Reg: no: 22BCS081

Exp: no: 1

Aim:

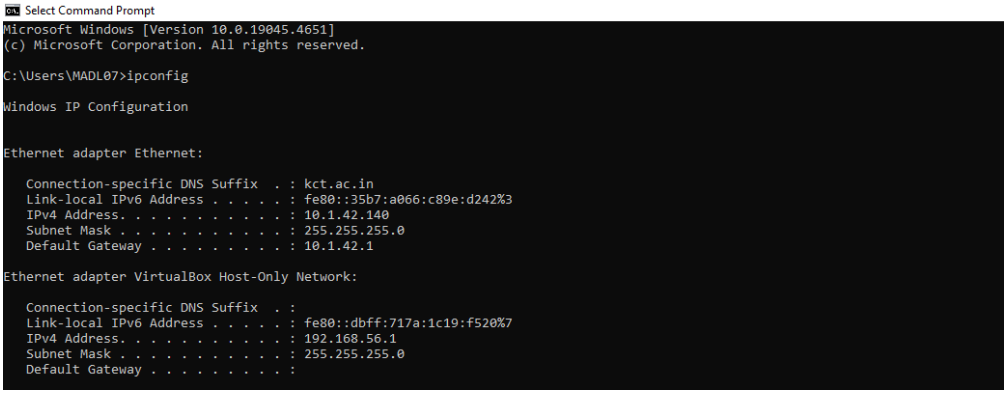
To demonstrate the working of network tools such as Ping, TCP Dump, Traceroute, Netstat and write the syntax, execute and place the screenshot for all the commands worked on.

a. Demonstrate the different *ipconfig* commands used to explore a system's IP address.

Sol:

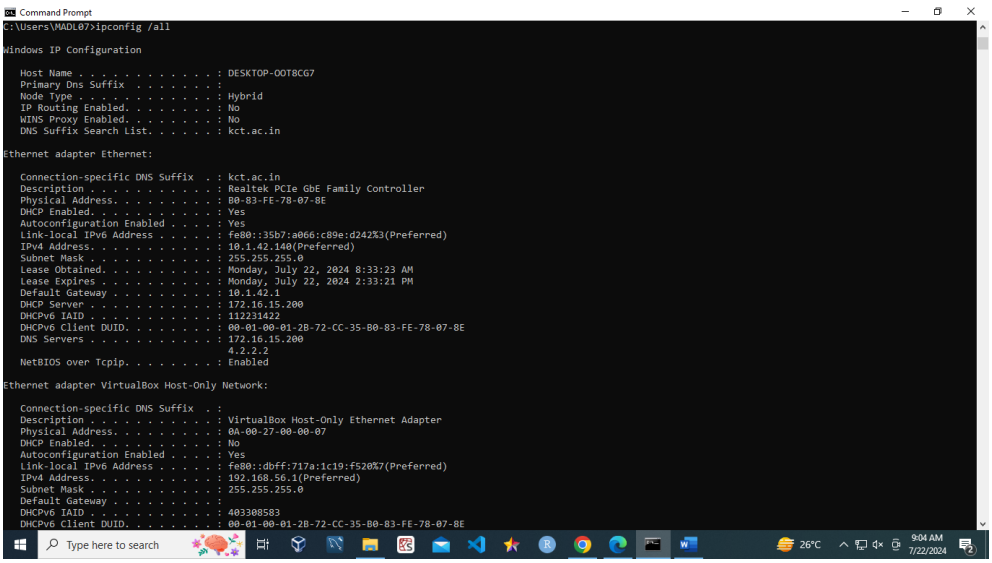
- **ipconfig**

ipconfig is a command-line utility in Windows that displays the current TCP/IP network configuration values for the computer. It is primarily used for troubleshooting network issues and managing network connections.



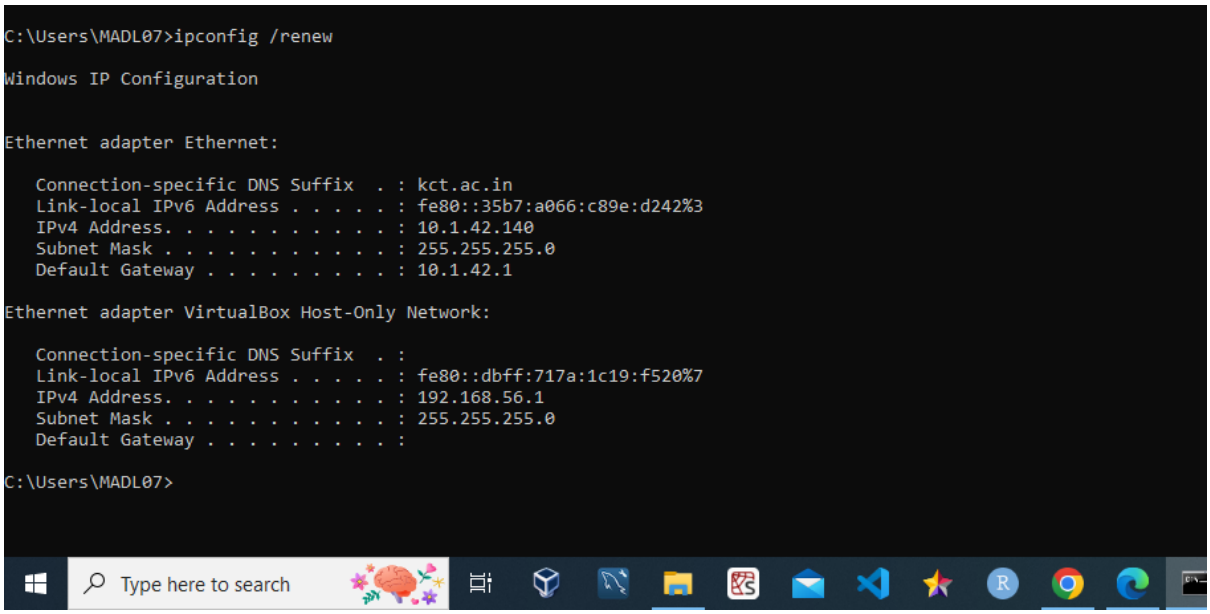
- **ipconfig /all**

Description: This command provides a detailed view of all network interfaces on the system, including physical (MAC) addresses, IP addresses, subnet masks, default gateways, DNS servers, and more.



- **ipconfig /renew**

The ipconfig /renew command requests a new IP address from a DHCP server for the specified network adapter in Windows.



b. To write the use of *ping* command and show the any 5 variants of the command.

Ping Command

The ping command is a fundamental network utility used to test connectivity between devices, measure round-trip time, diagnose network issues, perform name resolution testing, and monitor connection stability. The ping command is an essential tool for network diagnostics, providing valuable insights into the health and performance of network connections.

1. ping -t 10.1.42.140

- **Description**: Sends continuous ping requests to the specified host until interrupted (usually by pressing Ctrl + C).
- **Use Case**: Helpful for monitoring the stability of a connection over time or diagnosing intermittent connectivity issues.

```
C:\Users\MADL07>ping -t 10.1.42.140

Pinging 10.1.42.140 with 32 bytes of data:
Reply from 10.1.42.140: bytes=32 time<1ms TTL=128
Reply from 10.1.42.140: bytes=32 time<1ms TTL=128
Reply from 10.1.42.140: bytes=32 time<1ms TTL=128
Reply from 10.1.42.140: bytes=32 time<1ms TTL=128
Reply from 10.1.42.140: bytes=32 time<1ms TTL=128
Reply from 10.1.42.140: bytes=32 time<1ms TTL=128
Reply from 10.1.42.140: bytes=32 time<1ms TTL=128
Reply from 10.1.42.140: bytes=32 time<1ms TTL=128
Reply from 10.1.42.140: bytes=32 time<1ms TTL=128
Reply from 10.1.42.140: bytes=32 time<1ms TTL=128

Ping statistics for 10.1.42.140:
    Packets: Sent = 10, Received = 10, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
Control-C
^C
```

2. ping –a 10.1.42.140

- **Description**: Resolves the hostname of an IP address and displays it along with the ping results.
- **Use Case**: Useful for identifying the hostname of a given IP address, which can help in network diagnostics.

```
C:\Users\MADL07>ping -a 10.1.42.140

Pinging DESKTOP-OOT8CG7.kct.ac.in [10.1.42.140] with 32 bytes of data:
Reply from 10.1.42.140: bytes=32 time<1ms TTL=128
Reply from 10.1.42.140: bytes=32 time<1ms TTL=128
Reply from 10.1.42.140: bytes=32 time<1ms TTL=128
Reply from 10.1.42.140: bytes=32 time<1ms TTL=128

Ping statistics for 10.1.42.140:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

3. ping -h

- **Description**: Displays help information for the ping command, including a list of available options and their descriptions.
- **Use Case**: Useful for quickly referencing the command syntax and available options.

```
C:\Users\MADL07>ping -h
Bad option -h.

Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]
            [-r count] [-s count] [[-j host-list] | [-k host-list]]
            [-w timeout] [-R] [-S srcaddr] [-c compartment] [-p]
            [-4] [-6] target_name

Options:
    -t             Ping the specified host until stopped.
                   To see statistics and continue - type Control-Break;
                   To stop - type Control-C.
    -a             Resolve addresses to hostnames.
    -n count       Number of echo requests to send.
    -l size        Send buffer size.
    -f             Set Don't Fragment flag in packet (IPv4-only).
    -i TTL         Time To Live.
    -v TOS         Type Of Service (IPv4-only. This setting has been deprecated
                   and has no effect on the type of service field in the IP
                   Header).
    -r count       Record route for count hops (IPv4-only).
    -s count       Timestamp for count hops (IPv4-only).
    -j host-list   Loose source route along host-list (IPv4-only).
    -k host-list   Strict source route along host-list (IPv4-only).
    -w timeout     Timeout in milliseconds to wait for each reply.
    -R             Use routing header to test reverse route also (IPv6-only).
                   Per RFC 5095 the use of this routing header has been
                   deprecated. Some systems may drop echo requests if
                   this header is used.
    -S srcaddr     Source address to use.
    -c compartment Routing compartment identifier.
    -p             Ping a Hyper-V Network Virtualization provider address.
    -4             Force using IPv4.
    -6             Force using IPv6.
```

4. ping -l size 10.1.42.140

- **Description**: Specifies the size of the packet to be sent in bytes. The default size is typically 32 bytes.
- **Use Case**: Useful for testing how different packet sizes affect network performance and for diagnosing issues related to MTU (Maximum Transmission Unit) sizes.

```
C:\Users\MADL07>ping -l size 10.1.42.140

Pinging 10.1.42.140 with 0 bytes of data:
Reply from 10.1.42.140: bytes=0 time<1ms TTL=128
Reply from 10.1.42.140: bytes=0 time<1ms TTL=128
Reply from 10.1.42.140: bytes=0 time<1ms TTL=128
Reply from 10.1.42.140: bytes=0 time<1ms TTL=128

Ping statistics for 10.1.42.140:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

5. ping -n 5 google.com

- **Description**: Sends a specified number of ping requests to the target host instead of the default continuous pings.
- **Use Case**: Useful for conducting a limited test to measure response times without overwhelming the network.

```
C:\Users\MADL07>ping -n 5 google.com

Pinging google.com [142.250.70.46] with 32 bytes of data:
Reply from 142.250.70.46: bytes=32 time=32ms TTL=58
Reply from 142.250.70.46: bytes=32 time=31ms TTL=58
Reply from 142.250.70.46: bytes=32 time=31ms TTL=58
Reply from 142.250.70.46: bytes=32 time=31ms TTL=58
Reply from 142.250.70.46: bytes=32 time=35ms TTL=58

Ping statistics for 142.250.70.46:
    Packets: Sent = 5, Received = 5, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 31ms, Maximum = 35ms, Average = 32ms
```

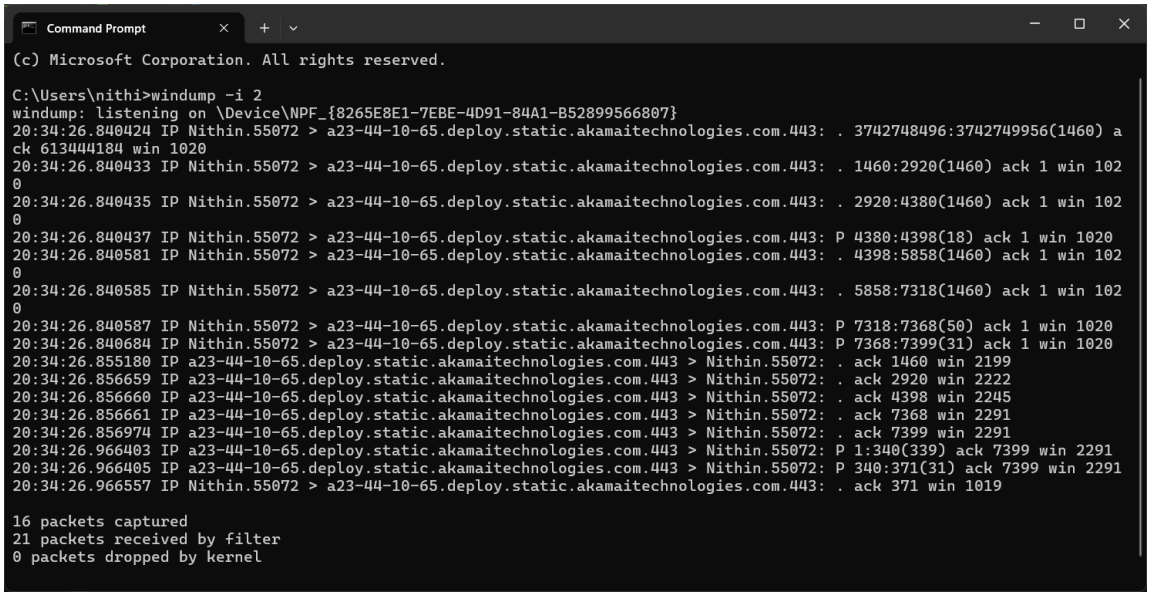
c. To write the use of *tcpdump(windump)* command execute the following commands.

windump Command

windump is the Windows port of the popular tcpdump tool used on Unix-like systems. It is a network packet analyzer that captures and displays the packet headers on a network interface. It can be used for network troubleshooting, monitoring, and analysis.

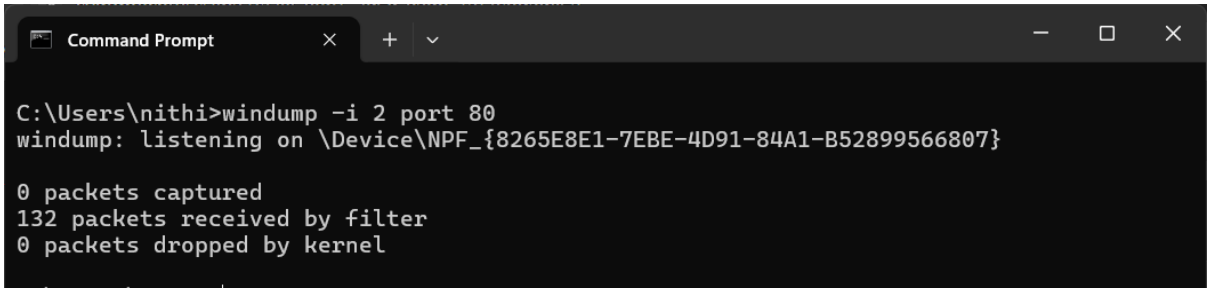
1. windump -i 1

- **Description:** Captures and displays all packets on the network interface with index 1 in real-time.
- **Use Case:** Useful for general network traffic monitoring and troubleshooting to understand the types and volume of traffic.



2. windump -i 1 port 80

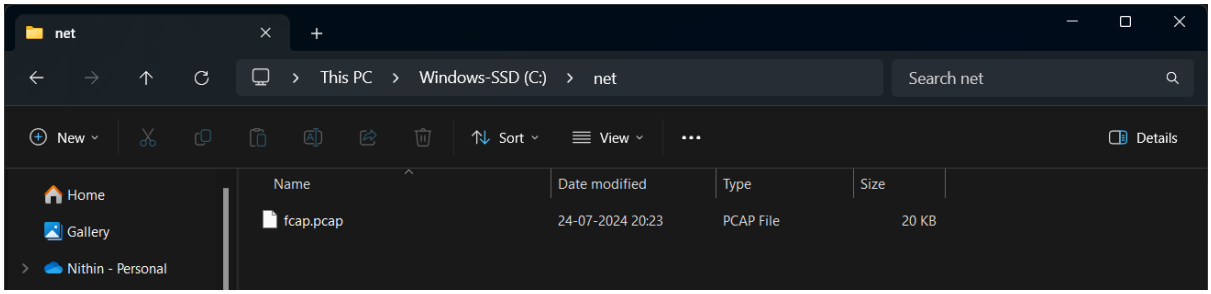
- **Description:** Captures packets on the network interface with index 1 that are destined for or originate from port 80 (HTTP traffic).
- **Use Case:** Ideal for monitoring and troubleshooting web traffic, allowing the analysis of HTTP communications between clients and servers.



3. windump -i 1 -w C:\net\fcap.pcap

- **Description:** Captures packets on the network interface with index 1 and writes them to a file named fcap.pcap in the C:\net directory.
- **Use Case:** Useful for saving network traffic data for later analysis with tools like Wireshark, aiding in detailed network forensic investigations.

```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nithi>windump -i 2 -w C:\net\fcap.pcap
windump: listening on \Device\NPF_{8265E8E1-7EBE-4D91-84A1-B52899566807}

222 packets captured
223 packets received by filter
0 packets dropped by kernel
```



d. To write the use of *traceroute* command and show the any 5 variants of the command.

Sol:

1. **Traceroute command**

The tracert command, known as traceroute in Unix-like systems, is a network diagnostic tool used to trace the path that packets take from a source to a destination over an IP network. It works by sending packets with incrementally increasing Time-To-Live (TTL) values, allowing it to identify each hop (router) along the route to the destination. This command helps network administrators troubleshoot connectivity issues by revealing where delays or packet losses occur, and it provides information about the round-trip time for each hop.

```
C:\Users\MADL07>tracert

Usage: tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout]
               [-R] [-S srcaddr] [-4] [-6] target_name

Options:
    -d                 Do not resolve addresses to hostnames.
    -h maximum_hops    Maximum number of hops to search for target.
    -j host-list       Loose source route along host-list (IPv4-only).
    -w timeout         Wait timeout milliseconds for each reply.
    -R                 Trace round-trip path (IPv6-only).
    -S srcaddr         Source address to use (IPv6-only).
    -4                 Force using IPv4.
    -6                 Force using IPv6.
```

2. tracert -j host-list 10.1.42.140

- **Description:** Specifies that the tracert command should use the "loose source routing" option to send packets to the destination. This allows you to specify a list of IP addresses that the packets should visit on the way to the destination.
- **Use Case:** Useful for testing connectivity through specific intermediate hosts or networks, or for bypassing certain network segments.

```
C:\Users\MADL07>tracert -j host-list 10.1.42.140
host-list is not a valid source route address.

C:\Users\MADL07>tracert -d 10.1.42.140

Tracing route to 10.1.42.140 over a maximum of 30 hops

  1    <1 ms    <1 ms    <1 ms  10.1.42.140

Trace complete.
```

3. tracert -w 2000 www.google.com

- **Description**: Sets the maximum number of milliseconds to wait for a response to each probe. The default timeout is 4000 milliseconds (4 seconds).
- **Use Case**: Helpful for reducing the time spent waiting for unresponsive hops or for speeding up the overall traceroute process.

