capturing Packets

- As soon as you double-click an interface, Wireshark begins capturing.
- You'll see packets live-streaming into the capture window.

Step 5: Apply Capture Filters (Optional)

- Use filters to focus on specific traffic:
    - `http` → shows only HTTP packets
    - `ip.addr == 192.168.1.10` → traffic to/from a specific IP
    - `tcp.port == 80` → only TCP traffic on port 80

Enter filter in the **"Display Filter"** bar and hit Enter.

Step 6: Stop the Capture

- Click the red square **(■)** in the top-left toolbar to stop capturing.

Step 7: Analyze the Packets

- Click on any packet to see:
    - **Frame details** (physical layer info)
    - **Ethernet header**
    - **IP header**
    - **TCP/UDP/ICMP details**
    - **Payload / Data (if unencrypted)**
- You can right-click a packet → **"Follow TCP stream"** to see complete conversations.

Step 8: Save the Capture (Optional)

- Go to **File → Save As**
- Save with `.pcapng` extension for later analysis or reporting.

```
v Frame 549: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits) on interface \Device\NPF_{43CB42C5-7217-4048-9BEA-C4FC92A98383}, id 0
  > Interface id: 0 (\Device\NPF_{43CB42C5-7217-4048-9BEA-C4FC92A98383})
    Encapsulation type: Ethernet (1)
    Arrival Time: Mar  2, 2020 15:15:51.718658000 India Standard Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1583142351.718658000 seconds
    [Time delta from previous captured frame: 0.000466000 seconds]
    [Time delta from previous displayed frame: 0.000466000 seconds]
    [Time since reference or first frame: 268.732174000 seconds]
    Frame Number: 549
    Frame Length: 174 bytes (1392 bits)
    Capture Length: 174 bytes (1392 bits)
```

Problem Statement 10 - To analyze complete TCP/IP protocol suite layer's headers using Wire Shark

Open Wireshark and Start a Capture

- Launch **Wireshark**.
- Select your **active network interface** (e.g., Wi-Fi or Ethernet).
- Double-click it to **begin capturing traffic**.

Step 2: Generate TCP/IP Traffic

- Open a browser and visit a website (e.g., `http://example.com`).
- This triggers an HTTP request, which involves all layers of the TCP/IP stack.

Step 3: Stop the Capture

- After a few seconds, click the **red square button (■)** to stop capturing.

Step 4: Apply a Display Filter for TCP

- In the filter bar, type: tcp (Press Enter to show only TCP packets.)

Step 5: Analyze Each Layer in a Packet

Click on any TCP packet in the list to expand its headers:

1. Frame (Layer 1 – Physical)

- Shows physical details like:
  - Interface used
  - Packet size (bytes on wire vs captured)
  - Time received

2. Ethernet II (Layer 2 – Data Link)

- Contains:
  - **Source MAC address**
  - **Destination MAC address**
  - **Type** (usually 0x0800 for IPv4)

3. Internet Protocol (IP) (Layer 3 – Network)

- Click "Internet Protocol Version 4":
  - **Source IP address**
  - **Destination IP address**
  - **Version**
  - **Header Length**
  - **TTL (Time To Live)**
  - **Protocol** (e.g., TCP = 6, UDP = 17)

4. Transmission Control Protocol (TCP) (Layer 4 – Transport)

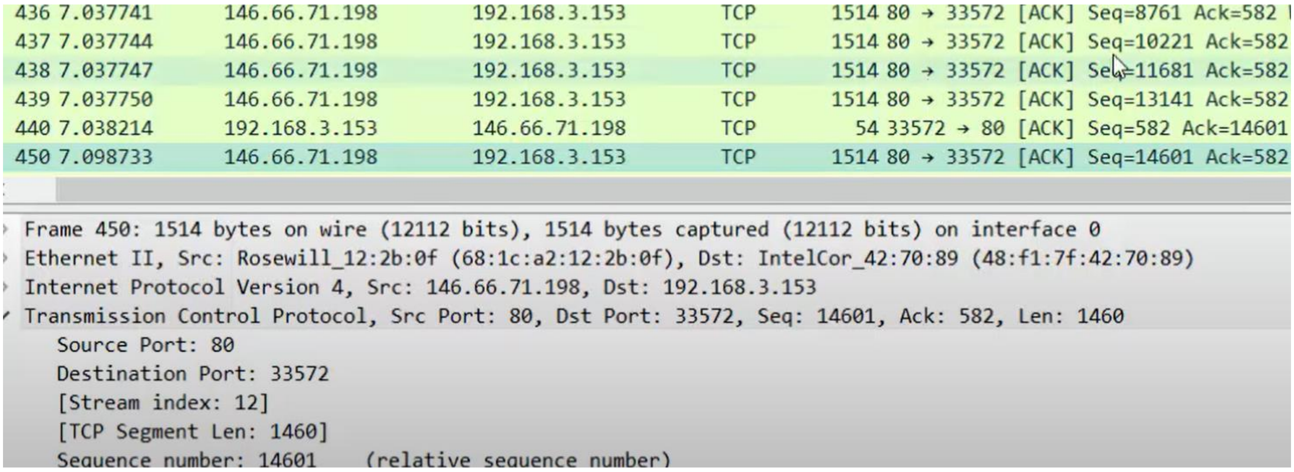
- Click "Transmission Control Protocol":
  - **Source port** and **Destination port**
  - **Sequence number**
  - **Acknowledgment number**
  - **Flags** (SYN, ACK, FIN, etc.)
  - **Window size**
  - **Checksum**

