



Department of Computer Application

National Institute of Technology Kurukshetra, Kurukshetra -136119



Computer Networks

MCA-231

(2024-27)

Submitted by:

Name: Suraj Pandey

Roll no: 524110023

Semester: 3rd

Section: Group 2

Submitted to:

Name: Dr. Munish Bhatia

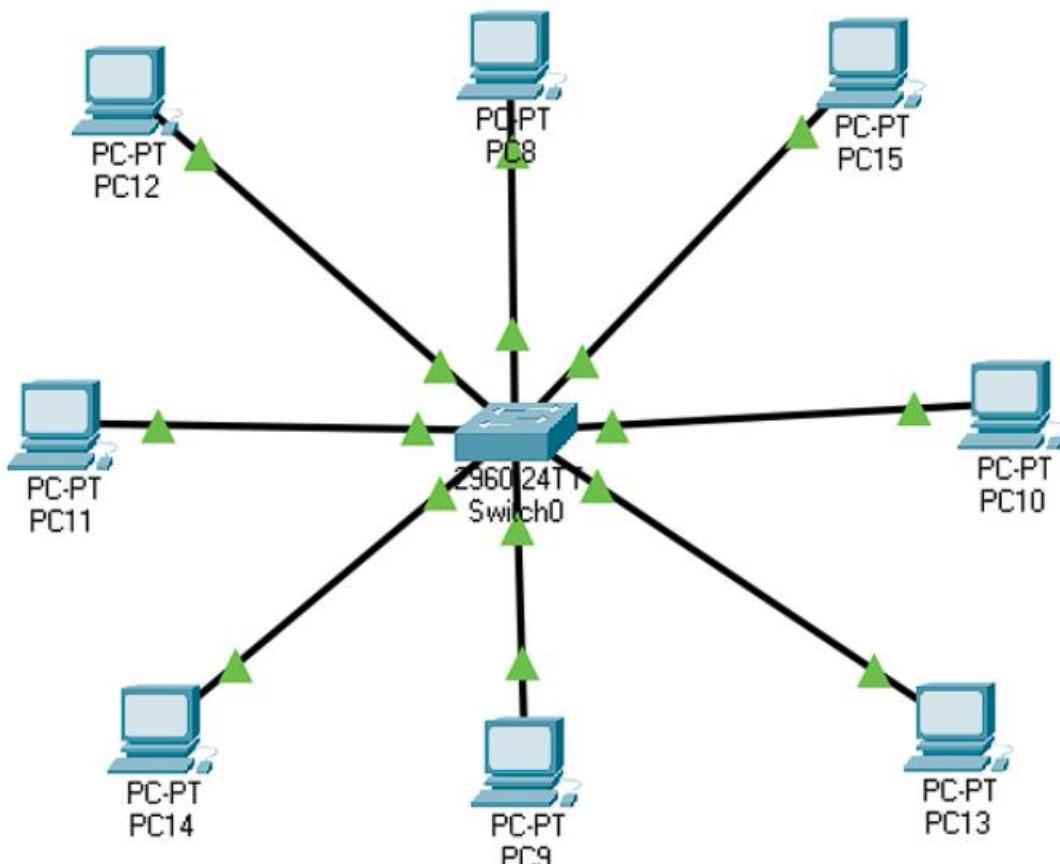
Date: 19-11-2025



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:

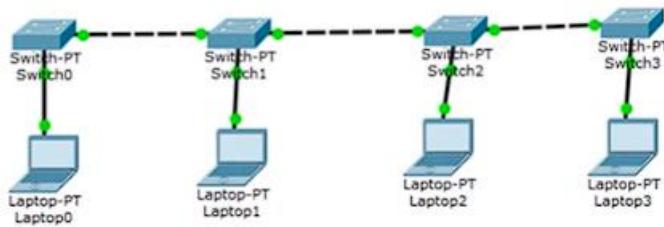
Reply from 10.10.10.7: bytes=32 time=1ms TTL=128
Reply from 10.10.10.7: bytes=32 time=4ms TTL=128
Reply from 10.10.10.7: bytes=32 time=4ms TTL=128
Reply from 10.10.10.7: bytes=32 time=4ms TTL=128