Command Prompt

```
C:\Users\MADL07>tracert -w 2000 www.google.com

Tracing route to www.google.com [142.250.70.68]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  10.1.42.1
  2     1 ms    <1 ms    <1 ms  103.196.28.194
  3    53 ms     6 ms     *     103.196.28.193
  4     1 ms     1 ms     1 ms  202.129.199.17
  5    11 ms    13 ms    10 ms  202.129.199.10
  6    12 ms    11 ms    11 ms  142.250.175.82
  7    14 ms    11 ms    11 ms  142.250.208.105
  8    10 ms    10 ms    11 ms  142.251.229.250
  9    26 ms    28 ms    27 ms  72.14.232.34
 10    26 ms    25 ms    26 ms  192.178.110.245
 11    31 ms    35 ms    33 ms  192.178.86.201
 12    28 ms    32 ms    27 ms  pnbomb-ab-in-f4.1e100.net [142.250.70.68]

Trace complete.
```

4. tracert -d www.google.com

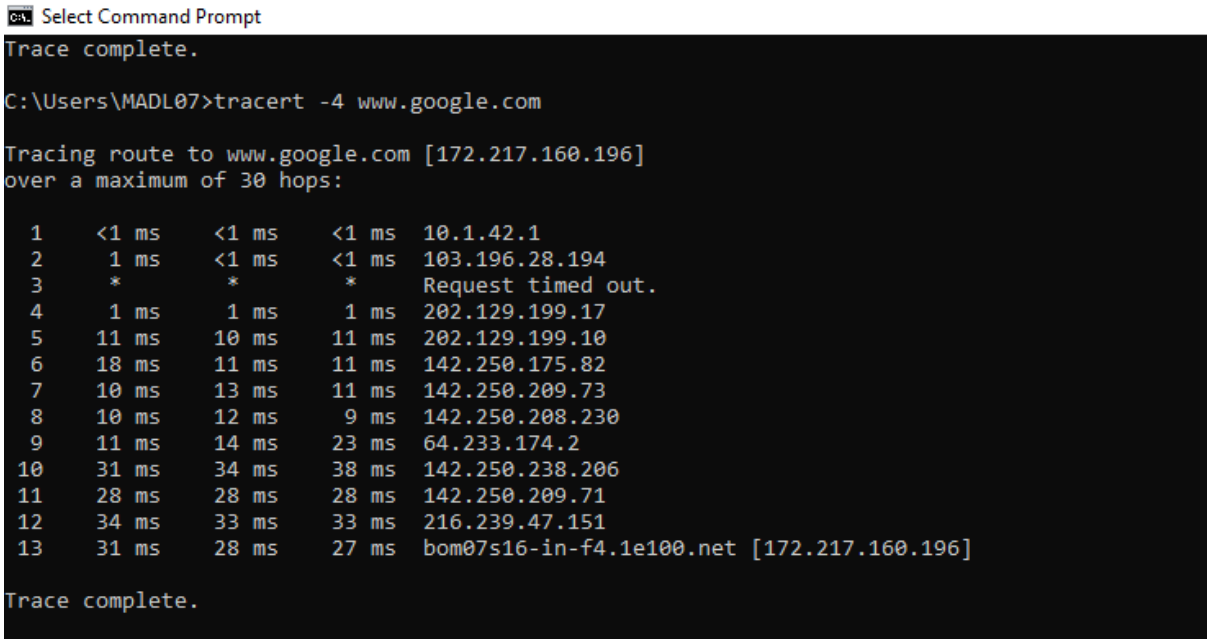
- **Description**: Prevents tracert from attempting to resolve the IP addresses of intermediate hops to hostnames.
- **Use Case**: Speeds up the traceroute process by skipping the hostname resolution step, which can be slow or fail if DNS is not working properly.

Command Prompt - tracert -4 www.google.com

```
C:\Users\MADL07>tracert -d www.google.com

Tracing route to www.google.com [142.250.70.68]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  10.1.42.1
  2     1 ms    <1 ms     1 ms  103.196.28.194
  3     *        *        *     Request timed out.
  4     2 ms     1 ms     1 ms  202.129.199.17
  5    10 ms    10 ms    11 ms  202.129.199.10
  6    12 ms    11 ms    22 ms  142.250.175.82
  7    11 ms    10 ms    11 ms  142.250.208.105
  8    11 ms    11 ms    10 ms  142.251.229.250
  9    28 ms    28 ms    26 ms  72.14.232.34
 10    26 ms    25 ms    26 ms  192.178.110.245
 11    33 ms    31 ms    31 ms  192.178.86.201
 12    26 ms    34 ms    33 ms  142.250.70.68

Trace complete.
```

5. tracert -4 www.google.com

- **Description**: Forces tracert to use IPv4 (-4) or IPv6 (-6) for the traceroute, even if the destination could be reached using the other protocol.
- **Use Case**: Useful for testing connectivity over a specific IP protocol version or for troubleshooting issues related to IPv4/IPv6 compatibility.



e. To write the use of *netstat* command and show the any 5 variants of the command.

Sol:

netstat command

The netstat command is a powerful networking tool used in Windows and other operating systems to display network connections, routing tables, interface statistics, and various network protocol statistics. It is essential for diagnosing network issues, monitoring network performance, and understanding the status of network connections.

1. netstat -a 10.1.42.140

- **Description**: Displays all active connections and listening ports, including both established connections and those that are waiting for incoming connections.
- **Use Case**: Useful for monitoring all network connections and identifying which ports are open and listening for connections.

```
C:\Users\MADL07>netstat -a 10.1.42.140

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            DESKTOP-OOT8CG7:0      LISTENING
  TCP    0.0.0.0:445            DESKTOP-OOT8CG7:0      LISTENING
  TCP    0.0.0.0:3306           DESKTOP-OOT8CG7:0      LISTENING
  TCP    0.0.0.0:5040           DESKTOP-OOT8CG7:0      LISTENING
  TCP    0.0.0.0:7680           DESKTOP-OOT8CG7:0      LISTENING
  TCP    0.0.0.0:33060          DESKTOP-OOT8CG7:0      LISTENING
  TCP    0.0.0.0:49664          DESKTOP-OOT8CG7:0      LISTENING
  TCP    0.0.0.0:49665          DESKTOP-OOT8CG7:0      LISTENING
  TCP    0.0.0.0:49666          DESKTOP-OOT8CG7:0      LISTENING
  TCP    0.0.0.0:49667          DESKTOP-OOT8CG7:0      LISTENING
  TCP    0.0.0.0:49668          DESKTOP-OOT8CG7:0      LISTENING
  TCP    0.0.0.0:49675          DESKTOP-OOT8CG7:0      LISTENING
  TCP    10.1.42.140:139        DESKTOP-OOT8CG7:0      LISTENING
  TCP    10.1.42.140:52759      20.198.162.78:https    ESTABLISHED
  TCP    10.1.42.140:52932      20.204.194.128:https   ESTABLISHED
  TCP    10.1.42.140:53016      52.123.253.136:https   ESTABLISHED
  TCP    10.1.42.140:53042      52.98.200.194:https    ESTABLISHED
  TCP    10.1.42.140:53065      52.98.200.194:https    ESTABLISHED
  TCP    10.1.42.140:53086      52.111.252.0:https     ESTABLISHED
  TCP    10.1.42.140:53507      40.99.9.34:https       ESTABLISHED
```

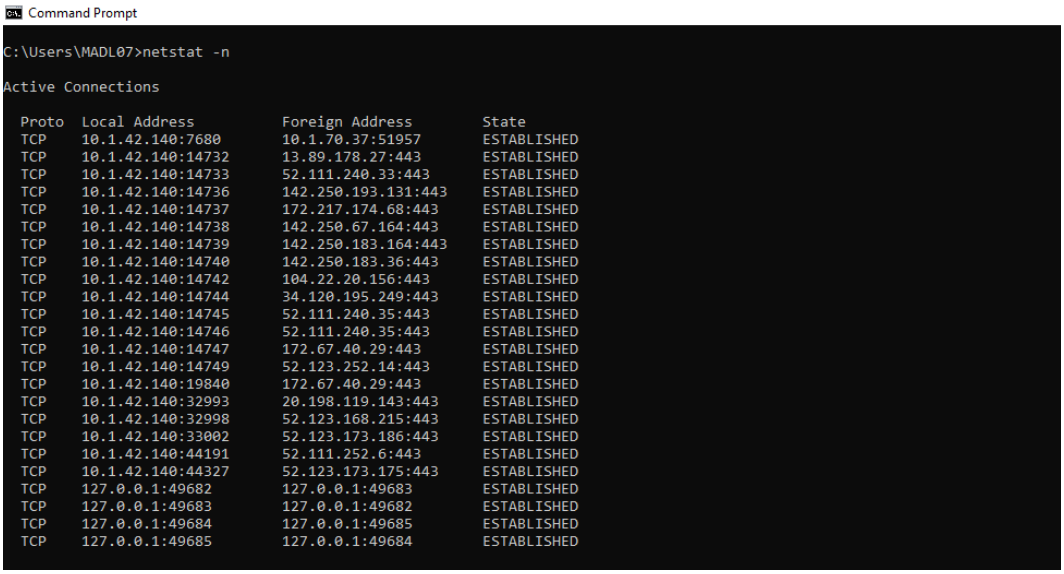
2. netstat -h

- **Description**: Displays help information about the netstat command, including available options and their descriptions.
- **Use Case**: Helpful for users who need guidance on how to use the command or want to explore its various options.

```
C:\Users\MADL07>netstat -h

Displays protocol statistics and current TCP/IP network connections.

NETSTAT [-a] [-b] [-e] [-f] [-n] [-o] [-p proto] [-r] [-s] [-t] [-x] [-y] [interval]

  -a            Displays all connections and listening ports.
  -b            Displays the executable involved in creating each connection or
                listening port. In some cases well-known executables host
                multiple independent components, and in these cases the
                sequence of components involved in creating the connection
                or listening port is displayed. In this case the executable
                name is in [] at the bottom, on top is the component it called,
                and so forth until TCP/IP was reached. Note that this option
                can be time-consuming and will fail unless you have sufficient
                permissions.
  -e            Displays Ethernet statistics. This may be combined with the -s
                option.
  -f            Displays Fully Qualified Domain Names (FQDN) for foreign
                addresses.
  -n            Displays addresses and port numbers in numerical form.
  -o            Displays the owning process ID associated with each connection.
  -p proto      Shows connections for the protocol specified by proto; proto
                may be any of: TCP, UDP, TCPv6, or UDPv6.  If used with the -s
```

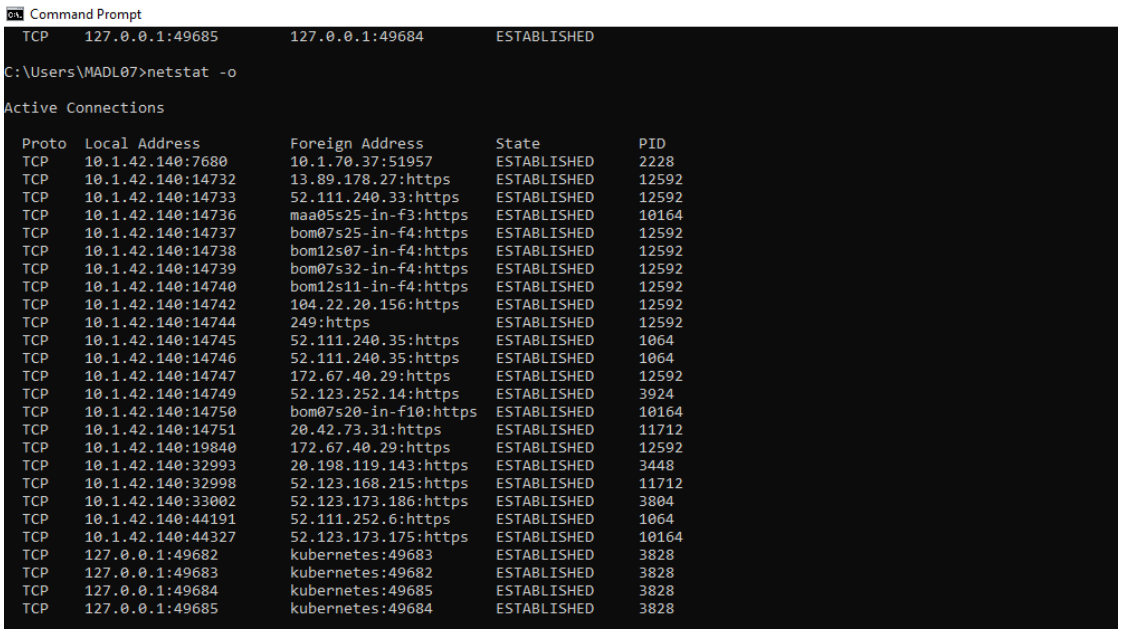
3. netstat -n

- **Description**: Displays addresses and port numbers in numerical format instead of resolving them to hostnames.
- **Use Case**: Useful for speeding up the output by avoiding DNS lookups, which can be slow or fail if there are DNS issues.



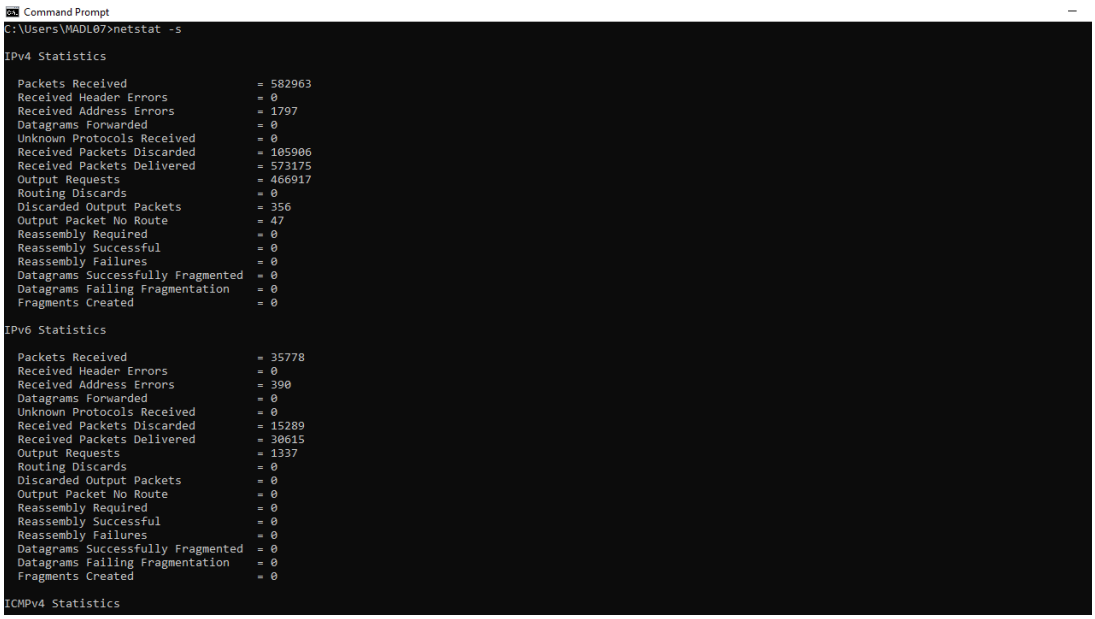
4. netstat -o

- **Description**: Displays active connections and includes the process ID (PID) associated with each connection.
- **Use Case**: Useful for identifying which applications are using specific network connections, aiding in troubleshooting and monitoring.



5. netstat -s

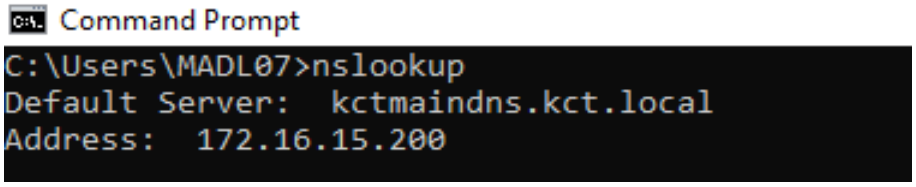
- **Description**: Displays statistics for each protocol (TCP, UDP, ICMP, etc.), providing detailed information about network activity and errors.
- **Use Case**: Helpful for diagnosing network issues by revealing statistics that indicate potential problems with specific protocols.



Post Lab:

nslookup command:

- **Description**: nslookup is a command-line tool used to query the Domain Name System (DNS) for information about domain names and IP addresses, enabling users to retrieve DNS records and perform reverse lookups.
- **Use Case:** It is commonly used for troubleshooting DNS issues by verifying domain name resolution, checking the IP address associated with a domain, and gathering DNS records to ensure proper configuration of web and email services.



Result:

All experiments have been successfully executed, with no errors or issues encountered. The expected results have been achieved, as demonstrated by the attached screenshots.

Reg: no: 22BCS081

Exp: no: 2

Analyze the network traffic using Packet tracer tool .

Aim:

To analyse the network traffic using Cisco Packet tracer tool and write the syntax, execute and place the screenshot for all the commands worked on.

a. Configure a simple network connecting two LANs using Cisco Packet Tracer.

Equipment:

- 2 x 2960-24TT Switches
- 1 x ISR4331 Router
- 4 x PCs

Syntax:

Step 1: Open Cisco Packet Tracer

1. Open Cisco Packet Tracer on your computer.

Step 2: Create the Network Topology

1. **Drag and Drop Devices**: Drag two 2960-24TT Switches, one ISR4331 Router, and four PCs onto the workspace.
2. **Connect Devices**:
    - Use the Copper Straight-Through cable to connect:
        - PC0 to Switch0 (Port FastEthernet0/1)
        - PC1 to Switch0 (Port FastEthernet0/2)
        - PC2 to Switch1 (Port FastEthernet0/1)
        - PC3 to Switch1 (Port FastEthernet0/2)
        - Switch0 (Port GigabitEthernet0/1) to Router (Port GigabitEthernet0/0)
        - Switch1 (Port GigabitEthernet0/1) to Router (Port GigabitEthernet0/1)

Step 3: Configure IP Addresses on PCs

1. **PC0**:
    - IP Address: 192.168.1.2
    - Subnet Mask: 255.255.255.0
    - Default Gateway: 192.168.1.1
2. **PC1**:
    - IP Address: 192.168.1.3
    - Subnet Mask: 255.255.255.0
    - Default Gateway: 192.168.1.1

3. **PC2**:
    - IP Address: 192.168.2.2
    - Subnet Mask: 255.255.255.0
    - Default Gateway: 192.168.2.1
4. **PC3**:
    - IP Address: 192.168.2.3
    - Subnet Mask: 255.255.255.0
    - Default Gateway: 192.168.2.1

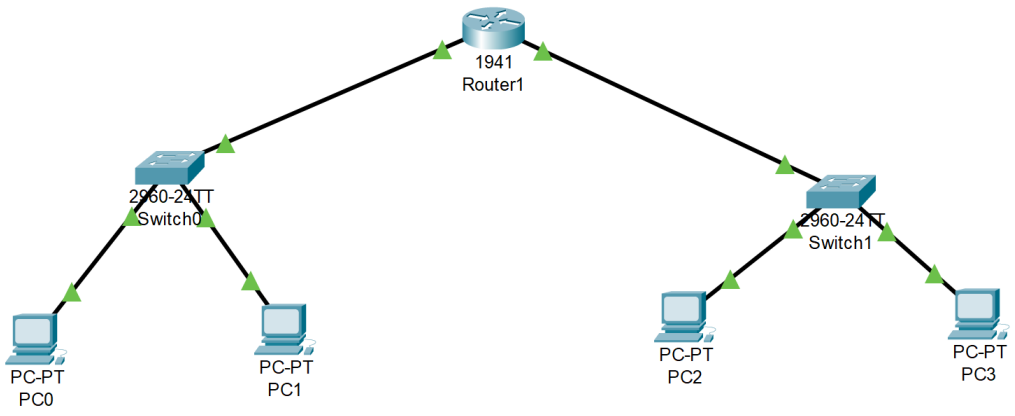
Step 4: Configure IP Addresses on Router

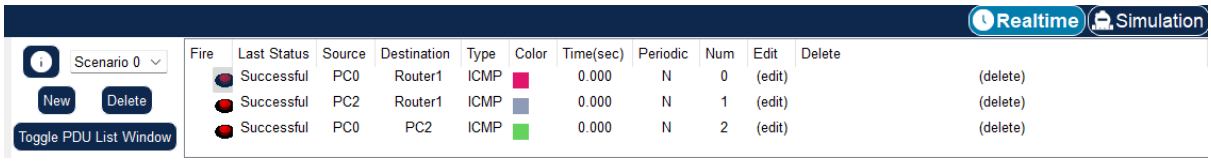
1. **Router**:
    - Enter the CLI of the ISR4331 Router.
    - Execute the following commands:

```
Router> enable

Router# configure terminal

Router(config)# interface GigabitEthernet0/0

Router(config-if)# ip address 192.168.1.1 255.255.255.0

Router(config-if)# no shutdown

Router(config-if)# exit

Router(config)# interface GigabitEthernet0/1

Router(config-if)# ip address 192.168.2.1 255.255.255.0

Router(config-if)# no shutdown

Router(config-if)# exit
```

Department of Computer Science and Engineering
U18CSI5201_Computer Networks Laboratory

Reg: no: 22BCS081





Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	Router1	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC2	Router1	ICMP		0.000	N	1	(edit)	(delete)
	Successful	PC0	PC2	ICMP		0.000	N	2	(edit)	(delete)

Nithin Prasath C - 22BCS081

b. Configure a network using Ring/Bus/Tree Topology using Packet tracer tool.

Equipment:

- 2960-24TT Switches
- ISR4331 Router
- PCs
- Appropriate cables (Copper Straight-Through and Copper Cross-Over)

Steps for Different Topologies:

1. Ring Topology:

Ring Topology connects each device to exactly two other devices, forming a single continuous pathway for signals through each device.

