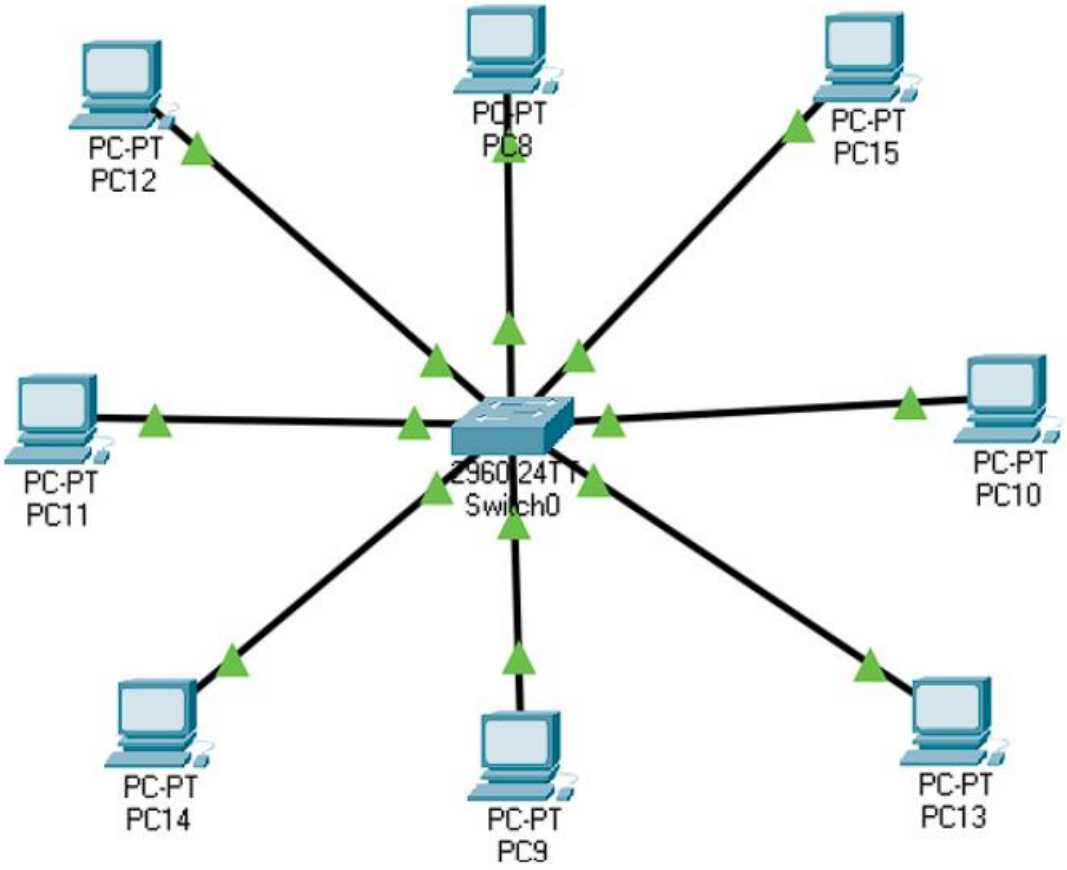
1. Implement of different Network Topologies. Locating different interfaces, routers and switches. Studying different pools of IP addresses.

Star Topology

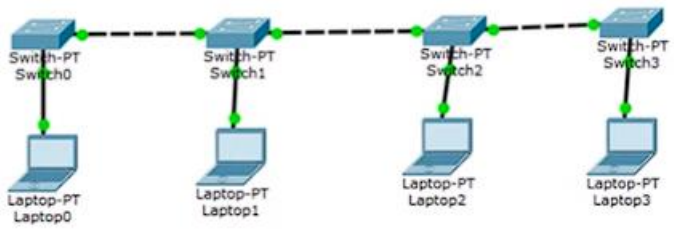
In a **star topology**, every device connects individually to a central hub using its own dedicated cable. This central hub acts as the main node, linking all devices together. The hub may be passive, simply relaying signals like a basic broadcasting unit, or active, with built-in repeaters to boost signal strength. Commonly, coaxial or RJ-45 cables are utilized for these connections. Star topology networks typically use Ethernet LAN protocols that support features like **collision detection** and **carrier sense multiple access (CSMA)** for efficient communication




```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 10.10.10.7

Pinging 10.10.10.7 with 32 bytes of data:

Reply from 10.10.10.7: bytes=32 time=1ms TTL=128
Reply from 10.10.10.7: bytes=32 time=4ms TTL=128
Reply from 10.10.10.7: bytes=32 time=4ms TTL=128
Reply from 10.10.10.7: bytes=32 time=4ms TTL=128

Ping statistics for 10.10.10.7:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 4ms, Average = 3ms
```

Bus Topology

In a **bus topology**, all computers and network devices are connected along a single communication line or backbone, forming one continuous cable that carries signals in both directions. This setup allows multiple devices to share the same transmission medium, but it has a major drawback: if the central cable experiences a failure, the entire network is disrupted. Bus networks are considered multipoint connections, where each device taps into the main cable. Several Media Access Control (MAC) methods used in this topology include techniques like **TDMA** (Time Division Multiple Access), **CDMA** (Code Division Multiple Access), and **Pure Aloha**, which help control access and manage potential data collisions within the network.




```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 10.10.10.22

Pinging 10.10.10.22 with 32 bytes of data:

Reply from 10.10.10.22: bytes=32 time=1ms TTL=128
Reply from 10.10.10.22: bytes=32 time=14ms TTL=128
Reply from 10.10.10.22: bytes=32 time=14ms TTL=128
Reply from 10.10.10.22: bytes=32 time=14ms TTL=128

Ping statistics for 10.10.10.22:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 14ms, Average = 10ms
```

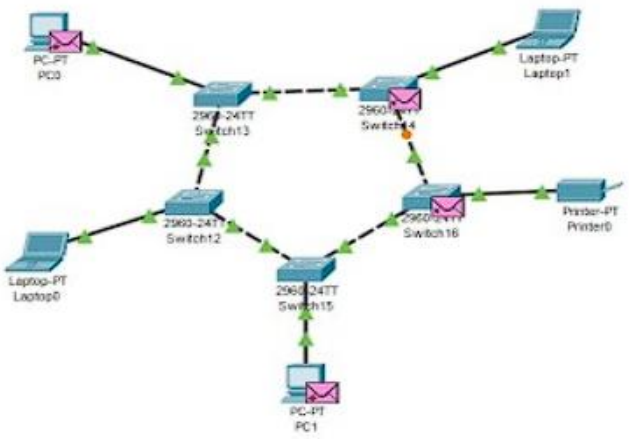
Ring Topology

In a **ring topology**, each device is linked to exactly two other devices, creating a closed loop that resembles a ring. When the network contains many devices, repeaters are often installed to boost signals as data may need to travel through several nodes—such as crossing 99 devices to reach the 100th. This helps prevent signal degradation or data loss.Computer_Networks_File.pdf

