MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

CSE 3143
Computer Networks Lab

Laboratory Manual

DEPARTMENT OF COMPUTER SCIENCE & ENGG.

Contents

Course objectives:

This laboratory course enable students to get practical experience in

1. To Understand writing Network Programs
2. To Capture and analyze different network traffic.
3. To design and simulate the working of computer networks.
4. To Evaluate Performance of network and protocols.

Course outcomes:

On the completion of this laboratory course, the students will be able to:

1. Learn to Develop Network Application Programs.
2. Learn to capture and analyze network traffic using network Monitoring Tools.
3. Learn to design and simulate the working of networks.
4. Analyze performance of different networks and protocols.

Evaluation plan

SN	Components	Marks	Duration	Comments
1	Mid Term Test	30	90 Minutes	Write-Up:20 M, Execution:10 M (Will be held Regular labs)
2	Continuous Evaluation	20	In Regular Labs	The assessment is based on punctuality, program execution, neatness of Lab Record, etc.
3	Viva	10		viva-voce/questions on content
	Total Internal	60		
4	End Semester	40	120 minutes	
	Total	100		

Instructions to the Students

Pre- Lab Session Instructions

- Students should carry the Lab Record and the required stationery to every lab session.
- Be on time and follow the institution's dress code.
- Must Sign in the log register provided.
- Make sure to occupy the allotted seat and answer the attendance.
- Adhere to the rules and maintain the decorum.

In- Lab Session Instructions

- Follow the instructions on the allotted exercises.
- Show the program and results to the instructors on completion of experiments.
- On receiving approval from the instructor, copy the program and results in the Lab record.
- Prescribed textbooks and class notes can be kept ready for reference if required.

General Instructions for the Exercises in Lab

- Implement the given exercise individually and not in a group.
- Lab records should be complete with proper design, Algorithms, and Flowcharts, related to the experiment they perform.
- Plagiarism (copying from others) is strictly prohibited and would invite severe penalties in evaluation.
- The exercises for each week are divided into : Solved exercise & Lab exercises - to be completed during lab hours.
- In case a student misses a lab class, he/ she must ensure that the experiment is completed during the repetition class with the permission of the faculty concerned but credit will be given only to one day's experiment(s).
- Questions for lab tests and examinations are not necessarily limited to the questions in the manual but may involve some variations and/or combinations of the questions.

Instructions for Mini Project:

- All the students must take up a mini-project and submit the project report.
- Mini projects can be taken up in a group of a maximum comprising of 2 or 3 students.

- Students may refer to Appendix-I for list of suggested topics for Mini Project.

The Students Should Not:

- Bring mobile phones or any other electronic gadgets to the lab.
- Go out of the lab without permission.

Week 1: Introduction to Networking tools

The lab activities aim to familiarize students with creating simple network topologies, configuring switches, simulating network traffic, capturing packets with TCPdump, and analyzing them using Wireshark. By the end of the week, students should be proficient in designing, troubleshooting basic networks, and analyzing network traffic.

PACKET TRACER

Packet Tracer is a powerful network simulation tool developed by Cisco, designed to provide hands-on experience in networking concepts and protocols. It offers a virtual environment where students and network professionals can design, configure, and troubleshoot networks without the need for physical hardware. With its intuitive graphical interface, Packet Tracer allows users to create complex network topologies, configure a wide range of devices such as routers, switches, and end devices, and simulate network traffic to observe how data flows through the network. This makes it an invaluable tool for learning and practicing networking skills, whether for educational purposes or professional development.

One of the key features of Packet Tracer is its ability to visualize network behavior and operations. Users can see real-time packet flow, analyze protocol interactions, and gain insights into network performance. Additionally, Packet Tracer supports a variety of network technologies and protocols, making it a versatile tool for exploring different networking scenarios.

Packet Tracer features two primary workspaces: the Logical Workspace and the Physical Workspace. The **Logical Workspace** is where most of the network design and configuration take place. In this workspace, users can drag and drop network devices, such as routers, switches, and PCs, onto the canvas to create network topologies. They can then configure these devices, set up connections, and simulate network operations. This workspace is ideal for visualizing and understanding the logical flow of data and the interaction between different network components. The Logical Workspace also includes various tools and options to customize network settings, test configurations, and troubleshoot issues.

The **Physical Workspace**, on the other hand, provides a more realistic representation of the network's physical layout. This workspace allows users to see how devices are physically arranged in different environments, such as in a building or across multiple locations. It includes representations of racks, cabling, and device placements, offering a more tangible perspective on network design. The Physical Workspace helps users understand the spatial aspects of network setup, such as cable management and device accessibility. Together, these workspaces provide a comprehensive view of network design and operations, enabling users to gain a deeper understanding of both the logical and physical aspects of networking.

Packet Tracer offers several modes to facilitate different aspects of network design, configuration, and troubleshooting. The two main modes are the Real-Time Mode and the Simulation Mode.
Real-Time Mode is the default operational mode in Packet Tracer, where network devices behave as they would in a real-world environment. Changes made to device configurations and network settings take effect immediately, allowing users to see the results of their configurations in real time. This mode is ideal for testing and validating network setups, as it provides an accurate representation of how devices will perform in an actual network. Users can interact with devices, send data across the network, and observe how packets are processed and routed.

Simulation Mode offers a more controlled environment for analyzing network behavior. In this mode, users can capture and examine the flow of packets through the network, providing detailed insights into protocol operations and interactions. Simulation Mode allows users to pause the network activity, view packet details, and step through the network processes at their own pace.

Switch and Router Modes in Packet Tracer

- ✓ **User EXEC Mode**:
    - **Switch Prompt**: Switch>
    - **Router Prompt**: Router>
    - This is the initial mode when a user connects to a switch or router.
    - Limited to basic monitoring commands.
    - It does not allow for configuration changes.
- ✓ **Privileged EXEC Mode**:
    - **Switch Prompt**: Switch#
    - **Router Prompt**: Router#

- Accessed from User EXEC Mode by entering the enable command.
- Provides access to all monitoring commands and the ability to enter configuration modes.
- Used for advanced diagnostics and troubleshooting.

✓ **Global Configuration Mode**:
- **Switch Prompt**: Switch(config)#
- **Router Prompt**: Router(config)#
- Accessed from Privileged EXEC Mode by entering the configure terminal command.
- Allows configuration changes to the device as a whole.
- From here, users can enter specific configuration sub-modes (e.g., interface, VLAN for switches, and routing protocol for routers).

✓ **Interface Configuration Mode**:
- **Switch Prompt**: Switch(config-if)#
- **Router Prompt**: Router(config-if)#
- A sub-mode of Global Configuration Mode, accessed by specifying an interface (e.g., interface FastEthernet0/1 for switches or interface GigabitEthernet0/0 for routers).
- Used to configure specific interfaces, such as setting IP addresses or enabling port security.

✓ **Routing Configuration Mode** (Routers Only):
- **Router Prompt**: Router(config-router)#
- A sub-mode of Global Configuration Mode, accessed by specifying a routing protocol (e.g., router ospf 1).
- Used to configure routing protocols and settings.

GNS3

GNS3 (Graphical Network Simulator-3) is a powerful network simulation tool that allows users to create virtual networks using real network operating systems, such as Cisco IOS, Juniper JunOS, or even custom virtual machines. It provides a graphical interface to design and configure complex network topologies, making it an invaluable tool for network engineers, administrators, and students to test and prototype network setups without needing physical hardware.

Key Features of GNS3:

1. **Virtualization**: GNS3 utilizes virtualization technologies to emulate network devices and appliances. This allows users to run actual network operating systems on their computers, mimicking real-world network scenarios.

2. **Graphical Interface**: GNS3 offers a user-friendly graphical interface where users can drag and drop network devices (routers, switches, firewalls, etc.) onto a workspace and interconnect them using virtual cables. This visual approach simplifies network design and configuration.
3. **Device Support**: It supports a wide range of network devices from various vendors, including Cisco, Juniper, Arista, HP, and more. Users can download and integrate device images (known as "appliances") from the GNS3 Marketplace or import their own.
4. **Integration with Virtual Machines**: GNS3 allows integration with external virtual machines (VMs), enabling users to run applications, servers, and additional network services alongside their simulated network infrastructure.
5. **Simulation and Emulation**: GNS3 can simulate network behaviors and traffic flows realistically. Users can configure device settings, simulate network protocols, and analyze packet flows to test network configurations and troubleshoot issues.
6. **Community and Resources**: GNS3 has a vibrant community of users who contribute device templates, configurations, and troubleshooting tips. It also provides extensive documentation, tutorials, and forums to help users learn and troubleshoot network designs.
7. **Scalability**: GNS3 is scalable, allowing users to create small lab setups for learning and testing purposes, as well as complex enterprise-level networks with multiple interconnected devices.

GNS3 Workspace: The GNS3 workspace is the area of GNS3 where you create topologies by adding devices and links.

IP address configuration:	**ip ip-address network-mask default-gateway(optional)**

In a computer network lab setting, understanding the roles of switches and hosts (end devices) is essential for setting up and troubleshooting network configurations.

Switches:

1. **Purpose**: Connect multiple devices (like computers, printers) within a local area network (LAN).
2. **Function**:
   - **Connectivity Hub**: Acts as a central point for devices to connect and communicate.

   - **Packet Forwarding**: Uses MAC addresses to send data only to the intended recipient, reducing network congestion.
   - **Segmentation**: Divides the network into smaller parts to improve performance and manage traffic.
3. **Key Features**:
   - **MAC Address Learning**: Automatically learns which devices are connected to each port.
   - **VLAN Support**: Allows networks to be divided into separate broadcast domains for security and efficiency.

A network switch is a hardware device that connects multiple devices within a local area network (LAN), facilitating communication between them. Switches operate primarily at the data link layer (Layer 2) of the OSI model, using MAC addresses to forward data to the correct destination. Unlike hubs, which broadcast data to all connected devices, switches intelligently direct data only to the specific device it is intended for, reducing unnecessary traffic and enhancing network efficiency. Managed switches offer advanced features such as VLANs, quality of service (QoS), and port security, allowing network administrators to optimize and secure their networks. They play a crucial role in both small and large networks, providing the backbone for data transmission and network management.

In a network, assigning an IP address to a switch is important for management purposes, such as remote configuration, monitoring, and troubleshooting.

```
Switch(config)# interface vlan 1
Switch(config-if)# ip address 192.168.1.2 255.255.255.0
Switch(config-if)# no shutdown
```

If the switch needs to communicate with devices outside its local subnet, configure a default gateway.

```
Switch(config)# ip default-gateway 192.168.1.1
```

Hosts:

End devices, also known as hosts, are the devices that users interact with in a network. They form the endpoints of communication and include a wide range of hardware.