Step 1: Open Cisco Packet Tracer

1. Open Cisco Packet Tracer on your computer.

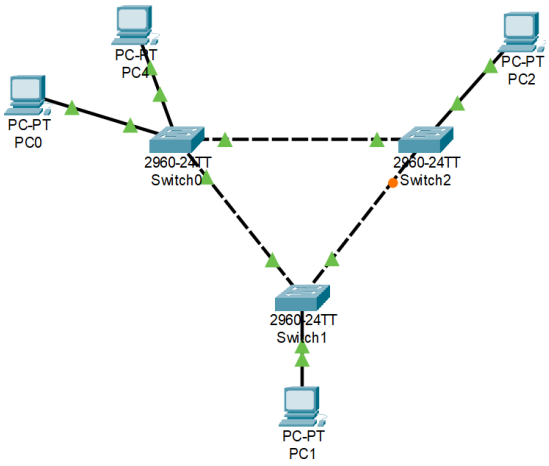
Step 2: Create the Network Topology

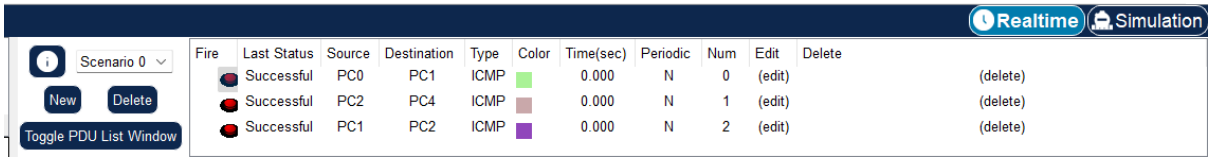
1. **Drag and Drop Devices**: Drag three 2960-24TT Switches and four PCs onto the workspace.
2. **Connect Devices**:
    - Use the Copper Cross-Over cable to connect:
        - Switch0 (Port GigabitEthernet0/1) to Switch1 (Port GigabitEthernet0/1)
        - Switch1 (Port GigabitEthernet0/2) to Switch2 (Port GigabitEthernet0/1)
        - Switch2 (Port GigabitEthernet0/2) to Switch0 (Port GigabitEthernet0/2)
    - Use the Copper Straight-Through cable to connect:
        - PC0 to Switch0 (Port FastEthernet0/1)
        - PC1 to Switch1 (Port FastEthernet0/1)
        - PC2 to Switch2 (Port FastEthernet0/1)
        - PC3 to Switch0 (Port FastEthernet0/2)

Step 3: Configure IP Addresses on PCs

1. **PC0**:
    - IP Address: 192.168.1.2
    - Subnet Mask: 255.255.255.0

2. **PC1**:
    - IP Address: 192.168.1.3
    - Subnet Mask: 255.255.255.0
3. **PC2**:
    - IP Address: 192.168.1.4
    - Subnet Mask: 255.255.255.0
4. **PC3**:
    - IP Address: 192.168.1.5
    - Subnet Mask: 255.255.255.0





2. Bus Topology:

Bus Topology connects all devices to a single central cable, called the bus or backbone.

Step 1: Open Cisco Packet Tracer

1. Open Cisco Packet Tracer on your computer.

Step 2: Create the Network Topology

1. **Drag and Drop Devices**: Drag one 2960-24TT Switch and four PCs onto the workspace.

2. **Connect Devices**:

    - Use the Copper Straight-Through cable to connect:
        - PC0 to Switch0 (Port FastEthernet0/1)
        - PC1 to Switch0 (Port FastEthernet0/2)
        - PC2 to Switch0 (Port FastEthernet0/3)
        - PC3 to Switch0 (Port FastEthernet0/4)

Step 3: Configure IP Addresses on PCs

1. **PC0**:

    - IP Address: 192.168.1.2
    - Subnet Mask: 255.255.255.0

2. **PC1**:

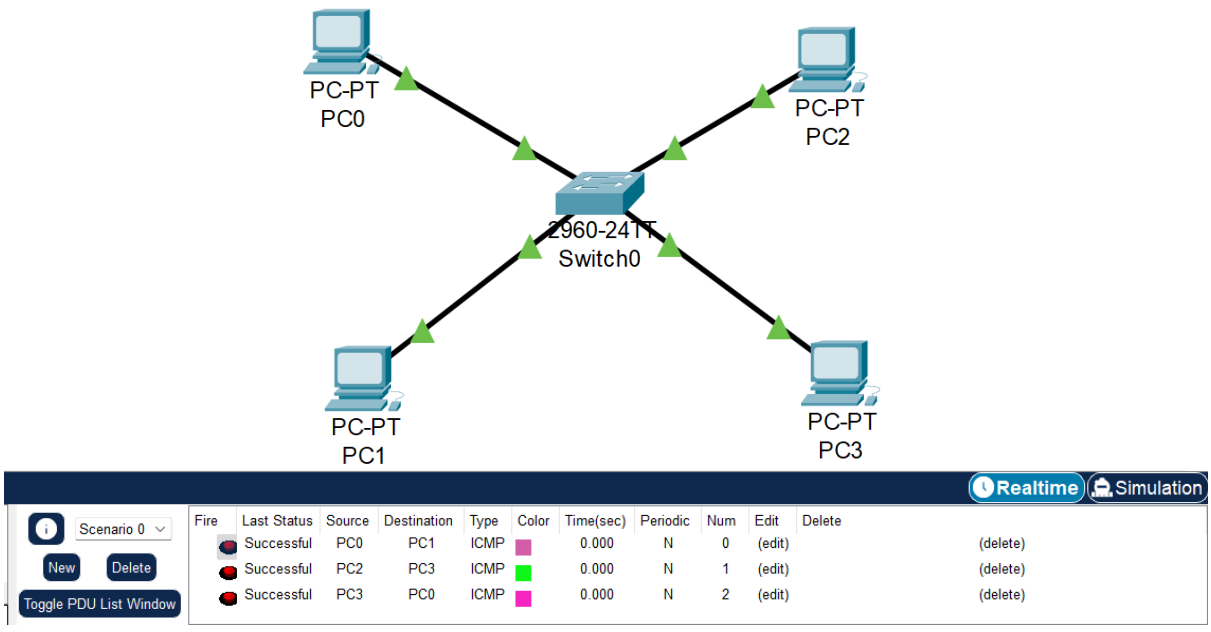
    - IP Address: 192.168.1.3
    - Subnet Mask: 255.255.255.0

3. **PC2**:

    - IP Address: 192.168.1.4
    - Subnet Mask: 255.255.255.0

4. **PC3**:

    - IP Address: 192.168.1.5
    - Subnet Mask: 255.255.255.0



3. Tree Topology:

Tree Topology combines characteristics of Star and Bus topologies. It consists of groups of star-configured networks connected to a linear bus backbone.

Step 1: Open Cisco Packet Tracer

1. Open Cisco Packet Tracer on your computer.

Step 2: Create the Network Topology

1. **Drag and Drop Devices**: Drag three 2960-24TT Switches, one ISR4331 Router, and six PCs onto the workspace.
2. **Connect Devices**:

    - Use the Copper Straight-Through cable to connect:
        - Router (Port GigabitEthernet0/0) to Switch0 (Port GigabitEthernet0/1)
        - Switch0 (Port GigabitEthernet0/2) to Switch1 (Port GigabitEthernet0/1)
        - Switch0 (Port GigabitEthernet0/3) to Switch2 (Port GigabitEthernet0/1)
        - PC0 to Switch1 (Port FastEthernet0/1)
        - PC1 to Switch1 (Port FastEthernet0/2)
        - PC2 to Switch1 (Port FastEthernet0/3)
        - PC3 to Switch2 (Port FastEthernet0/1)
        - PC4 to Switch2 (Port FastEthernet0/2)
        - PC5 to Switch2 (Port FastEthernet0/3)

Step 3: Configure IP Addresses on PCs

1. **PC0**:

    - IP Address: 192.168.1.2
    - Subnet Mask: 255.255.255.0

2. **PC1**:

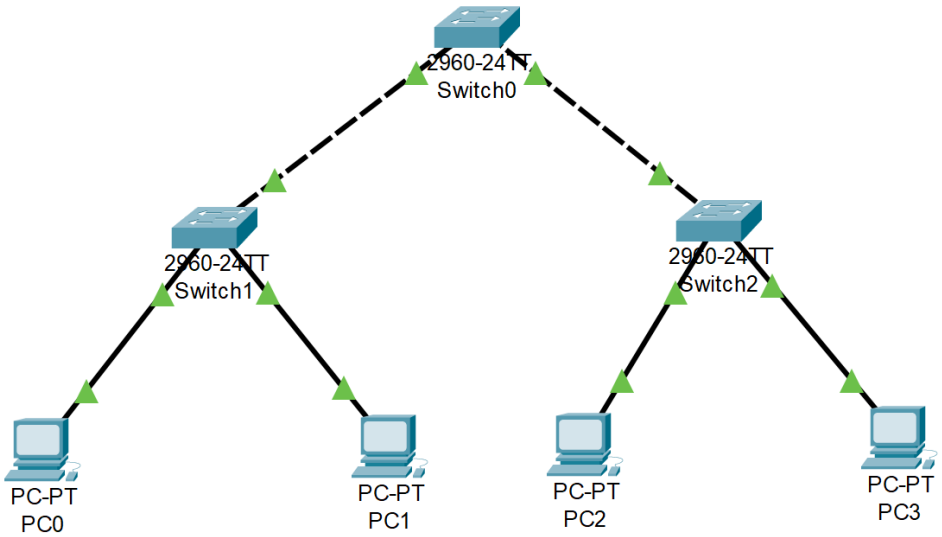
    - IP Address: 192.168.1.3
    - Subnet Mask: 255.255.255.0

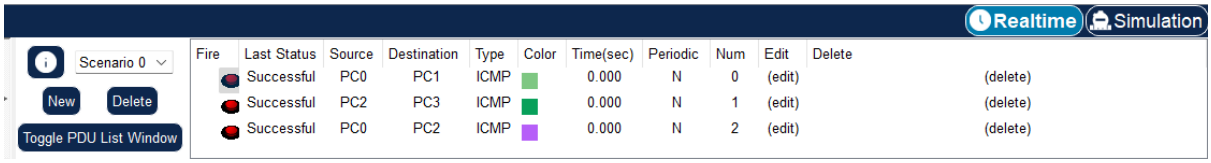
3. **PC2**:

    - IP Address: 192.168.1.4
    - Subnet Mask: 255.255.255.0

4. **PC3**:

- IP Address: 192.168.1.5
- Subnet Mask: 255.255.255.0





c. Configure a network for four departments in your college with a minimum of five PCs in a network.

Equipment:

- 2960-24TT Switches
- PCs
- Appropriate cables (Copper Straight-Through)

Steps:

Step 1: Open Cisco Packet Tracer

Step 2: Create the Network Topology

1. Drag and Drop Devices:
   - Drag four 2960-24TT Switches onto the workspace (one for each department).
   - Drag twenty PCs onto the workspace (five for each department).

Step 3: Connect Devices

1. **CSE Department:**
   - **Switch0 (CSE Department):**
     - **Use Copper Straight-Through cable to connect:**
       - PC0 to Switch0 (Port FastEthernet0/1)
       - PC1 to Switch0 (Port FastEthernet0/2)
       - PC2 to Switch0 (Port FastEthernet0/3)
       - PC3 to Switch0 (Port FastEthernet0/4)
       - PC4 to Switch0 (Port FastEthernet0/5)
2. **IT Department:**
   - **Switch1 (IT Department):**
     - **Use Copper Straight-Through cable to connect:**
       - PC5 to Switch1 (Port FastEthernet0/1)
       - PC6 to Switch1 (Port FastEthernet0/2)
       - PC7 to Switch1 (Port FastEthernet0/3)
       - PC8 to Switch1 (Port FastEthernet0/4)
       - PC9 to Switch1 (Port FastEthernet0/5)

3. **AIDS Department:**
    - **Switch2 (AIDS Department):**
        - **Use Copper Straight-Through cable to connect:**
            - PC10 to Switch2 (Port FastEthernet0/1)
            - PC11 to Switch2 (Port FastEthernet0/2)
            - PC12 to Switch2 (Port FastEthernet0/3)
            - PC13 to Switch2 (Port FastEthernet0/4)
            - PC14 to Switch2 (Port FastEthernet0/5)
4. **MECH Department:**
    - **Switch3 (MECH Department):**
        - **Use Copper Straight-Through cable to connect:**
            - PC15 to Switch3 (Port FastEthernet0/1)
            - PC16 to Switch3 (Port FastEthernet0/2)
            - PC17 to Switch3 (Port FastEthernet0/3)
            - PC18 to Switch3 (Port FastEthernet0/4)
            - PC19 to Switch3 (Port FastEthernet0/5)

Step 4: Configure IP Addresses on PCs

CSE Department (Subnet: 192.168.1.1/24):

1. PC0: IP Address: 192.168.1.2, Subnet Mask: 255.255.255.0
2. PC1: IP Address: 192.168.1.3, Subnet Mask: 255.255.255.0
3. PC2: IP Address: 192.168.1.4, Subnet Mask: 255.255.255.0
4. PC3: IP Address: 192.168.1.5, Subnet Mask: 255.255.255.0
5. PC4: IP Address: 192.168.1.6, Subnet Mask: 255.255.255.0

IT Department (Subnet: 192.168.2.1/24):

1. PC5: IP Address: 192.168.1.7, Subnet Mask: 255.255.255.0
2. PC6: IP Address: 192.168.1.8, Subnet Mask: 255.255.255.0
3. PC7: IP Address: 192.168.1.9, Subnet Mask: 255.255.255.0
4. PC8: IP Address: 192.168.1.10, Subnet Mask: 255.255.255.0
5. PC9: IP Address: 192.168.1.11, Subnet Mask: 255.255.255.0

AIDS Department (Subnet: 192.168.3.1/24):

1. PC10: IP Address: 192.168.1.12, Subnet Mask: 255.255.255.0
2. PC11: IP Address: 192.168.1.13, Subnet Mask: 255.255.255.0
3. PC12: IP Address: 192.168.1.14, Subnet Mask: 255.255.255.0
4. PC13: IP Address: 192.168.1.15, Subnet Mask: 255.255.255.0
5. PC14: IP Address: 192.168.1.16, Subnet Mask: 255.255.255.0

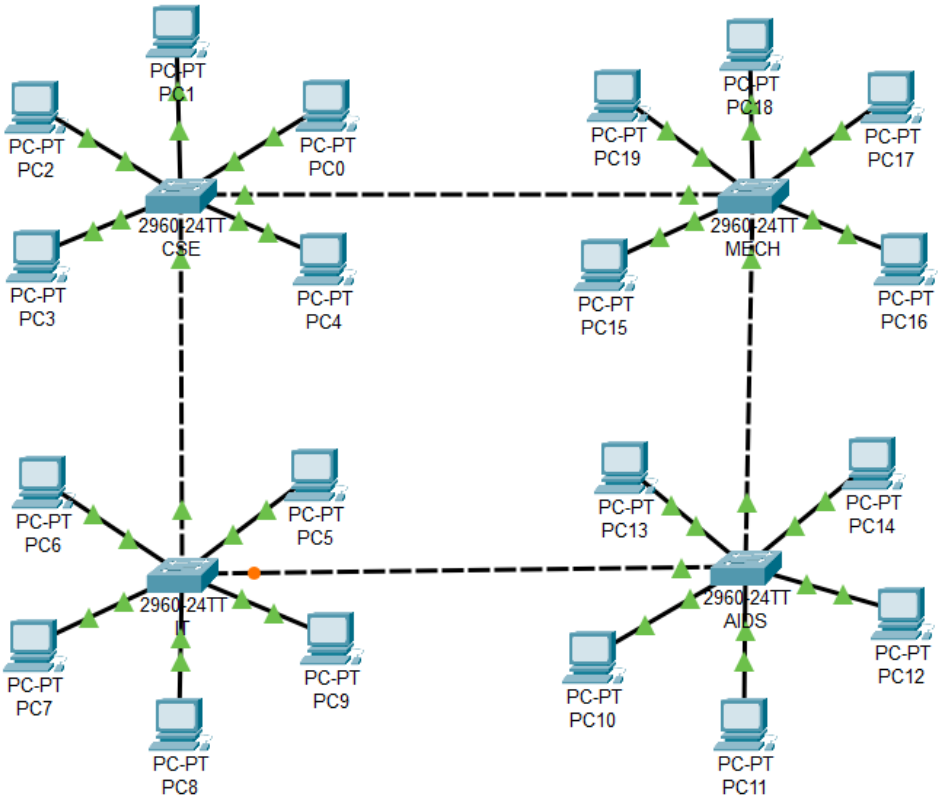
Reg: no: 22BCS081

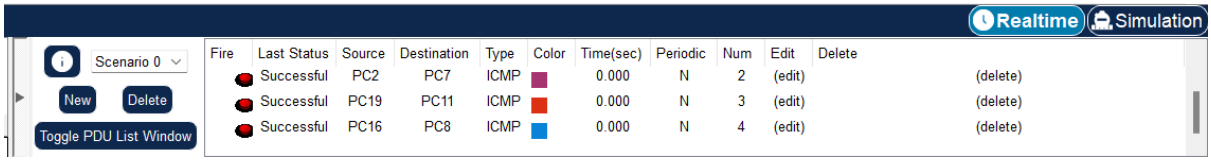
MECH Department (Subnet: 192.168.4.1/24):

1. PC15: IP Address: 192.168.1.17, Subnet Mask: 255.255.255.0
2. PC16: IP Address: 192.168.1.18, Subnet Mask: 255.255.255.0
3. PC17: IP Address: 192.168.1.19, Subnet Mask: 255.255.255.0
4. PC18: IP Address: 192.168.1.20, Subnet Mask: 255.255.255.0
5. PC19: IP Address: 192.168.1.20, Subnet Mask: 255.255.255.0

Step 5: Connect Switches

1. **Use Copper Straight-Through cables to interconnect the switches:**
    - Switch0 (Port GigabitEthernet0/1) to Switch1 (Port GigabitEthernet0/1)
    - Switch0 (Port GigabitEthernet0/2) to Switch2 (Port GigabitEthernet0/1)
    - Switch3 (Port GigabitEthernet0/1) to Switch1 (Port GigabitEthernet0/2)
    - Switch3 (Port GigabitEthernet0/2) to Switch2 (Port GigabitEthernet0/2)



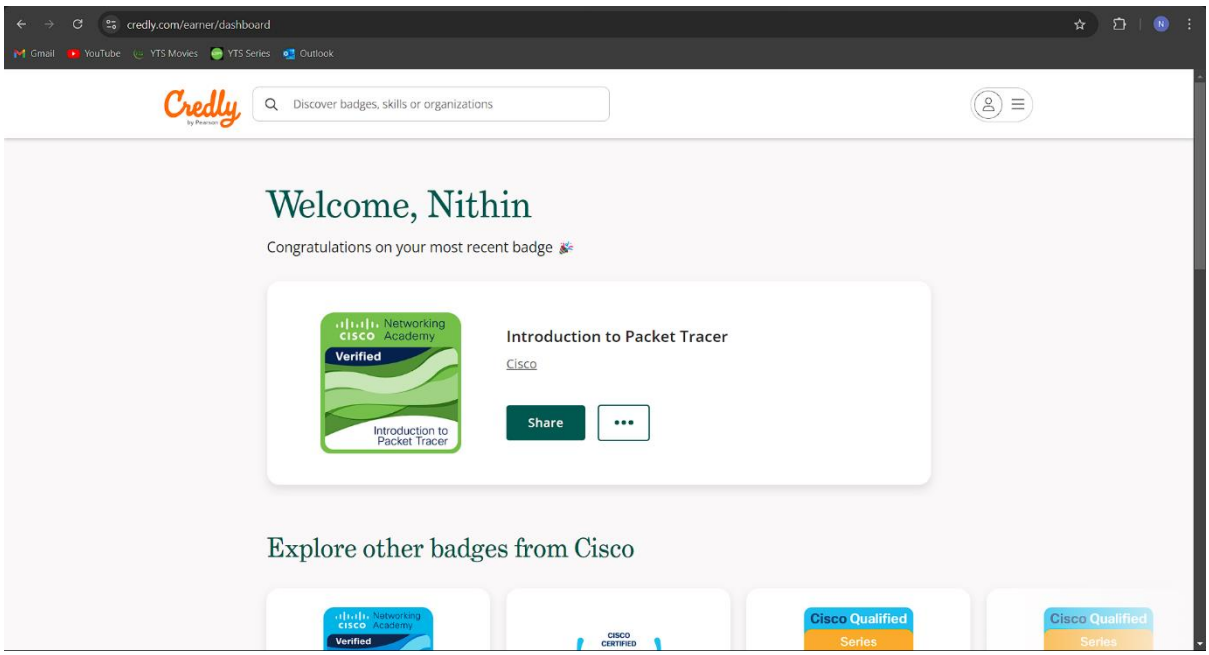


Reg: no: 22BCS081

PostLab:

Completion of the Cisco Packet tracer fundamental course and attachment of the certificate.

https://skillsforall.com/topics/cisco-packet-tracer?utm_source=n...



Certificate Verification Link:

https://www.credly.com/badges/dc101725-a8e7-412d-85de-5a96cdb14653/public_url

Result:

All experiments have been successfully executed, with no errors or issues encountered. The expected results have been achieved, as demonstrated by the attached screenshots.

Reg: no: 22BCS081

Exp: no: 3

Analyse the network traffic using Wireshark tool.

Aim:

To analyse the network traffic using Wireshark tool and write the syntax, execute and place the screenshot of all the commands worked on.

a. Explain what is a packet analyzer ?