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
```



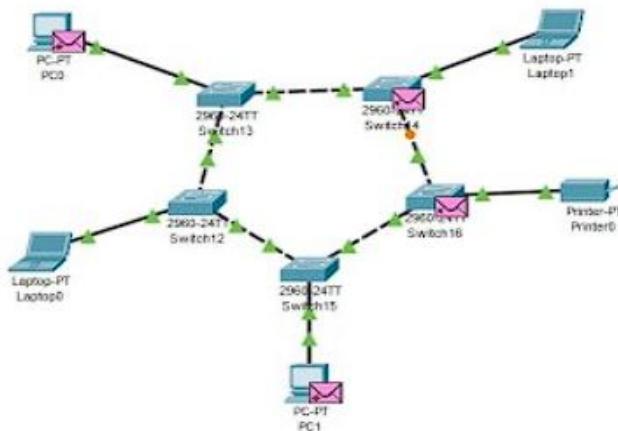
Ring Topology

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:

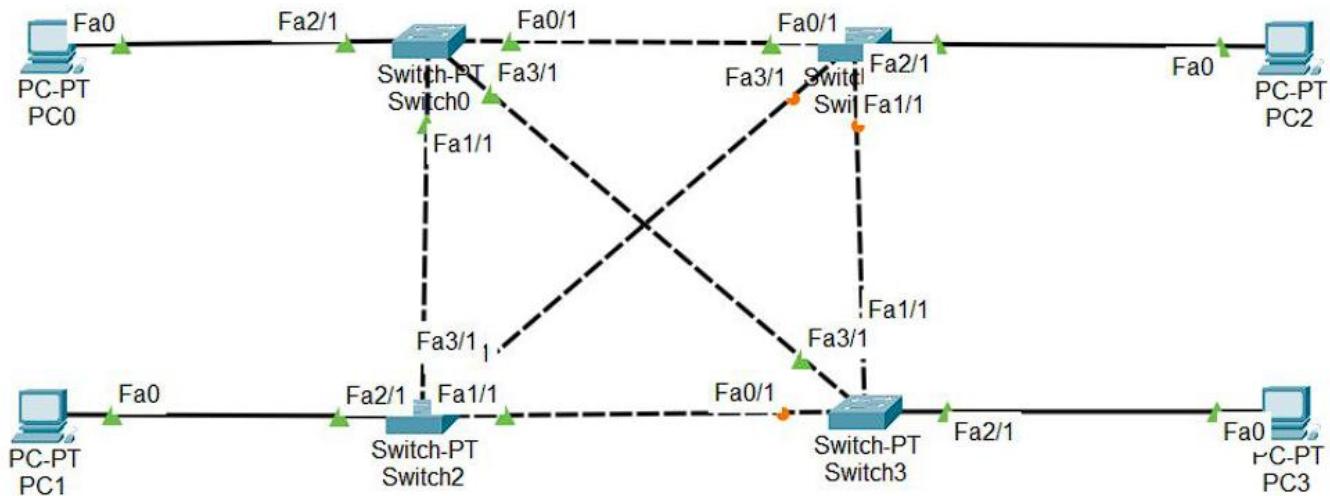
Reply from 10.10.10.38: bytes=32 time=7ms TTL=128
Request timed out.
Request timed out.
Reply from 10.10.10.38: bytes=32 time=8ms TTL=128

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:

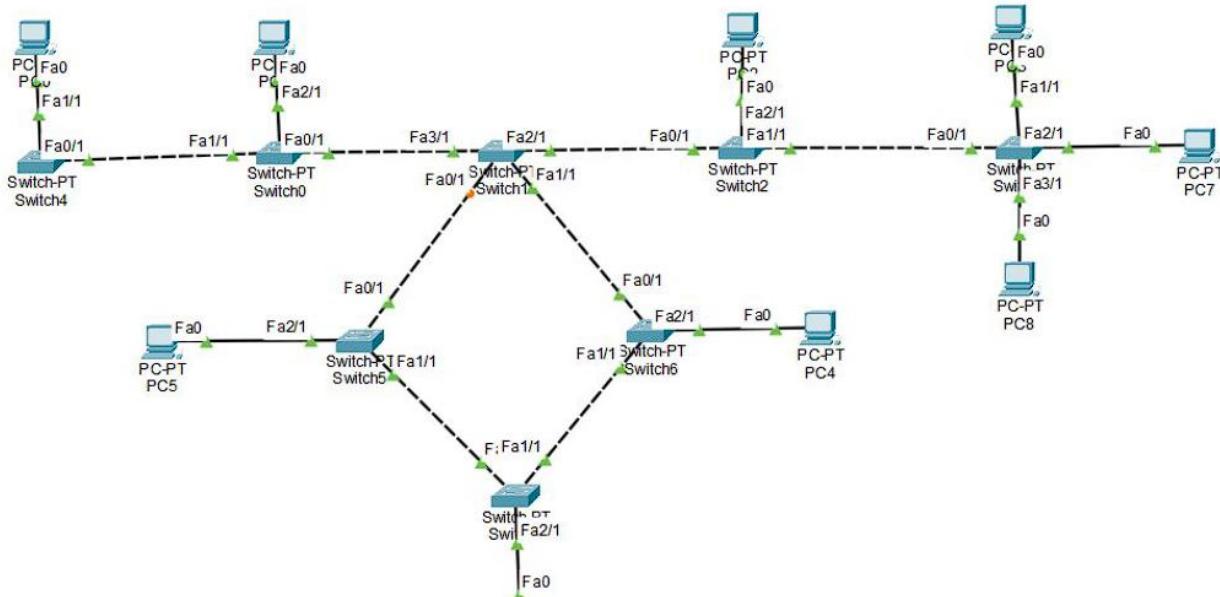
Reply from 10.10.10.15: bytes=32 time=16ms TTL=128
Reply from 10.10.10.15: bytes=32 time=8ms TTL=128
Reply from 10.10.10.15: bytes=32 time=8ms TTL=128
Reply from 10.10.10.15: bytes=32 time=8ms TTL=128

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
```



1. 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 firewall)):

```
Connection-specific DNS Suffix . :  
Link.local IPv6 Address . . . . . : fe50::b497:ba7a:3181:b93758  
IPv4 Address . . . . . : 172.22.176.1  
Subnet 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  
IPv4 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:

```
Pinging nitkkr.ac.in [14.139.60.10] with 32 bytes of data:  
Reply from 14.139.60.10: bytes=32 time<1ms TTL=62  
  
Ping statistics for 14.139.60.10:  
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
    Approximate round trip times in milli-seconds:  
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```



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:**

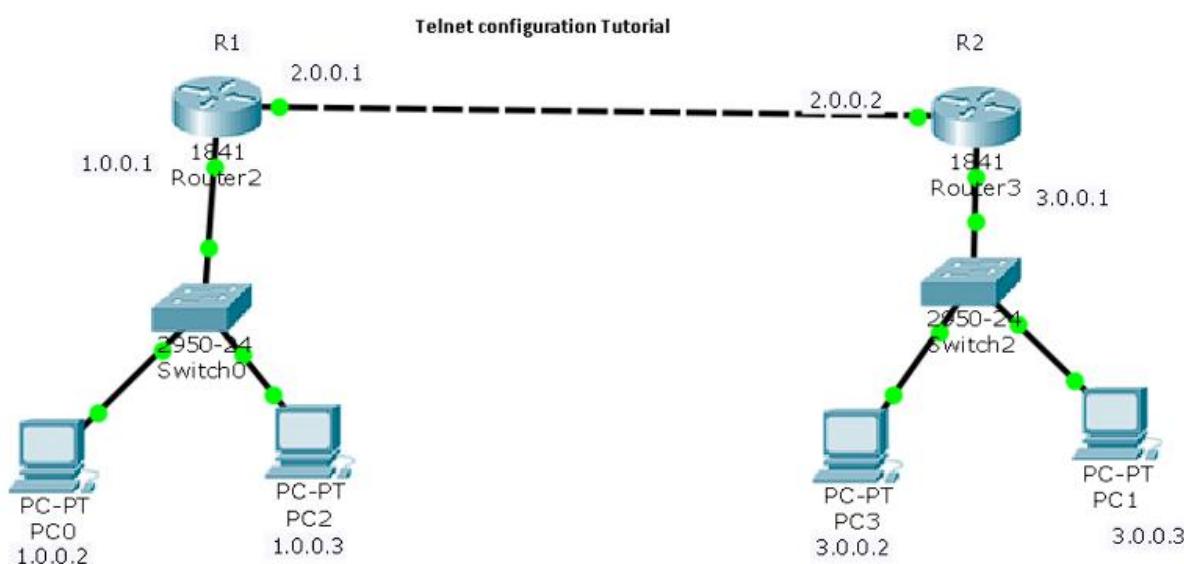
TELNET (Teletype Network) is a protocol used for remotely logging into another computer or network device over a network connection, enabling users to interact with the remote device as if they were directly connected via a terminal.