Data transmission in basic ring topologies is usually **unidirectional**, flowing in a single direction around the ring. However, the system can be upgraded to **dual ring topology** by connecting each device to its neighbors in both directions, enhancing redundancy and fault tolerance. The **token passing** method is the main access technique, where a special data packet, called a token, circulates the network. Only the device possessing the token is allowed to send data, minimizing the chance of packet collisions.Computer_Networks_File.pdf

Working Principles

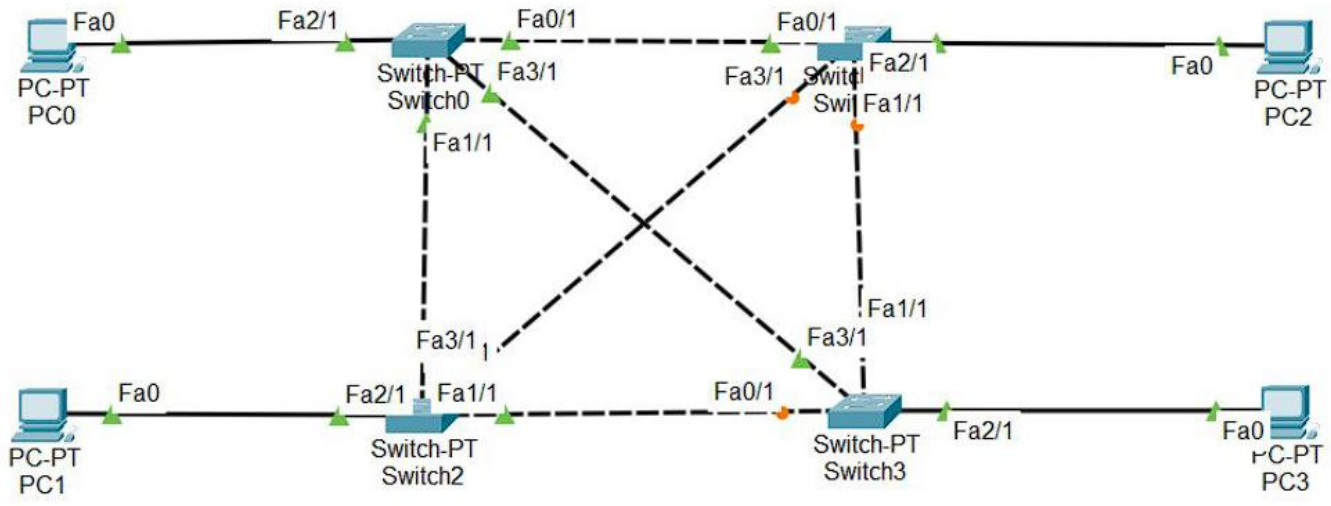
- A designated station acts as the **monitor**, overseeing management tasks and error control.
- To send data, a station must first acquire the token; once the transmission is complete, the token is released so other stations can transmit.
- If no station is transmitting, the token continues to circulate along the ring.
- There are two ways to release the token: **early token release** (right after data is sent) and **delayed token release** (after receiving acknowledgment from the recipient)




```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 10.10.10.38

Pinging 10.10.10.38 with 32 bytes of data:

Reply from 10.10.10.38: bytes=32 time=7ms TTL=128
Request timed out.
Request timed out.
Reply from 10.10.10.38: bytes=32 time=8ms TTL=128

Ping statistics for 10.10.10.38:
    Packets: Sent = 4, Received = 2, Lost = 2 (50% loss),
Approximate round trip times in milli-seconds:
    Minimum = 7ms, Maximum = 8ms, Average = 7ms
```

Mesh Topology

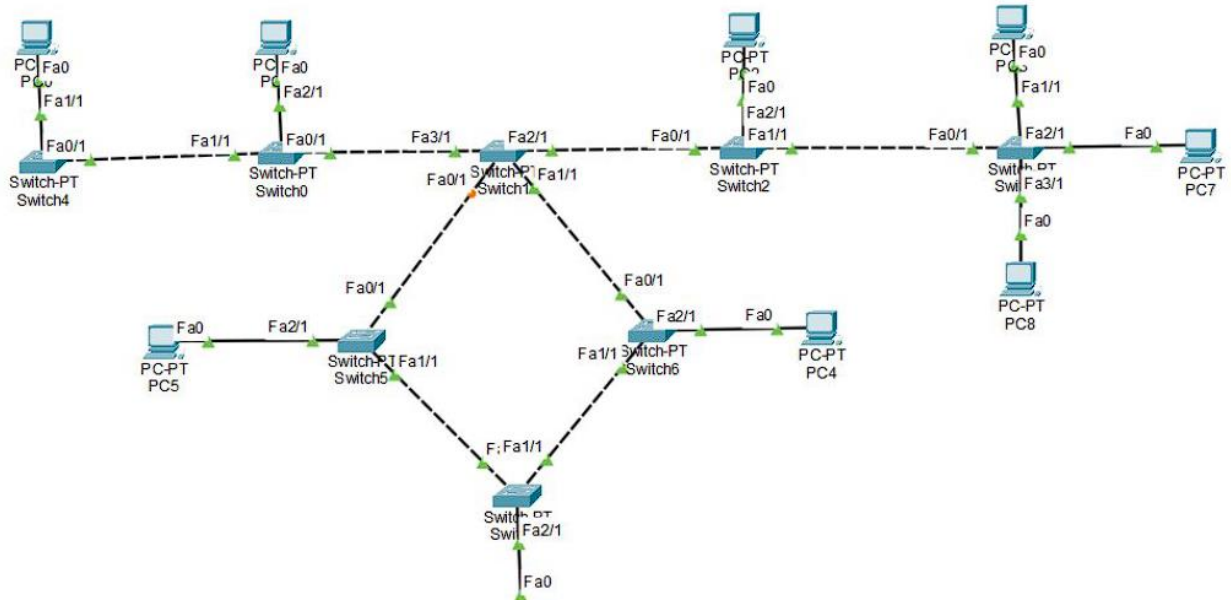
In a mesh topology, every device is connected to another device via a particular channel. Every device is connected to another via dedicated channels. These channels are known as links. In Mesh Topology, the protocols used are AHCP (Ad Hoc Configuration Protocols), DHCP (Dynamic Host Configuration Protocol).Suppose, the N number of devices are connected with each other in a mesh topology, the total number of ports that are required by each device is N-1.The total number of ports required = N * (N-1).




```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 10.10.10.15

Pinging 10.10.10.15 with 32 bytes of data:

Reply from 10.10.10.15: bytes=32 time=16ms TTL=128
Reply from 10.10.10.15: bytes=32 time=8ms TTL=128
Reply from 10.10.10.15: bytes=32 time=8ms TTL=128
Reply from 10.10.10.15: bytes=32 time=8ms TTL=128

Ping statistics for 10.10.10.15:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 8ms, Maximum = 16ms, Average = 10ms
```