1. **Role**: Devices like computers, printers, and servers that generate and consume data.
2. **Function**:
   - **Data Generation**: Create and send data packets (frames) across the network.

    - **Data Consumption**: Receive and process data sent from other devices.
3. **Features**:
    - **IP Addressing**: Each host has a unique IP address for identification and communication.
    - **Network Applications**: Run various applications like web browsers, email clients, and file sharing tools.

Interaction Between Switches and Hosts

- **Data Exchange**: Hosts communicate through switches by sending and receiving data frames.
- **MAC Address Handling**: Switches learn and store MAC addresses to efficiently deliver data within the network.
- **Efficiency**: Reduces network traffic and optimizes data transmission by directing packets only to the intended destination.

Lab Exercise 1:

A client has requested that you set up a simple network with two PCs connected to a switch. Verify that the hardware, along with the given configurations, meet the requirements of the client. Following are the steps to setup the above given scenario: (Packet Tracer & GNS3)

Set Up the Network Topology

- Add two PCs and a switch.
- Connect PC0 to Switch0 Fa0/1 and PC1 to Switch0 Fa0/2 with straight-through cables. (Ports can change)
- Configure PC0 with IP 192.168.10.10 and Subnet Mask 255.255.255.0.
- Configure PC1 with IP 192.168.10.11 and Subnet Mask 255.255.255.0.

- **Test Connectivity**

- Use the ping command from PC0 to PC1 to verify connectivity.

Additionally in the packet tracer configuration add an IP address 192.268.10.100 to the switch for remote management

Capturing Packets with GNS3

Wireshark is a versatile network protocol analyzer that allows users to capture and inspect network traffic in real-time. It's widely used by network administrators, security professionals, and developers to troubleshoot network issues, analyze security incidents, and debug network protocols.

Capturing Packets with GNS3

When using **GNS3** (Graphical Network Simulator-3), an emulator that allows you to simulate networks using real Cisco images, integrating Wireshark for packet capture enhances the simulation's capabilities. Here’s how you can effectively capture packets with Wireshark in a GNS3 network:

1. **Setting Up the Network in GNS3**:
   - Create a network topology in GNS3 by dragging and dropping devices such as routers, switches, and virtual PCs onto the workspace.
   - Connect these devices using virtual links that mimic physical connections.
2. **Adding a Virtual Machine (VM) with Wireshark**:
   - For capturing packets effectively, integrate a VM (e.g., Ubuntu, Windows) with Wireshark installed into your GNS3 topology.
   - Configure the VM's network adapter to bridge with the GNS3 network or use a cloud node in GNS3 to simulate external traffic.
3. **Connecting Wireshark to Capture Traffic**:
   - Start Wireshark on the VM and select the network interface corresponding to the GNS3 network.
   - Begin capturing packets by clicking on the interface or using Wireshark's capture options to filter specific types of traffic.
4. **Analyzing Network Traffic**:
   - Wireshark captures packets as they traverse the virtual network in GNS3.
   - Analyze captured packets in real-time or save them for offline analysis to troubleshoot issues, debug protocols, or monitor network performance.
5. **Applying Filters for Specific Analysis**:
   - Use Wireshark’s powerful filtering capabilities to focus on specific types of packets (e.g., ICMP, TCP, UDP).
   - Apply display filters to refine the view and concentrate on packets relevant to your analysis, such as packets between specific IP addresses or using particular protocols.
6. **Saving Captured Data**:
   - Save captured packets to a file for future reference or to share with colleagues for collaborative troubleshooting.
   - Export packets in various formats for further analysis or reporting.

Benefits of Using Wireshark with GNS3

- **Realistic Network Simulation**: Wireshark enhances GNS3's simulation capabilities by providing a detailed view of how virtual devices communicate and interact.
- **Troubleshooting and Debugging**: It allows network administrators and developers to diagnose issues, identify network anomalies, and debug protocol implementations within the virtual environment.
- **Educational Purposes**: In educational settings, combining GNS3 with Wireshark offers students hands-on experience in packet analysis, network monitoring, and understanding network behaviour.

TCPdump is a powerful command-line packet analyzer tool that captures and displays network traffic passing through a computer's network interface. It allows users to see the detailed contents of packets, making it a valuable tool for network diagnostics, monitoring, and security analysis.

Key Features of TCPdump

1. Packet Capture:
   - TCPdump captures packets from a network interface in real-time. It can capture all traffic or filter specific types of traffic based on criteria like protocol, source/destination IP, ports, etc.
2. Packet Filtering:
   - TCPdump uses the Berkeley Packet Filter (BPF) syntax to filter packets. This allows you to specify exactly which packets to capture, reducing the amount of data and focusing on relevant traffic.
3. Display Options:
   - Captured packets can be displayed in a human-readable format, showing detailed headers and payload data. You can choose the level of detail displayed.
4. Output to File:
   - TCPdump can save captured packets to a file in the PCAP format, which can be analyzed later using tools like Wireshark.
5. Protocol Analysis:
   - TCPdump supports analysis of various protocols, including TCP, UDP, ICMP, HTTP, DNS, and many more.

Basic TCPdump Commands

1. Capture All Packets on an Interface:

```
tcpdump -i eth0
```

- Captures all packets on the `eth0` interface.

2. Capture Packets and Save to a File:

```
tcpdump -i eth0 -w capture.pcap
```

- Captures packets and saves them to `capture.pcap`.

3. Read Packets from a Capture File:

```
tcpdump -r capture.pcap
```

- Reads packets from `capture.pcap`.

4. Filter by Protocol (TCP, UDP, ICMP):

```
tcpdump -i eth0 tcp
tcpdump -i eth0 udp
tcpdump -i eth0 icmp
```

- Captures only TCP, UDP, or ICMP packets on the `eth0` interface.

5. Filter by IP Address:

```
tcpdump -i eth0 host 192.168.10.10
```

- Captures packets to/from the specified IP address.

6. Filter by Port:

```
tcpdump -i eth0 port 80
```

- Captures packets to/from the specified port (e.g., HTTP traffic).

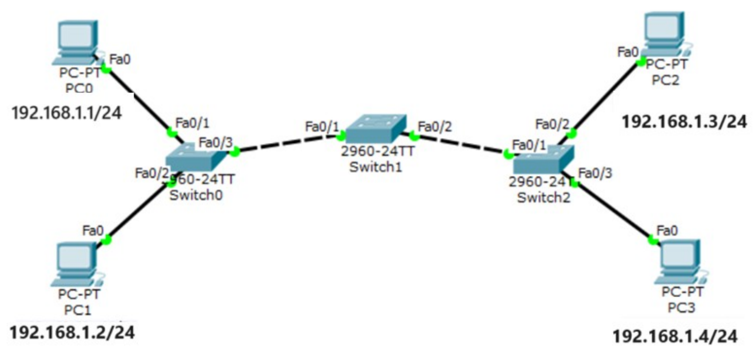
7. Detailed Packet Display:

```
tcpdump -i eth0 -vv
```

- Displays packets with detailed information (`-vv` for very verbose).

Lab Exercise 2

Set Up the Network Topology as shown and test connectivity between all end devices.
(Packet Tracer & GNS3)



Use Wireshark to capture and analyze network traffic in GNS3

- **Start Capturing Traffic on PC0:**
  - Open Wireshark on PC0 and Select the eth0 interface to start capturing packets.
  - Click on the start capture button.
- **Generate Traffic by Pinging from PC0 to PC1:**
  - Open a terminal on PC0 and ping PC1's IP address.
- **Stop the Capture:**
  - Go back to Wireshark and stop the capture by clicking on the red stop button.
- **Analyze the Captured Traffic:**
  - Look for ICMP packets in the capture. These packets represent the ping requests and replies.
  - You can use the filter icmp in Wireshark to display only the ICMP packets.

Additionally in GNS3, use `tcpdump` to capture and analyze network traffic:

- **Start Capturing Traffic on PC0:**

  - Open a terminal on PC0 and start capturing traffic on `eth0` interface:

```
tcpdump -i eth0 -w pc0_capture.pcap
```

  - This command will capture all traffic on `eth0` and save it to a file named `pc0_capture.pcap`.

- **Generate Traffic by Pinging from PC0 to PC1:**

  - Open another terminal on PC0 and ping PC1's IP address.

- **Stop the Capture**

  - Go back to the terminal where `tcpdump` is running on PC0 and stop it by pressing `Ctrl+C`.

- **Analyze the Captured Traffic:**

  - To analyze the captured traffic, you can use the following command to read the capture file:

```
tcpdump -r pc0_capture.pcap
```

Week 2: Router configuration in networks

This lab experiment introduces students to the fundamentals of routers in computer networks. The activities focus on understanding the role of routers, configuring basic router settings, and troubleshooting common router configurations. By the end of the experiment, students should be able to confidently design, configure, and troubleshoot basic router configurations in a simulated network environment.

Routers:

Routers are essential networking devices that connect multiple networks together, facilitating data transmission between them based on IP addresses. They serve as gateways, directing traffic efficiently across different network segments and enabling communication between devices in diverse network environments.

1. **Purpose**: Connect different networks together and manage traffic between them.
2. **Function**:
   - **Inter-Network Communication**: Route data packets between different networks using IP addresses.
   - **Gateway**: Provide a path for data to travel between networks, ensuring efficient data delivery.
   - **Network Address Translation (NAT)**: Translate private IP addresses to public IP addresses for internet access.
3. **Key Features**:
   - **Routing Table**: Stores information about paths to different networks and determines the best path for data transmission.
   - **Firewall**: Protects networks by filtering incoming and outgoing traffic based on security rules.

Interaction Between Devices

- **Switches and Hosts**: Switches manage local network traffic by directing data frames to their intended destinations based on MAC addresses.
- **Routers and Hosts**: Routers facilitate communication between different networks by using IP addresses to forward packets between them.

Lab Exercise 1

You are tasked with setting up a simple network for a client that includes three PCs connected to a routers. The objective is to ensure seamless communication between the PCs across these LANs using the configured router.

