COMPUTER NETWORKS PROGRAMS

1. Study different types of Network cables (Copper and Fiber) and prepare cables

(Straight and Cross) to connect Two or more jacks. Use LAN tester to connect the cables.

- Install and configure Network Devices: HUB, Switch and Routers. Consider both manageable and non-manageable switches. Do the logical configuration of the system. Set the bandwidth of different ports.

- Install and Configure Wired and Wireless NIC and transfer files between systems in Wired LAN and Wireless LAN. Consider both adhoc and infrastructure mode of operation.

2. Work with the commands Ping, Tracert, Ipconfig, pathping Hostname, Nbtstat, netdiag, and Nslookup

3. Find all the IP addresses on your network. Unicast, Multicast, and Broadcast on your

network.

4. Use Packet tracer software to build network topology and configure using Distance

vector routing protocol.

5. Use Packet tracer software to build network topology and configure using Link State

routing protocol.

6. Using JAVA RMI Write a program to implement Basic Calculator.

7. Implement a Chatting application using JAVA TCP and UDP sockets.

8. Hello command is used to know whether the machine at the other end is working or

not. Echo command is used to measure the round-rip time to the neighbor. Implement Hello and Echo commands using JAVA.

9. Using Wireshark perform the following operations:

- Inspect HTTP Tra

- Inspect HTTP Traffic from a Given IP Address,

- Inspect HTTP Traffic to a Given IP Address,

- Reject Packets to Given IP Address,

- Monitor Apache and MySQL Network Traffic

1. Study different types of network cables (copper and fiber)and prepare cables (straight and cross) to connect two or more systems. use crimping tool to connect jacks. use LAN tester to connect the cables

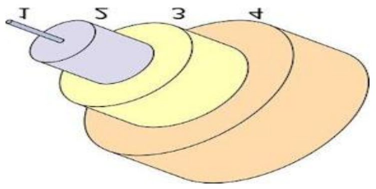
The primary types of cables used for connecting systems in a network are **copper cables** and **fiber optic cables**. Each type of cable has distinct characteristics, and there are different wiring standards (such as **straight-through** and **crossover**) for connecting devices.

1. Types of Network Cables

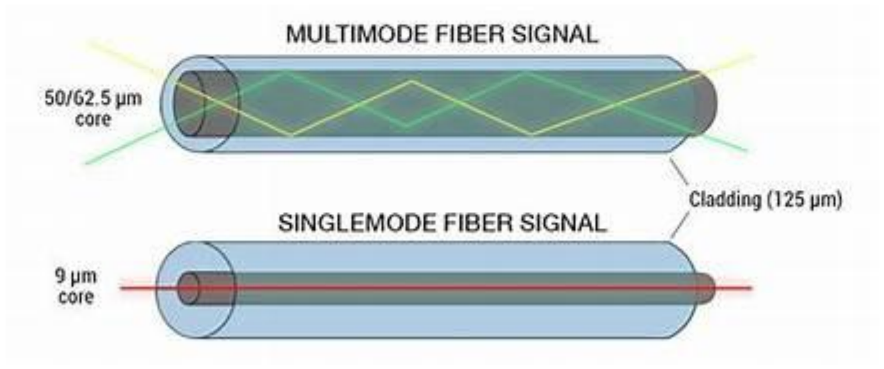
- **Copper Cables** (e.g., Ethernet cables)
  - **Unshielded Twisted Pair (UTP)**: This is the most common type of copper cable. It contains multiple pairs of twisted wires that help reduce electromagnetic interference (EMI).

  - **Shielded Twisted Pair (STP)**: Similar to UTP, but with an additional shielding to protect against EMI, often used in environments with high interference.

  - **Categories of Copper Cables**: Ethernet cables come in different categories, such as:
    - **Cat5e**: Enhanced version of Cat5, supports up to 1000 Mbps (Gigabit Ethernet).
    - **Cat6**: Supports 10 Gbps over shorter distances.
    - **Cat6a**: Enhanced Cat6, supports 10 Gbps up to 100 meters.
    - **Cat7 and Cat8**: Used for higher-speed networking and data centers.
- **Fiber Optic Cables**
  - **Single-mode fiber (SMF)**: Used for long-distance transmission, typically over kilometers.



- **Multi-mode fiber (MMF)**: Used for shorter distances within buildings and campuses.
- **Core and Cladding**: The core is the center part through which light travels, while the cladding reflects the light to keep it inside the core.



2. Types of Network Cable Configurations

- **Straight-through Cable**: This is the most commonly used cable for connecting different types of devices, like a computer to a switch or a router. Both ends of the cable are wired using the same pinout pattern (either T568A or T568B standard).
  - **T568A Pinout**:
    1. White/Green
    2. Green
    3. White/Orange
    4. Blue
    5. White/Blue
    6. Orange
    7. White/Brown
    8. Brown
  - **T568B Pinout** (alternative standard, commonly used in the US):
    1. White/Orange
    2. Orange
    3. White/Green
    4. Blue

5. White/Blue
6. Green
7. White/Brown
8. Brown

- **Crossover Cable**: Used for connecting similar devices, such as two computers or two switches, directly without needing a router or hub. It has different wiring on both ends:
  - One end is wired as T568A, and the other end is wired as T568B.

3. Using Crimping Tool

A crimping tool is essential for attaching connectors (RJ45) to the cable. Here's how to use it:

- **Steps to Crimp a Cable**:
  1. **Prepare the Cable**: Use a wire stripper to strip off about 1-2 inches of the outer insulation from the cable.
  2. **Untwist the Wires**: Separate the color-coded pairs of wires and untwist them.
  3. **Arrange the Wires**: Arrange the wires in the desired order (T568A or T568B). Ensure the wires are straight and in the correct sequence.
  4. **Trim the Wires**: Trim the wires so that they are all even and will fit properly into the RJ45 connector.
  5. **Insert Wires into the RJ45 Connector**: Insert the wires into the connector so that each wire goes to the end of the connector.
  6. **Crimp the Connector**: Place the connector in the crimping tool and squeeze the handle tightly to secure the wires into the connector.
  7. **Check the Connection**: Pull gently on the connector to make sure the wires are securely crimped.

4. Testing Cables Using a LAN Tester

A **LAN tester** is a device used to test the connectivity of network cables (Ethernet cables). Here's how to use a LAN tester:

- **Steps to Use a LAN Tester**:
  1. **Insert the Cable**: Plug one end of the cable into the tester's main unit and the other end into the remote unit.
  2. **Power On**: Turn on the tester.
  3. **Test the Cable**: The tester will check the continuity of each wire in the cable and display the results. It will show if there are any breaks in the wire or miswiring (e.g., if the wrong pinout standard is used).

4. **Interpret the Results**: The tester will show which pins are correctly connected. If any of the lights are out or incorrect, you may need to redo the cable crimping.

5. Practical Example: Creating and Testing a Cable

- **Straight-through Cable**:
  1. Prepare the cable by stripping the outer insulation.
  2. Untwist the wire pairs and arrange them in the T568A or T568B order.
  3. Insert the wires into the RJ45 connector and crimp it using the crimping tool.
  4. Use the LAN tester to check the cable for proper connectivity and wiring.
- **Crossover Cable**:
  1. Prepare the cable similarly, but use T568A on one end and T568B on the other.
  2. Crimp the cable and test it with the LAN tester.

Installing and configuring network devices like Hubs, Switches, and Routers is essential to creating a functional network. Additionally, configuring Network Interface Cards (NICs) for wired and wireless connections is crucial for connecting systems in both wired LANs and wireless LANs. Below is a guide to install and configure these devices and to set up file transfer between systems in Wired LAN and Wireless LAN, considering Ad-hoc and Infrastructure modes of operation.

1. Installing and Configuring Network Devices

a. Hub

A Hub is a basic networking device that connects multiple devices in a LAN. It operates at the Physical Layer and simply broadcasts data to all devices connected to it.

Installation:

Plug the hub into a power source.

Connect devices to the hub using Ethernet cables.

Ensure the devices are powered on and connected properly.

Configuration:

Non-manageable Hub: A hub typically requires no configuration; it is a simple plug-and-play device.

Manageable Hub: Some hubs may allow basic configuration (though this is more common with switches and routers). Configuration could involve settings like port mirroring or monitoring, which would be done via a web interface or command line.

b. Switch

A Switch operates at the Data Link Layer and can intelligently forward data only to the device that needs it, which improves network efficiency.

Installation:

Plug the switch into a power source.

Connect Ethernet cables from each device to the switch.

Ensure all devices are powered on.

Configuration:

Non-manageable Switch: Like the hub, a non-manageable switch is also plug-and-play and requires no configuration.

Manageable Switch: Manageable switches provide more advanced features, such as VLANs, port security, and QoS. To configure it:

Access the switch interface: Use a web interface or command line via Telnet or SSH.