- **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:\>
```



2. 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

```
{
  "args": {},
  "headers": {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
    "Accept-Encoding": "gzip, deflate",
    "Accept-Language": "en-US,en;q=0.9,ca;q=0.8",
    "Cache-Control": "max-age=0",
    "Host": "httpbin.org",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36",
    "X-Amzn-Trace-Id": "Root=1-68a205a8-43e619da5dfcaf465e48c885"
  },
  "origin": "122.15.204.70",
  "url": "http://httpbin.org/get"
}
```



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
```

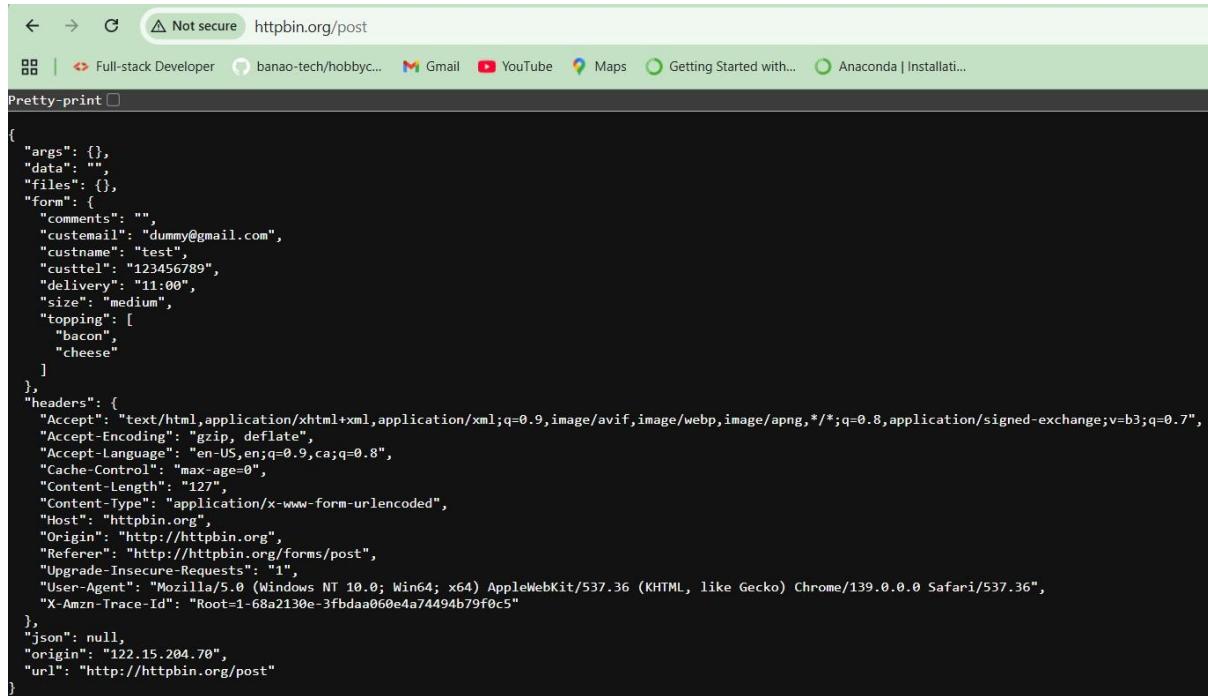
Captured HTTP Head request in Wireshark:

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]
```



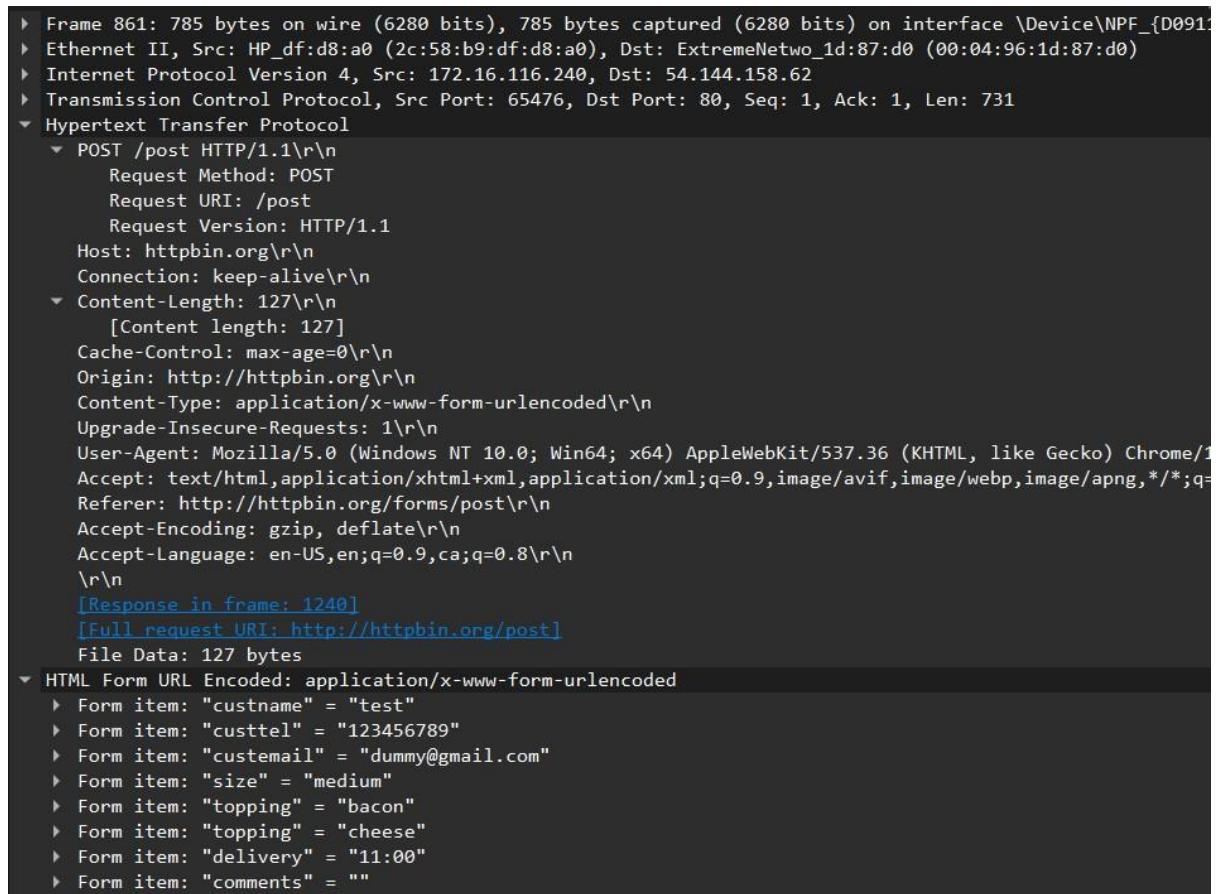
3. Post Request



```
{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "comments": "",
    "custemail": "dummy@gmail.com",
    "custname": "test",
    "custtel": "123456789",
    "delivery": "11:00",
    "size": "medium",
    "topping": [
      "bacon",
      "cheese"
    ]
  },
  "headers": {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
    "Accept-Encoding": "gzip, deflate",
    "Accept-Language": "en-US,en;q=0.9,ca;q=0.8",
    "Cache-Control": "max-age=0",
    "Content-Length": "127",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "Origin": "http://httpbin.org",
    "Referer": "http://httpbin.org/forms/post",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36",
    "X-Amzn-Trace-Id": "Root=1-68a2130e-3fbdaa060e4a/4494b79f0c5"
  },
  "json": null,
  "origin": "122.15.204.70",
  "url": "http://httpbin.org/post"
}
```

Captured HTTP POST request in Wireshark:

No.	Time	Source	Destination	Protocol	Length Info
588	4.637675	15.197.210.208	172.16.116.240	HTTP	824 HTTP/1.1 403 Forbidden (text/html)
1775...	45.250029	31.13.71.50	172.16.116.240	HTTP	824 HTTP/1.1 403 Forbidden (text/html)
→ 1880...	46.685954	172.16.116.240	52.202.31.94	HTTP	789 POST /post HTTP/1.1 (application/x-www-form-urlencoded)
← 1882...	48.416752	52.202.31.94	172.16.116.240	HTTP/J...	1414 HTTP/1.1 200 OK , JSON (application/json)