Set Up the Network Topology

1. **Add Devices**:
   - Add three PCs: PC0, PC1, and PC2.
   - Add a Cisco 1841 router.
2. **Connect Devices**:
   - Connect PC0 to Router0 Fa0/0 with a straight-through cable.
   - Connect PC1 to Router0 Fa0/1 with a straight-through cable.
   - Connect PC2 to Router0 Fa0/2 with a straight-through cable.
3. **Configure IP Addresses**:
   - Configure PC0 with IP 192.168.10.10 and Subnet Mask 255.255.255.0.
   - Configure PC1 with IP 192.168.20.10 and Subnet Mask 255.255.255.0.
   - Configure PC2 with IP 192.168.30.10 and Subnet Mask 255.255.255.0.
   - Configure Router0 interfaces:
     - Fa0/0 with IP 192.168.10.1 and Subnet Mask 255.255.255.0.
     - Fa0/1 with IP 192.168.20.1 and Subnet Mask 255.255.255.0.
     - Fa0/2 with IP 192.168.30.1 and Subnet Mask 255.255.255.0.
4. **Router Configuration**:
   - Access the router and configure the interfaces:

```
Router> enable
Router# configure terminal
Router(config)# interface fastethernet0/0
Router(config-if)# ip address 192.168.10.1 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface fastethernet0/1
Router(config-if)# ip address 192.168.20.1 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface fastethernet0/2
Router(config-if)# ip address 192.168.30.1 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# exit
Router# copy running-config startup-config
```

Test Connectivity

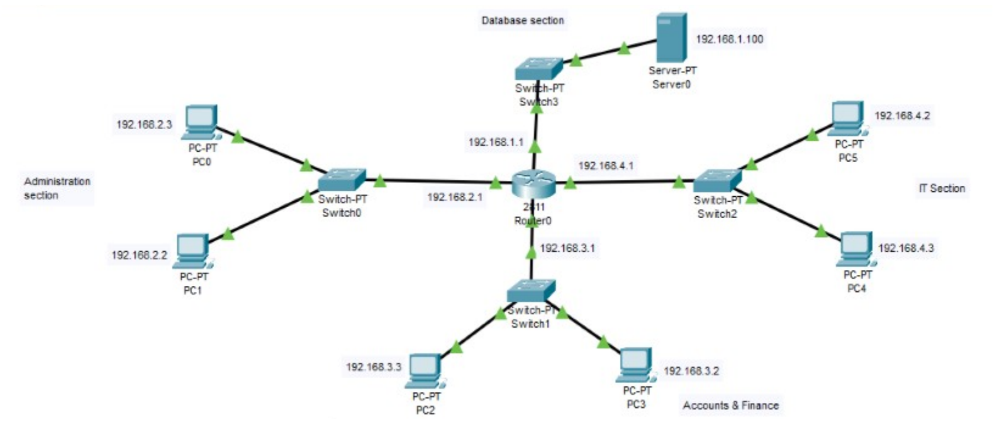
- Open the command prompt on PC0 and type:

```
ping 192.168.20.10
```

similarly ping between all hosts.

Lab Exercise 2

You have been tasked with designing and implementing a network infrastructure for a small organization with four distinct sections: Administration, Accounts and Finance, Information Technology (IT), and Database. The objective is to ensure efficient communication and data sharing among departments while maintaining security and scalability.



Week 3: The Web, HTTP and DNS

Week 4: Static Routing and Application Layer Protocols

In this lab exercise, students will configure a network using static routing to direct traffic between different network segments. They will also set up and test application layer protocols, including HTTP and DNS, to ensure proper web services and domain name resolution.

Introduction to Routing

Routing is the process of selecting paths in a network along which to send data packets. It enables communication between different network segments, such as different subnets or VLANs. Routers, which are network devices, perform routing by analyzing the destination IP address of a packet and forwarding it to the appropriate next-hop address.

Types of Routing

1. **Static Routing:**
    - Static routing involves manually configuring routes on a router.
    - It is suitable for small or simple networks where routes do not change frequently.
    - Advantages:
        - Simplicity and predictability.
        - No overhead of route computation.
    - Disadvantages:
        - Lack of scalability.
        - Manual updates are required for any network changes.
2. **Dynamic Routing:**
    - Dynamic routing uses protocols that automatically discover and maintain routes.
    - Common dynamic routing protocols include RIP (Routing Information Protocol), OSPF (Open Shortest Path First), and EIGRP (Enhanced Interior Gateway Routing Protocol).
    - Advantages:
        - Scalability and automatic adaptation to network changes.
        - Efficient route computation and management.
    - Disadvantages:
        - Higher complexity and overhead.
        - Potential for routing loops and convergence delays.

Introduction to Static Routing

Static routing is a type of network routing technique where routes are manually configured and maintained by a network administrator. Unlike dynamic routing, which automatically adjusts to network changes, static routing requires manual updates whenever the network topology changes. This method is straightforward and useful for small, simple networks with stable topologies.

Key Concepts

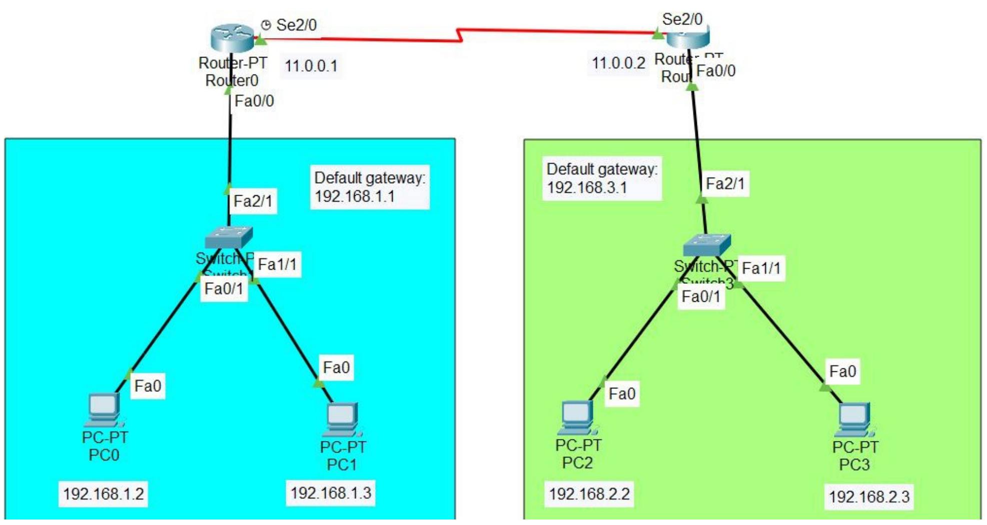
1. **Routing Table:**
   - A routing table is a data table stored in a router or a networked computer that lists the routes to particular network destinations.
   - Each route specifies a destination IP address, a subnet mask, a gateway (next-hop) address, and sometimes a metric.
2. **Next-Hop Address:**
   - The next-hop address is the IP address of the next device to which packets should be forwarded on their way to the destination.
3. **Administrative Distance:**
   - Administrative distance is a measure of the trustworthiness of the route source. Static routes typically have an administrative distance of 1, making them highly preferred.

Configuring Static Routing

To configure static routing, you need to manually add routes to the routing table of each router in the network. The configuration typically involves specifying the destination network, the subnet mask, and the next-hop address or exit interface.
CLI command : ip route <network id> <subnet mask><next hop>

Lab Exercise 1

Configure the PCs (hosts) with IPv4 address and Subnet Mask according to the IP addressing table given above.



Configure router with IP address and subnet mask.

Device	**Interface**	**IPv4 Addressing**	**Subnet Mask**
router 0	FastEthernet0/0	192.168.1.1	255.255.255.0
	Serial2/0	11.0.0.1	255.255.255.0
router 1	FastEthernet0/0	192.168.2.1	255.255.255.0
	Serial2/0	11.0.0.2	255.255.255.0

Static routing is not yet configured. Answer the following:

a. Can PC0 ping to PC1?

b. Can PC2 ping to PC3?

c. Can PC0 ping to PC3? Why not?

Configure the Static routes :

Static Routes for Router0 are given below:

Router(config)#ip route 192.168.2.0 255.255.255.0 11.0.0.2

Static Routes for Router1 are given below:

Router(config)#ip route 192.168.1.0 255.255.255.0 11.0.0.1

d. Can PC0 ping to PC3?

Introduction to Application Layer Services

The application layer in the OSI model provides various services to applications for effective communication. Two of the most essential application layer protocols are HTTP (Hypertext Transfer Protocol) and DNS (Domain Name System). These protocols facilitate web browsing and domain name resolution, respectively, which are crucial for modern network operations.

An HTTP server is a web server. It stores web resources that can be accessed by a web client. Your PC's browser(a web client) requests for web resources from a web server over
the internet. Web resources are files such as text and images that the server will give to the
client on request.

A Domain Name Service(DNS) server resolves host names into IP addresses. Although we can
access a network host using its IP address, DNS makes it easier by allowing us use domain
names which are easier to remember. For example its much easier to access google website by
typing http://www.google.com as compared to typing http://208.117.229.214. In either case,
you'll access google website, but using domain name is obviously easier.
Now, before any host can use a DNS service, we must configure a DNS server first. For example, when you type the URL http://www.google.com in your browser, the host will query
the DNS server for the IP address of http://www.google.com. The DNS server will resolve http://www.google.com into an IP address then answer back the host with the IP
address.

Lab Exercise 2

Configure a web server within the network, create a simple web page, and ensure HTTP access from PCs located in a different LAN.
Add a Server (Web Server) to the LAN in Router 0

- IP Address: 192.168.1.100
- Subnet Mask: 255.255.255.0

    - Default Gateway: 192.168.1.1 (Router0's FastEthernet0/0 IP)

Ensure that the server can ping PCs in the other LAN.

Create a basic HTML file (e.g., index.html) with the personalized content. Save the file. Enable the HTTP Service and ensure the web server service is running.

- Can PC0 access the web page hosted on the web server?
- Can PC2 access the web page hosted on the web server?
- If PC2 or PC3 cannot access the web page, what could be the issue?

Configure a DNS server within the network to resolve a domain name to the web server's IP address.

DNS Server:

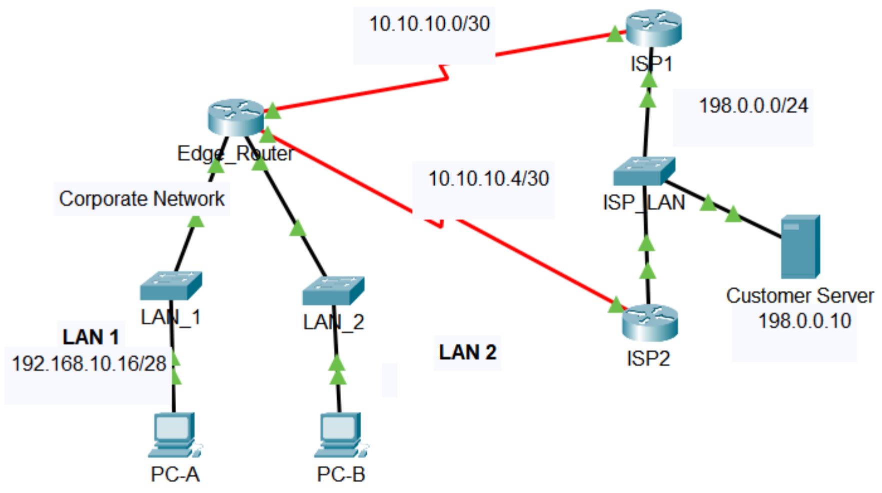
- IP Address: 192.168.1.200
- Subnet Mask: 255.255.255.0
- Default Gateway: 192.168.1.1 (Router0's FastEthernet0/0 IP)

Create an 'A' record for the domain name www.testing-cn.com and point it to the web server's IP address (192.168.1.100).
On each PC, set the DNS server IP to 192.168.1.200 in the network settings.