Packet Analyzer

A packet analyzer, also known as a network analyzer, protocol analyzer, or packet sniffer, is a tool used to capture, monitor, and analyze data packets traveling across a network. It allows network administrators, security professionals, and developers to examine the data being transmitted over a network in real-time or from saved packet capture files. By capturing all traffic on a network, a packet analyzer can provide detailed information about each packet, including its source and destination IP addresses, the protocols used, and the data payload. This capability makes packet analyzers indispensable for troubleshooting network issues, ensuring security, and optimizing network performance.

Beyond troubleshooting, packet analyzers are also crucial for security analysis. They can detect and help prevent unauthorized access, monitor for malicious activities, and identify vulnerabilities by examining the flow of data through a network. Additionally, developers use packet analyzers to test and debug network protocols, ensuring that applications interact with the network as intended. Popular tools like Wireshark, tcpdump, and SolarWinds offer varying degrees of functionality, from simple packet capture to advanced network traffic analysis, making packet analyzers versatile tools in both network management and cybersecurity.

Applications of Packet Analyzers

- **Network Monitoring and Troubleshooting:** Identifies network bottlenecks, misconfigurations, and connectivity issues.
- **Security Analysis:** Detects unauthorized access, malware, and suspicious activities.
- **Performance Optimization:** Analyzes traffic patterns to improve network performance.
- **Protocol Development and Debugging:** Assists developers in testing and debugging network protocols.

b. Explain the types of packet analyzers.

Types of Packet Analyzers

1. Network Tap Analyzers:

Explanation: Network Tap Analyzers rely on hardware devices called network taps to physically intercept network cables and capture all data passing through. They offer a passive monitoring approach, ensuring that the network's performance and integrity are unaffected during the analysis.

Example:

- Garland Technology Network Taps
- NetOptics Bypass Switch
- Datacom Systems TAPs

2. Software-Based Packet Analyzers:

Explanation: These packet analyzers are software applications installed on computers or servers to capture and examine network traffic on specific segments. They are versatile and widely used for various purposes, including troubleshooting and security monitoring.

Example:

- Wireshark
- tcpdump
- Microsoft Network Monitor

3. Inline Packet Analyzers:

Explanation: Inline packet analyzers are deployed directly within the network path, such as within firewalls or routers. They not only monitor network traffic but also have the capability to block or alter packets based on predefined rules, making them suitable for proactive security measures.

Example:

- Cisco Firepower
- Palo Alto Networks Next-Generation Firewall
- Juniper SRX Series Firewalls

4. Remote Packet Analyzers:

Explanation: Remote packet analyzers allow network administrators to capture and analyze traffic from distant locations without being physically present at the monitoring site. This capability is particularly useful for managing large-scale, distributed networks.

Example:

- SolarWinds Network Performance Monitor
- PRTG Network Monitor
- Nagios XI

5. Protocol-Specific Packet Analyzers:

Explanation: These analyzers focus on capturing and analyzing traffic for specific protocols like HTTP, FTP, or VoIP. They provide in-depth insights into the performance and behavior of specific applications or services on the network.

Example:

- SIPp (VoIP)
- Fiddler (HTTP)
- TShark (various protocols)

6. Capture Appliances:

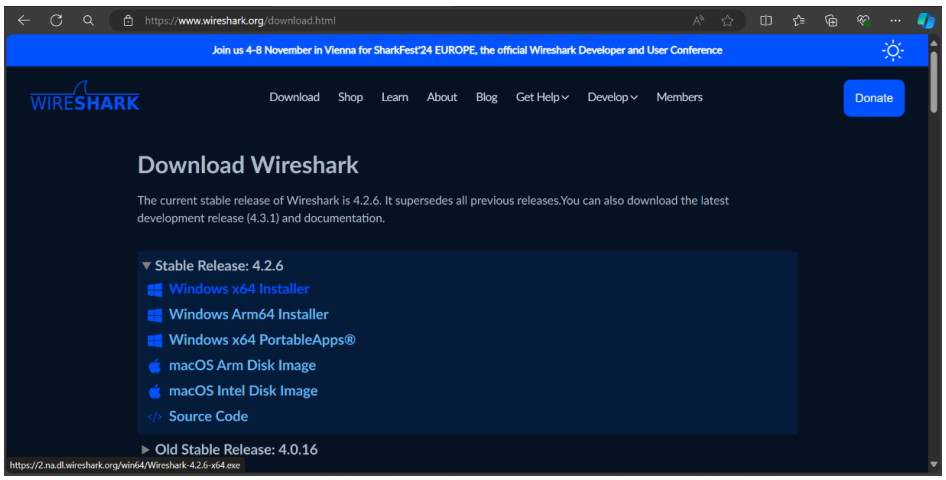
Explanation: Capture appliances are dedicated hardware devices designed for long-term traffic capture and storage. These devices are optimized for high-speed data capture and are often used in environments where detailed and prolonged traffic analysis is required.

Example:

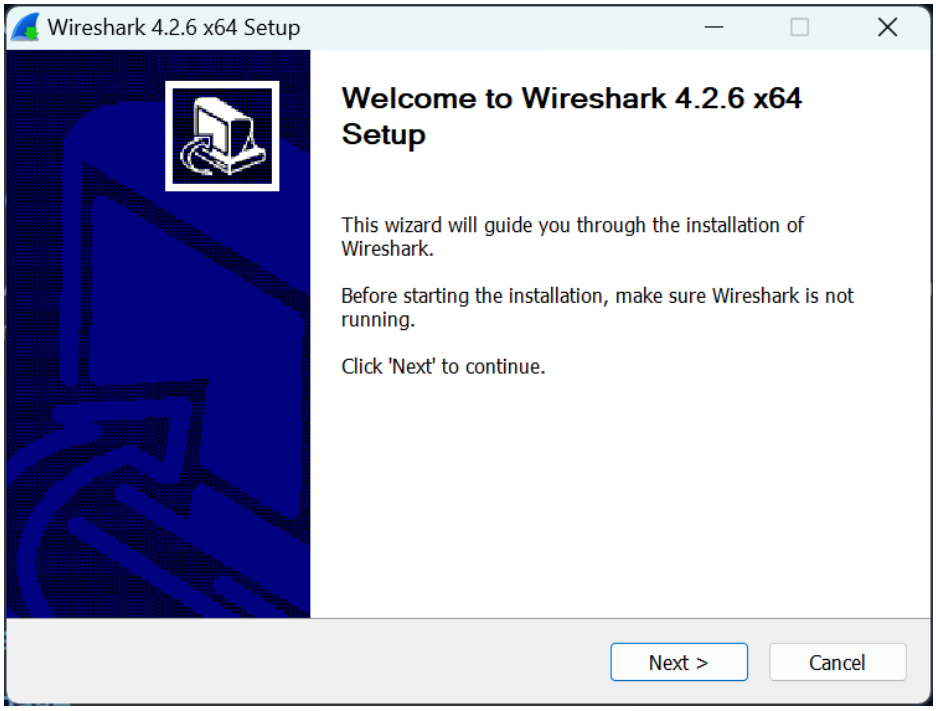
- EndaceProbe
- Riverbed SteelCentral
- VSS Monitoring Distributed Series

c. Dinesh a tech evangelist is interested to learn about packet analyzer tools and decided to use Wireshark. Dinesh trust in you, now your job is help Dinesh in learning Wireshark, now help Dinesh in installing Wireshark tool .

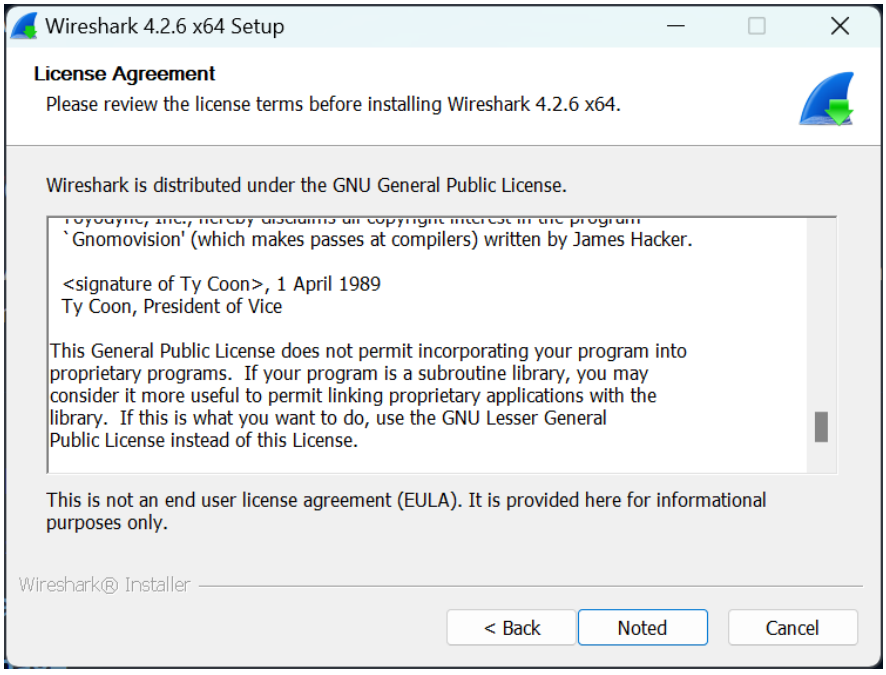
Step 1: Visit the official Wireshark website using any web browser. A page with different installers of Wireshark will be shown. Click on 'Windows x64 Installer'.



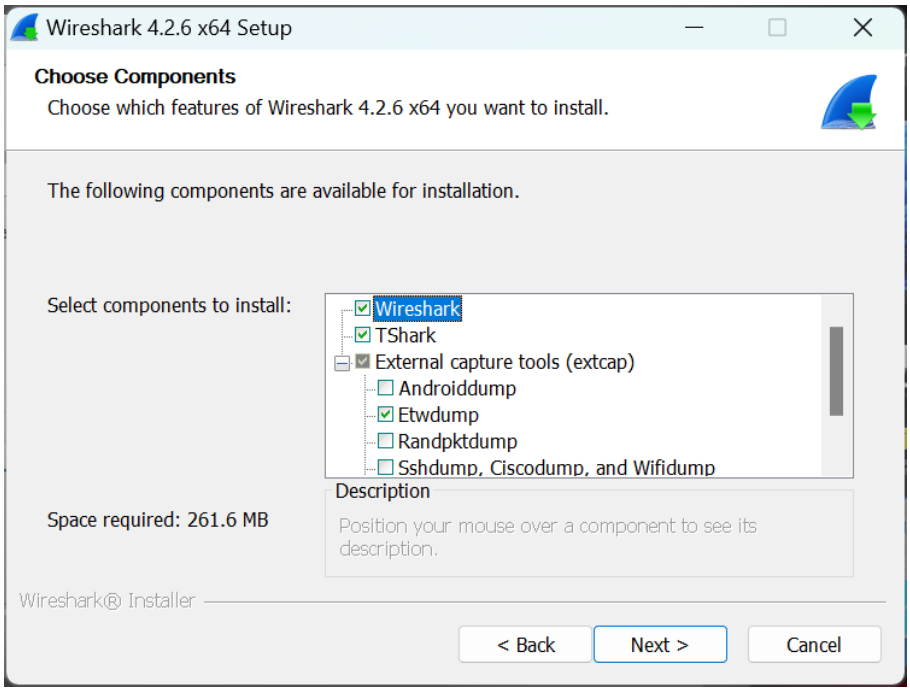
Step 2: Once the download is complete, open it. Setup screen will appear, click on Next.



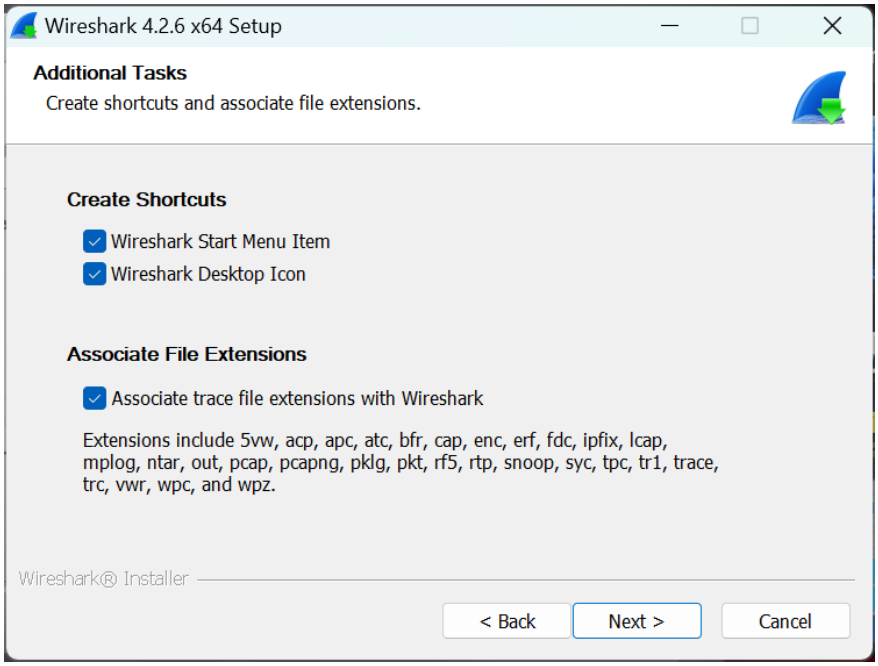
Step 3: The next screen will be of License Agreement, click on Noted.



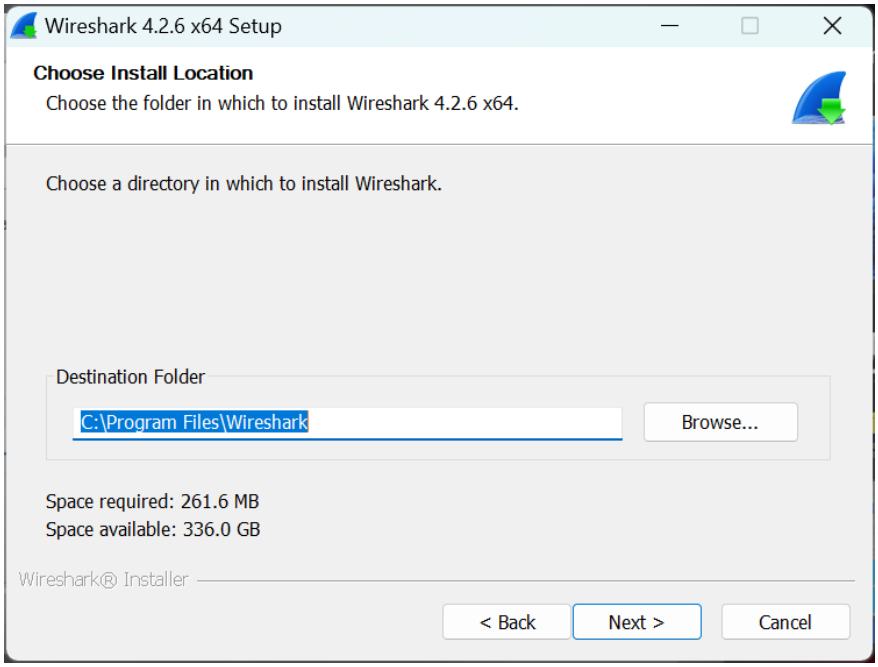
Step 4: This screen is for choosing components, all components are already marked so don't change anything just click on the Next button.



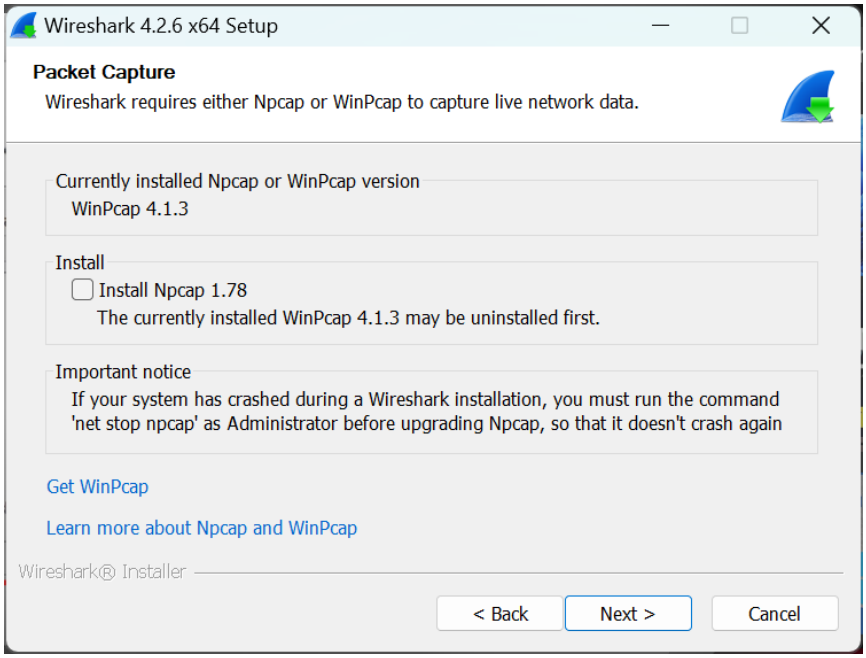
Step 5: This screen is of choosing shortcuts like start menu or desktop icon along with file extensions which can be intercepted by Wireshark, tick all boxes and click on Next button.



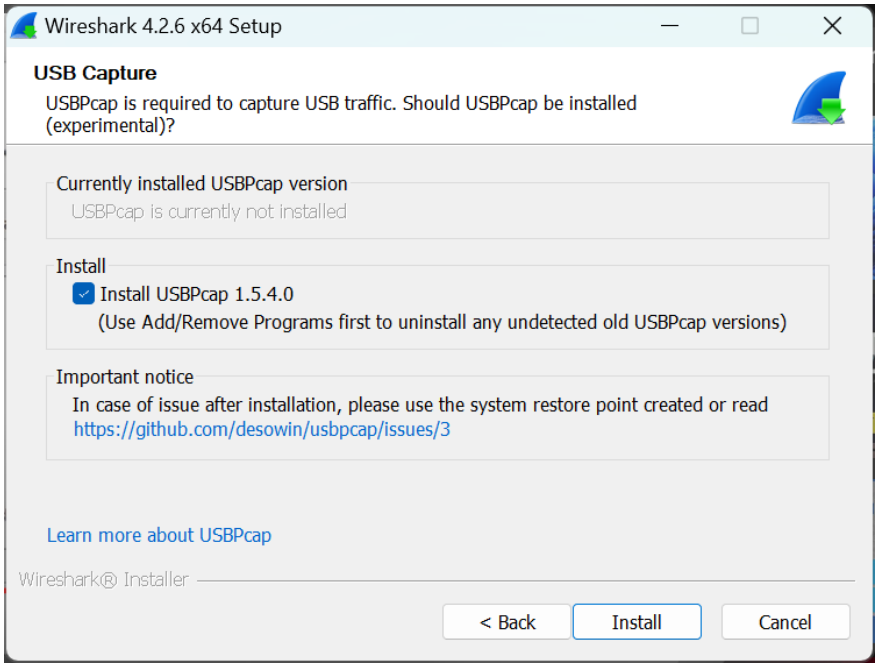
Step 6: The next screen will be of installation location so choose the drive which will have sufficient memory space for installation. It needs only a memory space of 261.6 MB.



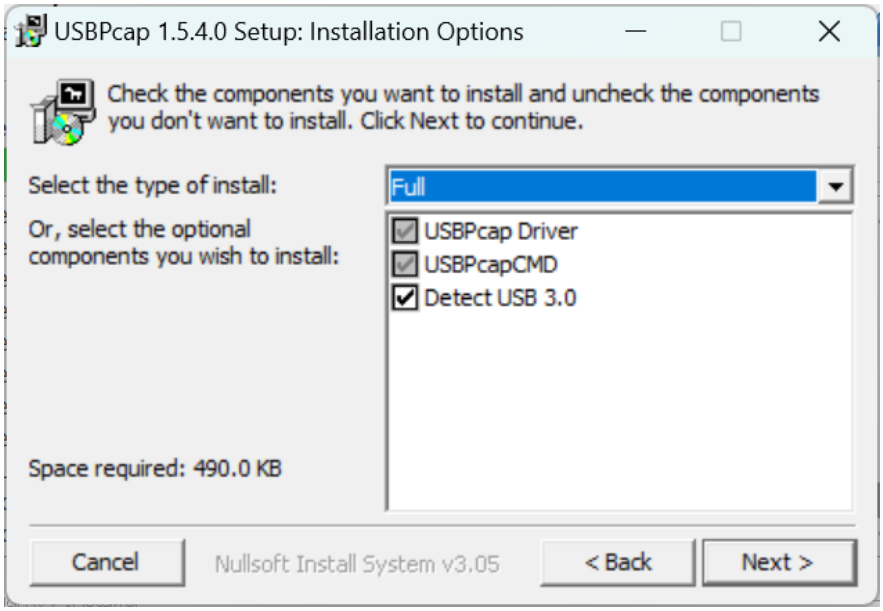
Step 7: Next screen has an option to install Npcap which is used with Wireshark to capture packets. pcap means packet capture so the install option is already checked don't change anything and click the next button.