```
Frame 861: 785 bytes on wire (6280 bits), 785 bytes captured (6280 bits) on interface \Device\NPF_{D0911
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: 54.144.158.62
Transmission Control Protocol, Src Port: 65476, Dst Port: 80, Seq: 1, Ack: 1, Len: 731
└ Hypertext Transfer Protocol
  └ POST /post HTTP/1.1\r\n
    Request Method: POST
    Request URI: /post
    Request Version: HTTP/1.1
    Host: httpbin.org\r\n
    Connection: keep-alive\r\n
    Content-Length: 127\r\n
      [Content length: 127]
    Cache-Control: max-age=0\r\n
    Origin: http://httpbin.org\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/1
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=
    Referer: http://httpbin.org/forms/post\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: en-US,en;q=0.9,ca;q=0.8\r\n
    \r\n
    [Response in frame: 1240]
    [Full request URI: http://httpbin.org/post]
    File Data: 127 bytes
  └ HTML Form URL Encoded: application/x-www-form-urlencoded
    └ Form item: "custname" = "test"
    └ Form item: "custtel" = "123456789"
    └ Form item: "custemail" = "dummy@gmail.com"
    └ Form item: "size" = "medium"
    └ Form item: "topping" = "bacon"
    └ Form item: "topping" = "cheese"
    └ Form item: "delivery" = "11:00"
    └ Form item: "comments" = ""
```



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:

The screenshot shows a web application for setting up a temporary FTP server. At the top, there are three colored circles (red, green, blue) and the URL 'sftpcloud.io'. Below the URL is a small icon of a server and the text 'Will be automatically deleted in 59 minutes'. There are three tabs: 'Connection details' (which is selected), 'Self deletion', and 'Info'. Under 'Connection details', the following fields are visible:

Host	Username
eu-central-1.sftpcloud.io	5b0420f21ceb4370b03e68bfceaeb37e

Under 'Password', the value is 'QqMmCOBjQUtQymDrrbz6GBY18VtlS4vB'. Under 'FTP Port', the value is '21'.



Connecting to FTP server:

```
ftp> open eu-central-1.sftpcloud.io
Connected to eu-central-1.sftpcloud.io.
220 SSH-2.0-SFTPCloud.io
200 I'm in UTF8 only anyway
User (eu-central-1.sftpcloud.io:(none)): 5b0420f21ceb4370b03e68bfceaeab37e
331 OK
Password:
230 Password ok, continue
```



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

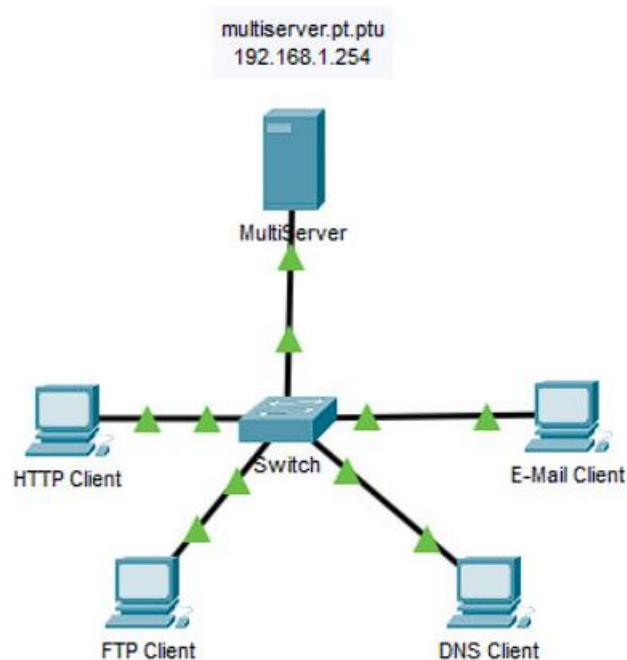
No.	Time	Source	Destination	Protocol	Length Info
1995	19.410779	172.16.116.240	159.69.223.221	FTP	83 Request: PORT 172,16,116,240,204,176
2028	19.606964	159.69.223.221	172.16.116.240	FTP	83 Response: 200 PORT command successful
2029	19.611188	172.16.116.240	159.69.223.221	FTP	60 Request: NLST
2039	20.106917	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	260.781198	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.

6. Client Server program using TCP Socket

Introduction

- Transmission Control Protocol (TCP) is one of the core protocols of the Internet Protocol Suite, responsible for reliable communication between devices.
- In Cisco Packet Tracer, a TCP client–server simulation demonstrates how two devices exchange data over a network using TCP.
- The setup involves a Client PC that initiates a connection and a Server that provides a TCP-based service such as HTTP, FTP, or custom TCP applications.
- Using Simulation Mode, the entire TCP communication process, including the three-way handshake, data transfer, and connection termination, can be observed in detail.

Objective

- To establish a TCP client–server communication using Cisco Packet Tracer.
- To understand the three-way handshake process (SYN, SYN-ACK, ACK).
- To visualize data transmission and acknowledgment between client and server.
- To study the behaviour of TCP segments and PDUs across different OSI layers.

Devices and Components Required

1. Client Device – PC0 (acts as the TCP Client).
2. Server Device – Server0 (configured to run a TCP-based service).
3. Networking Device – One Switch to interconnect devices.
4. Cables – Copper Straight-Through Cables for connections.
5. Software Tool – Cisco Packet Tracer (for simulation and analysis).

Network Topology Description

1. Client (PC0) and Server (Server0) are connected to a Switch using straight-through cables.
2. Each device is assigned a unique IP address within the same subnet (e.g., 192.168.1.0/24).
 - Client IP: 192.168.1.2
 - Server IP: 192.168.1.3
3. The Server is configured to run a TCP service such as HTTP (port 80) or FTP (port 21).
4. The Client initiates a TCP connection to the Server using its IP address and the appropriate port.
5. The Switch facilitates communication between both devices at Layer 2.

Steps to Configure the Setup

1. Open Cisco Packet Tracer and create a new project.
2. Place the required devices (1 PC, 1 Server, and 1 Switch) onto the workspace.
3. Connect the devices using Copper Straight-Through Cables.
4. Assign IP addresses:
 - o On the Client PC, go to *Desktop* → *IP Configuration* and assign 192.168.1.2.
 - o On the Server, assign 192.168.1.3.
 - o Use subnet mask 255.255.255.0 for both.
5. Configure TCP Service on the Server:
 - o Go to *Config* → *Services* and enable HTTP, FTP, or any desired TCP service.
6. Initiate Connection from Client:
 - o Use *Web Browser* or *Command Prompt* (*ping, ftp, etc.*) to connect to 192.168.1.3.
7. Switch to Simulation Mode:
 - o Observe the TCP three-way handshake process (SYN → SYN-ACK → ACK).
 - o Watch the PDU information to understand what happens at each OSI layer.

TCP Communication Process

1. Connection Establishment (Three-Way Handshake):
 - o Client sends SYN (synchronize) packet to Server to start communication.
 - o Server responds with SYN-ACK acknowledging the request.
 - o Client sends ACK to confirm, completing the connection setup.
2. Data Transmission:
 - o Once the connection is established, the client and server exchange data packets.
 - o TCP ensures reliable delivery, sequencing, and error correction.
3. Connection Termination:
 - o Either side (client or server) can initiate termination by sending a FIN packet.
 - o The other side responds with ACK and sends its own FIN, followed by a final ACK, completing a four-step termination process.

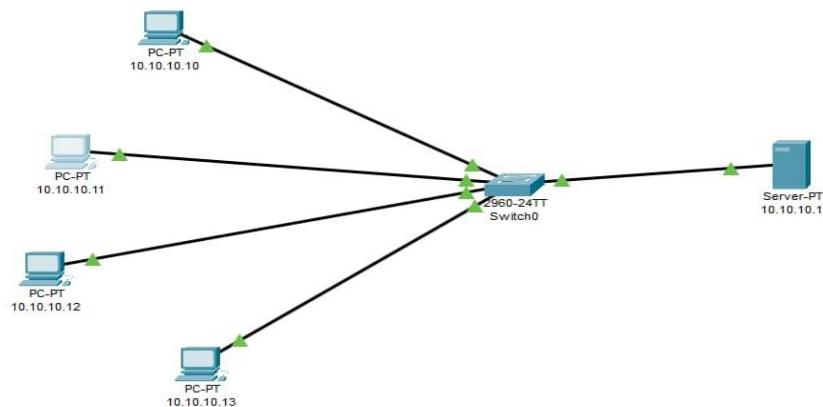
Observations in Simulation Mode

- Each packet can be inspected to view layer-wise encapsulation (Application → Transport → Network → Data Link → Physical).
- The TCP header fields such as Source Port, Destination Port, Sequence Number, and Acknowledgment Number are visible.
- You can see TCP flags (SYN, ACK, FIN) in the PDU details during handshake and termination.

- The flow of data is shown visually, helping to understand how TCP manages communication reliably.

Conclusion

- This experiment demonstrates the basic functioning of TCP in a client–server network.
- The three-way handshake ensures a reliable connection before data transfer begins.
- Using Cisco Packet Tracer's Simulation Mode, users can observe the entire TCP communication lifecycle, understand PDU structures, and visualize how data reliability and flow control are maintained.
- Such practical simulation enhances understanding of real-world networking concepts and TCP/IP communication.