Can PC3 resolve and access the web page using the domain name www.example.com? Use the nslookup command to check DNS resolution.

Additional Exercise:

The PT network requires static routes to provide internet access to the internal LAN users through the ISPs. In addition, the ISP routers require static routes to reach the internal LANs.



Device	Interface	IP Address / Prefix
Edge_Router	S0/0/0	10.10.10.2/30
Edge_Router	S0/0/1	10.10.10.6/30
Edge_Router	G0/0	192.168.10.17/28
Edge_Router	G0/1	192.168.11.33/27
ISP1	S0/0/0	10.10.10.1/30
ISP1	G0/0	198.0.0.1/24
ISP2	S0/0/1	10.10.10.5/30
ISP2	G0/0	198.0.0.2/24
PC-A	NIC	192.168.10.19/28
PC-B	NIC	192.168.11.4/27
Customer Server	NIC	198.0.0.10

Set up Customer Server as a web server and DNS Server.

Week 5: Iterative Socket Programming in UDP and TCP

Gain practical experience with socket programming, understand IPC principles with Unix TCP and UDP sockets, and learn to develop network programs.

INTRODUCTION TO WIRESHARK

Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible.

Purposes:

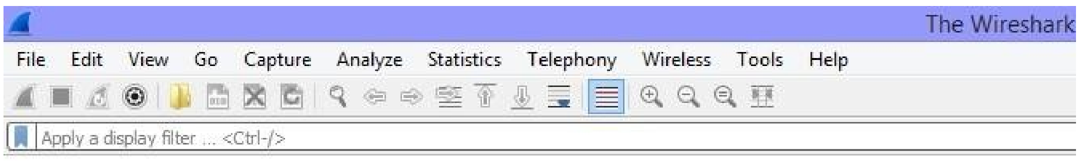
• Network administrators use it to troubleshoot network problems.

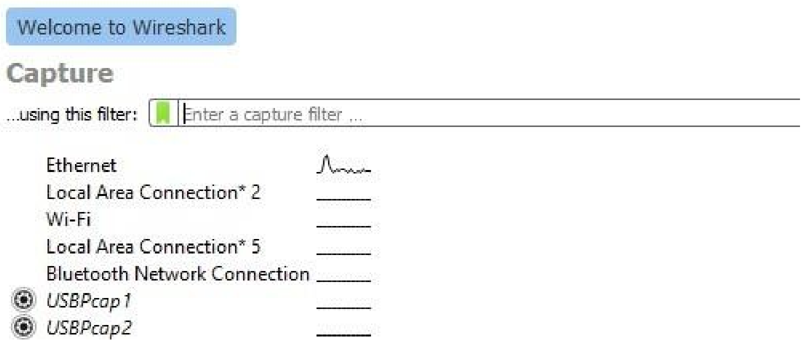
• Network security engineers use it to examine security problems.

• Developers use it to debug protocol implementations.

• People use it to learn network protocol internals.

Getting started with Wireshark:





• Choose the interface as "Ethernet". Once an interface is selected, if the computer is connected to the network and if there is some network traffic, the main window will be displayed. Click on the "start capturing packets" button on the main toolbar.

The capture is split into 3 parts:

1. Packet List Panel – this is a list of packets in the current capture. It colors the packets based on the protocol type. When a packet is selected, the details are shown in the two panels below.

2. Packet Details Panel – this shows the details of the selected packet. It shows the different protocols making up the layers of data for this packet. Layers include Frame, Ethernet, IP, TCP/UDP/ICMP, and application protocols such as HTTP.

3. Packet Bytes Panel – shows the packet bytes in Hex and ASCII encodings.

Wireshark uses two types of filters, capture filters and display filters. Capture filters are used to decide which packets should be kept. Only packets that meet filter criteria will be kept. Display filters work after the capture is completed. They restrict which packets are shown, but they don't discard any information. Capture filters would be more useful on very busy networks when you need to limit the amount of data your machine needs to process. On the other hand, display filters don't save any memory; display filters let you temporarily focus on an analysis without losing any underlying information.

Capture filters can be set in two different places. Go to the Capture menu and select "Options" and you will find a selection for capture filters. Alternatively, go to the Capture menu and select "Capture Filters". From the "Capture Filters" dialog box you will see a help menu that will explain how the function works. Display filters can be entered at the top of the display screen.

Display filters can be set as: Right click on the Source IP address field in the Packet Details Panel. Select Prepare a Filter->Selected. Wireshark automatically generates a Display Filter and applies it to the capture. The filter is shown in the Filter Bar, below the button toolbar. Only packets captured with a Source IP address of the value selected should be displayed. This same process can be performed on most fields within Wireshark and can be used to include or exclude traffic.

INTRODUCTION TO SOCKETS

Sockets allow communication between two different processes on the same or different machines. To be more precise, it's a way to talk to other computers using standard Unix file descriptors. In Unix, every I/O action is done by writing or reading a file descriptor. A file descriptor is just an integer associated with an open file and it can be a network connection, a text file, a terminal, or something else. To a programmer, a socket looks and behaves much like a low-level file descriptor. This is because commands such as read() and write() work with sockets in the same way they do with files and pipes.

Types of Sockets

There are four types of sockets available to the users. The first two are most commonly used and the last one is rarely used.

• **Stream Sockets:** Delivery in a networked environment is guaranteed. If you send through the stream socket three items "A, B, C", they will arrive in the same order – "A, B, C". These sockets use TCP (Transmission Control Protocol) for data transmission. If delivery is impossible, the sender receives an error indicator. Data records do not have any boundaries.

• **Datagram Sockets:** Delivery in a networked environment is not guaranteed. They're connectionless because you don't need to have an open connection as in Stream Sockets -you build a packet with the destination information and send it out. They use UDP (User Datagram Protocol).

• **Raw Sockets:** These provide users access to the underlying communication protocols, which support socket abstractions. Raw sockets are not intended for the general user; they have been provided mainly for those interested in developing new communication protocols, or for gaining access to some of the more cryptic facilities of an existing protocol.

Types of Servers

There are two types of servers you can have:

• **Iterative Server:** This is the simplest form of server where a server process serves one client and after completing the first request, it takes request from another client. Meanwhile, another client keeps waiting.

• **Concurrent Servers:** This type of server runs multiple concurrent processes to serve many requests at a time because one process may take longer and another client cannot wait for so long. The simplest way to write a concurrent server under Unix is to fork a child process to handle each client separately.

How to implement a client

The system calls for establishing a connection are somewhat different for the client and the server, but both involve the basic construct of a socket. Both the processes establish their own sockets. The steps involved in establishing a socket on the client side are as follows:

• Create a socket with the *socket()* system call.

• Connect the socket to the address of the server using the *connect()* system call.

• Send and receive data. There are a number of ways to do this, but the simplest way is to use the *read()* and *write()* system calls.

How to implement a Server Program in C using Linux APIs?

The steps involved in establishing a socket on the server side are as follows:

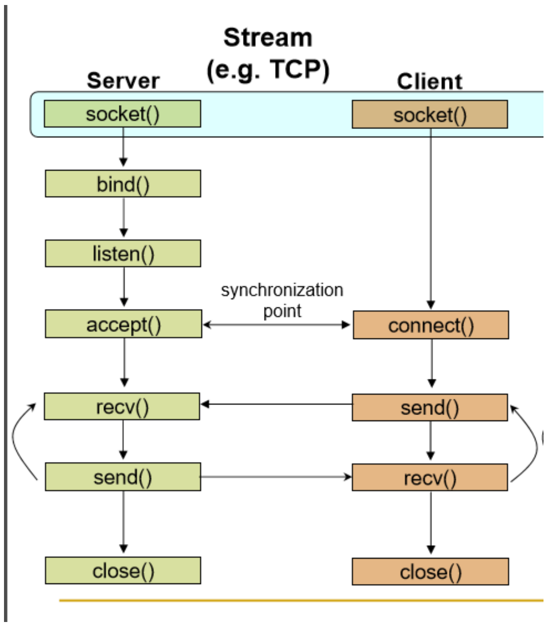
• Create a socket with the *socket()* system call.

• Bind the socket to an address using the *bind()* system call. For a server socket on the Internet, an address consists of a port number on the host machine.

• Listen for connections with the *listen()* system call.

• Accept a connection with the *accept()* system call. This call typically blocks the

• connection until a client connects with the server. Send and receive data using the *read()* and *write()* system calls.



TCP client server interactions

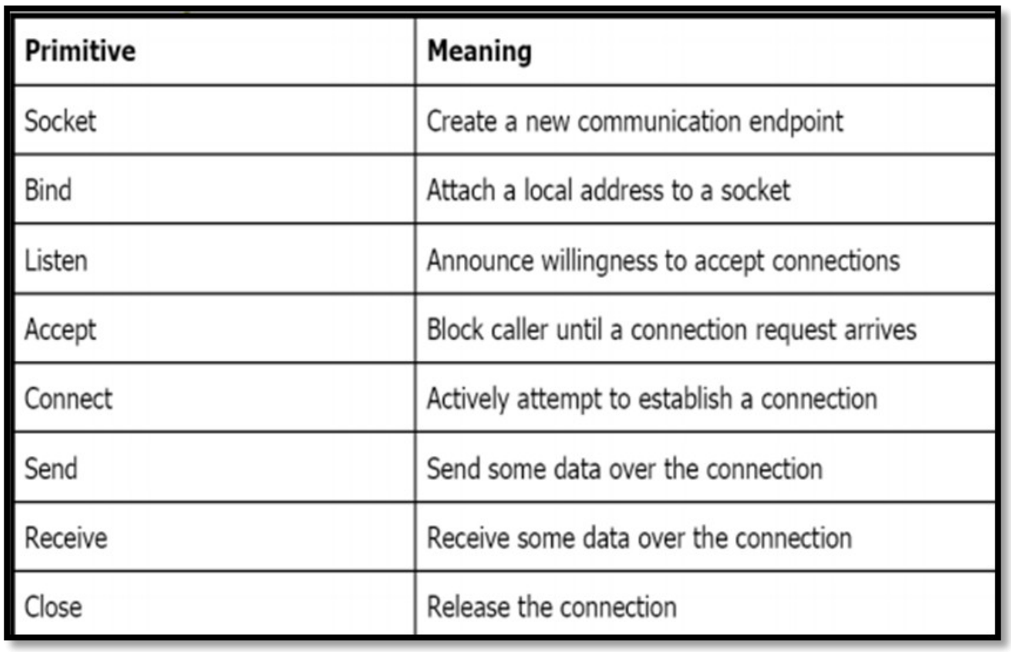
Basic data structures used in Socket programming:

Various structures are used in Unix Socket Programming to hold information about the address and port, and other information. Most socket functions require a pointer to a socket address structure as an argument. Structures defined in this chapter are related to Internet Protocol Family.

o **Socket Descriptor :** A simple file descriptor in Unix. Data type is integer.

o **Socket Address:** This construct holds the information for socket address.