Hybrid Topology

Hybrid Topology is the combination of all the various types of topologies we have studied above. Hybrid Topology is used when the nodes are free to take any form. It means these can be individuals such as Ring or Star topology or can be a combination of various types of topologies seen above.




```
Packet Tracer PC Command Line 1.0
C:\>ping 10.10.10.11

Pinging 10.10.10.11 with 32 bytes of data:

Reply from 10.10.10.11: bytes=32 time=1ms TTL=128
Reply from 10.10.10.11: bytes=32 time<1ms TTL=128
Reply from 10.10.10.11: bytes=32 time<1ms TTL=128
Reply from 10.10.10.11: bytes=32 time<1ms TTL=128

Ping statistics for 10.10.10.11:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

2. Network Commands

1. IPCONFIG / IFCONFIG Command

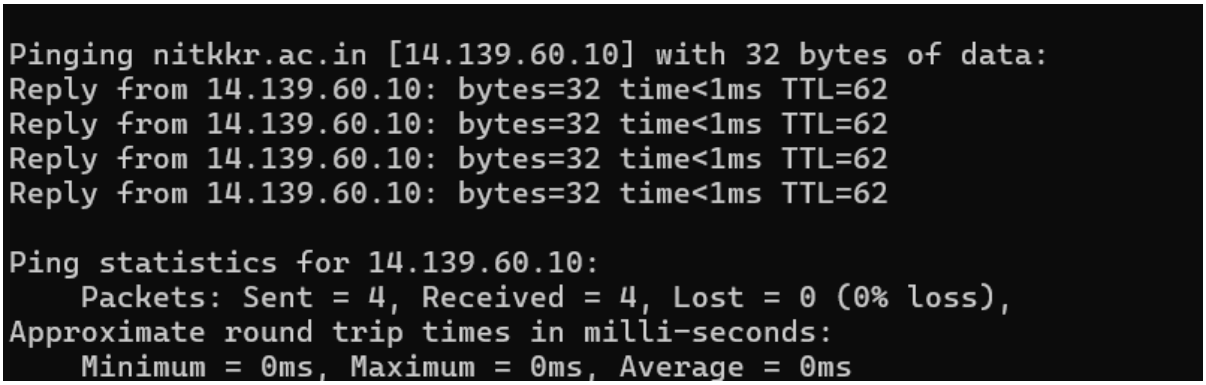
- **Definition:** ipconfig (Windows) and ifconfig (Linux) display and configure network interface details.
- **Working:**
  - Shows IPv4/IPv6 address, subnet mask, and default gateway.
  - Provides information about MAC address and DHCP configuration.
  - Used to troubleshoot connectivity issues.

```
Windows IP Configuration                    Suraj Pandey

Ethernet adapter VEthernet (WSL (Hyper-V firewali):

  Connection-specific DNS Suffix . :
  Link.local IPv6 Address . . . . : fe50::b497:ba7a:3181:b93758
  IPvu Address . . . . . . . . . . : 172.22.176.1
  Subaet Mask . . . . . . . . . . . : 255.255.240.0

Wireless LAN adapter Local Area Connection* 1:

  Media State . . . . . . . . . . . : Media disconnected
  Connection-specific DNS Suffix  :

Wireless LAN adapter Local Area Connection* 2:

  Media State . . . . . . . . . . . :
  Link:local IPv6 Address . . . . : fe50::7444:25b7:3e3c:b2e7%13
  IPvu Address   . . . . . . . . . . : 172.16.116.240
  Default Gateway . . . . . . . . . : 172.16.117.253
```

2. PING Command

- **Definition:** PING (Packet Internet Groper) is used to check connectivity between two hosts.
- **Working:** It uses ICMP Echo Request and Echo Reply messages to measure:
  - Whether the destination host is reachable
  - Round-trip time (latency)
  - Packet loss percentage

Example:



3. TRACEROUTE Command

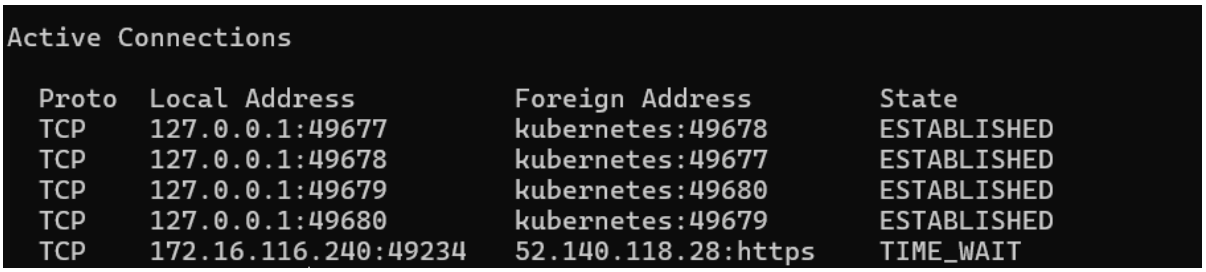
- **Definition:** TRACEROUTE is used to trace the path packets take from source to destination.
- **Working:**
  - It sends packets with increasing TTL (Time-to-Live) values.
  - Each router along the path decreases TTL by 1 and sends back an ICMP message when TTL reaches zero.
  - This way, all routers in the path can be identified.

Example:

```
Tracing route to nitkkr.ac.in [14.139.60.10]
over a maximum of 30 hops:

  1     3 ms     2 ms     2 ms  172.16.117.253
  2    <1 ms    <1 ms    <1 ms  172.16.0.1
  3    <1 ms    <1 ms    <1 ms  14.139.60.10

Trace complete.
```

4. NETSTAT Command

- **Definition:** NETSTAT (Network Statistics) displays active connections, routing tables, listening ports, and interface statistics.
- **Working:**
    - Helps administrators check open ports and running services.
    - Can be used to detect suspicious connections.
    - Displays TCP/UDP statistics and process IDs.