```

10.10.10.11

Physical Config Desktop Programming Attributes

Command Prompt

Packet Tracer PC Command Line 1.0
C:\>ftp demo.com
Trying to connect...demo.com
Connected to demo.com
220- Welcome to PT Ftp server
Username:Username
331- Username ok, need password
Password:
230- Logged in
(passive mode On)
ftp>get asa842-k8.bin

Reading file asa842-k8.bin from demo.com:
File transfer in progress...

[Transfer complete - 5571584 bytes]

5571584 bytes copied in 12.444 secs (102588 bytes/sec)
ftp>

C:\>ipconfig 10.10.10.11 255.0.0.0
C:\>ping 10.10.10.1

Pinging 10.10.10.1 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 10.10.10.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\>

```

7. Decentralized application like a Peer-to-Peer system

Introduction

- A Peer-to-Peer (P2P) network is a decentralized communication model in which each device (peer) can act as both a client and a server.
- Unlike the client-server model, there is no central server; all peers share resources and communicate directly with each other.
- In Cisco Packet Tracer, this setup can be simulated using multiple PCs that connect directly or through a switch, enabling direct data exchange and resource sharing.
- This simulation helps visualize how decentralized systems operate, where every device has equal privileges in communication and data sharing.

Objective

- To design and simulate a Peer-to-Peer network using Cisco Packet Tracer.
- To understand how devices communicate directly without a central server.
- To configure IP addressing for peers within the same subnet.
- To test connectivity and file/resource sharing between peers.

Devices and Components Required

1. End Devices – Two or more PCs (representing peers).
2. Networking Device (Optional) – A Switch (if connecting more than two peers).
3. Cables –
 - Copper Crossover Cable – for direct connection between two PCs.
 - Copper Straight-Through Cable – for connecting devices through a switch.
4. Software Tool – Cisco Packet Tracer (for network design and simulation).

Network Topology Overview

- Each PC acts as an independent peer, capable of sending and receiving data.
- Devices are directly connected using crossover cables or through a switch if more peers are added.
- All peers are assigned unique static IP addresses within the same subnet.
- The topology does not require a central server; instead, communication occurs directly between peers.

Steps to Configure the Peer-to-Peer Network

Step 1: Launch Packet Tracer

- Open Cisco Packet Tracer and create a new project workspace.
- Save the file with an appropriate name (e.g., P2P_Network_Simulation(pkt)).

Step 2: Add Devices

- From the End Devices section, drag and drop two or more PCs onto the workspace.
- Each PC will represent an individual peer in the network.

Step 3: Connect the Devices

- For two PCs only, use a Copper Crossover Cable to connect them directly (PC0 ↔ PC1).
- For three or more PCs, use a Switch and Copper Straight-Through Cables to connect all devices.
- Ensure all devices are linked properly — green dots on interfaces indicate active connections.

Step 4: Assign IP Addresses

- Click on each PC and open the Config tab → FastEthernet0 interface.
- Assign IP addresses manually, keeping them in the same subnet:
 - PC0 → 192.168.1.10
 - PC1 → 192.168.1.20
 - (Optional additional peers can use 192.168.1.30, 192.168.1.40, etc.)
- Use a subnet mask of 255.255.255.0 for all peers.

Step 5: Test Connectivity

- Open the Command Prompt on one PC (*Desktop* → *Command Prompt*).
- Use the ping command to test communication:
- ping 192.168.1.20
- A successful reply confirms proper connectivity.
- Repeat the ping test between all peers to verify full communication within the network.

Step 6: (Optional) Enable File Sharing