Configure VLANs: You can segment network traffic by creating Virtual Local Area Networks (VLANs).

Set up port security: This can prevent unauthorized devices from connecting.

Enable QoS: Set up Quality of Service to prioritize critical traffic (e.g., VoIP).

Save configuration: Once done, save the changes to prevent loss after a reboot.

c. Router

A Router is a device that connects different networks (usually a local network to the internet) and directs data between them. Routers operate at the Network Layer.

Installation:

Connect the router to a power source.

Connect one Ethernet cable from the router’s WAN port to your modem (for internet connectivity).

Connect the LAN ports of the router to devices or a switch for internal network connectivity.

Configuration:

Access the router’s configuration page: This is typically done via a browser. Type the router's default IP address (e.g., 192.168.1.1) in the address bar.

Login: Use the default credentials (often found on the router’s label).

Set up the router’s internet connection: This may involve entering details provided by your ISP (e.g., PPPoE, dynamic IP).

Configure the wireless settings (for routers with Wi-Fi):

Enable Wi-Fi and set the SSID (network name).

Choose the security type (WPA2 is recommended).

Set a strong password.

Configure routing settings: Set up static or dynamic routes if necessary, depending on your network.

Enable DHCP: The router can automatically assign IP addresses to devices on the network (typically enabled by default).

Save configuration.

2. Installing and Configuring Wired and Wireless NICs (Network Interface Cards)

a. Wired NIC Installation and Configuration

Installation:

Install the NIC: Insert the wired NIC into the computer's PCIe slot (for desktop) or connect the Ethernet port (for laptop).

The operating system will usually automatically detect and install the necessary drivers. If not, use the driver CD or download the driver from the manufacturer's website.

Configuration:

Static IP: In the Network Settings on Windows or Linux, set a static IP address if required. For example, in Windows:

Go to Control Panel > Network and Sharing Center > Change adapter settings.

Right-click the network connection, select Properties, then click Internet Protocol Version 4 (TCP/IPv4) and set the IP address, subnet mask, and gateway.

DHCP: Ensure the NIC is set to automatically obtain an IP if your router assigns addresses dynamically.

b. Wireless NIC Installation and Configuration

Installation:

Install the Wireless NIC: Insert the wireless card into the appropriate slot or use a USB Wi-Fi adapter.

Install the drivers: The operating system will usually detect the card and install the necessary drivers automatically. If not, use the provided driver disk or download them from the manufacturer's website.

Configuration:

Wireless Network Setup: Connect to an available wireless network from the network icon in the system tray (Windows) or Wi-Fi settings (Mac/Linux).

Enter SSID and Password: Choose your network and enter the Wi-Fi password.

Check Connection: Once connected, verify connectivity by opening a browser or pinging the gateway.

3. File Transfer Between Systems

a. Wired LAN File Transfer

File Sharing on Windows:

Enable file sharing: Right-click the folder you want to share, select Properties, then click the Sharing tab and select Share.

Access the Shared Folder: On another computer, open File Explorer, type \\<IP address> in the address bar to access the shared folder.

File Sharing on Linux:

Use Samba or NFS to share files between systems.

Share a folder using samba configuration or Samba GUI.

b. Wireless LAN File Transfer

Ad-hoc Mode (Direct Wireless Connection):

In Ad-hoc mode, devices connect directly to each other, without a router or access point.

Windows: Create an ad-hoc network by going to Control Panel > Network and Sharing Center > Set up a new connection or network and select Set up an ad-hoc network. Devices can connect to each other by selecting the ad-hoc network name and entering the password.

Linux: Use tools like nmcli or NetworkManager to create an ad-hoc network.

Infrastructure Mode (Via Router or Access Point):

In Infrastructure mode, devices connect to a central router or access point, which then routes traffic between them.

Windows: Set up a connection to the wireless network and use file sharing as explained above.

Linux: Similar to the wired LAN setup, connect to the network and share files via Samba or NFS.

4. Testing and Verifying Connectivity

Ping Test: After setting up devices, use the ping command to check connectivity between devices in the network.

Example: Open a command prompt and type ping <IP address> to verify if the device is reachable.

File Transfer: Try transferring files to verify the connection. For wireless setups, ensure both devices are on the same network (either Ad-hoc or Infrastructure mode).

2. Work with the commands Ping, Tracert, Ipconfig, pathping, telnet, ftp, getmac, ARP, Hostname, Nbtstat, netdiag, and Nslookup

1. Ping

- **Purpose**: Tests network connectivity by sending ICMP Echo Request packets to a target (IP address or domain name) and waits for an Echo Reply.
- **Syntax**: ping [target]

Example:

bash

```
ping google.com

ping 192.168.1.1
```

Output: Displays the round-trip time for packets sent and received. If the target is unreachable, it will show a timeout.

---

2. Traceroute

- **Purpose**: Traces the route taken by packets from your system to the target (IP address or domain), showing each hop along the way.
- **Syntax**: Traceroute [target]

Example:

bash

```
Traceroute google.com

Traceroute 8.8.8.8
```

Output: Lists each hop (router) and the time it took to reach it.

---

3. Ipconfig

- **Purpose**: Displays the network configuration of your computer (IP addresses, subnet mask, gateway, etc.).
- **Syntax**: ipconfig (basic), ipconfig /all (detailed)

Example:

bash

```
ipconfig

ipconfig /all
```

Output: Displays network interfaces, IP addresses, DNS servers, etc.

---

4. Pathping

- **Purpose**: Combines the features of ping and tracert by providing the route taken to a target and the latency at each hop. It also provides packet loss statistics.
- **Syntax**: pathping [target]

Example:

bash

```
pathping google.com

pathping 192.168.2.1
```

Output: Similar to traceroute, but with packet loss and latency statistics for each hop.

---

5. Telnet

- **Purpose**: Connects to a remote system or device via the Telnet protocol, typically used for remote terminal access. It is often used for troubleshooting port connectivity.
- **Syntax**: telnet [host] [port]

Example:

bash

```
telnet 192.168.2.1 80

telnet google.com 443
```

Output: If the connection is successful, a blank screen appears or a service banner is displayed (e.g., HTTP server response). If the connection fails, you'll see an error message.

---

6. FTP (File Transfer Protocol)

- **Purpose**: Transfers files between a client and server over a network.
- **Syntax**: ftp [host]

Example:

bash

```
ftp ftp.example.com

ftp 192.168.2.10
```

Output: Once connected, you can use commands like get to download files, put to upload files, and ls to list files.

---

7. Getmac

- **Purpose**: Displays the MAC (Media Access Control) address of network interfaces on your system.
- **Syntax**: getmac

Example:

```bash
getmac
```

Output: Lists the MAC addresses of network interfaces on your system.

---

8. ARP (Address Resolution Protocol)

- **Purpose**: Displays or modifies the ARP cache, which maps IP addresses to MAC addresses.
- **Syntax**: arp [command] [arguments]

Example:

```bash
arp -a
arp -d 192.168.2.1
```

Output: Displays the ARP table or deletes an entry from the table.

---

9. Hostname

- **Purpose**: Displays or sets the hostname of the computer.
- **Syntax**: hostname

Example:

```bash
hostname
```

Output: Displays the current hostname of the machine.

---

10. Nbtstat

- **Purpose**: Displays NetBIOS over TCP/IP statistics and the NetBIOS name table.
- **Syntax**: nbtstat [command]

Example:

bash

```
nbtstat -n
nbtstat -A 192.168.2.1
```

Output: Displays NetBIOS information for local machine or remote system.

---

11. Netdiag

- **Purpose**: A diagnostic tool in Windows that provides network diagnostics, especially for detecting and troubleshooting network problems.
- **Syntax**: netdiag

Example:

bash

```
netdiag
```

Output: Displays detailed information on network connections, including errors or issues found.

12. Nslookup

- **Purpose**: Queries DNS servers to obtain domain name or IP address information.
- **Syntax**: nslookup [domain]

Example:

bash

```
nslookup google.com
nslookup 8.8.8.8
```

Output: Displays the DNS information for the specified domain or IP.

3.Find all the IP addresses on your network. Unicast, Multicast, and Broadcast on your network

Understanding the Types of IP Addresses

- **Unicast:** A one-to-one communication. This is the most common type of IP address, used for communication between a specific device and another specific device. Examples are your computer talking to a web server.
- **Multicast:** One-to-many communication, where data is sent to a specific group of devices that have joined a particular multicast group. Examples include video streaming, group messaging, and some network discovery protocols.
- **Broadcast:** One-to-all communication within a specific network. Data is sent to all devices on the local network segment. Examples include DHCP requests, ARP broadcasts.