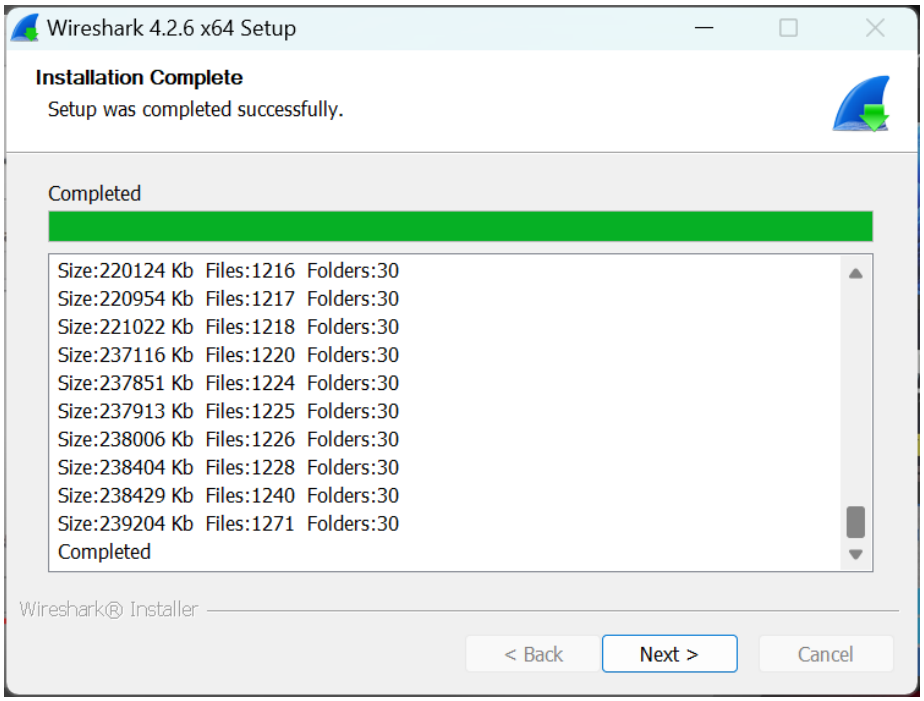
Step 8: Next screen is about USB network capturing so it is one's choice to use it or not, click on Install.



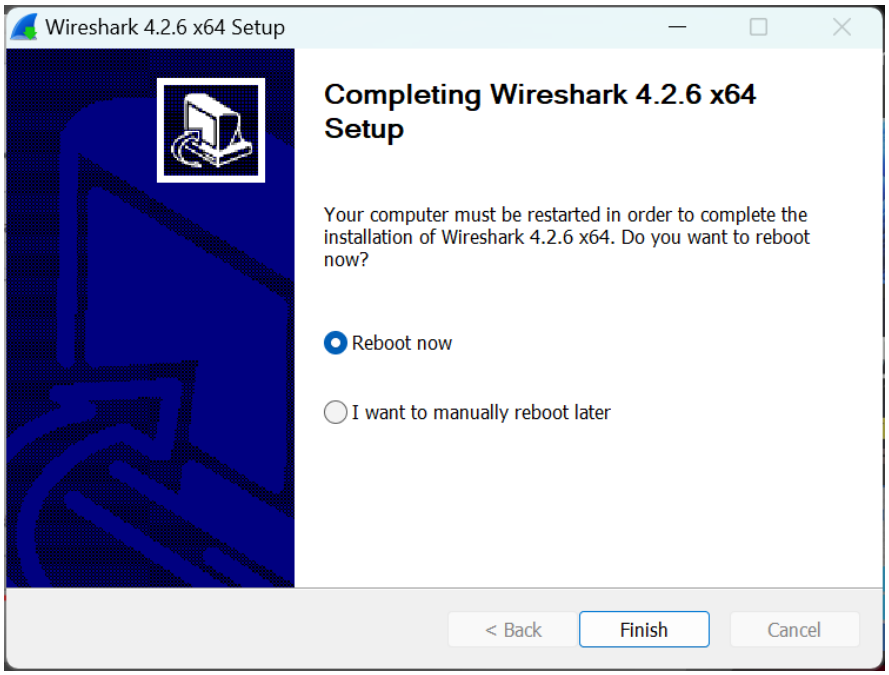
Step 9: After this installation process will start. This installation will prompt for Npcap and USBPcap installation. Agree to everything and click on next.



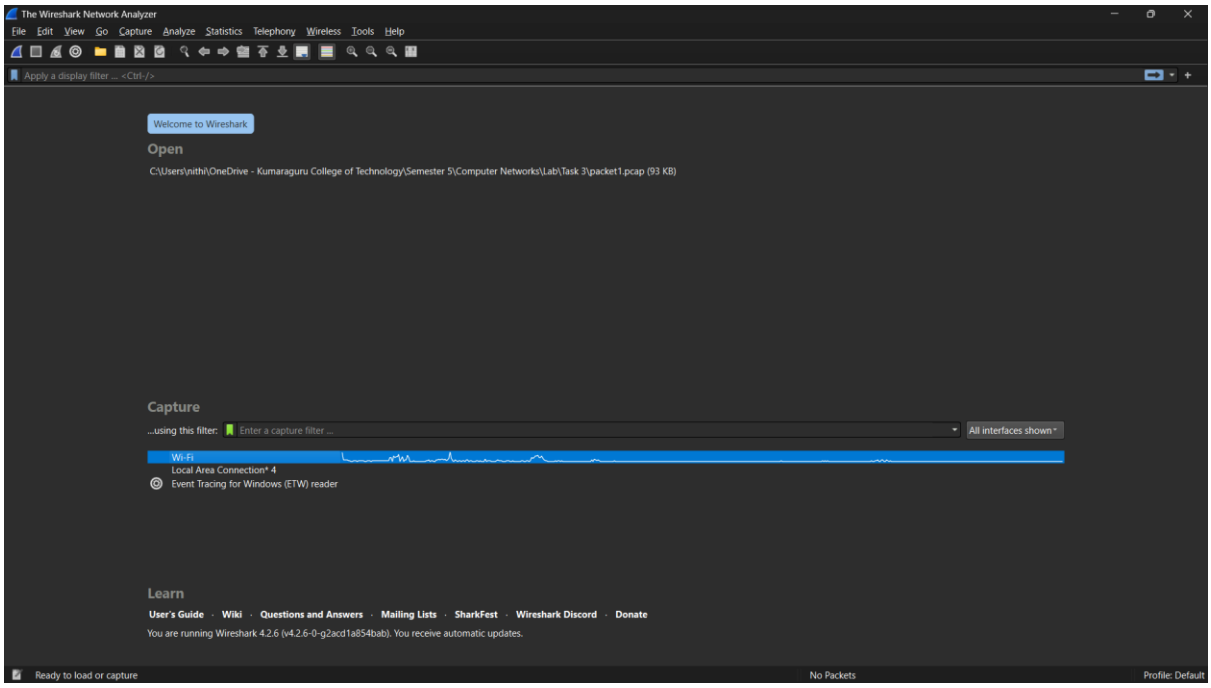
Step 10: After the installation process of Wireshark is complete click on the Next button.



Step 11: Click on Finish after the installation process of Wireshark is complete. Wireshark is successfully installed on the system and an icon is created on the desktop.



Step 12: Now run the software and see the interface. At this point, you have successfully installed Wireshark on your windows system



d. Now help Dinesh to understand the overview of Wireshark.

Overview of Wireshark

Wireshark is a powerful, widely used network protocol analyzer that allows users to capture and interactively browse the traffic running on a computer network. It is an essential tool for network administrators, security professionals, and developers who need to diagnose network problems, troubleshoot issues, or analyze network traffic for security purposes.

Key Features of Wireshark:

- **Packet Capture:** Wireshark captures network packets in real-time, providing detailed information about each packet, including headers, payloads, and metadata. Users can capture traffic from live networks or read from previously saved capture files.
- **Detailed Protocol Analysis:** Wireshark supports a vast array of network protocols, offering detailed analysis and decoding for over a thousand protocols, including TCP/IP, HTTP, DNS, FTP, and many more. This makes it invaluable for understanding how different protocols operate and interact within a network.
- **User-Friendly Interface:** Wireshark's graphical user interface (GUI) is designed to be user-friendly, making it easier to navigate through captured data. The interface allows users to apply filters, search for specific data, and highlight important information.

- **Powerful Filtering Capabilities:** Wireshark provides robust filtering tools, allowing users to focus on specific types of traffic or packets of interest. Display filters, capture filters, and coloring rules can be applied to quickly isolate relevant data.
- **Cross-Platform Availability:** Wireshark is available on multiple platforms, including Windows, macOS, and Linux, making it accessible to a broad range of users.
- **Open-Source and Extensible:** Wireshark is an open-source tool, meaning it is freely available and supported by a large community. Users can extend its functionality by writing custom dissectors or plugins to analyze new or proprietary protocols.

Common Use Cases:

- **Network Troubleshooting:** Wireshark helps identify network bottlenecks, latency issues, and misconfigurations by analyzing packet flows and protocol behaviors.
- **Security Analysis:** Security professionals use Wireshark to detect malicious traffic, investigate security incidents, and analyze attacks such as packet sniffing, man-in-the-middle attacks, and more.
- **Protocol Development:** Developers use Wireshark to debug and test network protocols, ensuring they function correctly during communication between devices.
- **Educational Purposes:** Wireshark is a valuable educational tool, helping students and professionals learn about network protocols and their operations by providing a hands-on, interactive experience.

e. While you are explaining the overview of Wireshark Dinesh observed some IP address in the packet capturing window, Dinesh want to extract the information of particular IP address and see where it is going and from where it is receiving the information, help Dinesh to filter particular IP address and its source and destination.

Step 1: Open Wireshark: Launch Wireshark and start capturing packets on the desired network interface or open a saved capture file.

Step 2: Start Packet Capture: Begin capturing packets or load an existing capture to analyze.

Step 3: Apply IP Address Filter:

- Use ip.addr == 192.168.70.99 to filter traffic for a specific IP address.
- For filtering by source or destination, use ip.src == 192.168.70.99

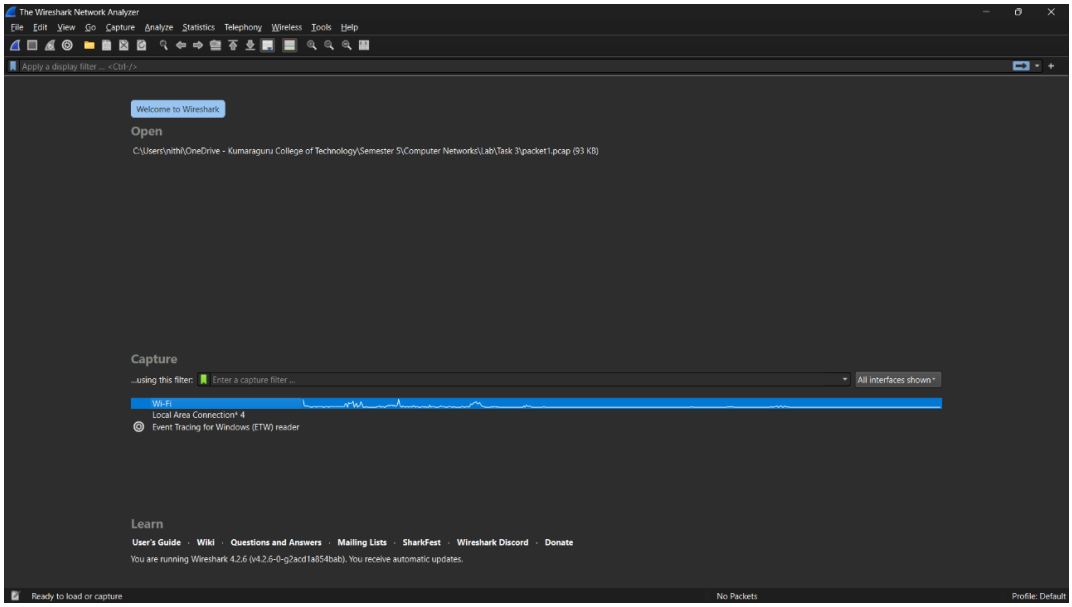
Step 4: Analyze Filtered Traffic: Review the source and destination addresses to understand where the traffic is coming from and going to.

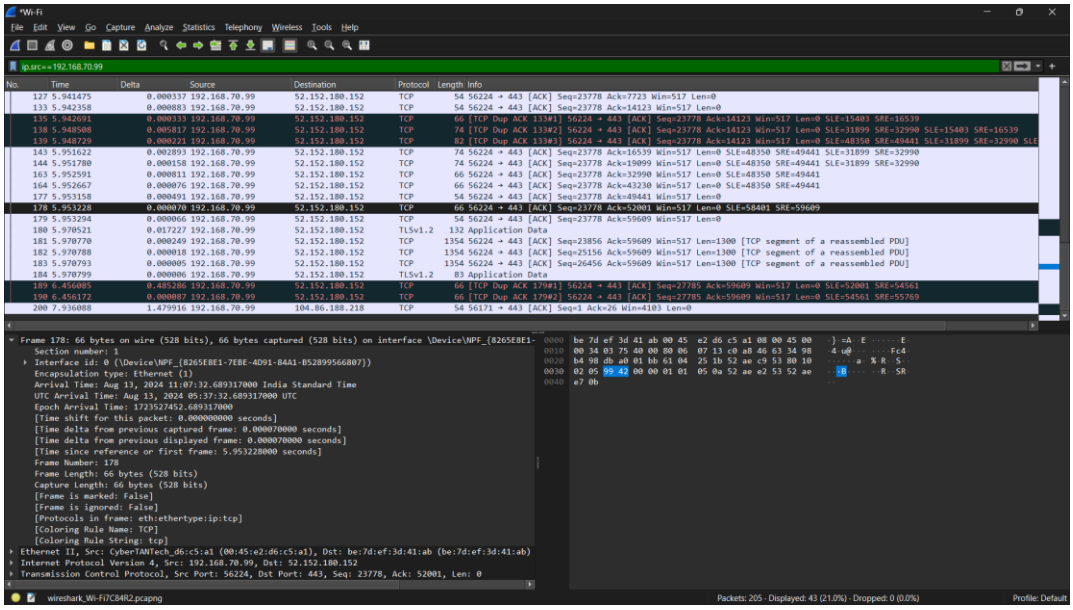
- **Time:** Timing of each packet captured during a network session.
- **Source:** The IP address that sent the packet.

- **Destination:** The IP address that received the packet.
- **Protocol:** The protocol used (e.g., TCP, UDP, HTTP).
- **Length:** Size of each packet captured during the network session.
- **Info:** Additional information about the packet, such as the ports used.

Step 5: Examine Packet Details: Click on any packet to view its detailed information in the packet details pane.

Step 6: Export Filtered Data: If needed, export the filtered packets for further analysis.





Result:

Thus, the network traffic was analysed using Wireshark tool and the commands were executed and verified successfully.

Department of Computer Science and Engineering
U18CSI5201_Computer Networks Laboratory

Reg: no: 22BCS081

Exp: no: 4

Simulation of Data Link layer and Network Layer Protocols.

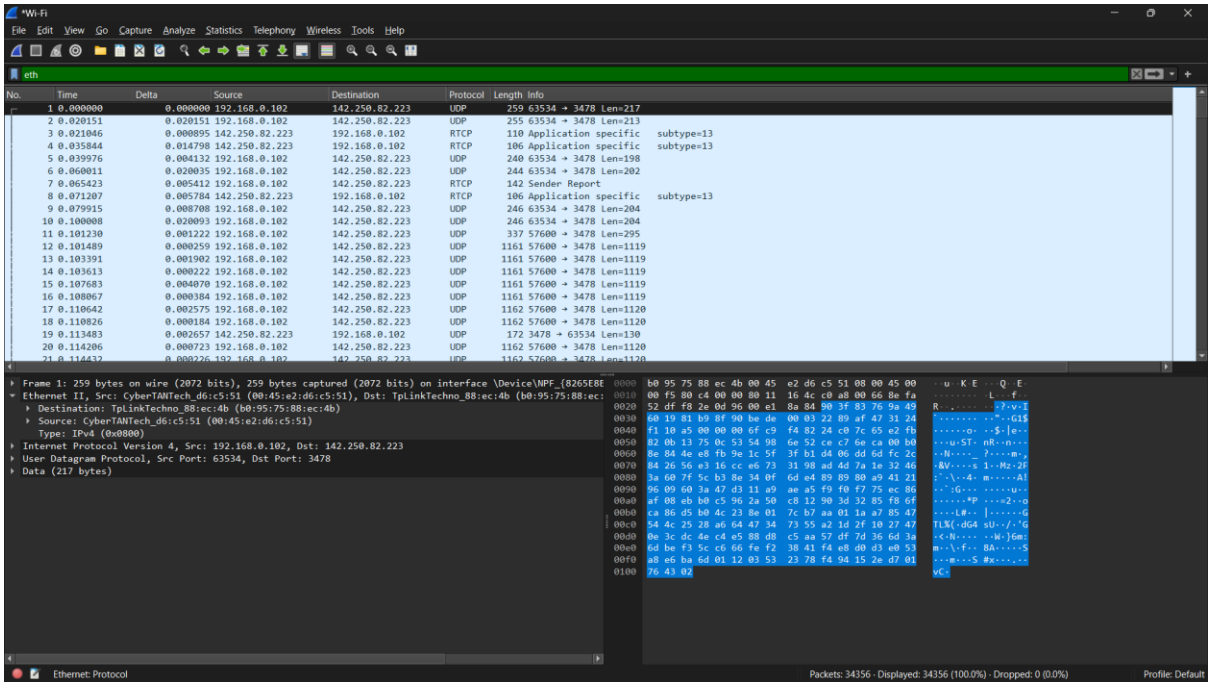
Aim:

To simulate Data Link layer and Network Layer Protocols and write the syntax, execute and place the screenshot for all the commands worked on.

a. Demonstrate the frame transmission scenario using Sniffing tool.

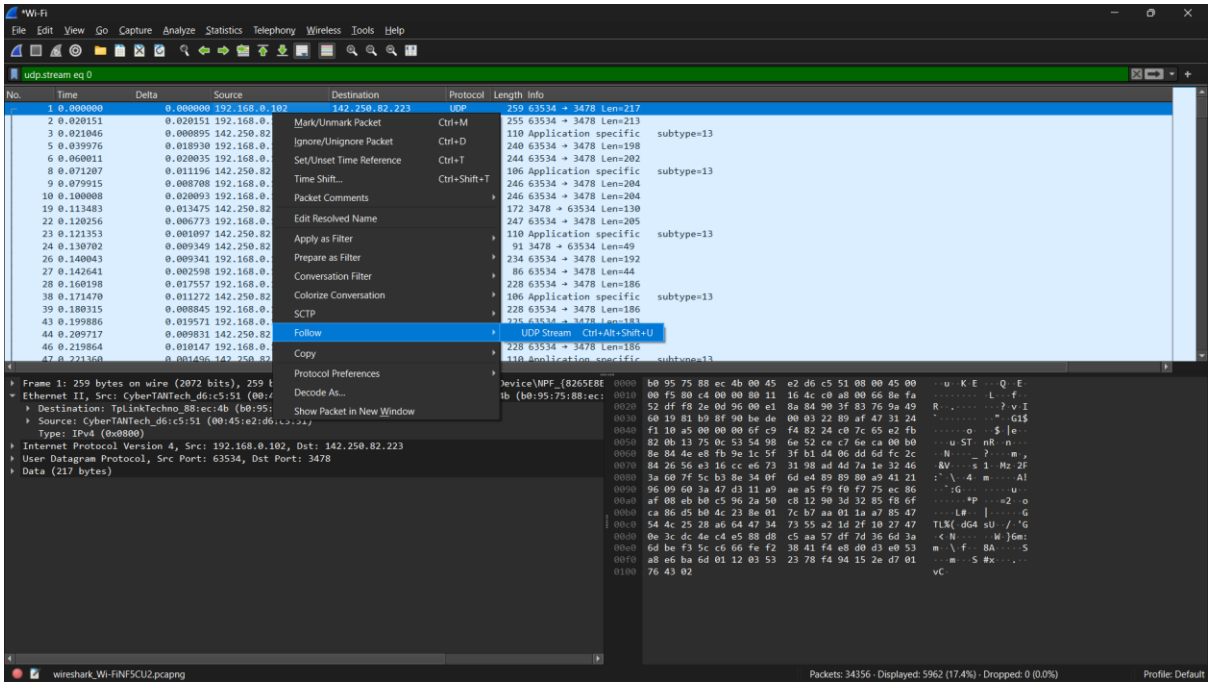
Syntax:

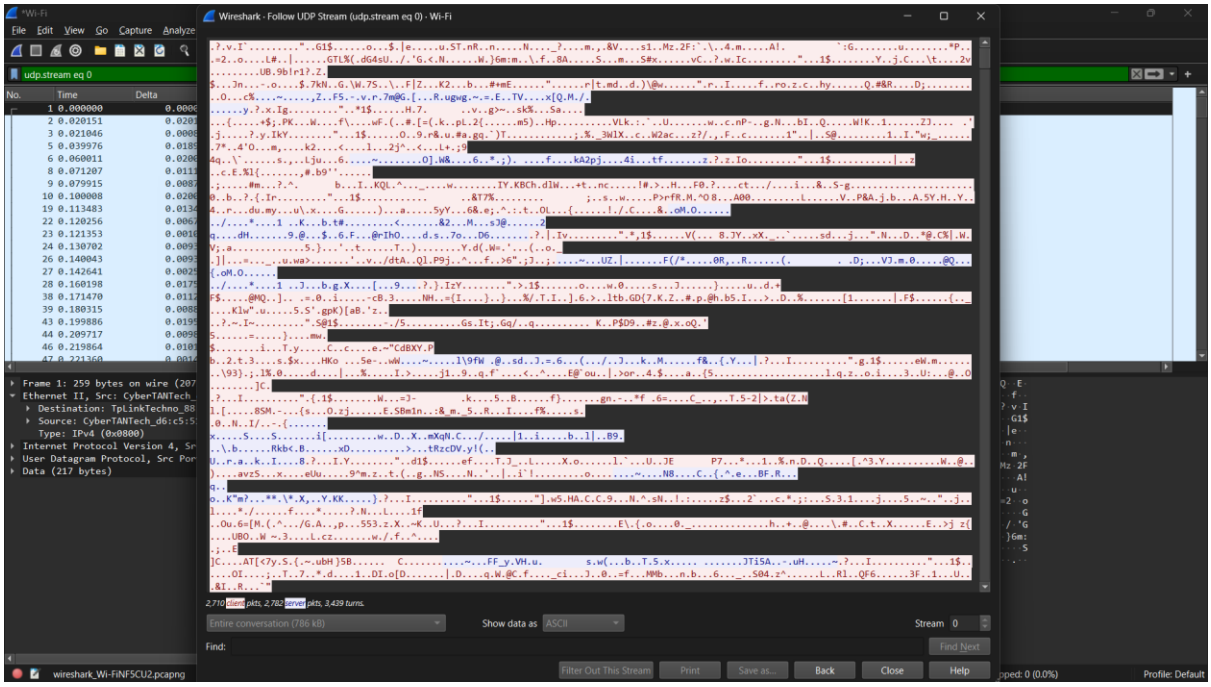
1. Open Wireshark and start a new capture on the network interface you want to monitor (e.g., Ethernet, Wi-Fi).
2. Perform a network activity that involves communication between devices, like accessing a shared folder on a networked computer or pinging another computer on the same network.
3. Stop the capture in Wireshark after some packets have been captured.
4. Look for frames in the capture:
    - Filter for Data Link layer protocols like Ethernet by typing eth in the filter bar.
    - You should see frames with information like source and destination MAC addresses, Ether Type, etc.
5. Select a frame to view its details, including the frame header and payload.