- On each PC, open the Desktop tab and select File Sharing (if available).
- Create shared folders and enable access permissions.
- This simulates resource sharing between peers, representing how files or data can be exchanged in real P2P systems.

Observations

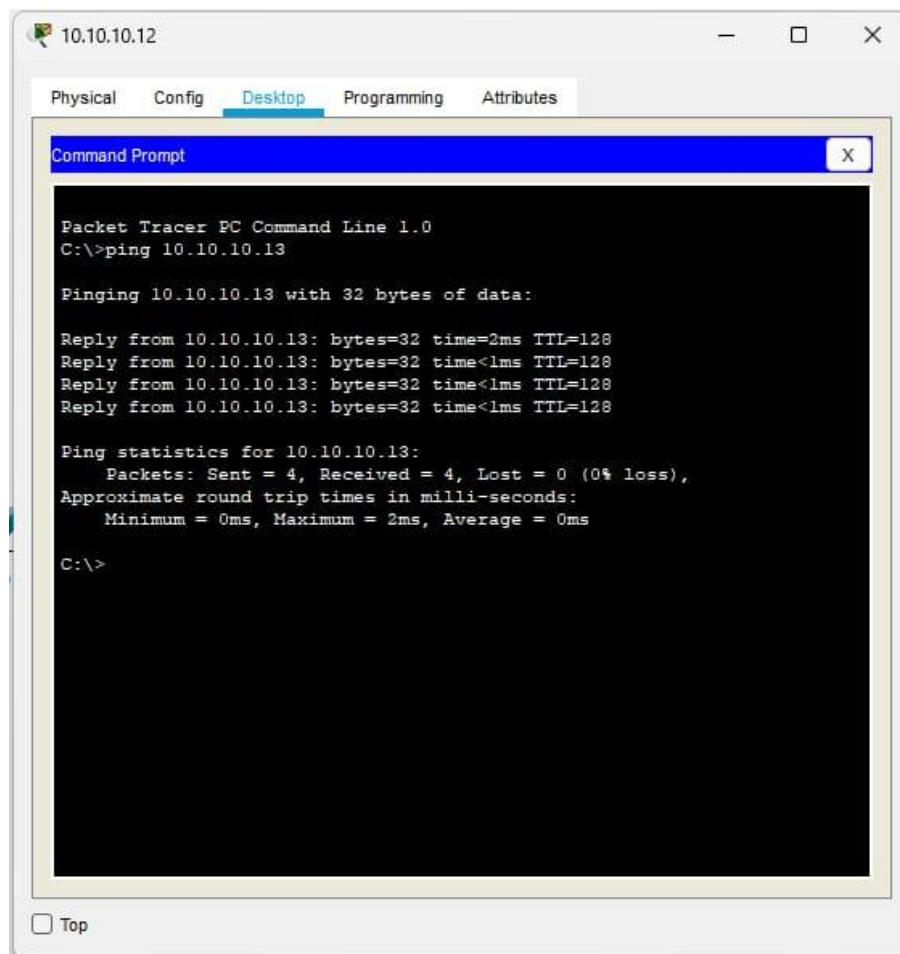
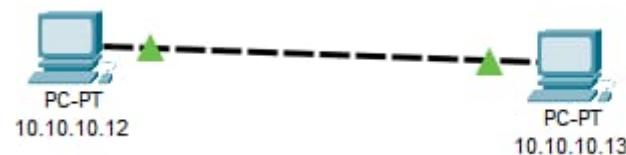
- All peers can communicate directly using their assigned IP addresses.
- The ping test verifies bidirectional connectivity and proper IP configuration.
- When file sharing is enabled, resources can be exchanged without any centralized control.
- Adding more PCs via a switch maintains network scalability while preserving the peer-to-peer nature.

Notes

- This setup illustrates the core principle of decentralization — no single point of control.
- The network can be expanded by adding more peers or integrating additional sharing applications.
- The model resembles real-world P2P systems such as BitTorrent, blockchain nodes, and LAN file-sharing setups.
- Cisco Packet Tracer's Simulation Mode can be used to visualize packet movement between peers, providing a clear understanding of how P2P communication works.

Conclusion

- The simulation successfully demonstrates a decentralized peer-to-peer network using Cisco Packet Tracer.
- Each peer operates independently, allowing direct communication and resource sharing.
- The configuration shows how P2P systems eliminate the need for centralized servers, emphasizing equal control and collaboration among nodes.
- This practical experiment provides insight into distributed networking concepts used in modern decentralized applications and file-sharing systems.



8. Open source firewall/proxy packages like iptables, ufw, squid etc.

Introduction

- Cisco Packet Tracer is a powerful network simulation tool developed by Cisco for building and testing virtual network topologies.
- While it offers an extensive range of Cisco routers, switches, and firewalls, it does not support direct installation of third-party open-source software such as iptables, ufw, or Squid proxy.
- Instead, Packet Tracer focuses on Cisco IOS (Internetwork Operating System) configurations that can simulate similar functionalities using Access Control Lists (ACLs), firewall modules, and policy-based routing.

Understanding Limitations

- Packet Tracer is designed specifically for Cisco-based environments and does not emulate non-Cisco operating systems, such as Linux.
- Therefore, software firewalls like iptables and ufw, or proxy applications like Squid, cannot be installed or executed within this simulator.
- However, users can still replicate similar behaviors through Cisco IOS features, achieving the same conceptual understanding of firewall and proxy mechanisms.

Linux-Based Firewalls (iptables and ufw)

a. Overview

- iptables and ufw (Uncomplicated Firewall) are popular Linux firewall tools used to control inbound and outbound network traffic based on defined rules.
- They work by managing the netfilter framework in Linux to filter, allow, or block packets.

b. Incompatibility in Packet Tracer

- Since Packet Tracer does not support Linux-based virtual machines, iptables or ufw cannot be installed or executed.
- No command-line shell for Linux or package installation is available within the tool.

c. Cisco Equivalent

- In Cisco environments, Access Control Lists (ACLs) perform similar functions as iptables or ufw.
- ACLs can be applied to router interfaces to filter network traffic based on source/destination IPs, ports, and protocols.
- Through ACL configuration, you can:
 - Permit or deny packets.
 - Create inbound or outbound filtering rules.
 - Simulate security behavior equivalent to a software firewall.

d. Example Configuration