System calls used in socket programming



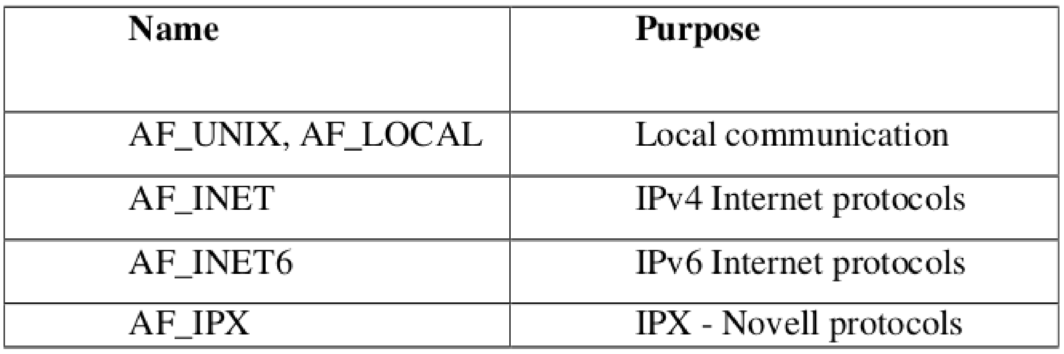
Primitive	**Meaning**
Socket	Create a new communication endpoint
Bind	Attach a local address to a socket
Listen	Announce willingness to accept connections
Accept	Block caller until a connection request arrives
Connect	Actively attempt to establish a connection
Send	Send some data over the connection
Receive	Receive some data over the connection
Close	Release the connection

Syntax:

```
struct sockaddrs {
unsigned short sa_family; // address family, AF_xxx or //PF_xxx
char sa_data[14];    // 14 bytes of protocol address
};
```

o AF stands for Address Family and PF stands for Protocol Family.

Table: Address Family



Name	**Purpose**
AF_UNIX, AF_LOCAL	Local communication
AF_INET	IPv4 Internet protocols
AF_INET6	IPv6 Internet protocols
AF_IPX	IPX - Novell protocols

o **struct sockaddr_in**

● This construct holds the information about the address family, port number, Internet address,and the size of the struct sockaddr.

struct sockaddr_in

{

short int sin_family; // Address family unsigned short int sin_port; // Port number

struct in_addr sin_addr; // Internet address

};

o The IP address structure, in_addr, is defined as follows

struct in_addr

{

unsigned long int s_addr;

};

Some of the System Calls Used for Conversion

Some systems (like x8086) are Little Endian i.e., least significant byte is stored in the higher address, whereas in Big-Endian systems most significant byte is stored in the higher address. Consider a situation where a Little-Endian system wants to communicate with a Big Endian one, if there is no standard for data representation then the data sent by one machine is misinterpreted by the other. So the standard has been defined for the data representation in the network (called Network Byte Order) which is the Big Endian.

The system calls that help us to convert a short/long from Host Byte order to Network Byte Order and vice versa are:

htons() -- "Host to Network Short"

htonl() -- "Host to Network Long"

ntohs() -- "Network to Host Short"

ntohl() -- "Network to Host Long"

To ensure correct byte ordering of the 16-bit port number, your server and client need to apply these functions to the port address.

For example

server_address.sin_addr.s_addr= htonl(INADDR_ANY);

server_address.sin_port = htons(9734);

IP address is a 32bit integer-not convenient for humans. So, the address is written in dotted

decimal representation.

inet_addr()

converts the Internet host address from the standard numbers-and-dots

notation into binary data. It returns nonzero if the address is valid, zero if not.

inet_aton() is also used for same purpose.

System calls used:

1. Socket creation in C using socket()

int sockid = socket(family, type, protocol);

- *sockid* is socket descriptor, an integer (like a file-handle)
- *family* is the communication domain, like PF_INET for IPv4 protocols and Internet addresses or PF_UNIX for Local communication and File addresses.
- *type* defines communication type such as SOCK_STREAM or SOCK_DGRAM.
- *protocol* specifies protocol used. It take values like IPPROTO_TCP or IPPROTO_UDP but usually set to 0 (i.e., use default protocol).

If the return value sockid is negative values, it means there is problem in socket creation. NOTE: socket call does not specify where data will be coming from, nor where it will be

going to – it just creates the interface!

2. Assign address to socket using bind(). bind() associates and reserves a port for use by the socket.

int status = bind(sockid, &addrport, size);

- *sockid* is a integer describing socket descriptor

- *addrport* is struct sockaddr which contains the (IP) address and port of the machine „ for TCP/IP server, internet address is usually set to INADDR_ANY, i.e., chooses any incoming interface size specifies the size (in bytes) of the addrport structure
- *status* will be assigned -1 returns on failure.

3. Listening to connection requests using *listen()*. This system call instructs TCP protocol implementation to listen for connections.

int status = listen(sockid, queueLimit);

- *sockid* is socket descriptor which is created using *socket()*
- *queueLimit* is an integer which specifies number of active participants that can "wait" for a connection
- *status* will be assigned -1when returns on failure.

Note: The listening socket (sockid) is never used for sending and receiving. It is used by the server only as a way to get new sockets.

4. Establish Connection using *connect()*. The client establishes a connection with the server by calling *connect()*.

int status = connect(sockid, &foreignAddr, addrlen);

- *sockid* is socket descriptor to be used in connection
- *foreignAddr* is *struct sockaddr* which contains address of the passive participant
- *addrlen* is sizeof(*foreignAddr*)
- *status* will be assigned -1 when returns on failure

Note: connect() is blocking where as listen() is non blocking.

5. *Accept incoming Connection using accept().*

The server gets a socket for an incoming client connection by calling accept()

int newsockid = accept(sockid, &clientAddr, &addrLen);

- *newsockid* is an integer, the new socket is created in server which is client specific and this new socket is used for data-transfer between server and client.
- *sockid* is the socket created using socket system call, which is used only to listen to incoming requests from clients.

- *clientAddr* is in the form of *struct sockaddr*, address of the active participant.
- *addrLen* is size of *clientAddr* parameter.

Note: accept() is blocking, it waits for connection before returning and dequeues the next connection on the queue for socket (sockid).

6. Exchanging data with stream socket

Application running in server and client(s) can transfer data using send() and receive() system call.

int count = send(sockid, msg, msgLen, flags);

- *sockid* is the new socket descriptor created by accept in server side and socket in client side, depending on where it is used.
- *msg* is an array holding message to be transmitted.
- *msgLen* holds length of message (in bytes) to transmit.
- *flags* are integer, special options, usually set 0
- return value count has number of bytes transmitted and is set to -1 on error

Syntax:

int count = recv(sockid, recvBuf, bufLen, flags);

- *recvBuf* stores received message
- *bufLen* holds number if bytes

7. Closing the socket using close()

When finished using a socket, the socket should be closed.

int status= close(sockid);

- sockid: the file descriptor (socket being closed)
- status: 0 if successful, -1 if error

Closing a socket closes a connection (for stream socket) and frees up the port used by the socket.

Sample Program: Write an iterative TCP client server program where client sends a message to server and server echoes back the message to client. Client should display the original message and echoed message. Note: As socket is also a file descriptor, we can use read and write system calls to receive and send data.

// TCP Server code:

// Make the necessary includes and set up the variables:

```
#include<stdio.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#define PORTNO 10200
int main()
{
int sockfd,newsockfd,portno,clilen,n=1;
struct sockaddr_in seraddr,cliaddr;
int i,value;
// create an unnamed socket for the server
sockfd = socket(AF_INET,SOCK_STREAM,0);
//Name the socket
seraddr.sin_family = AF_INET;
seraddr.sin_addr.s_addr = inet_addr("172.16.59.10");// **
seraddr.sin_port = htons(PORTNO);
bind(sockfd,(struct sockaddr *)&seraddr,sizeof(seraddr));
//Create a connection queue and wait for clients
listen(sockfd,5);
while (1) {
char buf[256];
printf("server waiting");
//Accept a connection
clilen = sizeof(clilen);
newsockfd=accept(sockfd,(struct sockaddr *)&cliaddr,&clilen);
//Read and write to client on client_sockfd (Logic for problem mentioned here)
n = read(newsockfd,buf,sizeof(buf));
printf(" \nMessage from Client %s \n",buf);
n = write(newsockfd,buf,sizeof(buf));
}
}
```

**- indicates replace this address with your systems IP address

```
//TCP Client Code:
//Make the necessary includes and set up the variables
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
int main()
{
int len,result,sockfd,n=1;
struct sockaddr_in address;
char ch[256],buf[256];
//Create a socket for the client
sockfd = socket(AF_INET, SOCK_STREAM, 0);
//Name the socket as agreed with the server
address.sin_family=AF_INET;
address.sin_addr.s_addr=inet_addr("172.16.59.10"); **
address.sin_port=htons(10200);
len = sizeof(address);
//Connect your socket to the server’s socket
result=connect(sockfd,(struct sockaddr *)&address,len);
if(result==-1)
{
perror("\nCLIENT ERROR");
exit(1);
}
//You can now read and write via sockfd (Logic for problem mentioned here)
printf("\nENTER STRING\t");
gets(ch);
ch[strlen(ch)]='\0';
write(sockfd,ch,strlen(ch));
printf("STRING SENT BACK FROM SERVER IS ..... ");
while(n){
```

```
n=read(sockfd,buf,sizeof(buf));
puts(buf);
}
}
```

**- indicates replace this address with your systems IP address

Steps to execute the program.

1. Open two terminal windows and open a text file from each terminal with .c extension using

command:

```
$ gedit filename.c
```

2. Type the client and server program in separate text files and save it before exiting the text

window.

3. First compile and run the server using commands mentioned below

```
$ gcc filename –o executablefileName //renaming the a.out file
$ ./ executablefileName
```

4. Compile and run the client using the same instructions as listed in Step 3.

Note: The ephemeral port number has to be changed every time the program is executed.