```
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:49677        kubernetes:49678       ESTABLISHED
  TCP    127.0.0.1:49678        kubernetes:49677       ESTABLISHED
  TCP    127.0.0.1:49679        kubernetes:49680       ESTABLISHED
  TCP    127.0.0.1:49680        kubernetes:49679       ESTABLISHED
  TCP    172.16.116.240:49234   52.140.118.28:https    TIME_WAIT
```

5. TELNET Command

- **Definition:**
  TELNET (Teletype Network) is a protocol used for remotely logging into another computer or network device over a network connection, enabling users to interact with the remote device as if they were directly connected via a terminal.

- **How It Works:**

  - TELNET typically operates over **port 23** on the target machine and uses a client-server model where the client initiates the session and the server responds.

  - Once connected, users gain access to a command-line interface on the remote system, allowing for configuration, management, or troubleshooting tasks.

  - All communication with TELNET, including usernames and passwords, occurs in plain, unencrypted text—meaning data can be intercepted by eavesdroppers.

  - Because of this security risk, TELNET is not recommended for sensitive tasks or public networks and has largely been replaced by SSH (Secure Shell), which encrypts communication.

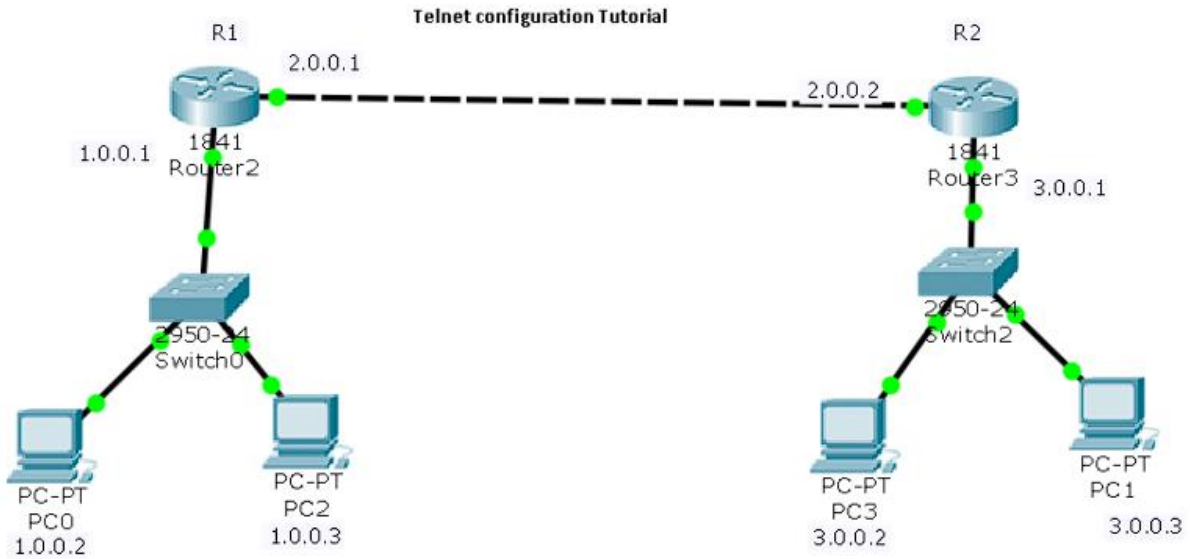
  - TELNET is still sometimes used to manage legacy systems, test network services, or verify if certain TCP ports are open on remote devices

- **Additional Points:**

  - Easily allows remote device management but exposes major vulnerabilities due to its lack of encryption and authentication strength.

  - Mainly found today in specialized, isolated, or legacy environments where newer protocols are unsupported.

  - SSH should be preferred whenever security is required, as it protects credentials and data from interception



IOS Command Line Interface

```
Cisco CISCO2911/K9 (revision 1.0) with 491520K/32768K bytes of memory.
Processor board ID FTX152400KS
3 Gigabit Ethernet interfaces
DRAM configuration is 64 bits wide with parity disabled.
255K bytes of non-volatile configuration memory.
249856K bytes of ATA System CompactFlash 0 (Read/Write)


         --- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]:

Press RETURN to get started!



Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface GigabitEthernet0/0
Router(config-if)#ip address 10.10.10.1 255.0.0.0
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0, changed state to up

Router(config-if)#line vty 0 2
Router(config-line)#login local
Router(config-line)#transport input telnet
Router(config-line)#exit
Router(config)#username admin password 1234
Router(config)#exit