How to Find IP Addresses on Your Network (Using Your Computer)

The methods vary slightly depending on your operating system:

1. Unicast IP Addresses (Your Devices on the Network)

- **Windows:**
    1. **Using ipconfig:**
        - Open Command Prompt (search for "cmd").
        - Type ipconfig /all and press Enter.
        - Look for the following:
            - **"Ethernet adapter Ethernet"** (if wired connection) or **"Wireless LAN adapter Wi-Fi"** (if wireless connection).
            - **"IPv4 Address"**: This is your computer's unicast IP address.
            - **"Subnet Mask"**: This helps define your local network.
            - **"Default Gateway"**: This is the IP address of your router.
    2. **Using Network Settings GUI:**
        - Open Settings (Win + I).
        - Go to "Network & Internet".
        - Click on "Ethernet" or "Wi-Fi", depending on your connection.
        - Click on "Properties".
        - Your IP address information will be displayed.
- **macOS:**
    1. **Using ifconfig (Terminal):**

        - Open Terminal (search for "Terminal").
        - Type ifconfig and press Enter.
        - Look for en0 (for wired Ethernet) or en1 (for Wi-Fi, may be different).
        - The inet line will show your computer's unicast IPv4 address (e.g., inet 192.168.1.10). The netmask is your subnet mask.
        - The router address can often be found in route get default | grep gateway
    2. **Using Network Settings GUI:**
        - Click the Apple menu and select "System Settings" (or "System Preferences").
        - Click "Network".
        - Select your active network interface (Ethernet or Wi-Fi).
        - The IP address and subnet mask will be displayed.
- **Linux:**
    1. **Using ifconfig or ip addr (Terminal):**
        - Open a terminal.
        - Type ifconfig (older systems) or ip addr (newer systems) and press Enter.
        - Look for your active interface (often eth0, wlan0, or enp0s3, etc.).
        - With ifconfig, the inet addr line will show the IPv4 address. With ip addr, look for the inet line under your interface.
        - The router address can often be found in ip route show default | grep gateway
    2. **Using GUI**
        - Open your system's network manager (typically through settings) and check the connection details for your chosen interface.

2. Finding Other Devices' Unicast IP Addresses on Your Local Network

- **Using arp:**
    - The Address Resolution Protocol (arp) maps IP addresses to physical (MAC) addresses on a local network. You can use this to see what other devices your computer is aware of.
    - **Windows (Command Prompt):** arp -a
    - **macOS or Linux (Terminal):** arp -a

    - The output will show the IP addresses (and corresponding MAC addresses) of other devices your computer has recently communicated with.
    - **Note:** arp only works with devices on your local broadcast domain.
- **Using a Network Scanner (Nmap):**
    - A network scanner can actively scan your network and find devices. nmap is a powerful and popular tool.
    - **Install nmap (using your OS's package manager or from the website):**
        - **Linux (Debian/Ubuntu):** sudo apt install nmap
        - **macOS:** brew install nmap (if you have Homebrew) or install from the Nmap website.
        - **Windows:** Download from the Nmap website.
    - **Scan your network (Terminal):**
        - nmap -sn <your_network_address>/<cidr_prefix>
        - Example, if you subnet is 192.168.2.0/24, use: nmap -sn 192.168.2.0/24. This will identify all devices on your subnet and output their IP addresses.

3. Multicast IP Addresses

- Multicast addresses are in the range of 224.0.0.0 to 239.255.255.255.
- **Specific Multicast Addresses:**
    - 224.0.0.1: All systems on the subnet.
    - 224.0.0.2: All routers on the subnet.
    - There are other specific multicast addresses related to different protocols.
- **How to Find Multicast Activity:**
    - You can use tools like tcpdump or Wireshark to capture network traffic, and then analyze to look for multicast packets.
    - **Example (Linux/macOS, in Terminal):** sudo tcpdump -i <interface> multicast
        - Replace <interface> with your network interface (eth0, wlan0, etc.)
- **Note:** You likely won't see a list of "multicast IP addresses" in the way you see unicast IPs. Instead, you'd see *traffic* using multicast addresses. You typically need specialized tools to find devices using multicast protocols, and also, be aware of multicast traffic on your network.

4. Broadcast IP Addresses

- The broadcast address is usually the last address in your network segment (determined by your subnet mask).
- **How to Find Your Broadcast Address:**
    1. Find your unicast IP address and subnet mask (using the methods above).
    2. Convert the subnet mask to binary form.
    3. Invert the binary form of the subnet mask and create the broadcast address using bitwise OR operation of your unicast address with the inverted binary form of subnet mask.
- **Example:**
    - If your IP address is 192.168.2.10 and your subnet mask is 255.255.255.0:
        - Subnet mask in binary: 11111111.11111111.11111111.00000000
        - Inverted subnet mask in binary: 00000000.00000000.00000000.11111111
        - Broadcast address in binary: 11000000.10101000.00000001.11111111
        - Broadcast address in decimal : 192.168.2.255

4. Use Packet tracer software to build network topology and configure using Distance vector routing protocol.

To build a network topology and configure it using a Distance Vector Routing Protocol (such as RIP) in Cisco Packet Tracer, follow these steps:

Step 1: Open Packet Tracer

1. Launch Cisco Packet Tracer.
2. Start a new project.

Step 2: Create the Network Topology

1. **Drag and Drop Devices**:
    - Add **routers** (e.g., Router-0, Router-1, and Router-2) to the workspace.
    - Add **PCs** (e.g., PC0, PC1, PC2) to the workspace for testing connectivity.
    - Add **Switches** (optional) if connecting multiple devices to one router interface.
2. **Connect Devices**:
    - Use the **copper straight-through cable** to connect PCs to switches or routers.
    - Use the **crossover cable** to connect router-to-router links.

Step 3: Configure IP Addresses

1. **Assign IP Addresses**:
    - Click on each router and go to the **CLI tab**.
    - Assign IP addresses to the interfaces (use a unique subnet for each link).

    Example Configuration for Router-0:

    plaintext
    CopyEdit
    ```
    Router> enable
    Router# configure terminal
    Router(config)# interface gigabitEthernet 0/0
    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 gigabitEthernet 0/1
    Router(config-if)# ip address 192.168.2.1 255.255.255.0
    Router(config-if)# no shutdown
    Router(config-if)# exit
    ```

2. **Assign IPs to PCs**:
    - Click on each PC, go to the **Desktop tab**, and configure the IP address and subnet mask. Use the gateway as the router's IP address.

    Example for PC0:

    - **IP Address**: 192.168.1.2
    - **Subnet Mask**: 255.255.255.0
    - **Default Gateway**: 192.168.1.1

Step 4: Enable RIP (Routing Information Protocol)

1. **Enable RIP on Routers**: Configure RIP on all routers to share routing information automatically.

    Example Configuration for Router-0:

    plaintext
    CopyEdit
    ```
    Router> enable
    Router# configure terminal
    Router(config)# router rip
    Router(config-router)# version 2
    Router(config-router)# network 192.168.1.0
    Router(config-router)# network 192.168.2.0
    Router(config-router)# exit
    ```

    Repeat for Router-1 and Router-2, adjusting the networks to match the topology.

2. Verify RIP Configuration: Use the following command on each router to check the RIP routing table:

    plaintext
    CopyEdit
    ```
    Router# show ip route
    ```

Step 5: Test Connectivity

1. **Ping Between Devices**:
    - Open the **Command Prompt** on any PC.
    - Ping another PC to ensure connectivity:

        plaintext
        CopyEdit
        ```
        ping 192.168.x.x
        ```

  - If pings succeed, the Distance Vector Routing protocol (RIP) is functioning correctly.

Step 6: Save Configuration

1. Save the running configuration to the startup configuration on each router:

   plaintext
   CopyEdit
   ```
   Router# copy running-config startup-config
   ```

5. Use Packet tracer software to build network topology and configure using Link State routing protocol.

Objective

To create a network topology and configure routers using OSPF (Open Shortest Path First) protocol in Cisco Packet Tracer.

Step 1: Launch Packet Tracer

- Open the Cisco Packet Tracer application.

Step 2: Add Network Devices

1. Drag and drop the following devices into the workspace:
    - **3 Routers** (e.g., 2911 series).
    - **2 Switches** (optional for connecting PCs).
    - **4 PCs** (2 for testing on each side of the network).

Step 3: Connect the Devices

- Use the appropriate cables:
    - **Straight-through cables** for connecting PCs to switches and switches to routers.
    - **Serial or crossover cables** for connecting routers to each other.

Example Topology Connections:

- **Router 1 (R1)** connects to **Router 2 (R2)** using a serial link.
- **Router 2 (R2)** connects to **Router 3 (R3)** using a serial link.
- **PC1 and PC2** connect to **Switch1**, which connects to **Router 1 (R1)**.
- **PC3 and PC4** connect to **Switch2**, which connects to **Router 3 (R3)**.

Step 4: Assign IP Addresses

Assign IP addresses to router interfaces, switches, and PCs.