Department of Computer Science and Engineering
U18CSI5201_Computer Networks Laboratory

Reg: no: 22BCS081





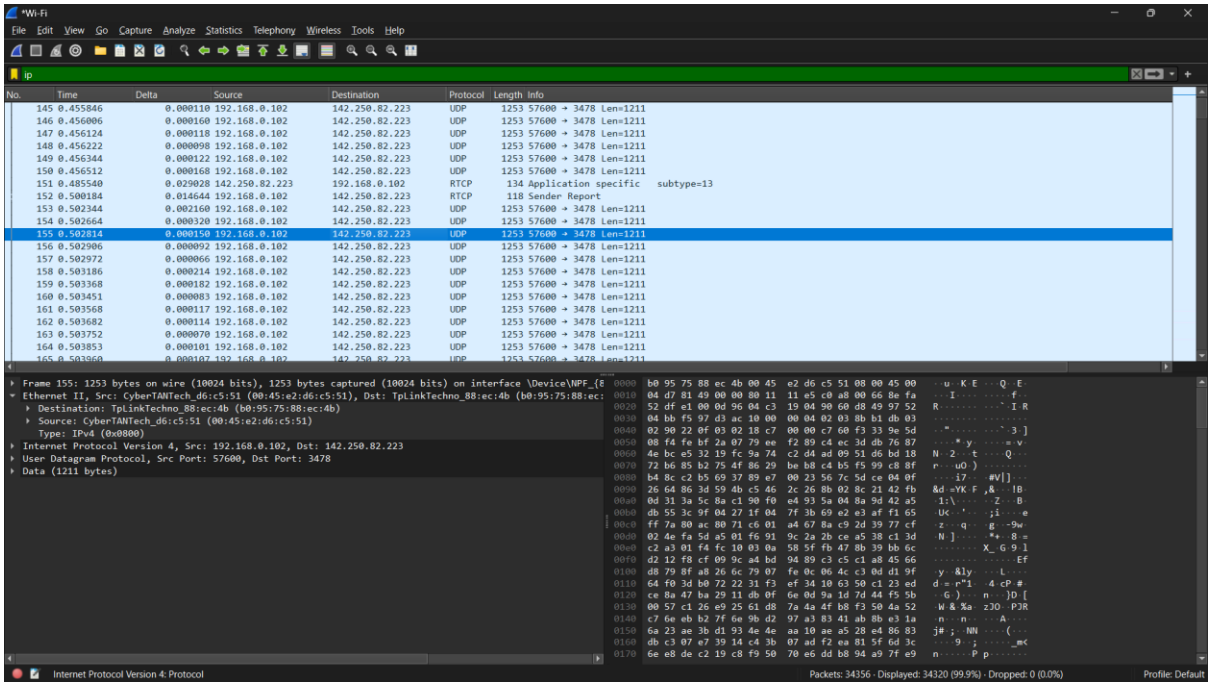
Nithin Prasath C - 22BCS081

Reg: no: 22BCS081

b. Demonstrate the packet transmission scenario using Sniffing tool.

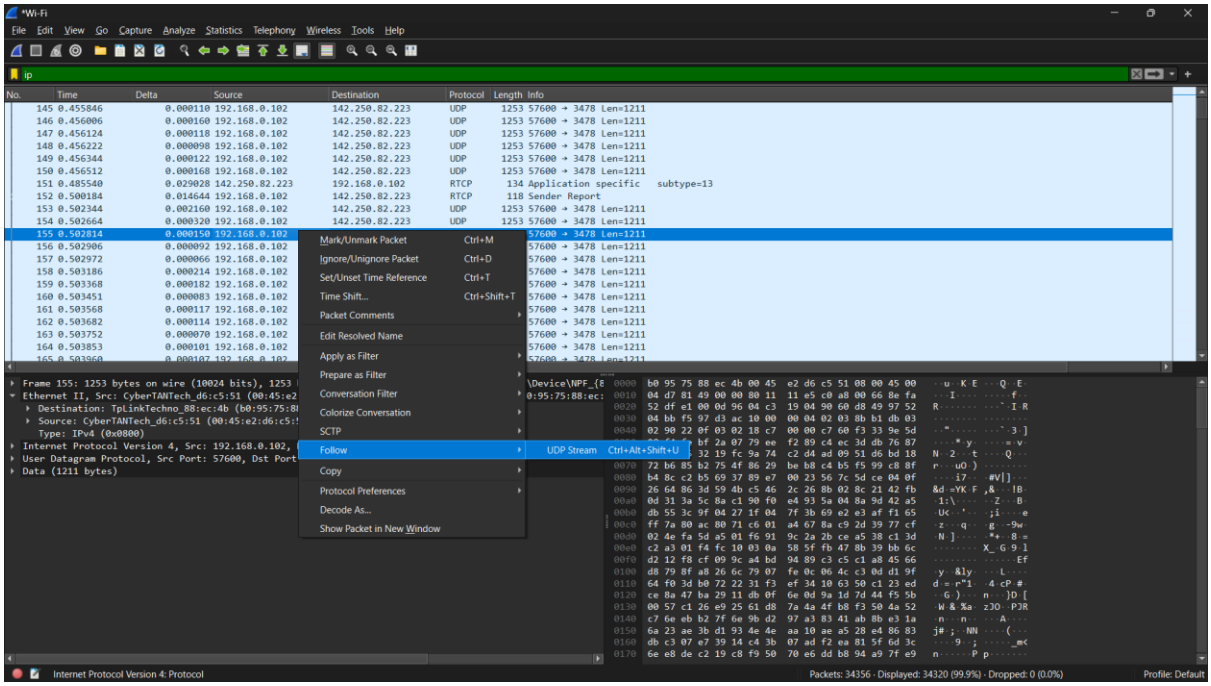
Syntax:

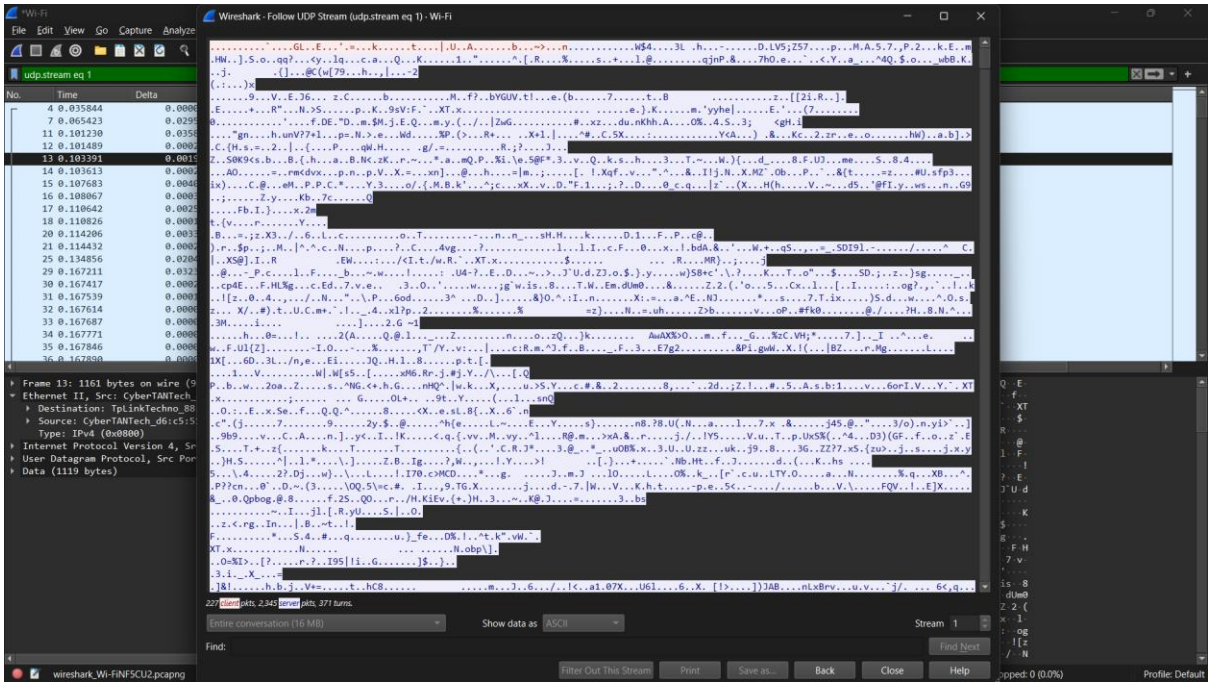
1. In Wireshark, start a new capture again on the same network interface.
2. Perform a network activity that involves communication over IP, such as opening a website in a web browser or using ping to contact an external server (e.g., ping google.com).
3. Stop the capture after a few packets are captured.
4. Look for packets in the capture:
    - Filter for Network layer protocols like IP by typing ip in the filter bar.
    - You should see packets containing information like source and destination IP addresses, protocol types (e.g., TCP, UDP), etc.
5. Select a packet to view its details, including the IP header and encapsulated segment data.
6. To filter for specific protocols, use more detailed filters like:
    - tcp for TCP packets.
    - udp for UDP packets.
    - icmp for ICMP packets (used in ping).



Department of Computer Science and Engineering
U18CSI5201_Computer Networks Laboratory

Reg: no: 22BCS081

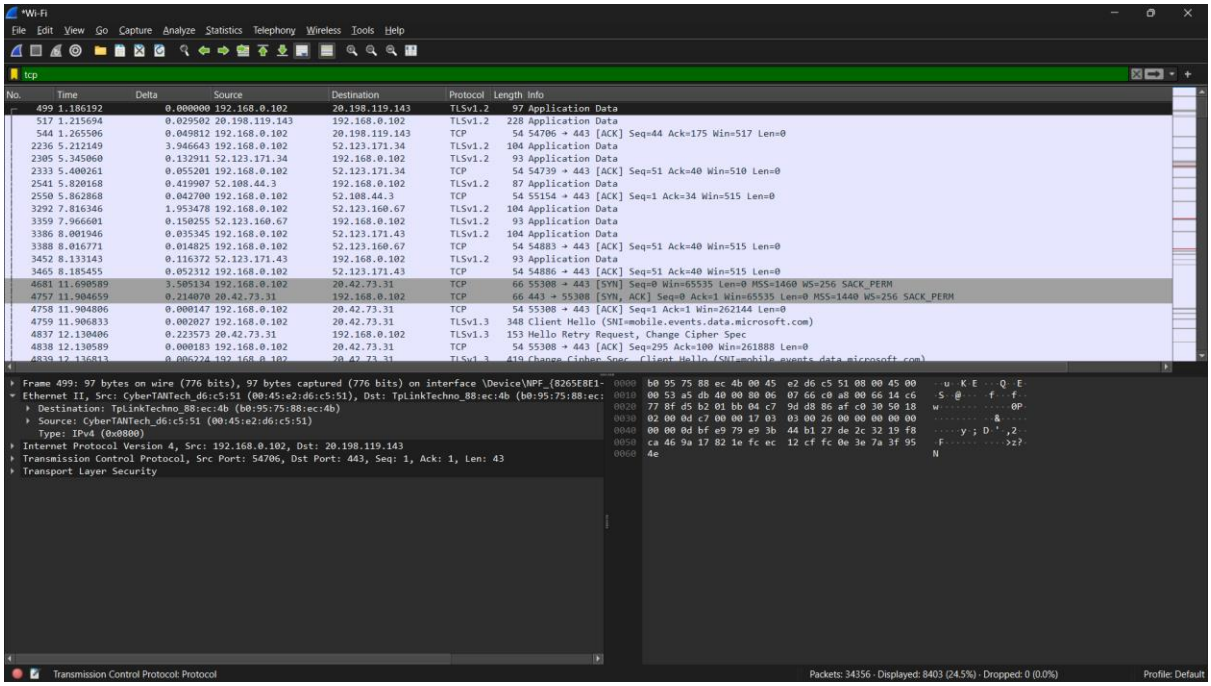


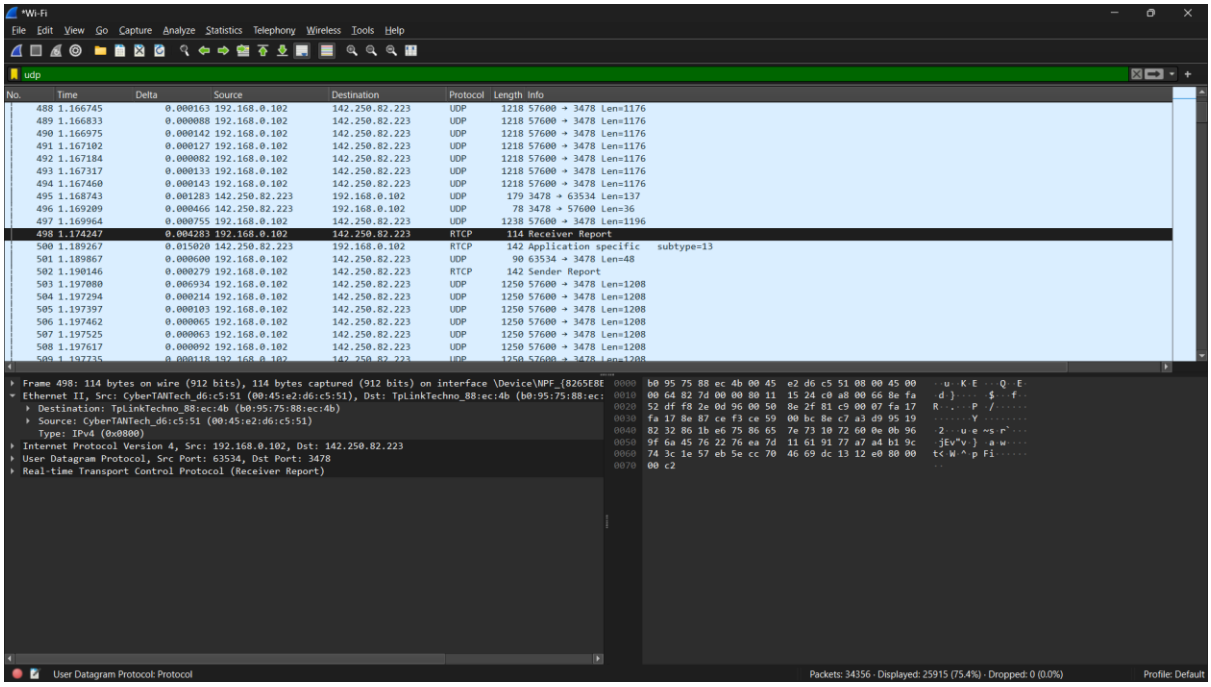


Nithin Prasath C - 22BCS081

Department of Computer Science and Engineering
U18CSI5201_Computer Networks Laboratory

Reg: no: 22BCS081





Result:

All experiments have been successfully executed, with no errors or issues encountered. The expected results have been achieved, as demonstrated by the attached screenshots.

Nithin Prasath C - 22BCS081

Reg: no: 22BCS081

Exp: no: 5

Performance analysis of routing protocols using simulation tool.

Aim:

To simulate routing protocols for Data Link layer and Network Layer, and to write the syntax, execute the commands, and capture the screenshots

a. Construct 3 to 4 networks each connected by a router and enable packet transmission between all the networks.

Syntax:

Step 1: Setting up Devices

Components in the Topology:

1. 3 Routers (1841) - Router 4, Router 5, Router 6
2. 3 Switches (2960) - Switch 0, Switch 1, Switch 2
3. 6 PCs - PC0, PC1 (Network A), PC2, PC3 (Network B), PC4, PC5 (Network C)

Step 2: Device Connections

Connecting the Devices:

- **PCs to Switches:**
  - Network A:
    - PC0 -> Switch0 (FastEthernet0/1)
    - PC1 -> Switch0 (FastEthernet0/2)
  - Network B:
    - PC2 -> Switch1 (FastEthernet0/1)
    - PC3 -> Switch1 (FastEthernet0/2)
  - Network C:
    - PC4 -> Switch2 (FastEthernet0/1)
    - PC5 -> Switch2 (FastEthernet0/2)
- **Switches to Routers:**
  - Switch0 -> Router 4 (FastEthernet0/0)
  - Switch1 -> Router 5 (FastEthernet0/0)
  - Switch2 -> Router 6 (FastEthernet0/0)

- **Router-to-Router Connections (using Serial Interfaces):**
    - Router 4 (Se0/0/0) -> Router 5 (Se0/0/0)
    - Router 5 (Se0/0/1) -> Router 6 (Se0/0/0)
    - Router 6 (Se0/0/1) -> Router 4 (Se0/0/1)

Step 3: Configuring PCs

For Each PC:

Go to: Desktop -> IP Configuration.

- **Network A (PC0, PC1):**
    - IP Addresses:
        - PC0: 192.168.1.2
        - PC1: 192.168.1.3
    - Subnet Mask: 255.255.255.0
    - Default Gateway: 192.168.1.1 (Router 4's IP)
- **Network B (PC2, PC3):**
    - IP Addresses:
        - PC2: 192.168.2.2
        - PC3: 192.168.2.3
    - Subnet Mask: 255.255.255.0
    - Default Gateway: 192.168.2.1 (Router 5's IP)
- **Network C (PC4, PC5):**
    - IP Addresses:
        - PC4: 192.168.3.2
        - PC5: 192.168.3.3
    - Subnet Mask: 255.255.255.0
    - Default Gateway: 192.168.3.1 (Router 6's IP)

Step 4: Configuring the Routers

Router Configuration Steps:

Router 4 (Network A):

1. Configure FastEthernet Interface (Fa0/0):
    - IP Address: 192.168.1.1
    - Subnet Mask: 255.255.255.0
    - Turn on the interface.

2. Configure Serial Interface (Se0/0/0) - Link to Router 5:
    - IP Address: 10.0.0.2
    - Subnet Mask: 255.0.0.0
    - Clock Rate: 148000
    - Turn on the interface.
3. Configure Serial Interface (Se0/0/1) - Link to Router 6:
    - IP Address: 11.0.0.1
    - Subnet Mask: 255.0.0.0
    - Clock Rate: 148000
    - Turn on the interface.

Router 5 (Network B):

1. Configure FastEthernet Interface (Fa0/0):
    - IP Address: 192.168.2.1
    - Subnet Mask: 255.255.255.0
    - Turn on the interface.
2. Configure Serial Interface (Se0/0/0) - Link to Router 4:
    - IP Address: 10.0.0.1
    - Subnet Mask: 255.0.0.0
    - Turn on the interface.
3. Configure Serial Interface (Se0/0/1) - Link to Router 6:
    - IP Address: 12.0.0.2
    - Subnet Mask: 255.0.0.0
    - Clock Rate: 148000
    - Turn on the interface.