```
Router(config)# access-list 100 deny tcp any any eq 80
```

```
Router(config)# access-list 100 permit ip any any
```

```
Router(config)# interface gig0/0
```

```
Router(config-if)# ip access-group 100 in
```

- This example blocks HTTP (port 80) traffic while allowing all other packets—similar to a basic iptables rule.

Squid Proxy Server Simulation

a. Overview

- Squid is a popular open-source proxy server used for caching, content filtering, and web access control on Linux systems.
- It supports HTTP, HTTPS, and FTP protocols and helps improve performance and control user access.

b. Limitations in Packet Tracer

- Packet Tracer does not emulate Linux operating systems, and therefore Squid cannot be installed or configured.
- The simulator lacks the ability to run third-party proxy software or emulate application-level caching and filtering.

c. Cisco Alternatives

- Cisco routers and switches support WCCP (Web Cache Communication Protocol), which can be used to integrate real Squid proxies in actual networks.
- In Packet Tracer, you can mimic proxy-like behavior by using:
 - Route-maps and policy-based routing to redirect traffic.
 - QoS (Quality of Service) and Access Control Lists to simulate traffic control or filtering.
- However, true proxy caching, authentication, or web filtering cannot be demonstrated within Packet Tracer alone.

How to Simulate Firewall and Proxy Concepts in Packet Tracer

Firewall Simulation:

1. Use Cisco routers or ASA firewalls available in Packet Tracer.
2. Configure Standard or Extended Access Control Lists (ACLs) to permit or block traffic.
3. Apply the ACLs to interfaces in the inbound or outbound direction.
4. Observe how filtered traffic behaves during simulation mode.

Proxy Simulation:

1. Use policy-based routing to redirect packets through a simulated “proxy router.”
2. Configure static routes to represent traffic redirection.

3. Apply ACLs to simulate filtering rules.
4. Understand that this only conceptually represents proxy behavior and not true caching or authentication.

Key Takeaways

- Packet Tracer focuses on Cisco IOS configurations, not third-party software integration.
- Firewall simulation is fully possible using ACLs on Cisco routers and ASA firewalls.
- Proxy server behavior can only be approximated, not fully emulated.
- For hands-on practice with iptables, ufw, or Squid, you need a real or virtual Linux environment (e.g., VirtualBox, GNS3, or EVE-NG).

Conclusion

- Although Cisco Packet Tracer does not support installing open-source packages like iptables, ufw, or Squid, it provides alternative ways to simulate their concepts using IOS features.
- By using Access Control Lists (ACLs), firewall appliances, and policy-based configurations, learners can effectively understand and visualize firewall and proxy operations.
- For more advanced and realistic proxy or Linux-based firewall experiments, external virtual labs should be used alongside Packet Tracer.
- This combination ensures a complete grasp of both Cisco networking and open-source network security tools.

10.10.10.10

Physical Config Desktop Programming Attributes

Command Prompt X

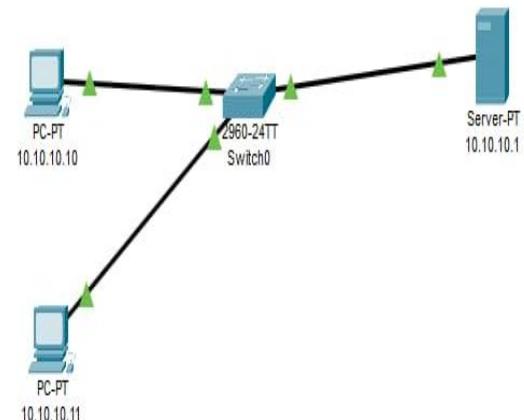
```
Packet Tracer PC Command Line 1.0
C:\>ftp 10.10.10.1
Trying to connect...10.10.10.1
Connected to 10.10.10.1
220- Welcome to PT Ftp server
Username:Username
331- Username ok, need password
Password:
230- Logged in
(passive mode On)
ftp>get asa842-k8.bin

Reading file asa842-k8.bin from 10.10.10.1:
File transfer in progress...

[Transfer complete - 5571584 bytes]

5571584 bytes copied in 12.377 secs (103144 bytes/sec)
ftp>
```

Top



9. Experiments with Emulator like Netkit, Emulab,Kathara etc.

Kathará is an open-source network emulation framework that allows users to design, configure, and test complex network topologies using lightweight containers (Linux network namespaces). It is the modern successor of Netkit, built to provide a more flexible and efficient environment for networking experiments, protocol testing, and educational simulations.

Kathará enables users to create virtual network devices such as PCs, routers, and switches, and connect them using virtual links that behave like real network cables. Each node in Kathará runs its own isolated Linux environment with its own processes, interfaces, and routing tables, making it ideal for simulating real networking scenarios.

Key Features of Kathará

1. Lightweight and Fast – Uses Linux namespaces instead of heavy virtual machines, resulting in minimal resource usage.
2. Easy Topology Design – Network topologies are described using simple text files (lab.conf, startup.sh, etc.), making setup and deployment very quick.
3. Realistic Network Behavior – Supports routing, bridging, VLANs, and network services such as DNS, DHCP, and HTTP.
4. Multi-Node Simulation – Allows emulation of networks consisting of multiple PCs, routers, and switches interconnected in any desired topology.
5. Integration with Mininet and Docker – Provides flexibility for both academic experiments and large-scale testing.
6. Cross-Platform Support – Works on Linux natively and can run on Windows or macOS through virtualization tools.

How Kathará Works -

Kathará uses network namespaces to isolate each node. Every node behaves like an independent Linux system.

- Configuration files define how these nodes are connected.
- When the kathara lstart command is executed, it creates the entire network automatically.
- Each node can then be accessed using kathara connect <node_name> and standard networking commands (like ifconfig, ping, route, etc.) can be used.

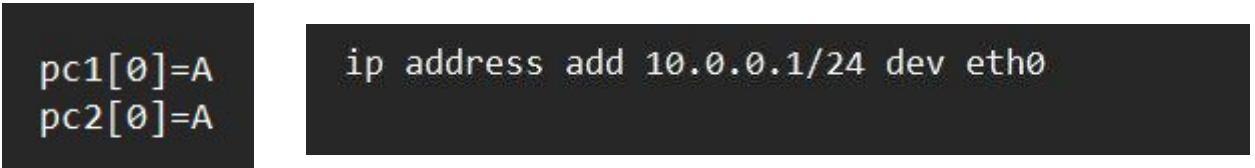
This makes it a powerful educational and research tool for understanding network concepts such as IP addressing, routing, switching, subnetting, and inter-network communication.

Applications of Kathará

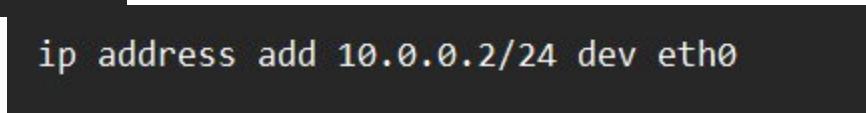
- Learning and demonstrating computer network concepts in academic labs.
- Testing and debugging networking protocols.
- Simulating enterprise or ISP networks before real deployment.
- Practicing configuration of routers and network devices in a safe virtual environment.

Conclusion

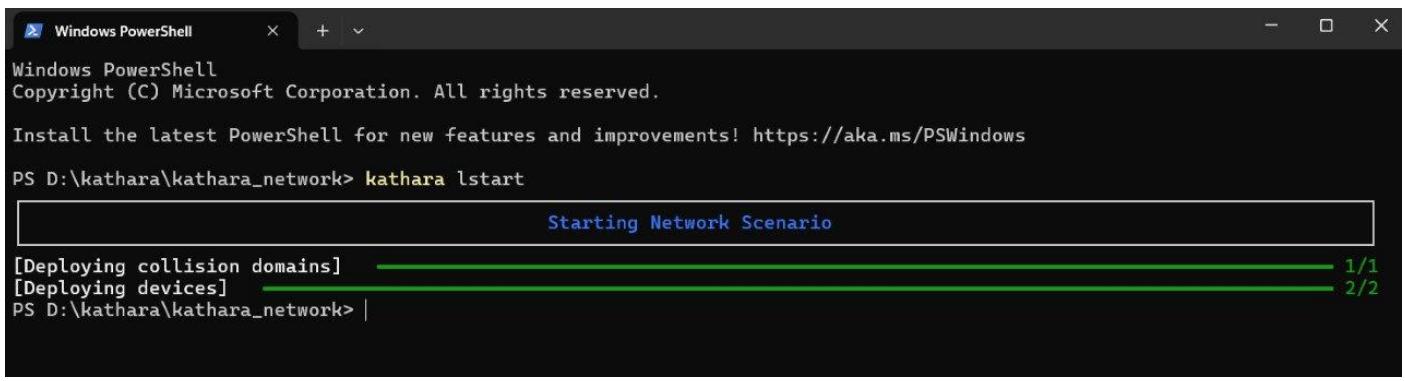
Kathará is a highly effective and user-friendly network emulator designed for education, research, and experimentation. It offers the simplicity of Netkit with improved performance, scalability, and support for modern networking features. By using Kathará, students and professionals can gain hands-on experience in configuring and troubleshooting real-world network setups without the need for physical hardware.