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#configure
Configuring from terminal, memory, or network [terminal]? terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#enable password 456
Router(config)#
```

```
C:\>telnet 10.0.0.1
Trying 10.0.0.1 ...Open


User Access Verification

Username: admin
Password:
Router>enable
Password:
Router#configure
Configuring from terminal, memory, or network [terminal]?
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname R1
R1(config)#exit
R1#exit

[Connection to 10.0.0.1 closed by foreign host]
C:\>
```

3. HTTP And FTP Protocol

HTTP (Hypertext Transfer Protocol)

HTTP (Hypertext Transfer Protocol) is a fundamental protocol used on the web to transfer various types of content such as web pages, images, videos, and other files between a client (usually a web browser) and a web server. It operates on a client-server model where the client sends requests and the server responds with the requested resources or error messages.

Key Features:

- **Stateless Protocol**: HTTP treats each request independently, without retaining any memory of previous interactions, making it scalable and simple. Sessions and state management are typically handled using cookies or other mechanisms outside the core protocol.
- **Client-Server Model**: The client initiates communication by sending requests, and the server processes those requests and returns a response, which might be the requested data or an error message.

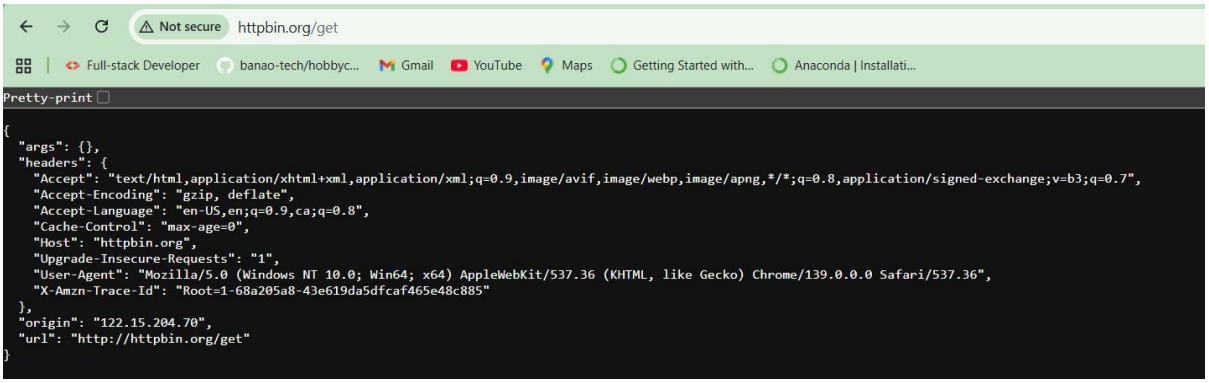
Common HTTP Methods:

- **GET**: Retrieves data from the server without modifying it; commonly used to request web pages or files.
- **HEAD**: Similar to GET but only requests headers, useful for checking resource metadata without downloading the body.
- **POST**: Sends data to the server, often used for submitting forms, and can create or update server-side resources.
- **PUT**: Replaces an existing resource or creates one if it does not exist.
- **DELETE**: Removes a specified resource from the server.
- **OPTIONS**: Requests information about communication options supported by the server for a specific URL or the server itself.

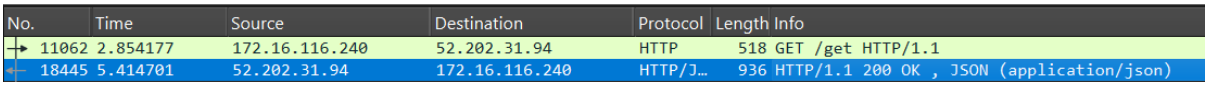
Overall, HTTP facilitates efficient and flexible communication on the web, enabling the seamless exchange of content between clients and servers.

Analyzing HTTP Requests and Responses Using Wireshark

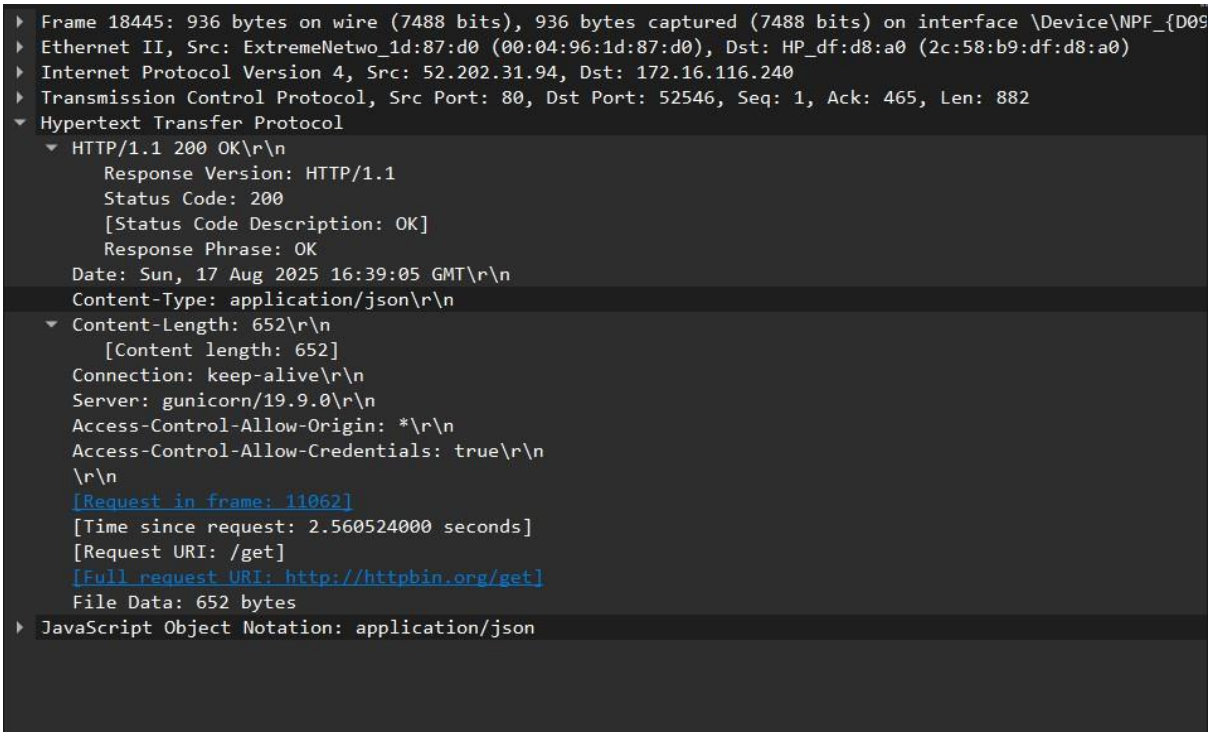
1. Get Request



Captured HTTP GET request in Wireshark:



No.	Time	Source	Destination	Protocol	Length	Info
11062	2.854177	172.16.116.240	52.202.31.94	HTTP	518	GET /get HTTP/1.1
18445	5.414701	52.202.31.94	172.16.116.240	HTTP/J…	936	HTTP/1.1 200 OK , JSON (application/json)