Router 6 (Network C):

1. Configure FastEthernet Interface (Fa0/0):
    - IP Address: 192.168.3.1
    - Subnet Mask: 255.255.255.0
    - Turn on the interface.
2. Configure Serial Interface (Se0/0/0) - Link to Router 5:
    - IP Address: 12.0.0.1
    - Subnet Mask: 255.0.0.0
    - Turn on the interface.

3. Configure Serial Interface (Se0/0/1) - Link to Router 4:
    - IP Address: 11.0.0.2
    - Subnet Mask: 255.0.0.0
    - Turn on the interface.

Step 5: Configuring Routing (RIP)

Enabling RIP on Each Router:

1. Router 4 (Network A):
    - Network: 10.0.0.0 (Router 4 and Router 5 connection)
    - Network: 11.0.0.0 (Router 4 and Router 6 connection)
2. Router 5 (Network B):
    - Network: 10.0.0.0 (Router 4 and Router 5 connection)
    - Network: 12.0.0.0 (Router 5 and Router 6 connection)
3. Router 6 (Network C):
    - Network: 11.0.0.0 (Router 4 and Router 6 connection)
    - Network: 12.0.0.0 (Router 5 and Router 6 connection)

Save configurations after setting up RIP for all routers.

Step 6: Testing the Network

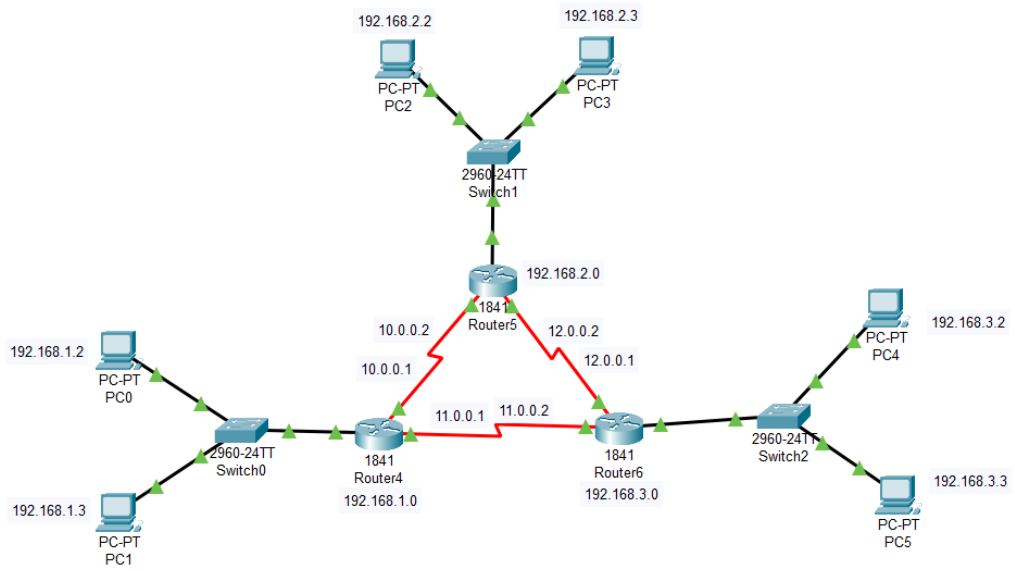
To test the connectivity:

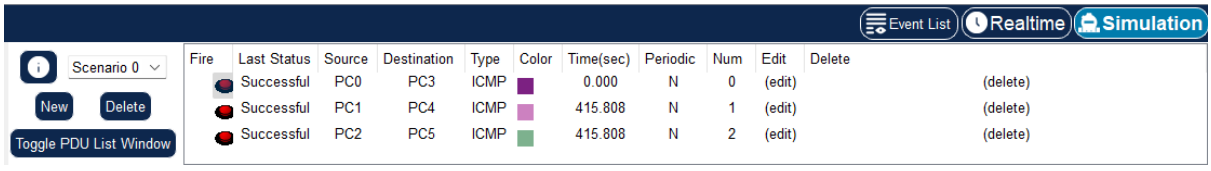
1. Ping from a PC in one network to the router in the same network:
    - For example, from PC0 (192.168.1.2) to Router 4 (192.168.1.1).
2. Ping from a PC in one network to a router in a different network:
    - For example, from PC0 (192.168.1.2) to Router 6 (192.168.3.1).
3. Ping between PCs across networks:
    - Ping from PC0 (192.168.1.2) in Network A to PC4 (192.168.3.2) in Network C.
    - If the ping is successful, the routing between networks is configured properly.

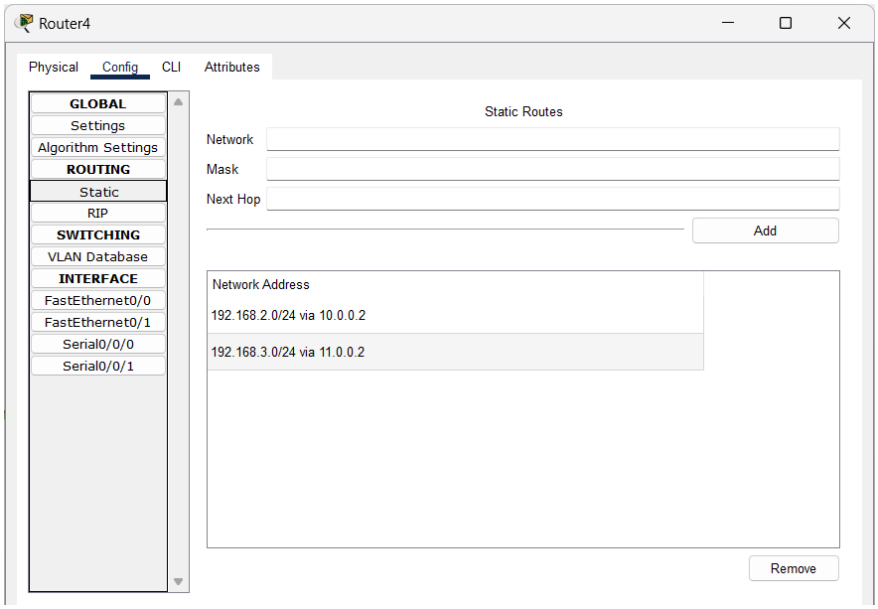
Department of Computer Science and Engineering
U18CSI5201_Computer Networks Laboratory

Reg: no: 22BCS081

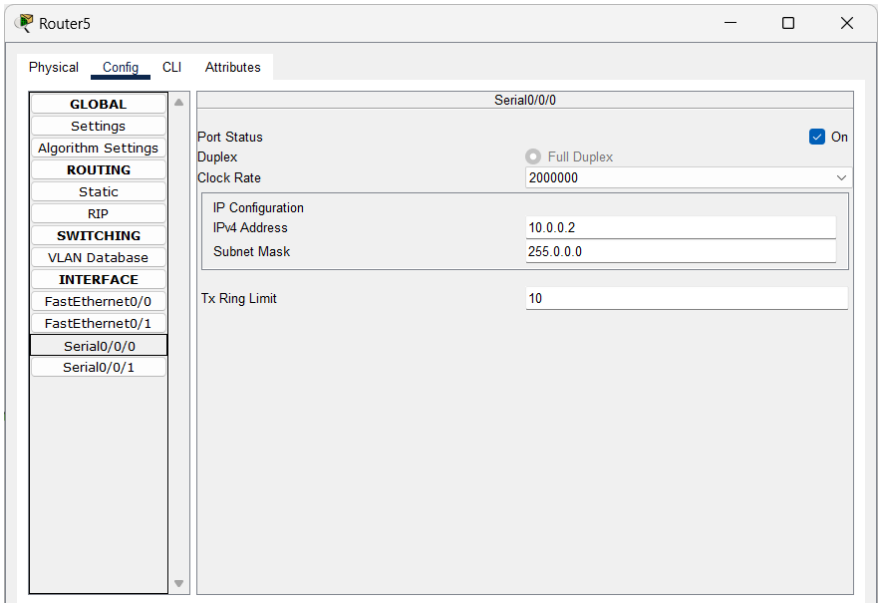
Output:

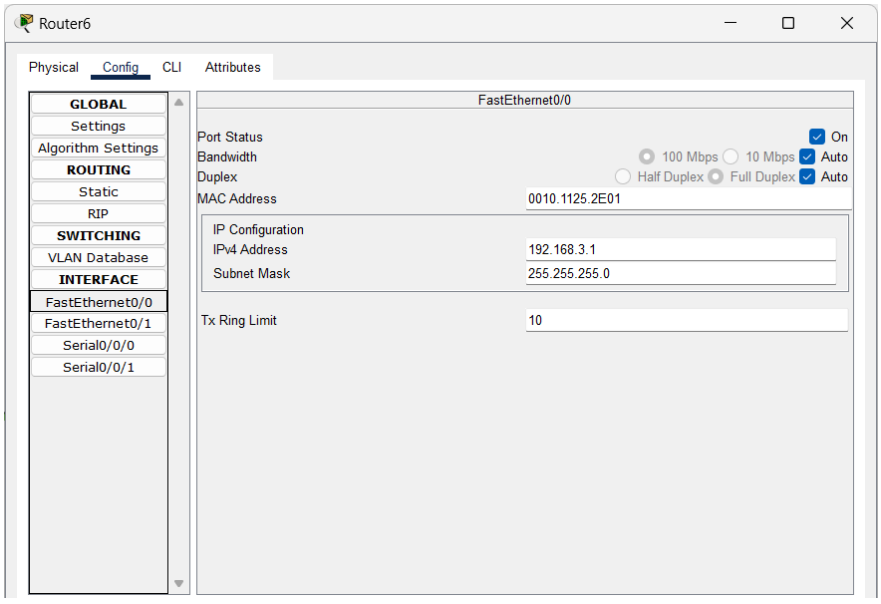






Reg: no: 22BCS081





Result:

All experiments have been successfully executed, with no errors or issues encountered. The expected results have been achieved, as demonstrated by the attached screenshots.

Reg: no: 22BCS081

Exp: no: 6

Performance analysis of TCP and UDP protocol.

Aim:

To analyse the performance of TCP and UDP protocols by comparing their data segments, frame formats, and key differences.

a. Draw the UDP frame format.

Procedure:

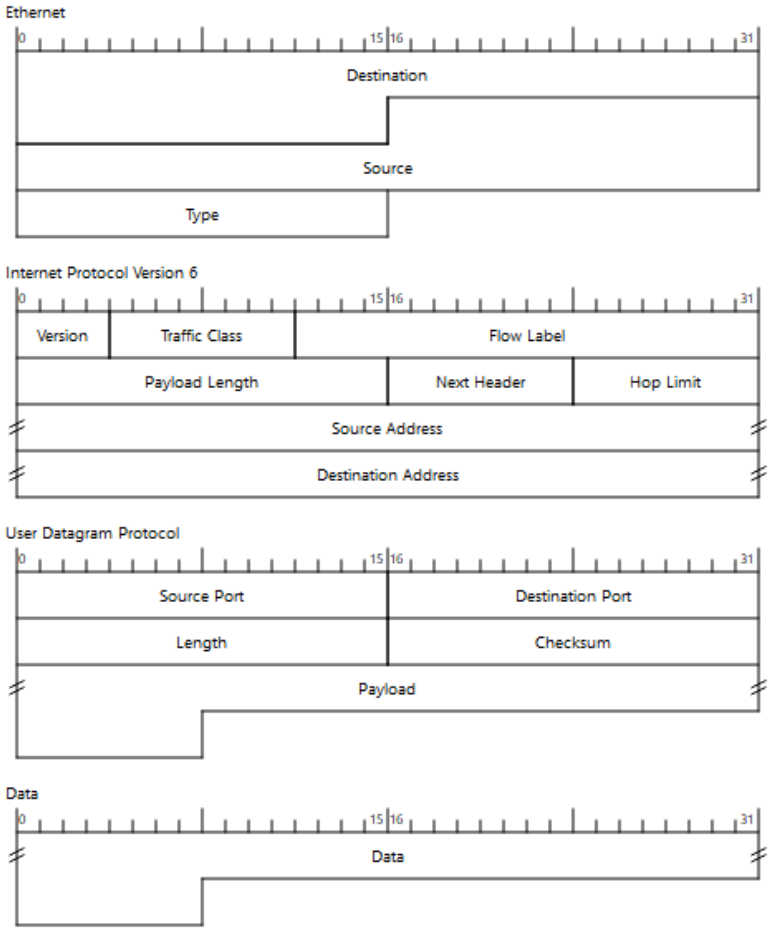
i. **Install and Open Wireshark**:
   a. Ensure Wireshark is installed on your system and launch it.
ii. **Start a Capture**:
   a. Select the appropriate network interface (e.g., Wi-Fi or Ethernet) where you expect the UDP traffic to be.
   b. Click the **Start** button to begin capturing network traffic.
iii. **Filter for UDP Packets**:
   a. In the **filter bar** at the top of the Wireshark window, enter the filter `udp` and press **Enter**. This filter will show only UDP packets.
iv. **Initiate a UDP Transmission**:
   a. Open a network application that uses UDP (e.g., a video streaming service or VoIP application) to generate UDP traffic.
v. **Capture and Stop**:
   a. Once the UDP traffic is visible in Wireshark, click the **Stop** button to halt the capture.
vi. **Select a UDP Packet**:
   a. From the captured packets list, select any UDP packet to analyze. Click on it to see its details.
vii. **Examine the UDP Frame Format**:
   a. In the **Packet Details** pane, expand the **User Datagram Protocol (UDP)** section to view the fields of the UDP frame:
      i. **Source Port**: Port from which the packet was sent.
      ii. **Destination Port**: Port to which the packet is being sent.
      iii. **Length**: The length of the UDP header and the encapsulated data.
      iv. **Checksum**: Used for error checking.
   b. You can also view the **Data Payload** carried by the UDP packet.
viii. **Screenshot**: Take a screenshot of the UDP frame format for future reference or reports.

Department of Computer Science and Engineering
U18CSI5201_Computer Networks Laboratory

Reg: no: 22BCS081

Output:

```
Frame 2525: 1307 bytes on wire (10456 bits), 1307 bytes captured (10456 bits) on interface \Device\NPF_{26351557-741B-4644-99B9-EA4747BF405A}, id 0
    Section number: 1
  > Interface id: 0 (\Device\NPF_{26351557-741B-4644-99B9-EA4747BF405A})
    Encapsulation type: Ethernet (1)
    Arrival Time: Oct 14, 2024 09:23:14.278233000 India Standard Time
    UTC Arrival Time: Oct 14, 2024 03:53:14.278233000 UTC
    Epoch Arrival Time: 1728877994.278233000
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.043023000 seconds]
    [Time delta from previous displayed frame: 0.043023000 seconds]
    [Time since reference or first frame: 23.061342000 seconds]
    Frame Number: 2525
    Frame Length: 1307 bytes (10456 bits)
    Capture Length: 1307 bytes (10456 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ipv6:udp:data]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
```



Nithin Prasath C - 22BCS081

b. List out the differences between TCP and UDP.

Feature	TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
Connection Type	Connection-oriented (Requires handshake)	Connectionless (No handshake required)
Reliability	Provides reliability through error checking, acknowledgments, and retransmissions.	No reliability, no retransmissions. Once sent, data is not guaranteed to arrive.
Flow Control	Uses flow control to manage data transmission rate	No flow control mechanisms
Congestion Control	Employs congestion control algorithms like TCP Reno, TCP Tahoe	No congestion control
Overhead	Higher due to error-checking, sequencing, and acknowledgment mechanisms	Lower overhead due to minimal header fields
Use Cases	File transfer (e.g., FTP), web browsing (e.g., HTTP), email (e.g., SMTP)	Real-time services like video streaming, VoIP, DNS queries, and online gaming
Speed	Slower due to connection setup and error-checking	Faster as no connection setup is required
Data Segmentation	Segments data into smaller packets and reassembles them at the receiver's end	No data segmentation, sends data in the form of datagrams
Error Detection	Provides error detection and correction	Error detection is limited (only checksum)
Header Size	20-60 bytes	8 bytes

c. Analyse the TCP and UDP data segments with the data parameters.

Procedure for TCP Segment Analysis:

i. **Start Wireshark and Capture Network Traffic**:
   a. Select your network interface and click the **Start** button to begin capturing packets.
ii. **Filter for TCP Traffic**:
   a. In the **filter bar**, type **tcp** and press **Enter** to display only TCP traffic.
iii. **Generate TCP Traffic**:
   a. Open a TCP-based application (e.g., a web browser, download a file via HTTP, or initiate a file transfer using FTP) to generate TCP traffic.
iv. **Capture and Stop**:
   a. Once you have some TCP packets captured, click **Stop**.
v. **Select a TCP Packet**:
   a. Select a TCP packet from the list of captured packets and click on it to see the detailed breakdown.
vi. **Examine the TCP Data Segment**:
   a. In the **Packet Details** pane, expand the **Transmission Control Protocol (TCP)** section to view the following fields:
      i. **Source Port & Destination Port**: Identifies the sending and receiving applications.
      ii. **Sequence Number**: Ensures data packets are received in order.
      iii. **Acknowledgment Number**: Used to acknowledge the successful receipt of data.
      iv. **Flags**: Shows TCP control flags (SYN, ACK, FIN, etc.).
      v. **Window Size**: Indicates how much data the receiver can accept.
      vi. **Checksum**: Ensures data integrity.
   b. **Data Payload**: You can also analyze the data segment being transmitted by the TCP packet (i.e., the actual data from the application layer).
vii. **Take Notes/Screenshot**: Take a screenshot of the segment analysis or record important data parameters.

Procedure for UDP Segment Analysis:

i. **Start Wireshark and Capture Network Traffic**:
   a. As before, select your network interface and click **Start**.
ii. **Filter for UDP Traffic**:
   a. In the **filter bar**, type **udp** and press **Enter** to focus on UDP traffic.
iii. **Generate UDP Traffic**:
   a. Open a UDP-based application (e.g., online gaming, VoIP service, or DNS queries).

iv. **Capture and Stop**:
   a. Once UDP packets are visible, click **Stop** to halt the capture.

v. **Select a UDP Packet**:
   a. Choose any UDP packet from the capture list and click on it to view its details.

vi. **Examine the UDP Data Segment**:
   a. In the **Packet Details** pane, expand the **User Datagram Protocol (UDP)** section to view:
      i. **Source Port & Destination Port**: The application ports involved in communication.
      ii. **Length**: Total length of the UDP packet, including the header and payload.
      iii. **Checksum**: Ensures data integrity for error detection.
   b. **Data Payload**: You can inspect the application data encapsulated within the UDP segment.

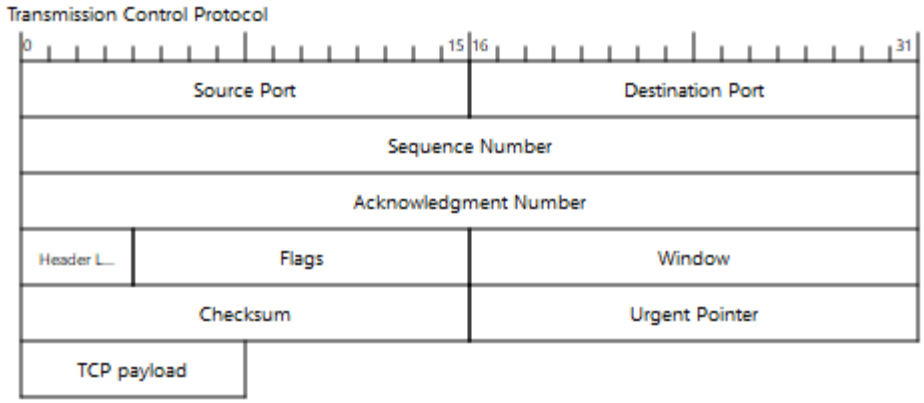
vii. **Take Notes/Screenshot**: Like before, capture a screenshot or take notes for documentation.

Output:

TCP:

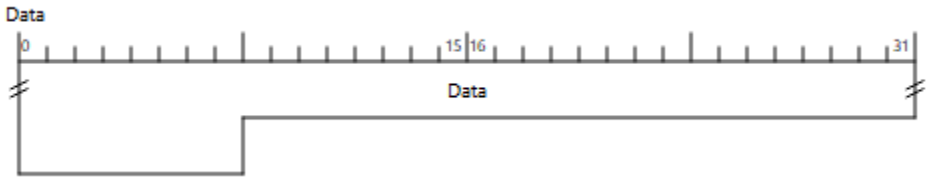
```
Transmission Control Protocol, Src Port: 60559, Dst Port: 443, Seq: 1, Ack: 1, Len: 1
    Source Port: 60559
    Destination Port: 443
    [Stream index: 38]
  > [Conversation completeness: Incomplete (12)]
    [TCP Segment Len: 1]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 2646827587
    [Next Sequence Number: 2    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1619227582
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window: 1026
    [Calculated window size: 1026]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x9057 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
    TCP payload (1 byte)
    TCP segment data (1 byte)
```

Reg: no: 22BCS081



UDP:

```
User Datagram Protocol, Src Port: 3702, Dst Port: 64100
    Source Port: 3702
    Destination Port: 64100
    Length: 1253
    Checksum: 0xf6e8 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 32]
  > [Timestamps]
    UDP payload (1245 bytes)
```



Result:

The performance of TCP and UDP protocols was successfully analysed by examining their frame formats, data segments, and identifying the key differences.