Introduction to User Datagram Protocol

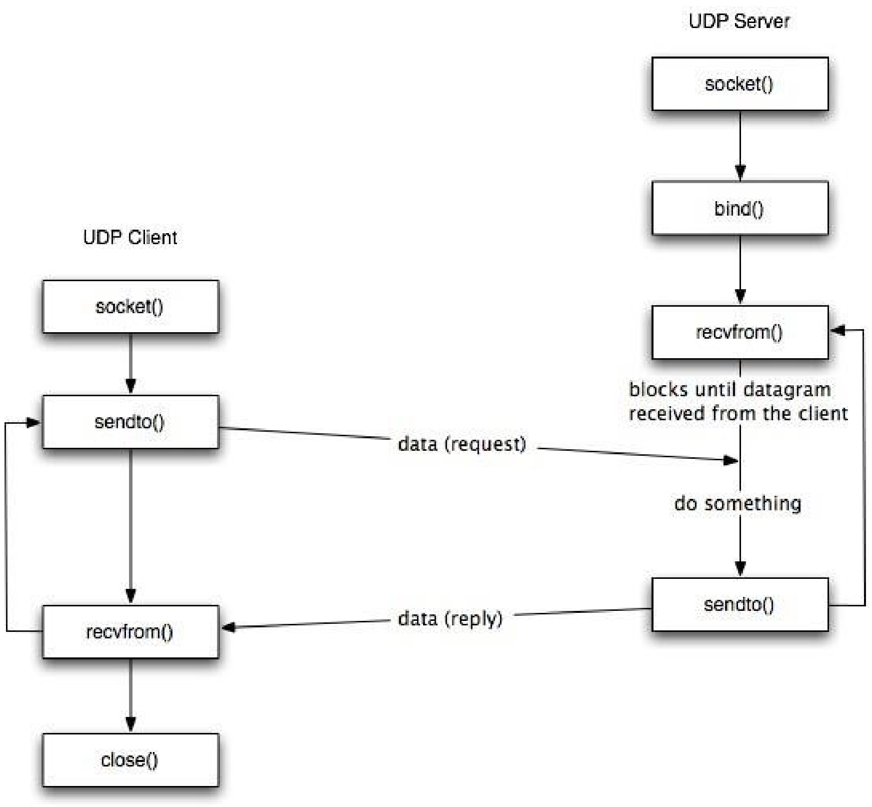


Figure: Interaction between UDP client and Server.

System calls used in UDP:

1. Socket creation using socket() : Only difference in this call is its second parameter. For TCP, we use stream sockets as data is transferred in the form of the stream whereas in UDP, data is transmitted in the form of datagrams. So, the type of socket used for UDP transmission is DATAGRAM socket. The system call should be used as follows.

sockfd = socket(AF_INET, SOCK_DGRAM, 0);

2. Exchange of data using sendto() and recvfrom(): As UDP does not establish the connection, while sending datagram, it should specify the address of the receiver.

int sendto(int sockfd, const void *msg, int len, unsigned int flags,const struct sockaddr *to, int tolen);

- sockfd: It is a socket descriptor returned by the socket function.
- msg: It is a pointer to the data you want to send.
- len: It is the length of the data you want to send (in bytes).
- flags: It is set to 0.

- to: It is a pointer to struct sockaddr for the host where data has to be sent.
- tolen: It is set it to sizeof(struct sockaddr).

Return value will be number of bytes sent or -1 for error. Similarly for receiving data, recvfrom is used and syntax is as follows.

int recvfrom(int sockfd, void *buf, int len, unsigned int flags,struct sockaddr *from, int

****fromlen);***

- sockfd: It is a socket descriptor returned by the socket function.
- buf: It is the buffer to read the information into.
- len: It is the maximum length of the buffer.
- flags: It is set to 0.
- from: It is a pointer to struct sockaddr for the host where data has to be read.
- fromlen: It is set it to sizeof(struct sockaddr).

Return value will be number of bytes recieved or -1 for error.

```
//UDP Server Program:
#include<stdio.h>
#include<fcntl.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<unistd.h>
int main()
{
int sd;
char buf[25];
struct sockaddr_in sadd,cadd;
//Create a UDP socket
sd=socket(AF_INET,SOCK_DGRAM,0);
//Construct the address for use with sendto/recvfrom... */
sadd.sin_family=AF_INET;
sadd.sin_addr.s_addr=inet_addr("172.16.56.10");//**
sadd.sin_port=htons(9704);
int result=bind(sd,(struct sockaddr *)&sadd,sizeof(sadd));
int len=sizeof(cadd);
int m=recvfrom(sd,buf,sizeof(buf),0,(struct sockaddr *)&cadd,&len);
printf("the server send is\n");
puts(buf);
int n=sendto(sd,buf,sizeof(buf),0,(struct sockaddr *)&cadd,len);
return 0;
}
```

**- indicates replace this address with your systems IP address

```
//UDP Client program
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<unistd.h>
int main()
{
int sd;
struct sockaddr_in address;
sd=socket(AF_INET,SOCK_DGRAM,0);
address.sin_family=AF_INET;
address.sin_addr.s_addr=inet_addr("172.16.56.10");//**
address.sin_port=htons(9704);
char buf[25],buf1[25];
printf("enter buf\n");
gets(buf);
int len=sizeof(address);
int m=sendto(sd,buf,sizeof(buf),0,(struct sockaddr *)&address, len);
int n=recvfrom(sd,buf,sizeof(buf),0,(struct sockaddr *)&address,&len);
printf("the server echo is\n");
puts(buf);
return 0;
}
```

**- indicates replace this address with your systems IP address

Lab Exercises:

1. Write an iterative TCP client-server program where the client accepts a sentence from the user and sends it to the server. The server will check for duplicate words in the string. Server will find number of occurrences of duplicate words present and remove the duplicate words by retaining single occurrence of the word and send the resultant sentence to the client. The client displays the received data on the client screen. The process repeats until the user enter the string "Stop". Then both the processes terminate.

2. Write an iterative UDP client-server program where the client sends rows of a matrix, and the server combines them together as a matrix.

Additional Exercises:

1. To illustrate encryption and decryption of messages using TCP. The client accepts messages to be encrypted through standard input device. The client will encrypt the string by adding 4 (random value) to ASCII value of each alphabet. The encrypted message is sent to the server. The server then decrypts the message and displays both encrypted and decrypted forms of the string. Program terminates after one session.

2. Write a client program to send a manually crafted HTTP request packet to a Web Server and display all fields received in HTTP Response at client Side.

Week 6: Concurrent Socket Programming in TCP

Learn to implement concurrent servers capable of handling multiple client requests simultaneously.

. Introduction to Socket Programming in ‘C’ using TCP/IP – Concurrent Servers There might be a need to consider the case of multiple, simultaneous clients connecting to a server. The fact that the original socket is still available and that sockets behave as file descriptors gives you a method of serving multiple clients at the same time. If the server calls fork to create a second copy of itself, the open socket will be inherited by the new child process. It can then communicate with the connecting client while the main server continues to accept further client connections. This is, in fact, a fairly easy change to make to your server program, which is shown in the following.

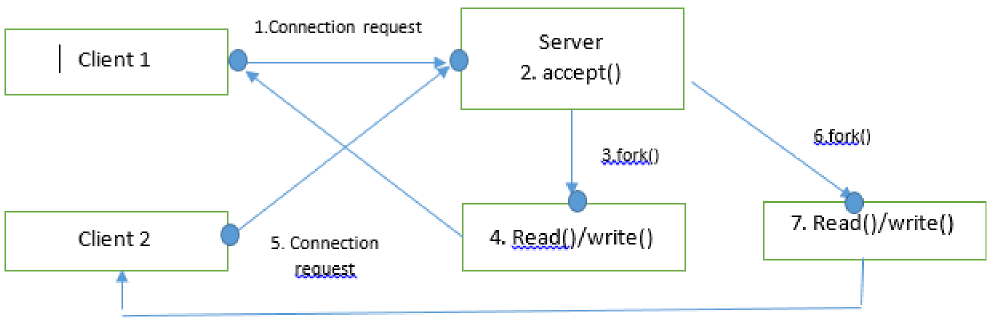


Figure: TCP concurrent server and clients interactions

Function Description: fork() The fork command creates a new separate process for each client. The fork() command splits the current process into two processes: a parent and a child. The new process (child process) is an almost exact copy of the process that calls it (the parent process). The fork() command returns 0 when called in the child process, returns the process ID of the newly created (child) process when called in the parent process, and –1 on error. Therefore, the return value of the function call to fork() tells the process whether it is the parent or the child. For a parent to keep track of its children, it should record the return values from call to fork(). (Note: If it is desired to get the process ID of the parent, the child can obtain it by calling getppid command.) From this point one can easily program the child process to serve the client’s request while the parent can keep accepting other requests. However, when a child finishes and exits it needs to notify the parent that it is done This is where the waitpid() command comes to screen.(Please do a man page on this function to learn more about it.)

We have seen how fork()can be used to handle multiple clients. But forking a new process is expensive, it duplicates the entire state (memory, stack, file/socket descriptors …).Threads decrease this cost by allowing multitasking within the same process. Both process and thread incurs overhead as they include creation, scheduling and context switching and also as their numbers increases, this overhead increases.