```
▸ Frame 18445: 936 bytes on wire (7488 bits), 936 bytes captured (7488 bits) on interface \Device\NPF_{D09
▸ Ethernet II, Src: ExtremeNetwo_1d:87:d0 (00:04:96:1d:87:d0), Dst: HP_df:d8:a0 (2c:58:b9:df:d8:a0)
▸ Internet Protocol Version 4, Src: 52.202.31.94, Dst: 172.16.116.240
▸ Transmission Control Protocol, Src Port: 80, Dst Port: 52546, Seq: 1, Ack: 465, Len: 882
▾ Hypertext Transfer Protocol
  ▾ HTTP/1.1 200 OK\r\n
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
    Date: Sun, 17 Aug 2025 16:39:05 GMT\r\n
    Content-Type: application/json\r\n
  ▾ Content-Length: 652\r\n
      [Content length: 652]
    Connection: keep-alive\r\n
    Server: gunicorn/19.9.0\r\n
    Access-Control-Allow-Origin: *\r\n
    Access-Control-Allow-Credentials: true\r\n
    \r\n
    [Request in frame: 11062]
    [Time since request: 2.560524000 seconds]
    [Request URI: /get]
    [Full request URI: http://httpbin.org/get]
    File Data: 652 bytes
▸ JavaScript Object Notation: application/json
```

2. Head Request

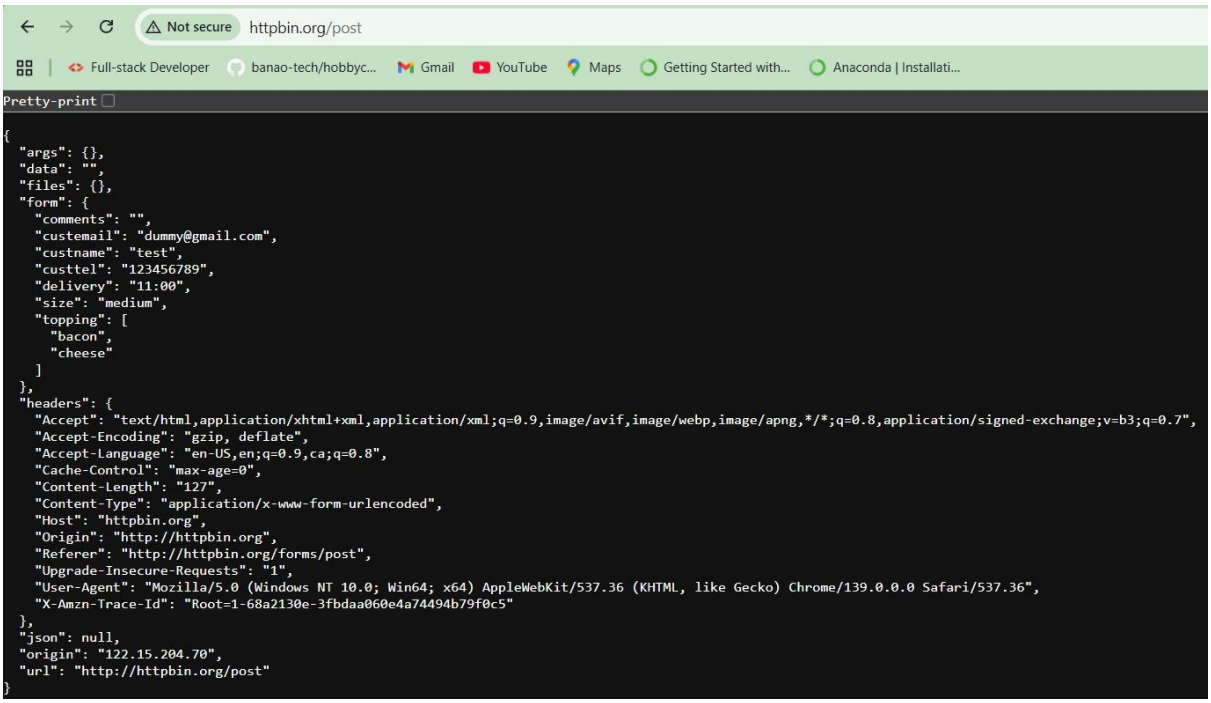
```
HTTP/1.1 200 OK
Date: Sun, 17 Aug 2025 16:44:34 GMT
Content-Type: application/json
Content-Length: 254
Connection: keep-alive
Server: gunicorn/19.9.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
```

Captured HTTP Head request in Wireshark:

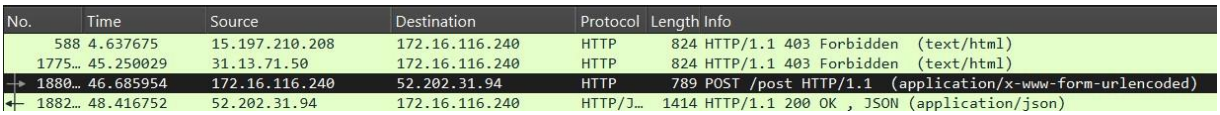
No.	Time	Source	Destination	Protocol	Length	Info
3548	27.871199	172.16.116.240	172.217.26.99	HTTP	293	GET /generate_204 HTTP/1.1
3552	27.890198	172.217.26.99	172.16.116.240	HTTP	212	HTTP/1.1 204 No Content
3620	28.081067	172.16.116.240	54.209.231.157	HTTP	133	HEAD /get HTTP/1.1
3671	28.407781	172.16.116.240	20.47.89.120	HTTP	269	GET /generate_204 HTTP/1.1
3675	28.479941	20.47.89.120	172.16.116.240	HTTP	173	HTTP/1.1 204 No Content
3690	28.631372	54.209.231.157	172.16.116.240	HTTP	284	HTTP/1.1 200 OK

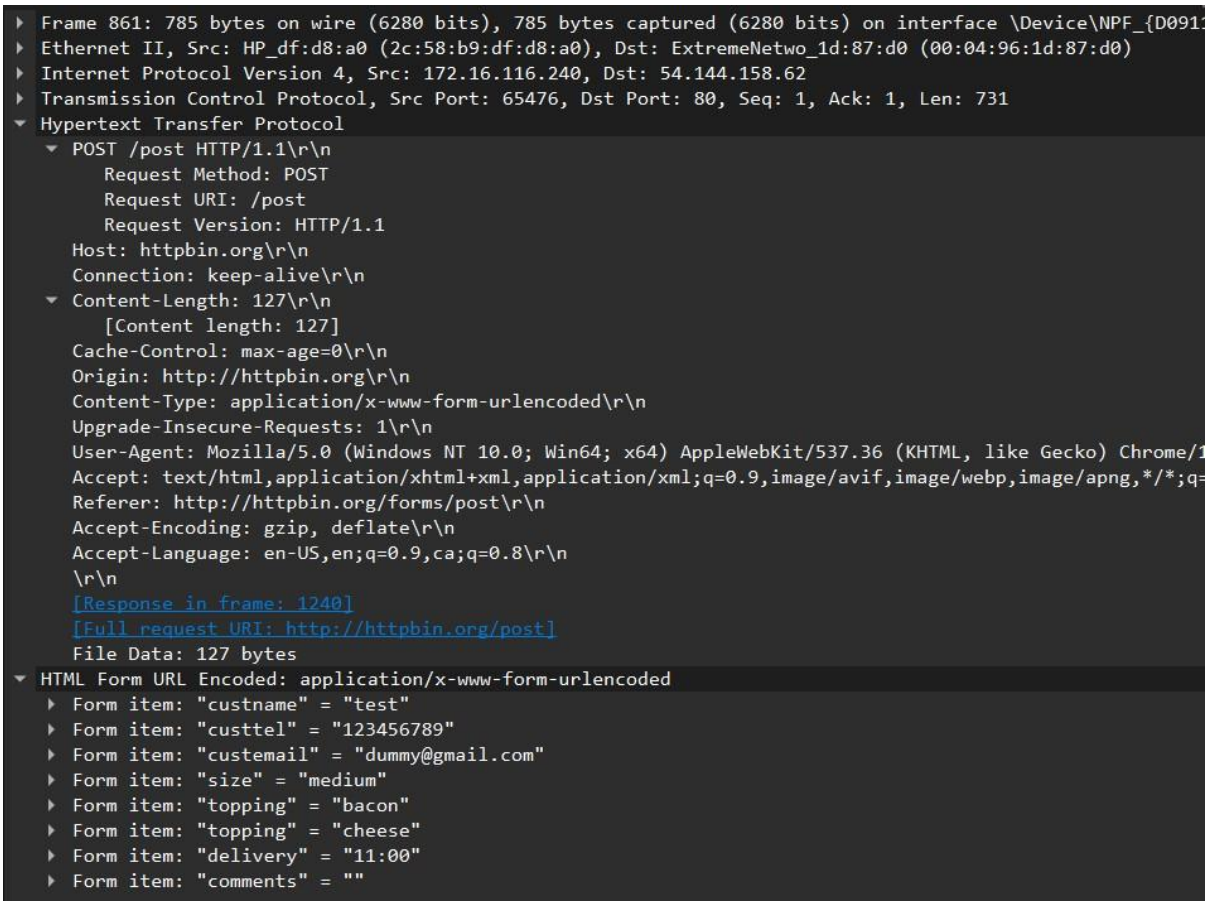
```
▸ Frame 3620: 133 bytes on wire (1064 bits), 133 bytes captured (1064 bits) on interface \Device\NPF_{D091
▸ Ethernet II, Src: HP_df:d8:a0 (2c:58:b9:df:d8:a0), Dst: ExtremeNetwo_1d:87:d0 (00:04:96:1d:87:d0)