Reg: no: 22BCS081

Exp: no: 7

Demonstration of Client server-based TCP applications using socket programming

Aim:

To implement and demonstrate a client-server application using **TCP (Transmission Control Protocol)** for communication between two systems, showcasing the basics of socket programming and TCP's reliable, connection-oriented nature and to write the syntax, execute and place the screenshot for all the commands worked on.

a. **Write a client-server program to implement a TCP application.**

Description:

In this lab task, we'll create a simple Client-Server application using TCP. TCP is a connection-oriented protocol that provides reliable communication by establishing a connection before data transfer begins. TCP guarantees that data packets arrive in sequence and without errors, making it suitable for applications where data integrity and reliability are essential.

The client will establish a connection with the server, send a message, and the server will respond back with a confirmation message. This application demonstrates TCP socket programming fundamentals, connection establishment, data transfer, and termination of the connection.

Procedure:

1. **Set Up the Environment:**
   - Use Java to implement the client-server application. Ensure JDK is installed, and use a text editor or terminal to write and execute the code.
2. **Server Implementation:**
   - The server binds to a specific port (5000) using ServerSocket and waits for a client to connect.
   - Once a connection is established, it reads messages from the client using DataInputStream and prints them.
   - The server continues receiving messages until it gets the termination signal "Over".
3. **Client Implementation:**
   - The client connects to the server using its IP (127.0.0.1) and port (5000) via Socket.
   - It sends messages to the server using DataOutputStream, taking input from the console.
   - The client terminates the connection when "Over" is sent.

4. **Run the Server and Client:**
   - Start the Server.java program first to make it ready for incoming connections.
   - Then, run the Client.java program to establish the connection and exchange messages.

Code:

- Server Code (Server.java):

```
import java.net.*;
import java.io.*;
 public class Server
{
    private Socket          socket   = null;
    private ServerSocket    server   = null;
    private DataInputStream in       =  null;
    public Server(int port)
    {
        try{
            server = new ServerSocket(port);
            System.out.println("Server started");
            System.out.println("Waiting for a client ...");
            socket = server.accept();
            System.out.println("Client accepted");
            in = new DataInputStream(
                new BufferedInputStream(socket.getInputStream()));
            String line = "";
            while (!line.equals("Over"))
            {
                try{
                    line = in.readUTF();
                    System.out.println(line);

                }
                catch(IOException i)
                {
                    System.out.println(i);
                }
            }
            System.out.println("Closing connection");
            socket.close();
            in.close();
        }
        catch(IOException i)
        {
            System.out.println(i);
        }
    }
```

```
    public static void main(String args[])
    {
        Server server = new Server(5000);
    }
}
```

- Client Code (Client.java):

```
import java.io.*;
import java.net.*;
 public class Client {
    private Socket socket = null;
    private DataInputStream input = null;
    private DataOutputStream out = null;
    public Client(String address, int port)
    {
        try {
            socket = new Socket(address, port);
            System.out.println("Connected");
            input = new DataInputStream(System.in);
            out = new DataOutputStream(
                socket.getOutputStream());
        }
        catch (UnknownHostException u) {
            System.out.println(u);
            return;
        }
        catch (IOException i) {
            System.out.println(i);
            return;
        }
        String line = "";
        while (!line.equals("Over")) {
            try {
                line = input.readLine();
                out.writeUTF(line);
            } catch (IOException i) {
                System.out.println(i);
            }
        }
        try {
            input.close();
            out.close();
            socket.close();
        }  catch (IOException i) {
            System.out.println(i);
        }
    }
```

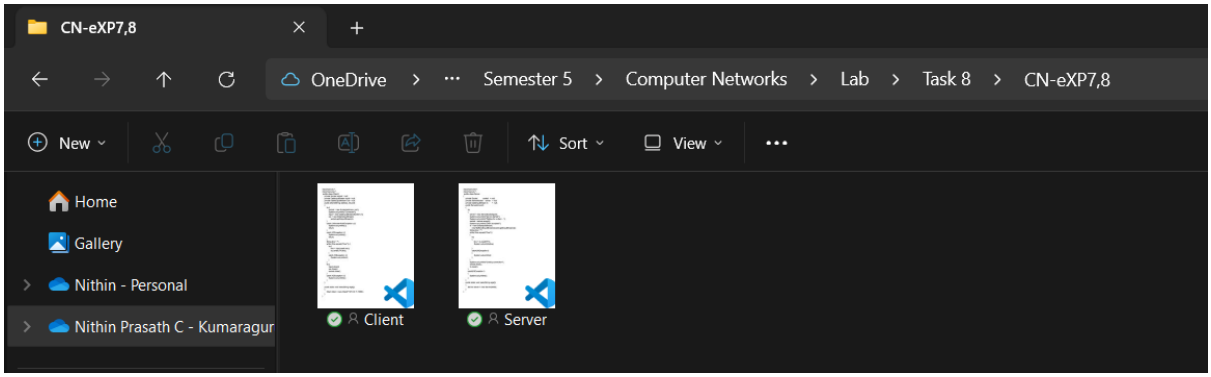
```
    public static void main(String args[])
    {
        Client client = new Client("127.0.0.1", 5000);
    }
}
```

Explanation of the Code

1. **Server Code:**
    - The server creates a ServerSocket on port 5000 to listen for incoming client connections.
    - Once a connection is established using accept(), it uses DataInputStream to receive messages from the client.
    - Messages are printed on the server console, and the connection is closed after receiving "Over".
2. **Client Code:**
    - The client creates a Socket to connect to the server at IP 127.0.0.1 and port 5000.
    - It uses DataOutputStream to send messages and System.in to take input from the user.
    - The client terminates the connection after sending "Over" and closes all resources.

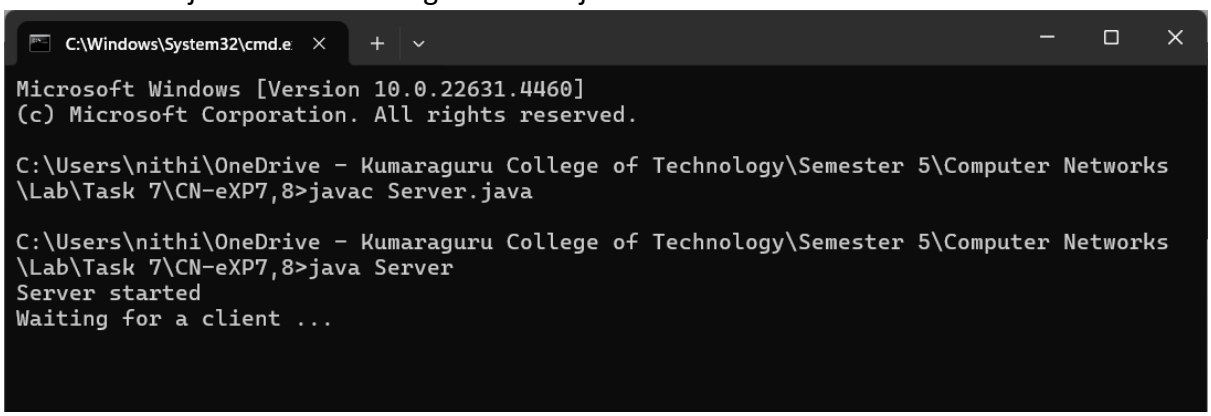
Output:

- Server and Client java files stored in a separate folder

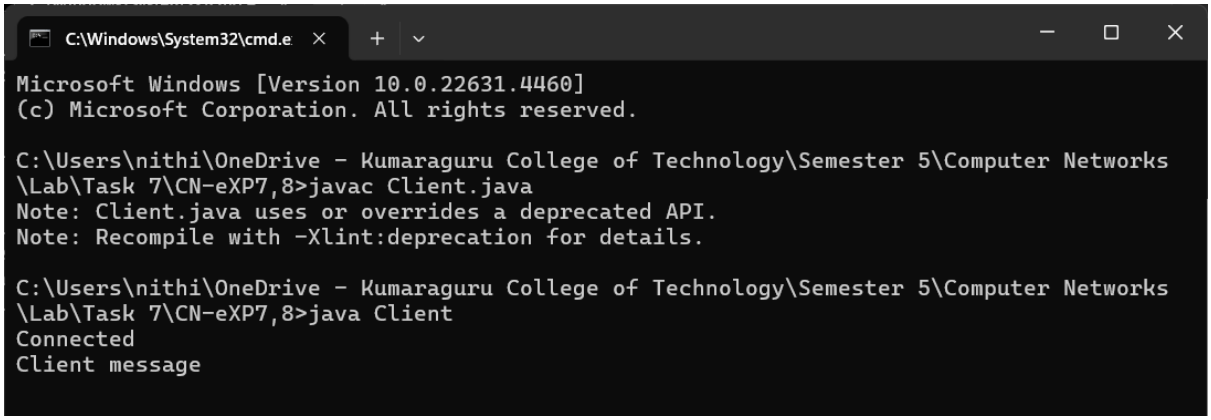


Reg: no: 22BCS081

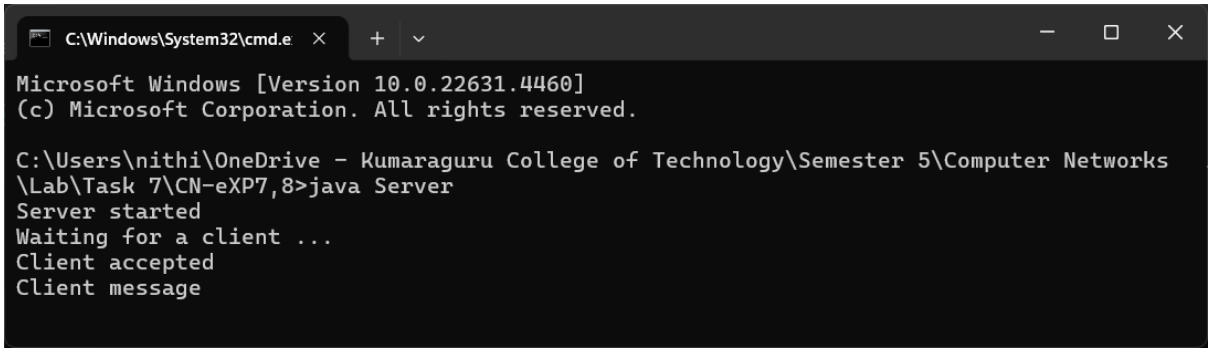
- Server.java before starting the Client.java file



- Running Client.java file



- Server.java after starting the Client.java file



Result:

Thus, Client server-based TCP applications was demonstrated using socket programming and wrote the syntax, executed and placed the screenshot for all the commands worked on.

Reg: no: 22BCS081

Exp: no: 8

Demonstration of Client server-based UDP applications using socket programming

Aim:

To implement and demonstrate a client-server application using **UDP (User Datagram Protocol)** for communication between two systems, showcasing the basics of socket programming and UDP's connectionless, unreliable nature and to write the syntax, execute and place the screenshot for all the commands worked on.

a. **Write a client-server program to implement a UDP application.**

Description:

In this lab task, we'll create a simple Client-Server application using UDP. UDP is a connectionless protocol that allows data to be sent without establishing a prior connection. This makes it faster but also less reliable than TCP, as it doesn't guarantee message delivery or order.

The client will send a message to the server, and the server will respond back with a confirmation message. This application will help understand the basics of UDP socket programming, data packet transmission, and handling message exchanges over a network.

Procedure:

1. **Set Up the Environment:**
    - Use Java for the client-server application. Ensure JDK is installed and use an editor like Eclipse or a terminal.
2. **Server Implementation:**
    - The server binds to a port (5000), listens for client connections, receives messages using DataInputStream, and prints them.
    - The connection remains active until the client sends "Over".
3. **Client Implementation:**
    - The client connects to the server using its IP (127.0.0.1) and port (5000).
    - It reads user input, sends it to the server using DataOutputStream, and terminates on "Over".
4. **Run the Programs:**
    - Compile and run the Server.java file first. Then, compile and run Client.java to establish the connection and exchange messages.

Reg: no: 22BCS081

Code:

- Server Code (Server.java):

```
import java.net.*;
import java.io.*;
 public class Server
{
    private Socket          socket   = null;
    private ServerSocket    server   = null;
    private DataInputStream in       =  null;
    public Server(int port)
    {
        try{
            server = new ServerSocket(port);
            System.out.println("Server started");
            System.out.println("Waiting for a client ...");
            socket = server.accept();
            System.out.println("Client accepted");
            in = new DataInputStream(
                new BufferedInputStream(socket.getInputStream()));
            String line = "";
            while (!line.equals("Over"))
            {
                try{
                    line = in.readUTF();
                    System.out.println(line);

                }
                catch(IOException i)
                {
                    System.out.println(i);
                }
            }
            System.out.println("Closing connection");
            socket.close();
            in.close();
        }
        catch(IOException i)
        {
            System.out.println(i);
        }
    }
    public static void main(String args[])
    {
        Server server = new Server(5000);
    }
}
```

- Client Code (Client.java):

```
import java.io.*;
import java.net.*;
 public class Client {
    private Socket socket = null;
    private DataInputStream input = null;
    private DataOutputStream out = null;
    public Client(String address, int port)
    {
        try {
            socket = new Socket(address, port);
            System.out.println("Connected");
            input = new DataInputStream(System.in);
            out = new DataOutputStream(
                socket.getOutputStream());
        }
        catch (UnknownHostException u) {
            System.out.println(u);
            return;
        }
        catch (IOException i) {
            System.out.println(i);
            return;
        }
        String line = "";
        while (!line.equals("Over")) {
            try {
                line = input.readLine();
                out.writeUTF(line);
            } catch (IOException i) {
                System.out.println(i);
            }
        }
        try {
            input.close();
            out.close();
            socket.close();
        }  catch (IOException i) {
            System.out.println(i);
        }
    }

    public static void main(String args[])
    {
        Client client = new Client("127.0.0.1", 5000);
    }
}
```

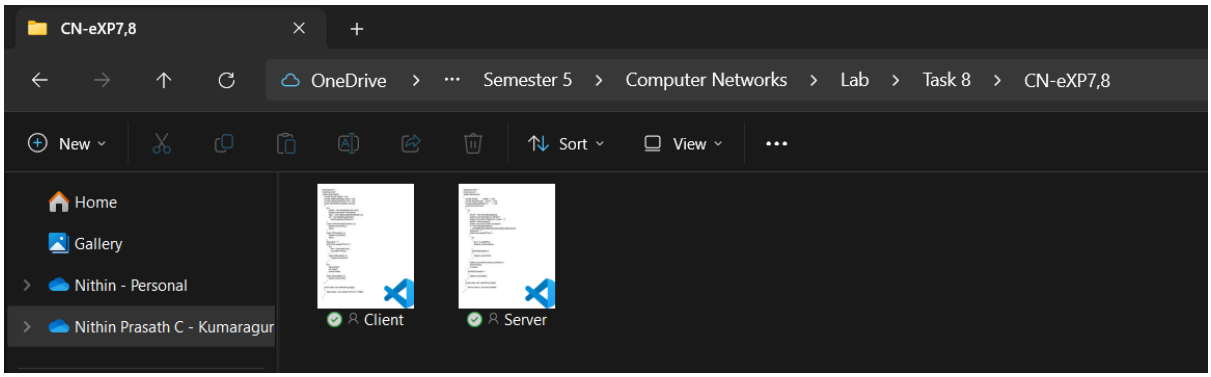
Reg: no: 22BCS081

Explanation of the Code

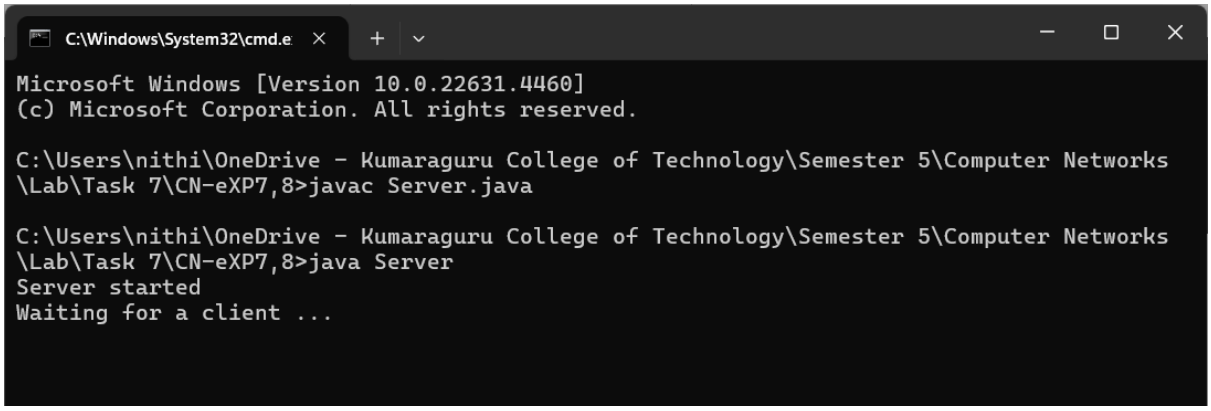
1. **Server Code:**
   - Creates a ServerSocket on port 5000, accepts client connections, and uses readUTF() to receive messages.
   - Prints client messages and closes the connection when "Over" is received.
2. **Client Code:**
   - Connects to the server at 127.0.0.1:5000 using a Socket.
   - Reads user input via readLine() and sends it using writeUTF().
   - Closes the connection after sending "Over".

Output:

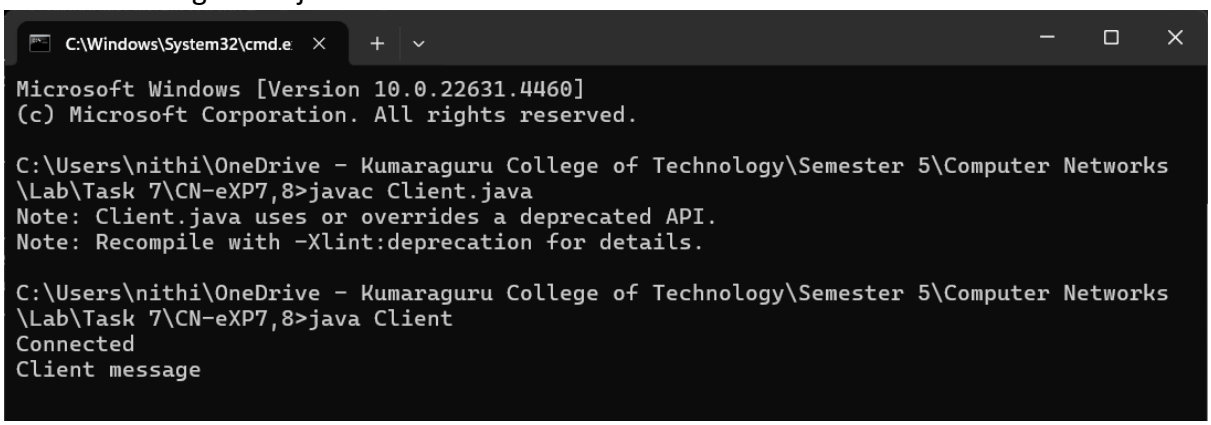
- Server and Client java files stored in a separate folder



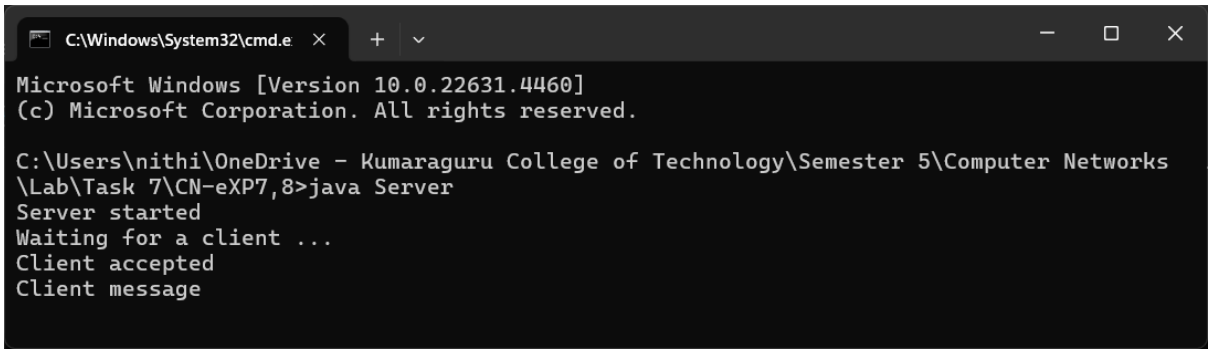
- Server.java before starting the Client.java file



- Running Client.java file



- Server.java after starting the Client.java file



Result:

Thus, Client server-based UDP applications was demonstrated using socket programming and wrote the syntax, executed and placed the screenshot for all the commands worked on.