Example : fork()

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
int main(void)
{
pid_t pid = fork();
if(pid == 0) {
printf("Child => PPID: %d PID: %d\n", getppid(), getpid());
exit(EXIT_SUCCESS);
}
else if(pid > 0) {
printf("Parent => PID: %d\n", getpid());
printf("Waiting for child process to finish.\n");
wait(NULL);
printf("Child process finished.\n");
}
else {
printf("Unable to create child process.\n");
}
return 0;
}
```

Output:

```
Parent => PID: 229
Waiting for the child process to finish.
Child => PPID: 229 PID: 230
Child process finished.
```

```
//TCP Server Program
```

```
#include<stdio.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#define PORTNO 10200
int main()
{
int sockfd,newsockfd,portno,clilen,n=1;
char buf[256];
struct sockaddr_in seraddr,cliaddr;
int i,value;
sockfd = socket(AF_INET,SOCK_STREAM,0);
seraddr.sin_family = AF_INET;
seraddr.sin_addr.s_addr = inet_addr("172.16.59.10"); //**
seraddr.sin_port = htons(PORTNO);
bind(sockfd,(struct sockaddr *)&seraddr,sizeof(seraddr));
// Create a connection queue, ignore child exit details, and wait for clients
listen(sockfd,5);
while(1){
//Accept the connection
clilen = sizeof(clilen);
newsockfd=accept(sockfd,(struct sockaddr *)&cliaddr,&clilen);
//Fork to create a process for this client and perform a test to see whether
//you're the parent or the child:
if(fork()==0){
// If you're the child, you can now read/write to the client on newsockfd.
n = read(newsockfd,buf,sizeof(buf));
printf(" \nMessage from Client %s \n",buf);
n = write(newsockfd,buf,sizeof(buf));
close(newsockfd);
exit(0);
}
//Otherwise, you must be the parent and your work for this client is finished
else
close(newsockfd);
}
}
```

**- indicates replace this address with your systems IP address

```
//TCP Client program
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<unistd.h>
int main()
{
int sd;
struct sockaddr_in address;
sd=socket(AF_INET,SOCK_DGRAM,0);
address.sin_family=AF_INET;
address.sin_addr.s_addr=inet_addr("172.16.56.10");//**
address.sin_port=htons(9704);
char buf[25],buf1[25];
printf("enter buf\n");
gets(buf);
int len=sizeof(address);
int m=sendto(sd,buf,sizeof(buf),0,(struct sockaddr *)&address, len);
int n=recvfrom(sd,buf,sizeof(buf),0,(struct sockaddr *)&address,&len);
printf("the server echo is\n");
puts(buf);
return 0;
}
```

**- indicates replace this address with your systems IP address

Steps for execution

1. Open three terminals
2. Terminal 1: $gcc server.c –o server
./server
3. Terminal 2 : $gcc client.c –o client1
./client1
4. Terminal 3: $gcc client.c –o client2
./client2

Lab Exercises:

1. Implement concurrent Remote Math Server To perform arithmetic operations in the server and display the result to the client. The client accepts two integers and an operator from the user and sends it to the server. The server then receives integers and operator. The server will perform the operation on integers and sends the result back to the client which is displayed on the client screen. Then both the processes terminate.

2. Write a concurrent TCP client server ‘C’ program where the client accepts a sentence from the user and sends it to the server. The server will check for duplicate words in the string. Server will find the number of occurrences of duplicate words present and remove the duplicate words by retaining single occurrence of the word and send the resultant sentence to the client.

Additional Exercises:

1. Write a concurrent TCP daytime server ‘C’ program. Along with the result, server should also send the process id to the client.

Week 7: Socket Programming Applications

The objective of this lab is to introduce students to the practical applications of socket programming.

Scenario:
1. You are tasked with developing a secure communication system where a client and a server exchange a symmetric encryption key securely over a network. The exchanged key will then be used to encrypt and decrypt messages between the client and server.

2. Implement a basic chat application where multiple clients can send and receive messages through a server.

Server Requirements:
1. The server should be able to accept multiple client connections.
2. The server should broadcast received messages to all connected clients.
3. Each client should be able to join the chat by connecting to the server and should see all messages sent by other clients.

Client Requirements:
1. Each client should be able to connect to the server.
2. Clients should be able to send messages to the server.
3. Clients should be able to receive and display messages from other clients.

Week 8: Advanced IP Configuration Lab: Subnetting, Static Routing

Students will learn to design and configure an IP addressing scheme for a network based on specific requirements, focusing on subnetting. They will also implement and verify static routing to enable communication between different subnets.

IP Address Classes

IP addresses are divided into classes to accommodate different sizes of networks. The class of an IP address determines its default subnet mask.

Classful Addressing

1. **Class A**:
   - **Range**: 1.0.0.0 to 126.0.0.0
   - **Default Subnet Mask**: 255.0.0.0 (/8)
   - **Network Bits**: 8
   - **Host Bits**: 24
   - **Number of Networks**: 128 (2^7 - 2 for network and broadcast addresses)
   - **Number of Hosts per Network**: 16,777,214 (2^24 - 2 for network and broadcast addresses)
2. **Class B**:
   - **Range**: 128.0.0.0 to 191.255.0.0
   - **Default Subnet Mask**: 255.255.0.0 (/16)
   - **Network Bits**: 16
   - **Host Bits**: 16
   - **Number of Networks**: 16,384 (2^14 - 2 for network and broadcast addresses)
   - **Number of Hosts per Network**: 65,534 (2^16 - 2 for network and broadcast addresses)
3. **Class C**:
   - **Range**: 192.0.0.0 to 223.255.255.0
   - **Default Subnet Mask**: 255.255.255.0 (/24)
   - **Network Bits**: 24
   - **Host Bits**: 8
   - **Number of Networks**: 2,097,152 (2^21 - 2 for network and broadcast addresses)

    - **Number of Hosts per Network**: 254 (2^8 - 2 for network and broadcast addresses)
4. **Class D (Multicast)**:
    - **Range**: 224.0.0.0 to 239.255.255.255
    - **Not used for subnetting or host addressing**
5. **Class E (Experimental)**:
    - **Range**: 240.0.0.0 to 255.255.255.255
    - **Not used for subnetting or host addressing**

Subnetting is the process of dividing a larger network into smaller, more manageable sub-networks (subnets). This segmentation improves network performance and security by limiting broadcast domains and organizing IP address allocation.

Classless Addressing (CIDR)

Classless Inter-Domain Routing (CIDR) allows for more flexible allocation of IP addresses than the fixed class system. Instead of being limited to the fixed classful subnet masks, CIDR uses variable-length subnet masks (VLSM) to allocate IP addresses more efficiently.

CIDR Notation:

- The IP address is followed by a slash (/) and the number of bits used for the network portion.
- Example: 192.168.1.0/24 means the first 24 bits are the network portion, leaving the remaining 8 bits for host addresses.

Detailed Subnetting Explanation

1. Subnet Mask Calculation

A subnet mask is a 32-bit number that masks an IP address, dividing it into the network and host addresses. It consists of a series of consecutive 1s followed by consecutive 0s. The 1s represent the network part, and the 0s represent the host part.

Example:

- Subnet mask of 255.255.255.0 (or /24 in CIDR notation) means the first 24 bits are the network part, and the last 8 bits are for hosts.

2. Calculating Subnet Masks

To determine the subnet mask, consider the number of required hosts per subnet. The formula to calculate the number of hosts is:

Number of hosts=2n−2\text{Number of hosts} = 2^n - 2Number of hosts=2n−2

Where nnn is the number of bits for hosts.

Example:

- For 25 hosts, you need at least 5 bits (since 25−2=302^5 - 2 = 3025−2=30). This gives a subnet mask of 255.255.255.224 (/27).

3. Determining Subnet Addresses

The subnet address is calculated by applying the subnet mask to the base network address. This can be done by performing a bitwise AND operation between the IP address and the subnet mask.

Example:

- Base network address: 192.168.0.0/16
- Subnet mask: 255.255.255.224 (/27)
- The first subnet address: 192.168.0.0
- The second subnet address: 192.168.0.32

4. Determining the Number of Subnets and Hosts per Subnet

To determine how many subnets and hosts per subnet are required, you can borrow bits from the host portion of the address to create additional subnets.

Steps:

1. **Identify the Base Network**:
   - Example: 192.168.0.0/24
2. **Determine the Number of Subnets Needed**:
   - If you need 4 subnets, you need at least 2 bits (since 22=42^2 = 422=4).
3. **Calculate the New Subnet Mask**:
   - Borrow 2 bits from the host portion: /24 + 2 = /26.
   - New subnet mask: 255.255.255.192.
4. **Calculate the Number of Hosts per Subnet**:
   - Remaining bits for hosts: 32 - 26 = 6 bits.
   - Number of hosts per subnet: 26−2=622^6 - 2 = 6226−2=62.
5. **Creating Subnet Ranges**

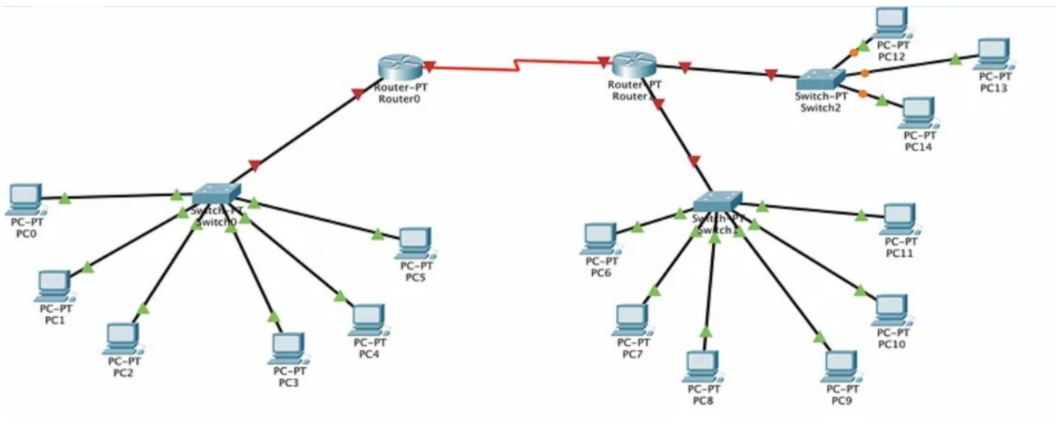
Once you have the new subnet mask, you can create the subnet ranges.

Example:

- For 192.168.0.0/26:
  - Subnet 1: 192.168.0.0 - 192.168.0.63
  - Subnet 2: 192.168.0.64 - 192.168.0.127
  - Subnet 3: 192.168.0.128 - 192.168.0.191
  - Subnet 4: 192.168.0.192 - 192.168.0.255

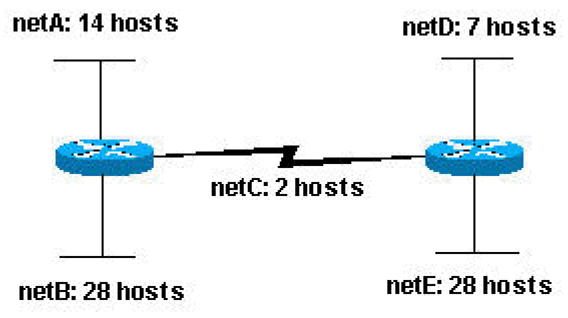
Lab Exercise 1

As a network Engineering, you are required to subnet the 192.168.1.0 network to provide IP addresses to the given network. Show all calculations. What is the Then configure all devices and ensure that PCs can ping each other.



Lab Exercise 2:

You are a network technician assigned to design and implement a new network for a client. Given the Class C network of 192.168.5.0/24, subnet the network to create the network as shown with the host requirements shown. Show sample hosts and ensure connectivity.



Additional Exercise

You are a network technician assigned to design and implement a new network for a client. The client has a Class C network of 192.168.5.0/24 and needs to create multiple subnets to meet the following requirements:

1. **LAN-A** connected to **Router0** needs to support a minimum of 30 hosts.
2. **LAN-B** connected to **Router1** needs to support a minimum of 20 hosts.
3. **LAN-C** connected to **Router2** needs to support a minimum of 10 hosts.
4. Each router will be interconnected with serial links.

5. Ensure there are sufficient subnets for future expansion and additional connections.

Requirements:

- Subnet the 192.168.5.0/24 network to meet the given requirements.
- Assign IP addresses to each device. Assign IP addresses to the routers' interfaces connected to the LANs.
- Ensure that each LAN has enough IP addresses for the required hosts.
- Verify connectivity between all devices with static routing.

Network Layout:

1. **Router0**:
   - LAN-A
   - Serial link to Router1
   - Serial link to Router2
2. **Router1**:
   - LAN-B
   - Serial link to Router0
   - Serial link to Router2
3. **Router2**:
   - LAN-C
   - Serial link to Router0
   - Serial link to Router1

Questions:

1. **Subnet Calculation:**
   - How many subnets are needed to meet the requirements?
   - What is the subnet mask for each LAN?
   - List the subnets and their ranges.
2. **IP Address Assignment:**
   - What IP address will you assign to Router0's interface connected to LAN-A?
   - What IP address will you assign to Router1's interface connected to LAN-B?
   - What IP address will you assign to Router2's interface connected to LAN-C?
   - Provide sample IP addresses for at least two hosts in each LAN.
   - What IP addresses will you assign to the serial interfaces connecting the routers?

Week 9: VLAN Configuration and InterVLAN Communication

Configure VLANs on multiple switches to segment network traffic and enhance security. Set up inter-VLAN routing using a router-on-a-stick configuration to enable communication between different VLANs.

VLANs (Virtual Local Area Networks)

VLANs are a method of creating logically separate networks within a single physical network infrastructure. By grouping devices into VLANs, network administrators can improve security, manageability, and performance. Each VLAN is a distinct broadcast domain, meaning that broadcasts sent by a device in one VLAN are only received by devices in the same VLAN. This segmentation helps reduce unnecessary traffic on the network, which enhances efficiency. VLANs also offer security advantages by isolating sensitive data and systems, limiting the reach of potential threats within the network. For instance, an HR department could be on a separate VLAN from the IT department, ensuring that sensitive HR data is only accessible to authorized personnel.

Switch Ports: Switch ports are the physical interfaces on a network switch where devices like computers, printers, and other networked devices connect. These ports can be configured in different modes, the most common being access and trunk modes. Access ports are designed to handle traffic for a single VLAN. When a device connects to an access port, it is assigned to the VLAN configured on that port, making it part of that specific network segment. This is useful for segmenting different departments or groups within an organization. For example, all computers in the marketing department could be connected to access ports assigned to the marketing VLAN, ensuring they are part of the same network segment and can easily share resources.

Trunk Ports: Trunk ports play a crucial role in networks that use VLANs by allowing multiple VLANs to traverse a single physical link between switches or between a switch and a router. Unlike access ports, which belong to a single VLAN, trunk ports are configured to carry traffic for several VLANs simultaneously. This is achieved by tagging the traffic with a VLAN identifier, so each packet can be directed to the appropriate VLAN once it reaches its destination. Trunk ports are essential for maintaining VLAN continuity across the network infrastructure. For example, if two switches each have devices in multiple VLANs, a trunk link between them allows all VLANs to communicate across that single link, simplifying the cabling requirements and improving network efficiency.

Suppose we have made 2 logical groups of devices (VLAN) named sales and finance. If a device in the sales department wants to communicate with a device in the finance department, inter-VLAN routing has to be performed. These can be performed by either router or layer 3 switches.

Switch Virtual Interface (SVI): SVI is a logical interface on a multilayer switch that provides layer 3 processing for packets to all switch ports associated with that VLAN. A single SVI can be created for a VLAN. SVI on the layer 3 switch provides both management and routing services while SVI on layer 2 switch provides only management services like creating VLANs or telnet/SSH services.

Configuring VLANs on Cisco switches involves creating VLANs,