▸ Internet Protocol Version 4, Src: 172.16.116.240, Dst: 54.209.231.157
▸ Transmission Control Protocol, Src Port: 64045, Dst Port: 80, Seq: 1, Ack: 1, Len: 79
▾ Hypertext Transfer Protocol
  ▾ HEAD /get HTTP/1.1\r\n
      Request Method: HEAD
      Request URI: /get
      Request Version: HTTP/1.1
    Host: httpbin.org\r\n
    User-Agent: curl/8.14.1\r\n
    Accept: */*\r\n
    \r\n
    [Response in frame: 3690]
    [Full request URI: http://httpbin.org/get]
```

3. Post Request



Captured HTTP POST request in Wireshark:





2.FTP (File Transfer Protocol)

FTP (File Transfer Protocol) is a widely used network protocol designed to transfer files between a client and a server over TCP/IP networks, including the Internet and private networks.

How FTP Operates:

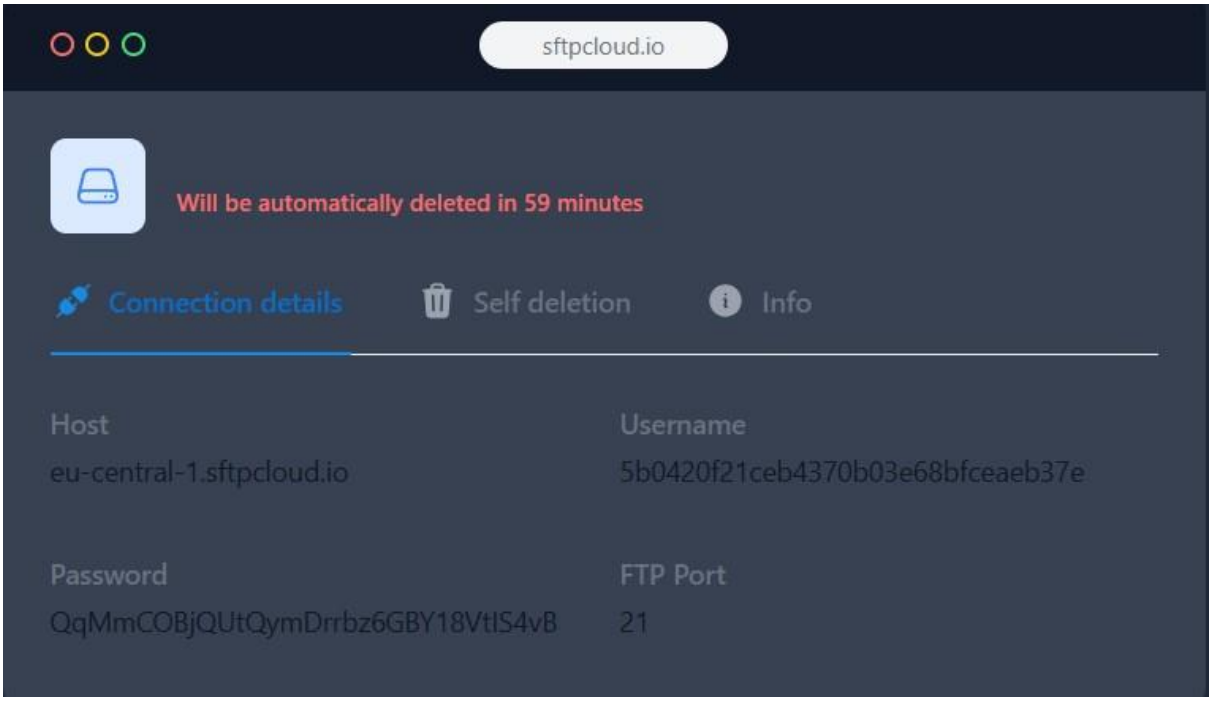
- FTP functions on a **client-server model**: the client initiates communication to upload or download files, while the server manages and responds to these requests.
- FTP uses **two distinct communication channels**:

    1. **Command Channel (Port 21):** This channel handles control information such as user authentication, directory navigation, and commands for file operations like uploading, downloading, renaming, or deleting files.
    2. **Data Channel (Port 20):** This separate channel is specifically used for the actual transfer of file data between the client and server.

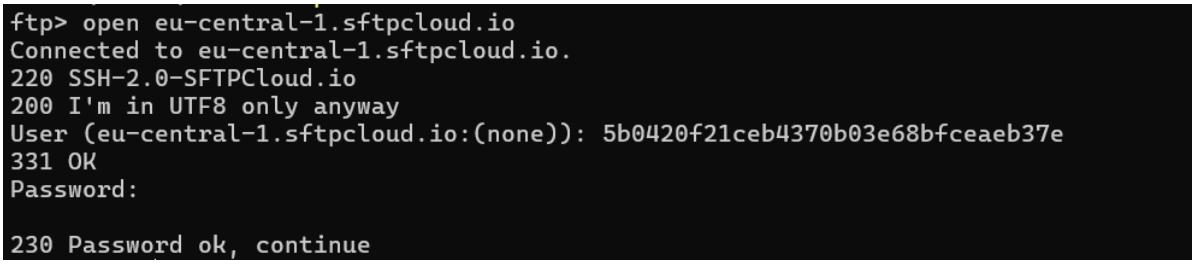
This separation allows FTP to manage commands and data transfers independently, resulting in efficient and organized file transmission. FTP supports both active and passive transfer modes to accommodate different network configurations, especially those with firewalls. Users usually authenticate with a username and password, but some servers allow anonymous access with limited permissions. FTP remains popular for managing large file transfers, website content updates, and remote file operations.

Analyzing FTP Requests and Responses Using Wireshark

Setting up a temporary FTP server:



Connecting to FTP server:



Uploading file to FTP server