Example IP Scheme:

Device	Interface	IP Address	Subnet Mask
Router 1	Gig0/0	192.168.1.1	255.255.255.0

Device	**Interface**	**IP Address**	**Subnet Mask**
Router 1	Serial0/0/0	10.0.0.1	255.255.255.252
Router 2	Serial0/0/0	10.0.0.2	255.255.255.252
Router 2	Serial0/0/1	10.0.1.1	255.255.255.252
Router 3	Serial0/0/1	10.0.1.2	255.255.255.252
Router 3	Gig0/0	192.168.3.1	255.255.255.0

- Configure PCs with the IP address and gateway corresponding to the router.

Step 5: Configure Routers

Use the following commands to configure OSPF on each router:

Router 1:

```
plaintext
enable
configure terminal
router ospf 1
router-id 1.1.1.1
network 192.168.1.0 0.0.0.255 area 0
network 10.0.0.0 0.0.0.3 area 0
exit
write memory
```

Router 2:

```
plaintext
enable
configure terminal
router ospf 1
router-id 2.2.2.2
network 10.0.0.0 0.0.0.3 area 0
network 10.0.1.0 0.0.0.3 area 0
exit
write memory
```

Router 3:

```
plaintext
enable
configure terminal
router ospf 1
router-id 3.3.3.3
network 192.168.3.0 0.0.0.255 area 0
network 10.0.1.0 0.0.0.3 area 0
exit
```

write memory

Step 6: Verify Configuration

Run the following commands on each router:

1. **Verify OSPF Neighbors:**

   ```
   plaintext
   show ip ospf neighbor
   ```

   - Ensure all routers recognize each other as OSPF neighbors.
2. **Verify Routing Table:**

   ```
   plaintext
   show ip route
   ```

   - Look for O routes in the routing table.
3. **Test Connectivity:**
   - Ping from **PC1** (192.168.1.x) to **PC4** (192.168.3.x).

   ```
   plaintext
   ping 192.168.3.x
   ```

Step 7: Save the Configuration

Ensure all routers save their configurations:

```
plaintext
copy running-config startup-config
```

6.Using JAVA RMI Write a program to implement Basic Calculator.

Here’s a step-by-step implementation of a basic calculator using **Java RMI (Remote Method Invocation)**.

The program includes the following components:

1. **Calculator Interface**: Defines the remote methods.
2. **Calculator Implementation**: Implements the interface methods.
3. **Server**: Registers the calculator service.
4. **Client**: Invokes the remote methods on the server.

Step 1: Create the Calculator Interface

The interface defines the methods that the remote calculator supports.

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Calculator extends Remote {
    // Declare methods for basic operations
    double add(double a, double b) throws RemoteException;
    double subtract(double a, double b) throws RemoteException;
    double multiply(double a, double b) throws RemoteException;
    double divide(double a, double b) throws RemoteException;
}
```

Step 2: Implement the Calculator Interface

The implementation provides the actual logic for the calculator.

```
java
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class CalculatorImpl extends UnicastRemoteObject implements Calculator {
    // Constructor
    protected CalculatorImpl() throws RemoteException {
        super();
    }

    @Override
    public double add(double a, double b) throws RemoteException {
        return a + b;
    }
    @Override
    public double subtract(double a, double b) throws RemoteException {
        return a - b;
    }

    @Override
    public double multiply(double a, double b) throws RemoteException {
        return a * b;
    }

    @Override
    public double divide(double a, double b) throws RemoteException {
        if (b == 0) {
            throw new ArithmeticException("Division by zero is not allowed.");
        }
        return a / b;
    }
}
```

Step 3: Create the Server

The server registers the Calculator service with the RMI registry.

java
```
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;

public class CalculatorServer {
    public static void main(String[] args) {
        try {
            // Create an instance of CalculatorImpl
            Calculator calculator = new CalculatorImpl();

            // Start the RMI registry
            LocateRegistry.createRegistry(1099); // Default RMI registry port is 1099

            // Bind the calculator object to a name in the RMI registry
            Naming.rebind("rmi://localhost/CalculatorService", calculator);

            System.out.println("Calculator Service is running...");
        } catch (Exception e) {
            System.err.println("Server exception: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Step 4: Create the Client

The client connects to the server and uses the calculator service.

```
java
import java.rmi.Naming;

public class CalculatorClient {
    public static void main(String[] args) {
        try {
            // Lookup the remote calculator object
            Calculator calculator = (Calculator)
Naming.lookup("rmi://localhost/CalculatorService");

            // Perform calculations
            System.out.println("Addition: " + calculator.add(10, 5));
            System.out.println("Subtraction: " + calculator.subtract(10, 5));
            System.out.println("Multiplication: " + calculator.multiply(10, 5));
            System.out.println("Division: " + calculator.divide(10, 5));
        } catch (Exception e) {
            System.err.println("Client exception: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Step 5: Compile and Run the Program

1. Compile the Code

```
javac Calculator.java CalculatorImpl.java CalculatorServer.java CalculatorClient.java
```

2. Start the RMI Registry

Start the RMI registry in a terminal:

```
rmiregistry
```

3. Start the Server

Run the server in a separate terminal:

```
java CalculatorServer
```

4. Start the Client

Run the client in another terminal:

```
java CalculatorClient
```

Output

The client will display the results of the calculations:

```plaintext
Addition: 15.0
Subtraction: 5.0
Multiplication: 50.0
Division: 2.0
```

7. Implement a Chatting application using JAVA TCP and UDP sockets.

To implement a chatting application using **Java** with **TCP** and **UDP sockets**, you can follow these steps. Below is a guide for building both **TCP** and **UDP** socket-based chat applications.

---

1. TCP Chatting Application

A TCP chat application ensures reliable communication between the client and the server.

Code: TCP Server

```
import java.io.*;
import java.net.*;

public class TCPServer {
    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(5000)) {
            System.out.println("Server started. Waiting for clients...");
            Socket socket = serverSocket.accept();
            System.out.println("Client connected!");

            BufferedReader input = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter output = new PrintWriter(socket.getOutputStream(), true);

            BufferedReader consoleInput = new BufferedReader(new
InputStreamReader(System.in));

            String clientMessage, serverResponse;
            while (true) {
                clientMessage = input.readLine();
                if (clientMessage.equalsIgnoreCase("exit")) {
                    System.out.println("Client disconnected!");
                    break;
                }
                System.out.println("Client: " + clientMessage);
                System.out.print("You: ");
                serverResponse = consoleInput.readLine();
                output.println(serverResponse);
            }

            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Code: TCP Client

```
import java.io.*;
import java.net.*;

public class TCPClient {
    public static void main(String[] args) {
        try (Socket socket = new Socket("localhost", 5000)) {
            System.out.println("Connected to the server!");

            BufferedReader input = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter output = new PrintWriter(socket.getOutputStream(), true);

            BufferedReader consoleInput = new BufferedReader(new
InputStreamReader(System.in));

            String serverMessage, clientMessage;
            while (true) {
                System.out.print("You: ");
                clientMessage = consoleInput.readLine();
                output.println(clientMessage);
                if (clientMessage.equalsIgnoreCase("exit")) {
                    System.out.println("Disconnected from the server!");
                    break;
                }
                serverMessage = input.readLine();
                System.out.println("Server: " + serverMessage);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

---

2. UDP Chatting Application

A UDP chat application allows for faster but less reliable communication between a client and a server.

Code: UDP Server

```
import java.net.*;

public class UDPServer {
    public static void main(String[] args) {
        try (DatagramSocket socket = new DatagramSocket(5000)) {
            System.out.println("UDP Server started. Waiting for clients...");

            byte[] buffer = new byte[1024];
            DatagramPacket packet = new DatagramPacket(buffer, buffer.length);

            while (true) {
                socket.receive(packet);
                String clientMessage = new String(packet.getData(), 0, packet.getLength());
                System.out.println("Client: " + clientMessage);

                if (clientMessage.equalsIgnoreCase("exit")) {
                    System.out.println("Client disconnected!");
                    break;
                }