5. Hypertext Transfer Protocol (HTTP) (Layer 7 – Application)

(Visible only if the website is HTTP, not HTTPS)

- Shows:
  - **GET/POST requests**
  - **User-Agent**
  - **Host**
  - **Accept types**

If you're visiting **HTTPS** sites, Layer 7 will be **encrypted** (won't show readable headers).



Problem Statement 11 - TCP Client-Server Communication: Implement a TCP client program that sends a message to a TCP server program. Implement the corresponding TCP server program that receives the message and displays it. Test the communication between the client and server by exchanging.

1. Start Wireshark

- Open Wireshark.
- Select the active **network interface** (e.g., Ethernet, Wi-Fi, or loopback).

2. Start Capturing

- Click the network interface to begin packet capture.

3. Run Your TCP Server

- In one terminal, start the TCP server (e.g., using `tcp_server.py`).

4. Run Your TCP Client

- In another terminal, start the TCP client (e.g., using `tcp_client.py`).
- It sends a message to the server.

5. Stop Capturing

- Click the red square (■) to stop capturing after the message is exchanged.

6. Filter TCP Traffic

- In the **filter bar**, type:

  tcp.port == 12345 (Replace 12345 with your actual port if different)

7. Analyze the TCP Packets

Click on the relevant packets to see:

TCP Handshake:

- `[SYN]` → Client initiates connection
- `[SYN, ACK]` → Server acknowledges
- `[ACK]` → Client completes handshake

Message Exchange:

- Packet with payload: "Hello, TCP Server!"

TCP Teardown:

- `[FIN, ACK]` → Connection closing

8. Drill into Layers

Click on any packet and expand:

- **Ethernet II** (Layer 2)
- **IP** (Layer 3)
- **TCP** (Layer 4)
- **Data** → contains your actual message (Layer 7)

```
Server listening on port 8080...
Received from client: Hello from TCP Client!
```

Problem Statement 12 - UDP Client-Server Communication: Implement a UDP client program that sends a message to a UDP server program. Implement the corresponding UDP server program that receives the message and displays it (Using 'C' Language)

Save as `udp_server.c`:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <arpa/inet.h>

#include <unistd.h>

#define PORT 8080

#define MAXLINE 1024

int main() {

    int sockfd;

    char buffer[MAXLINE];

    struct sockaddr_in servaddr, cliaddr;

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    if (sockfd < 0) {

        perror("socket creation failed");

        exit(EXIT_FAILURE);

    }

    memset(&servaddr, 0, sizeof(servaddr));

    memset(&cliaddr, 0, sizeof(cliaddr));

    servaddr.sin_family = AF_INET;        // IPv4

    servaddr.sin_addr.s_addr = INADDR_ANY;

    servaddr.sin_port = htons(PORT);

    if (bind(sockfd, (const struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {

        perror("bind failed");

        exit(EXIT_FAILURE);

    }

    socklen_t len = sizeof(cliaddr);

    int n = recvfrom(sockfd, buffer, MAXLINE, 0, (struct sockaddr *)&cliaddr, &len);
```

```
    buffer[n] = '\0';
    printf("Received message: %s\n", buffer);
    close(sockfd);
    return 0;
}
```

Save as `udp_client.c`:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <unistd.h>
#define PORT 8080
#define MAXLINE 1024
int main() {
    int sockfd;
    char *message = "Hello from UDP Client!";
    struct sockaddr_in servaddr;
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }
    memset(&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = INADDR_ANY;

        sendto(sockfd, message, strlen(message), 0, (const struct sockaddr *) &servaddr,
sizeof(servaddr));
    printf("Message sent.\n");
```

```
    close(sockfd);

    return 0;

}
```

Code for the terminal –

gcc udp_server.c -o udp_server

gcc udp_client.c -o udp_client

Run the server in one terminal

./udp_server

Run the client in another terminal

./udp_client

Server output –

```
Server is ready...

Received message: Hello from UDP Client!
```

Client Output –

```
Message sent.
```