```
ftp> put D:\cn\dummy.txt
200 PORT command successful
150 Using transfer connection
226 Closing transfer connection
```

Listing files in FTP server

```
ftp> ls
200 PORT command successful
150 Using transfer connection
dummy.txt
226 Closing transfer connection
ftp: 14 bytes received in 0.00Seconds 14.00Kbytes/sec.
```

Downloading files from FTP server

```
ftp> get dummy.txt Downloads\dummy.txt
200 PORT command successful
150 Using transfer connection
226 Closing transfer connection
ftp: 19 bytes received in 0.05Seconds 0.40Kbytes/sec.
```

Closing connection to FTP server

```
ftp> close
221 Goodbye
ftp> quit
```

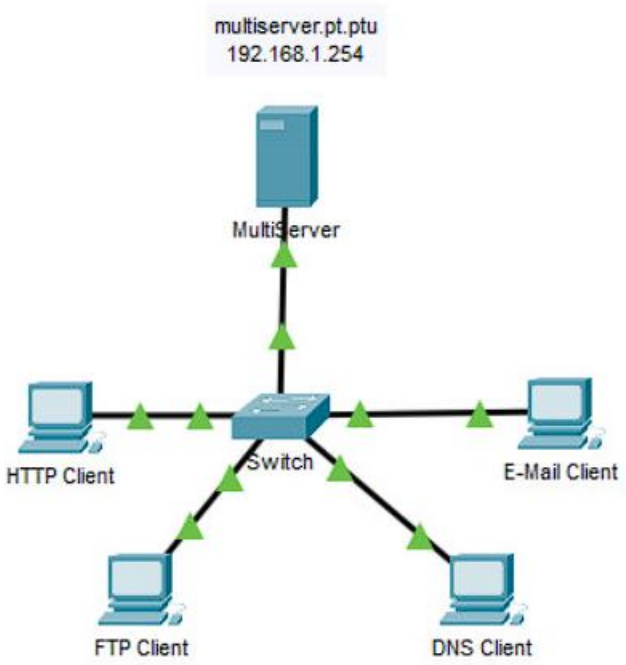
Captured FTP requests in Wireshark

No.	Time	Source	Destination	Protocol	Length	Info
1995	19.410779	172.16.116.240	159.69.223.221	FTP	83	Request: PORT 172,16,116,240,204,176
2028	19.606964	159.69.223.221	172.16.116.240	FTP	83	Response: 200 PORT command successful
2029	19.611188	172.16.116.240	159.69.223.221	FTP	60	Request: NLST
2039	20.106917	159.69.223.221	172.16.116.240	FTP	85	Response: 150 Using transfer connection
2040	20.106917	159.69.223.221	172.16.116.240	FTP	87	Response: 226 Closing transfer connection
7154	75.607745	172.16.116.240	159.69.223.221	FTP	83	Request: PORT 172,16,116,240,249,156
7156	75.803965	159.69.223.221	172.16.116.240	FTP	83	Response: 200 PORT command successful
7157	75.809963	172.16.116.240	159.69.223.221	FTP	70	Request: RETR dummy.txt
7167	76.203020	159.69.223.221	172.16.116.240	FTP	85	Response: 150 Using transfer connection
7168	76.203020	159.69.223.221	172.16.116.240	FTP	87	Response: 226 Closing transfer connection
13501	145.938438	172.16.116.240	159.69.223.221	FTP	82	Request: PORT 172,16,116,240,227,21
13504	146.134958	159.69.223.221	172.16.116.240	FTP	83	Response: 200 PORT command successful
17494	193.374807	172.16.116.240	159.69.223.221	FTP	82	Request: PORT 172,16,116,240,227,22
17519	193.571711	159.69.223.221	172.16.116.240	FTP	83	Response: 200 PORT command successful
17520	193.584975	172.16.116.240	159.69.223.221	FTP	70	Request: RETR dummy.txt
17541	193.977511	159.69.223.221	172.16.116.240	FTP	85	Response: 150 Using transfer connection
17542	193.977511	159.69.223.221	172.16.116.240	FTP	87	Response: 226 Closing transfer connection
24436	260.781108	172.16.116.240	159.69.223.221	FTP	82	Request: PORT 172,16,116,240,227,25
24439	260.977766	159.69.223.221	172.16.116.240	FTP	83	Response: 200 PORT command successful
24440	260.983361	172.16.116.240	159.69.223.221	FTP	70	Request: STOR dummy.txt
24458	261.643463	159.69.223.221	172.16.116.240	FTP	85	Response: 150 Using transfer connection
24492	262.382169	159.69.223.221	172.16.116.240	FTP	87	Response: 226 Closing transfer connection
26958	292.935322	172.16.116.240	159.69.223.221	FTP	83	Request: PORT 172,16,116,240,242,246
26963	293.135920	159.69.223.221	172.16.116.240	FTP	83	Response: 200 PORT command successful
26964	293.142701	172.16.116.240	159.69.223.221	FTP	70	Request: RETR dummy.txt
26979	293.545479	159.69.223.221	172.16.116.240	FTP	85	Response: 150 Using transfer connection
26987	293.598363	159.69.223.221	172.16.116.240	FTP	87	Response: 226 Closing transfer connection
35618	389.874405	172.16.116.240	159.69.223.221	FTP	83	Request: PORT 172,16,116,240,226,205
35623	390.070073	159.69.223.221	172.16.116.240	FTP	83	Response: 200 PORT command successful
35624	390.077989	172.16.116.240	159.69.223.221	FTP	70	Request: RETR dummy.txt
35668	390.470420	159.69.223.221	172.16.116.240	FTP	85	Response: 150 Using transfer connection
35669	390.470420	159.69.223.221	172.16.116.240	FTP	87	Response: 226 Closing transfer connection
55012	585.110098	172.16.116.240	159.69.223.221	FTP	60	Request: QUIT
55026	585.305781	159.69.223.221	172.16.116.240	FTP	67	Response: 221 Goodbye

4.Working with client-server scenario. Observing the difference between UDP and TCP servers.

Client-server communication is a core concept in computer networking where a client requests services or data, and the server responds by providing the requested resources. This interaction typically takes place over transport layer protocols, mainly Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), each with its distinct features and suitability for different applications.

This practical experiment aims to explore the behavioral distinctions between TCP and UDP within a simulated client-server setup using Cisco Packet Tracer. By examining how these protocols handle data transmission in a controlled network environment, one can better understand their strengths, weaknesses, and the scenarios where they are most effective. TCP offers reliable, ordered, and error-checked communication, making it ideal for applications requiring data integrity. In contrast, UDP provides faster, connectionless transmission with lower overhead, favored in time-sensitive uses where some data loss is acceptable, such as streaming or gaming. This comparison enhances comprehension of how protocol choice impacts network performance and application behavior.



Understanding TCP and UDP

TCP (Transmission Control Protocol)

- **Connection-oriented:** Establishes a reliable connection before data transmission through a three-way handshake.
- **Reliability:** Ensures data packets are delivered correctly and in order; lost packets are retransmitted.
- **Ordered delivery:** Data arrives at the destination in the same sequence it was sent.
- **Error checking and recovery:** Uses acknowledgments (ACK) and checksums to detect and correct errors.
- **Flow and congestion control:** Regulates data transmission rates to avoid overwhelming the network.
- **Use cases:** Ideal for applications that require accuracy and completeness, such as:
    - Web browsing (HTTP/HTTPS)
    - Email (SMTP, IMAP, POP3)
    - File transfers (FTP)
- **Higher overhead:** Due to connection setup, reliability, and control mechanisms, TCP uses more bandwidth and processing.
- **Slower speed:** The added reliability mechanisms result in slower transmission compared to UDP.

UDP (User Datagram Protocol)

- **Connectionless:** No formal connection is established before data is sent; packets are sent independently.
- **Unreliable:** No guarantee of packet delivery, ordering, or retransmission of lost packets.
- **No flow or congestion control:** Transmits data as quickly as possible without regulating traffic.
- **Lightweight:** Minimal protocol overhead, resulting in faster transmissions.
- **Best-effort delivery:** Suitable for applications where speed is critical and some data loss is acceptable.
- **Use cases:**
    - Voice over IP (VoIP)
    - Online gaming
    - Live video streaming
    - DNS (Domain Name System)
- **No acknowledgment:** Does not perform error correction or require acknowledgments from the receiver.
- **Faster speed:** Due to reduced control mechanisms and no connection setup, UDP transmits data faster.

Summary

- TCP is preferred where data integrity and reliability are paramount.
- UDP is chosen when low latency and speed are more important than perfect accuracy.

This detailed contrast helps in choosing the suitable protocol based on application needs and network conditions

Client-Server Implementation in Cisco Packet Tracer

Network Setup

In this practical, a simple network was created in Cisco Packet Tracer with the following components:

- One server configured with both **TCP** and **UDP** services.
- Two clients (PCs) connected to the server through a switch or router.
- IP addressing done statically for easy configuration.
- Protocol-specific services enabled:
    - **TCP Service:** Web Server (HTTP)
    - **UDP Service:** DNS Server

Steps Performed

1. **Configured Server:**
    - Assigned IP address (e.g., 192.168.1.1)
    - Enabled HTTP and DNS services.
2. **Configured Clients:**
    - Assigned IP addresses (e.g., 192.168.1.2 and 192.168.1.3)
    - Configured DNS settings to point to server IP
    - Used the browser to access the server's web page (TCP)
    - Used the command prompt to perform DNS lookup (UDP)
3. **Captured Packet Flow:**
    - Used the simulation mode to capture and analyze TCP and UDP packets.
    - Observed the three-way handshake in TCP.
    - Observed the single-request-response behavior in UDP.