                String response = "Message received: " + clientMessage;
                buffer = response.getBytes();
                InetAddress clientAddress = packet.getAddress();
                int clientPort = packet.getPort();
                packet = new DatagramPacket(buffer, buffer.length, clientAddress, clientPort);
                socket.send(packet);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Code: UDP Client

```
import java.net.*;
import java.util.Scanner;

public class UDPClient {
    public static void main(String[] args) {
        try (DatagramSocket socket = new DatagramSocket()) {
            InetAddress serverAddress = InetAddress.getByName("localhost");
            Scanner scanner = new Scanner(System.in);

            byte[] buffer = new byte[1024];
            while (true) {
                System.out.print("You: ");
                String message = scanner.nextLine();
                buffer = message.getBytes();
                DatagramPacket packet = new DatagramPacket(buffer, buffer.length,
serverAddress, 5000);
                socket.send(packet);

                if (message.equalsIgnoreCase("exit")) {
                    System.out.println("Disconnected from the server!");
                    break;
                }

                packet = new DatagramPacket(buffer, buffer.length);
                socket.receive(packet);
                String serverResponse = new String(packet.getData(), 0, packet.getLength());
                System.out.println("Server: " + serverResponse);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Steps to Run the Application

1. Save the server and client codes in separate .java files.
   - TCP server: TCPServer.java
   - TCP client: TCPClient.java
   - UDP server: UDPServer.java
   - UDP client: UDPClient.java
2. Compile the files:

   ```
   javac TCPServer.java TCPClient.java UDPServer.java UDPClient.java
   ```

3. Start the server:
   - For TCP: Run TCPServer first.
   - For UDP: Run UDPServer first.
4. Start the client:
   - For TCP: Run TCPClient.
   - For UDP: Run UDPClient.
5. Test by exchanging messages between the server and the client.

8. Hello command is used to know whether the machine at the other end is working or not. Echo command is used to measure the round-trip time to the neighbor. Implement Hello and Echo commands using JAVA.

To implement **Hello** and **Echo** commands in Java, you can create a simple client-server program using **UDP sockets**. Here's how:

Concept Explanation

1. **Hello Command**:
   - The client sends a "HELLO" message to the server.
   - The server replies with "HELLO RECEIVED" to confirm it is working.
2. **Echo Command**:
   - The client sends an "ECHO <message>" to the server.
   - The server echoes the same message back to the client.
   - The client measures the round-trip time (RTT).

Code Implementation

Hello and Echo UDP Server

```
java
import java.net.*;

public class HelloEchoServer {
    public static void main(String[] args) {
        try (DatagramSocket serverSocket = new DatagramSocket(5000)) {
            System.out.println("Server is running and waiting for client...");

            byte[] buffer = new byte[1024];
            DatagramPacket requestPacket = new DatagramPacket(buffer, buffer.length);

            while (true) {
                // Receive packet from client
                serverSocket.receive(requestPacket);
                String receivedMessage = new String(requestPacket.getData(), 0,
requestPacket.getLength());
                System.out.println("Received: " + receivedMessage);

                String response = "";

                // Process "HELLO" command
                if (receivedMessage.equalsIgnoreCase("HELLO")) {
                    response = "HELLO RECEIVED";
                }
                // Process "ECHO <message>" command
                else if (receivedMessage.startsWith("ECHO")) {
                    response = receivedMessage.substring(5); // Extract the message to echo
                } else if (receivedMessage.equalsIgnoreCase("exit")) {
                    System.out.println("Client disconnected.");
                    break;
                } else {
                    response = "UNKNOWN COMMAND";
                }

                // Send response back to the client
                byte[] responseData = response.getBytes();
                InetAddress clientAddress = requestPacket.getAddress();
                int clientPort = requestPacket.getPort();
                DatagramPacket responsePacket = new DatagramPacket(responseData,
responseData.length, clientAddress, clientPort);
                serverSocket.send(responsePacket);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Hello and Echo UDP Client

```
import java.net.*;
import java.util.Scanner;

public class HelloEchoClient {
    public static void main(String[] args) {
        try (DatagramSocket clientSocket = new DatagramSocket()) {
            InetAddress serverAddress = InetAddress.getByName("localhost");
            Scanner scanner = new Scanner(System.in);

            byte[] buffer = new byte[1024];

            while (true) {
                System.out.print("Enter command (HELLO / ECHO <message> / exit): ");
                String command = scanner.nextLine();
                long startTime = 0, endTime;

                // Send command to server
                buffer = command.getBytes();
                DatagramPacket requestPacket = new DatagramPacket(buffer, buffer.length,
serverAddress, 5000);
                startTime = System.nanoTime(); // Start measuring RTT
                clientSocket.send(requestPacket);

                // Exit condition
                if (command.equalsIgnoreCase("exit")) {
                    System.out.println("Disconnected from server.");
                    break;
                }

                // Receive response from server
                DatagramPacket responsePacket = new DatagramPacket(buffer, buffer.length);
                clientSocket.receive(responsePacket);
                endTime = System.nanoTime(); // End measuring RTT

                // Display server response and round-trip time
                String serverResponse = new String(responsePacket.getData(), 0,
responsePacket.getLength());
                System.out.println("Server Response: " + serverResponse);

                if (command.startsWith("ECHO")) {
                    double rtt = (endTime - startTime) / 1_000_000.0; // RTT in milliseconds
                    System.out.printf("Round-Trip Time: %.2f ms%n", rtt);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Steps to Run

1. Save the server code as HelloEchoServer.java and the client code as HelloEchoClient.java.
2. Compile the programs:

   ```
   javac HelloEchoServer.java HelloEchoClient.java
   ```

3. Start the server:

   ```
   java HelloEchoServer
   ```

4. Start the client in another terminal:

   ```
   java HelloEchoClient
   ```

5. Test the commands:

- For **HELLO**:
  - Client: HELLO
  - Server: Responds with HELLO RECEIVED.
- For **ECHO**:
  - Client: ECHO <your message>
  - Server: Responds with <your message>.
- Exit the client using exit.

Output Example

Client:

plaintext

```
Enter command (HELLO / ECHO <message> / exit): HELLO
Server Response: HELLO RECEIVED
Enter command (HELLO / ECHO <message> / exit): ECHO Testing Echo Command
Server Response: Testing Echo Command
Round-Trip Time: 1.23 ms
Enter command (HELLO / ECHO <message> / exit): exit
Disconnected from server.
```

Server:

plaintext

```
Server is running and waiting for client...
Received: HELLO
Received: ECHO Testing Echo Command
Client disconnected.
```

9. Using Wireshark perform the following operations: - Inspect HTTP Traffic
- Inspect HTTP Traffic from a Given IP Address,
- Inspect HTTP Traffic to a Given IP Address,
- Reject Packets to Given IP Address,
- Monitor Apache and MySQL Network Traffic.

Objectives

- **Part 1: Capture and view HTTP traffic**
- **Part 2: Capture and view HTTPS traffic**

Background / Scenario

HyperText Transfer Protocol (HTTP) is an application layer protocol that presents data via a web browser. With HTTP, there is no safeguard for the exchanged data between two communicating devices.

With HTTPS, encryption is used via a mathematical algorithm. This algorithm hides the true meaning of the data that is being exchanged. This is done through the use of certificates that can be viewed later in this lab.

Regardless of HTTP or HTTPS, it is only recommended to exchange data with websites that you trust. Just because a site uses HTTPS does not mean it is a trustworthy site. Threat actors commonly use HTTPS to hide their activities.

In this lab, you will explore and capture HTTP and HTTPS traffic using Wireshark.

Required Resources

- CyberOps Workstation VM
- Internet connection

Instructions

Part 1: Capture and View HTTP Traffic

In this part, you will use tcpdump to capture the content of HTTP traffic. You will use command options to save the traffic to a packet capture (pcap) file. These records can then be analyzed using different applications that read pcap files, including Wireshark.

Step 1: Start the virtual machine and log in.

Start the CyberOps Workstation VM. Use the following user credentials:

Username: **analyst**
Password: **cyberops**

Step 2: Open a terminal and start tcpdump.

a. Open a terminal application and enter the command `ip address`.

```
[analyst@secOps ~]$ ip address
```

b. List the interfaces and their IP addresses displayed in the **ip address** output.
enp0s3 with 10.0.2.15 and lo with 127.0.0.1 (answers for enp0s3 will vary).

c. While in the terminal application, enter the command sudo tcpdump –i enp0s3 –s 0 –w httpdump.pcap. Enter the password **cyberops** for the user analyst when prompted.