```
pc1[0]=A
pc2[0]=A
ip address add 10.0.0.1/24 dev eth0
```



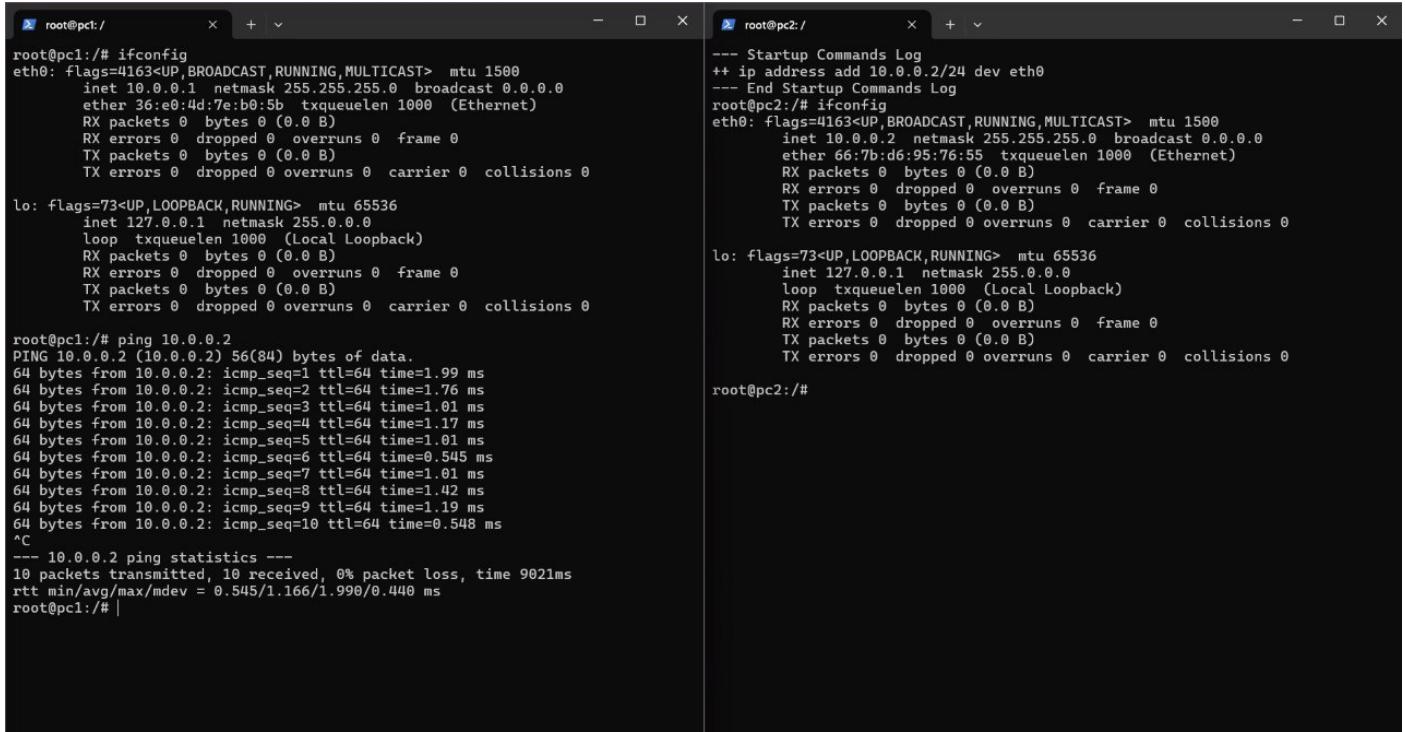
```
ip address add 10.0.0.2/24 dev eth0
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\kathara\kathara_network> kathara lstart
Starting Network Scenario
[Deploying collision domains] 1/1
[Deploying devices] 2/2
PS D:\kathara\kathara_network> |
```



```
root@pc1:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.255.255.0 broadcast 0.0.0.0
        ether 36:e0:4d:7e:b0:5b txqueuelen 1000 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@pc1:/# ping 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.99 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.76 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=1.01 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=1.17 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=1.01 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.545 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=1.01 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=1.42 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=1.19 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.548 ms
^C
--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9021ms
rtt min/avg/max/mdev = 0.545/1.166/1.990/0.440 ms
root@pc1:/#
```

```
--- Startup Commands Log
++ ip address add 10.0.0.2/24 dev eth0
--- End Startup Commands Log
root@pc2:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.2 netmask 255.255.255.0 broadcast 0.0.0.0
        ether 66:7b:d6:95:76:55 txqueuelen 1000 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@pc2:/#
```

10. Experiments with Simulator like GNS3, NS2, NCTU NS etc.

GNS3 (Graphical Network Simulator-3) is a powerful open-source network simulation tool that allows users to design, configure, and test complex network topologies using real or emulated network devices. It supports both virtual and physical networking devices, making it ideal for learning, testing, and certification training such as CCNA, CCNP, and CCIE.

A Point-to-Point (P2P) connection refers to a direct communication link between two network devices, such as routers, switches, or PCs. In this setup, data is transmitted directly from one device to another without passing through any intermediary device. This type of connection is often used in WAN links, leased lines, or simple two-node communication setups.

In GNS3, a P2P connection can be established by connecting two devices using a single cable (like a FastEthernet or Serial link) and assigning IP addresses from the same subnet to their respective interfaces. Once the IP configuration is done, connectivity is verified using the ping command.

Steps Involved:

1. **Open GNS3** and create a new project.
2. **Add two PCs (or routers)** to the workspace.
3. **Connect them** using an Ethernet or Serial cable.
4. **Start the devices** and open the console for each.
5. **Assign IP addresses** to both ends of the link (from the same subnet).
 - o Example:
 - PC1 → 192.168.1.1/24
 - PC2 → 192.168.1.2/24

6. **Check connectivity** using the command:

```
ping 192.168.1.2
```

from PC1 or vice versa.

If the configuration is correct and both devices are in the same network, the ping replies will confirm that the **P2P link is successfully established**.

Advantages of P2P Connection:

- Simple and easy to configure.
- Provides dedicated bandwidth between two nodes.
- Low latency and high reliability.
- Ideal for testing and learning basic networking concepts.

Conclusion:

The P2P connection in GNS3 demonstrates basic network communication between two nodes using direct cabling. This experiment helps understand how IP addressing and direct communication work in networks.

By successfully pinging one PC from another, we verify that the devices are properly connected and configured for communication over a point-to-point link.