```
[analyst@secOps ~]$ sudo tcpdump –i enp0s3 –s 0 –w httpdump.pcap

[sudo] password for analyst:

tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
```

This command starts tcpdump and records network traffic on the **enp0s3** interface.
The -i command option allows you to specify the interface. If not specified, the tcpdump will capture all traffic on all interfaces.
The -s command option specifies the length of the snapshot for each packet. You should limit snaplen to the smallest number that will capture the protocol information in which you are interested. Setting snaplen to 0 sets it to the default of 262144, for backwards compatibility with recent older versions of tcpdump.
The -w command option is used to write the result of the tcpdump command to a file. Adding the extension .pcap ensures that operating systems and applications will be able to read to file. All recorded traffic will be printed to the file httpdump.pcap in the home directory of the user analyst.
Use the man pages for tcpdump to determine the usage of the -s and -w command options.

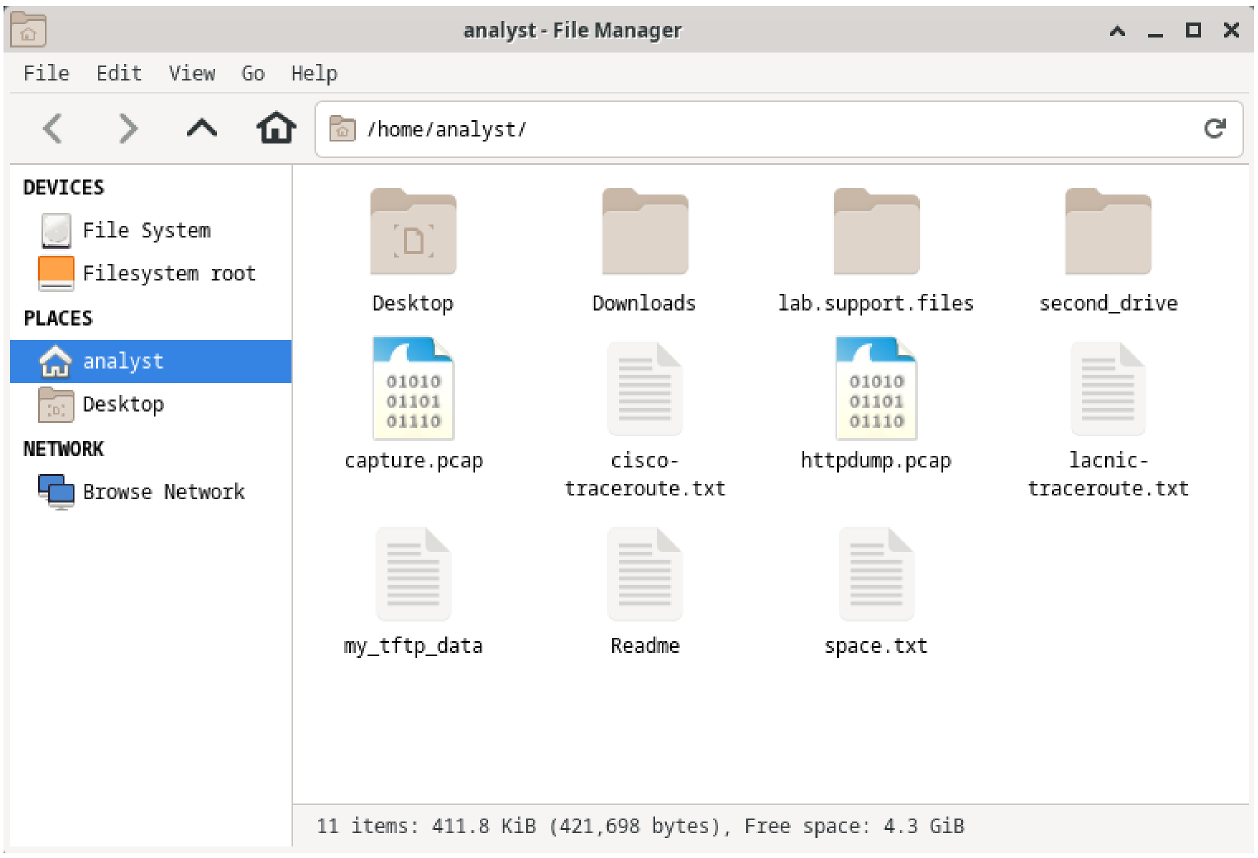
d. Open a web browser from the launch bar within the CyberOps Workstation VM. Navigate to http://www.altoromutual.com/login.jsp
Because this website uses HTTP, the traffic is not encrypted. Click the Password field to see the warning pop up.

e. Enter a username of **Admin** with a password of **Admin** and click **Login**.
f. Close the web browser.

g. Return to the terminal window where tcpdump is running. Enter **CTRL+C** to stop the packet capture.

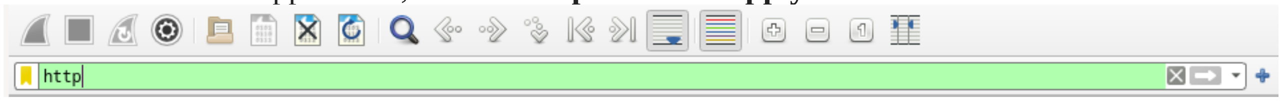
Step 3: View the HTTP capture.

The tcpdump, executed in the previous step, printed the output to a file named httpdump.pcap. This file is located in the home directory for the user **analyst**.
a. Click the File Manager icon on the desktop and browse to the home folder for the user **analyst**. Double-click the **httpdump.pcap** file, in the Open With dialog box scroll down to Wireshark and then click **Open**.

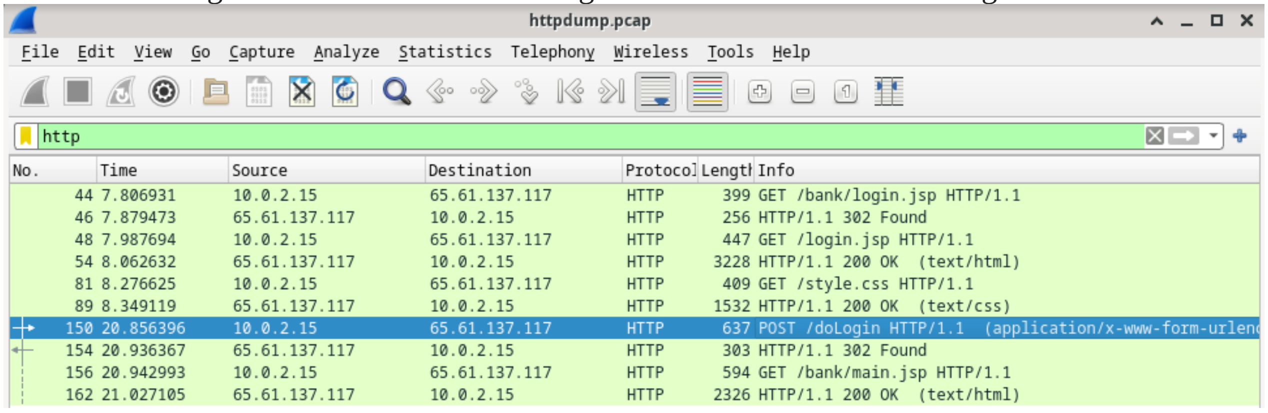


```
11 items: 411.8 KiB (421,698 bytes), Free space: 4.3 GiB
```

b. In the Wireshark application, filter for **http** and click **Apply**.

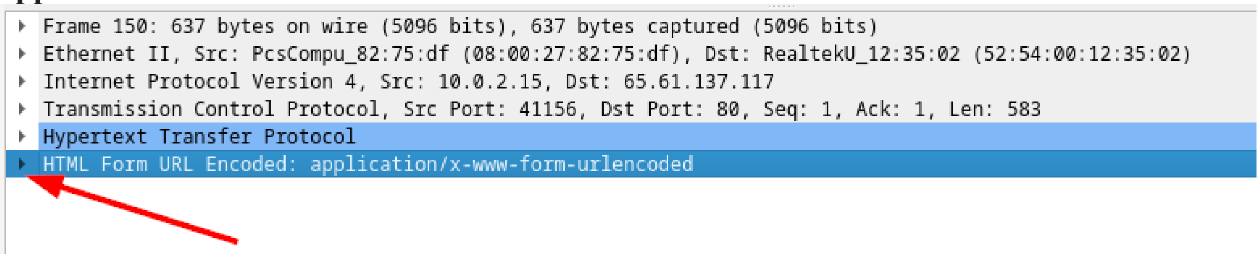


c. Browse through the different HTTP messages and select the **POST** message.


httpdump.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

```
No.    Time        Source         Destination    Protocol Length Info
    44 7.806931    10.0.2.15      65.61.137.117  HTTP      399 GET /bank/login.jsp HTTP/1.1
    46 7.879473    65.61.137.117  10.0.2.15      HTTP      256 HTTP/1.1 302 Found
    48 7.987694    10.0.2.15      65.61.137.117  HTTP      447 GET /login.jsp HTTP/1.1
    54 8.062632    65.61.137.117  10.0.2.15      HTTP     3228 HTTP/1.1 200 OK  (text/html)
    81 8.276625    10.0.2.15      65.61.137.117  HTTP      409 GET /style.css HTTP/1.1
    89 8.349119    65.61.137.117  10.0.2.15      HTTP     1532 HTTP/1.1 200 OK  (text/css)
   150 20.856396   10.0.2.15      65.61.137.117  HTTP      637 POST /doLogin HTTP/1.1  (application/x-www-form-urlen
   154 20.936367   65.61.137.117  10.0.2.15      HTTP      303 HTTP/1.1 302 Found
   156 20.942993   10.0.2.15      65.61.137.117  HTTP      594 GET /bank/main.jsp HTTP/1.1
   162 21.027105   65.61.137.117  10.0.2.15      HTTP     2326 HTTP/1.1 200 OK  (text/html)
```

d. In the lower window, the message is displayed. Expand the **HTML Form URL Encoded: application/x-www-form-urlencoded** section.



```
▸ Frame 150: 637 bytes on wire (5096 bits), 637 bytes captured (5096 bits)
▸ Ethernet II, Src: PcsCompu_82:75:df (08:00:27:82:75:df), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 65.61.137.117
▸ Transmission Control Protocol, Src Port: 41156, Dst Port: 80, Seq: 1, Ack: 1, Len: 583
▸ Hypertext Transfer Protocol
▸ HTML Form URL Encoded: application/x-www-form-urlencoded
```

What two pieces of information are displayed?

The uid of Admin and passw of Admin

e. Close the Wireshark application.

Part 2: Capture and View HTTPS Traffic

You will now use tcpdump from the command line of a Linux workstation to capture HTTPS traffic. After starting tcpdump, you will generate HTTPS traffic while tcpdump records the contents of the network traffic. These records will again be analyzed using Wireshark.

Step 1: Start tcpdump within a terminal.

a. While in the terminal application, enter the command `sudo tcpdump –i enp0s3 –s 0 –w httpsdump.pcap`. Enter the password cyberops for the user analyst when prompted.

```
[analyst@secOps ~]$ sudo tcpdump –i enp0s3 –s 0 –w httpsdump.pcap

[sudo] password for analyst:

tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
```

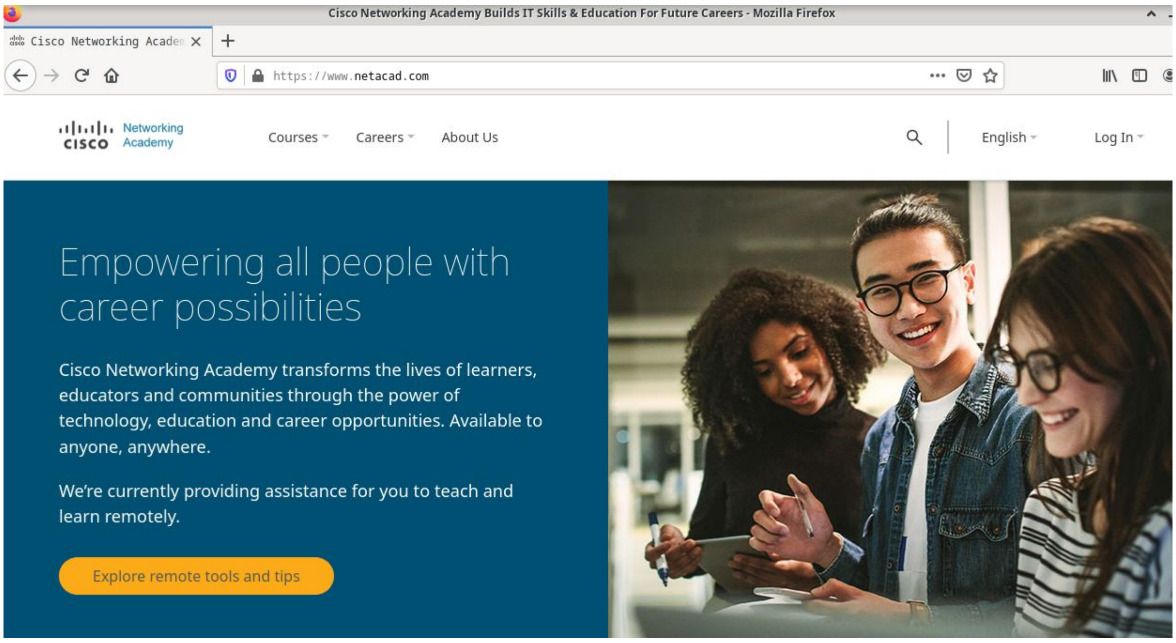
This command will start tcpdump and record network traffic on the **enp0s3** interface of the Linux workstation. If your interface is different than enp0s3, please modify it when using the above command.
All recorded traffic will be printed to the file **httpsdump.pcap** in the home directory of the user analyst.
b. Open a web browser from the launch bar within the CyberOps Workstation VM. Navigate to www.netacad.com.
Note: If you receive a “Secure Connection Failed” webpage it probably means the date and time are incorrect. Update the day and time with the following command, changing to the current day and time:

```
[analyst@secOps ~]$ sudo date -s "12 MAY 2020 21:38:20
```

What do you notice about the website URL?

Answers will vary. The website is using HTTPS, and there is a lock.

c. Click **Log in**.


Cisco Networking Academy Builds IT Skills & Education For Future Careers - Mozilla Firefox

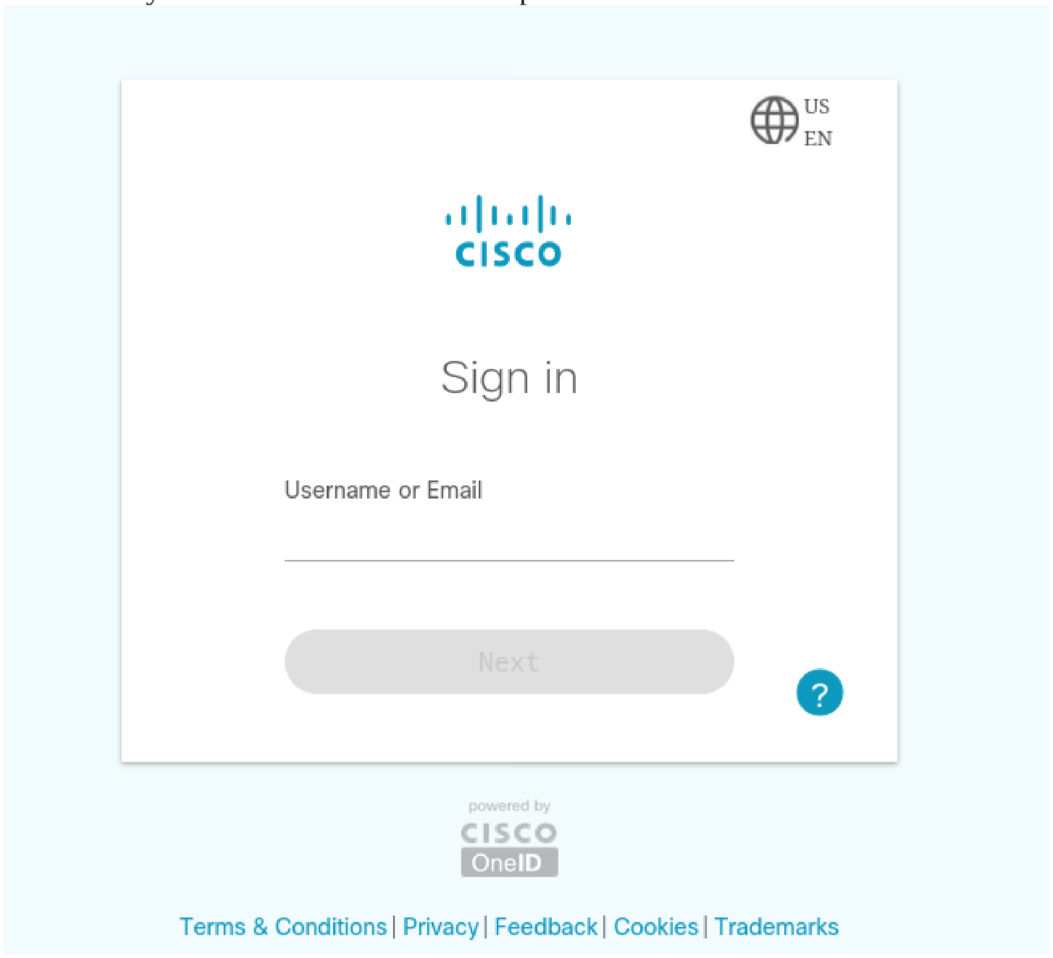
https://www.netacad.com

Empowering all people with career possibilities

Cisco Networking Academy transforms the lives of learners, educators and communities through the power of technology, education and career opportunities. Available to anyone, anywhere.

We're currently providing assistance for you to teach and learn remotely.

d. Enter in your NetAcad username and password. Click **Next**.


Sign in

Username or Email

US
EN
Next
powered by
CISCO OneID
Terms & Conditions | Privacy | Feedback | Cookies | Trademarks

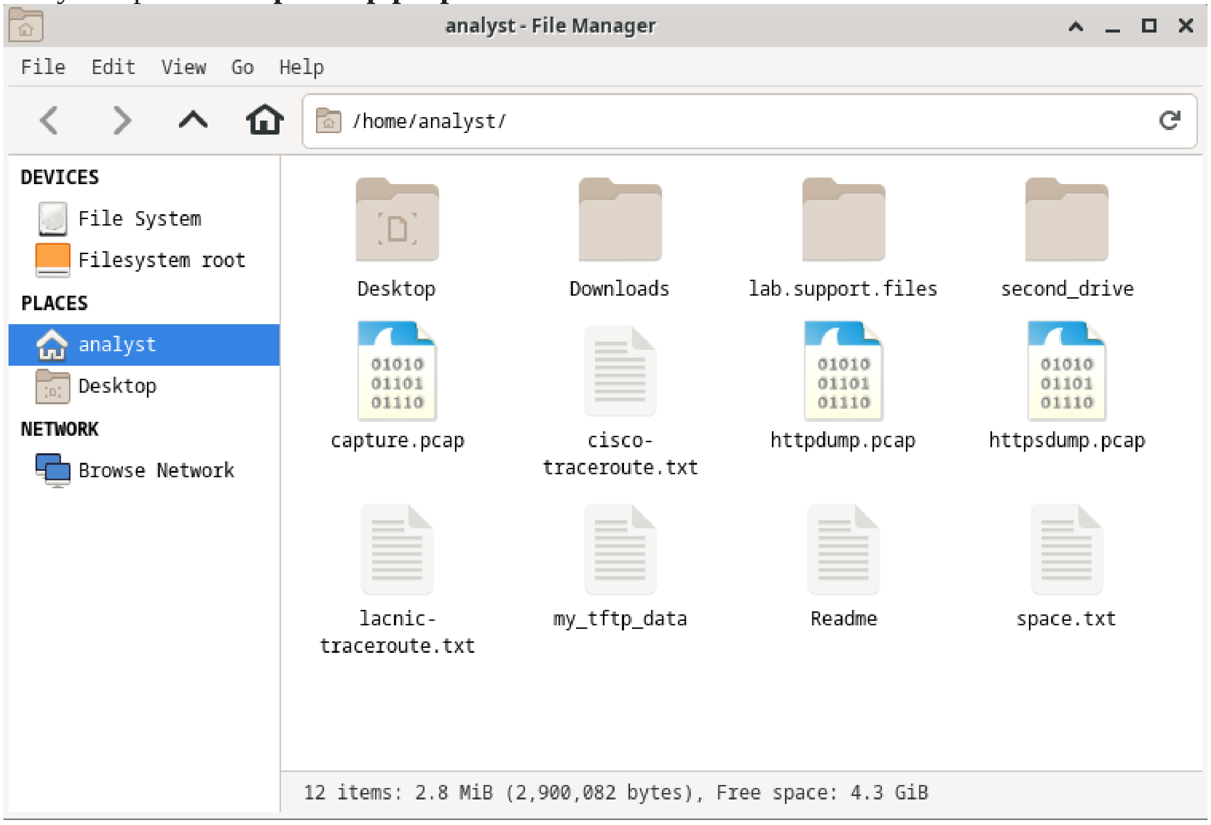
e. Close the web browser in the VM.

f. Return to the terminal window where tcpdump is running. Enter **CTRL+C** to stop the packet capture.

Step 2: View the HTTPS capture.

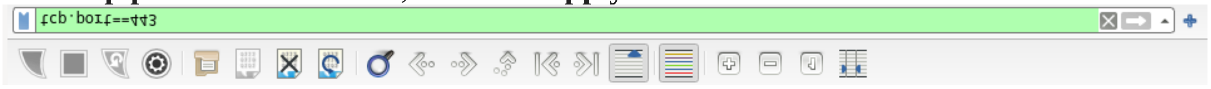
The tcpdump executed in Step 1 printed the output to a file named httpsdump.pcap. This file is located in the home directory for the user **analyst**.

a. Click the Filesystem icon on the desktop and browse to the home folder for the user analyst. Open the **httpsdump.pcap** file.

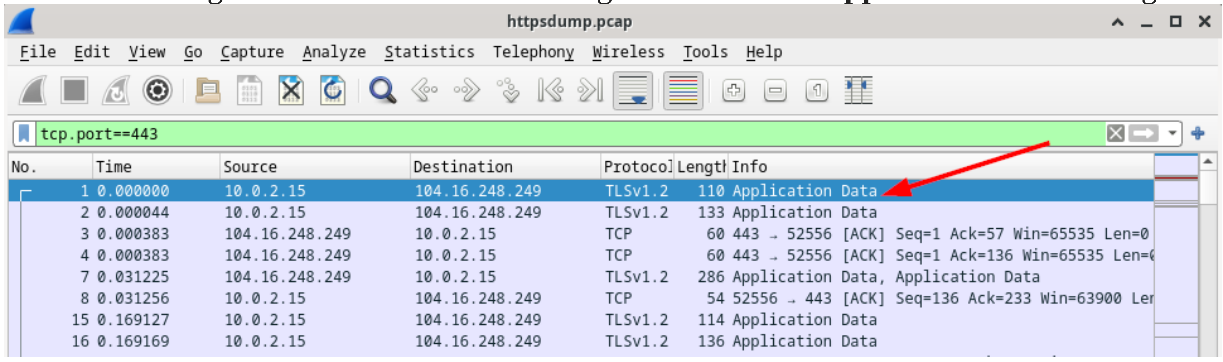


b. In the Wireshark application, expand the capture window vertically and then filter by HTTPS traffic via port 443.

Enter **tcp.port==443** as a filter, and click **Apply**.



c. Browse through the different HTTPS messages and select an **Application Data** message.


httpsdump.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port==443

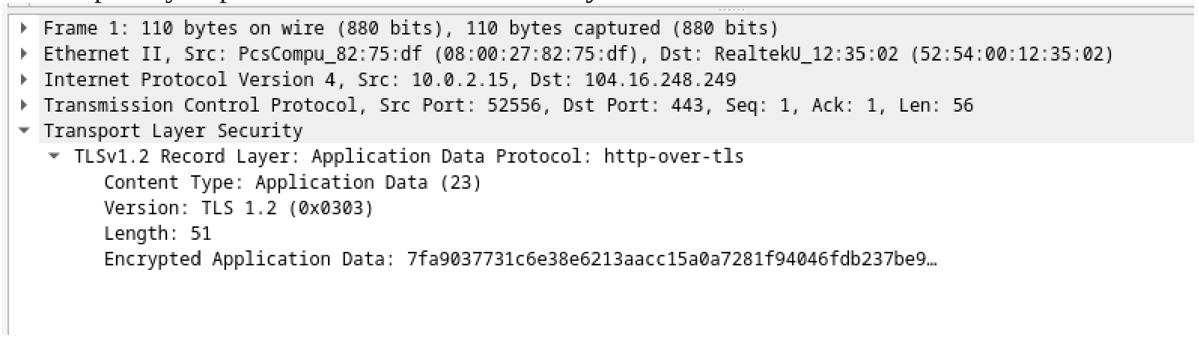
```
No.   Time      Source          Destination     Protocol Length Info
    1 0.000000  10.0.2.15       104.16.248.249  TLSv1.2   110 Application Data
    2 0.000044  10.0.2.15       104.16.248.249  TLSv1.2   133 Application Data
    3 0.000383  104.16.248.249  10.0.2.15       TCP        60 443 → 52556 [ACK] Seq=1 Ack=57 Win=65535 Len=0
    4 0.000383  104.16.248.249  10.0.2.15       TCP        60 443 → 52556 [ACK] Seq=1 Ack=136 Win=65535 Len=
    7 0.031225  104.16.248.249  10.0.2.15       TLSv1.2   286 Application Data, Application Data
    8 0.031256  10.0.2.15       104.16.248.249  TCP        54 52556 → 443 [ACK] Seq=136 Ack=233 Win=63900 Le
   15 0.169127  10.0.2.15       104.16.248.249  TLSv1.2   114 Application Data
   16 0.169169  10.0.2.15       104.16.248.249  TLSv1.2   136 Application Data
```

d. In the lower window, the message is displayed.

What has replaced the HTTP section that was in the previous capture file?

After the TCP section, there is now a Secure Sockets Layer (SSL/TLS 1.2) section instead of HTTP.

e. Completely expand the **Secure Sockets Layer** section.



```
▸ Frame 1: 110 bytes on wire (880 bits), 110 bytes captured (880 bits)
▸ Ethernet II, Src: PcsCompu_82:75:df (08:00:27:82:75:df), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 104.16.248.249
▸ Transmission Control Protocol, Src Port: 52556, Dst Port: 443, Seq: 1, Ack: 1, Len: 56
▾ Transport Layer Security
  ▾ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 51
      Encrypted Application Data: 7fa9037731c6e38e6213aacc15a0a7281f94046fdb237be9…
```

f. Click the **Encrypted Application Data**.
Is the application data in a plaintext or readable format?

The data payload is encrypted using TLSv1.2 and cannot be viewed.

g. Close all windows and shut down the virtual machine.