```
Switch(config)# vlan 3
Switch(config-vlan)# name Management
Switch(config-vlan)# exit
```

assigning switch ports to those VLANs,

```
Switch(config)# interface range f0/18
Switch(config-if-range)# switchport mode access
Switch(config-if-range)# switchport access vlan 4
Switch(config-if-range)# exit
```

and configuring trunk ports to allow VLAN traffic between switches.

```
Switch(config)# interface f0/1
Switch(config-if)# switchport mode trunk
```

Inter-VLAN Routing

Router-on-a-Stick is a method of implementing inter-VLAN routing using a single physical interface on a router. This approach is useful when you have multiple VLANs that need to communicate with each other, typically within a Layer 2 switch environment where the router does not have direct access to each VLAN.

In this configuration, a single router interface is connected to a switch port configured as a trunk port. The trunk port allows traffic from multiple VLANs to pass through a single physical connection between the switch and the router. VLAN tagging (802.1Q encapsulation) is used on the trunk port to distinguish traffic from different VLANs. Each frame traversing the trunk port is tagged with a VLAN ID, indicating its VLAN membership. On the router, the physical interface connected to the trunk port is divided into logical sub-interfaces, one for each VLAN. Each sub-interface is configured with an IP address belonging to the respective VLAN subnet. The router performs routing between these sub-interfaces, effectively enabling inter-VLAN communication.

Commands for Router-on-a-Stick Configuration (Cisco IOS):
1. Configure the physical interface on the router and enable it.

```
Router(config)# interface GigabitEthernet0/0   // Example interface, use appropriate interface
Router(config-if)# no shutdown
```

2. Create sub-interfaces for each VLAN that needs to communicate. Each sub-interface is associated with a VLAN ID and configured with an IP address from the VLAN subnet. Enable 802.1Q encapsulation on each sub-interface to match the VLAN tagging used on the trunk port.

```
Router(config)# interface GigabitEthernet0/0.3   // Sub-interface for VLAN 3 (for every VLAN)
Router(config-subif)# encapsulation dot1Q 3     // VLAN ID 3
Router(config-subif)# ip address 192.168.3.1 255.255.255.0   // IP address for VLAN 3
Router(config-subif)# exit
Router(config)# interface GigabitEthernet0/0.4   // Sub-interface for VLAN 4
```

3. Enable routing

```
Router(config)# ip routing
```

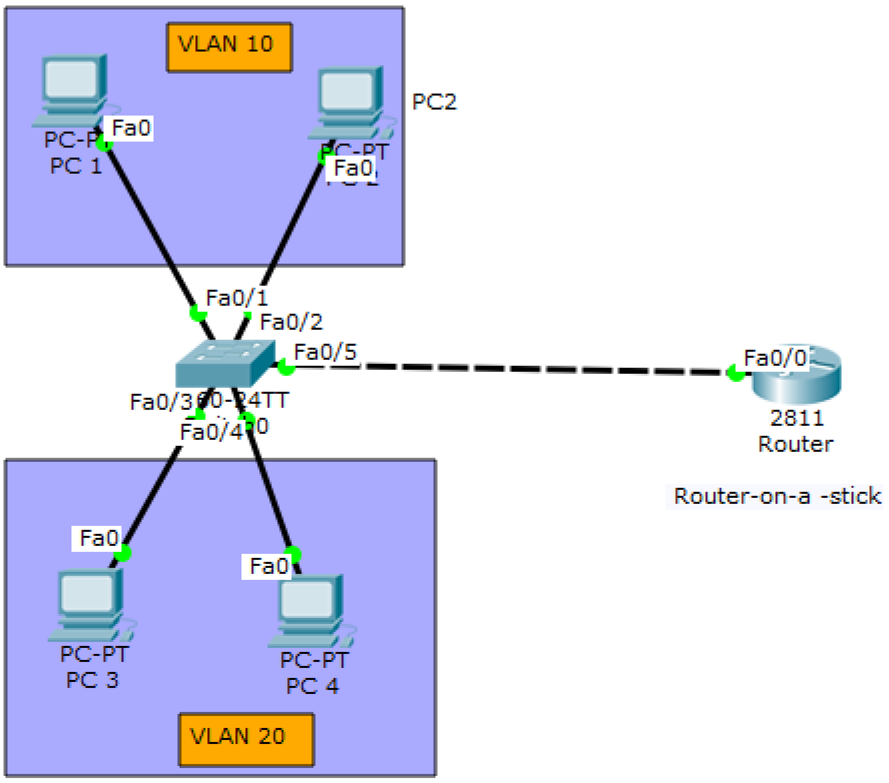
Physical Interface: The main physical interface on the router (e.g., GigabitEthernet0/0) remains up and is used for trunking VLAN traffic.
Sub-Interfaces: Each sub-interface is configured with a VLAN ID and IP address, effectively creating logical interfaces for each VLAN.
Encapsulation: 802.1Q encapsulation tags frames with VLAN IDs, allowing the router to distinguish between VLANs on a single physical interface.

Routing: The router performs inter-VLAN routing between the sub-interfaces, forwarding traffic based on IP routing table entries.

Router-on-a-Stick is a scalable and efficient method for handling inter-VLAN routing in environments with multiple VLANs and helps in reducing the number of physical interfaces needed on the router.

Lab Exercise 1

Create a topology as shown and Create 2 VLANs on the switch: VLAN 10 and VLAN 20. You can give them custom names.



Assign switch ports to the VLANs.

¬⅄ An *access port* is assigned to a single VLAN . These ports are configured for switch ports that connect to devices with a normal network card, for example a PC in a network.

¬⅄ A *trunk port* on the other hand is a port that can be connected to another switch or router. This port can carry traffic of multiple VLANs

Assign static IP addresses to the four PCs which are located in the separate VLANs. PC1 and PC2 fall in VLAN 10 while PC3 and PC4 fall in VLAN 20.
PC1 IP address 192.168.1.10 **Subnet mask** 255.255.255.0 **Default gateway** 192.168.1.1
PC2: IP address 192.168.1.20 **Subnet mask** 255.255.255.0 **Default gateway** 192.168.1.1
PC3: IP address 192.168.2.10 **Subnet mask** 255.255.255.0 **Default gateway** 192.168.2.1
PC4: IP address 192.168.2.20 **Subnet mask** 255.255.255.0 **Default gateway** 192.168.2.1

To test communication between hosts in the same VLAN:
Ping PC2 from PC1 both in VLAN 10. Ping test should be successful.
To test connectivity between hosts in different VLANs:
Ping PC3 in VLAN 20 from PC1 in VLAN 10. Ping here will definitely fail. Why?

PC1 in VLAN 10 needs to communicate with PC3 in VLAN 20. Facilitate inter-VLAN communication using Router-on-a-Stick.
Configure the router with Router-on-a-Stick:

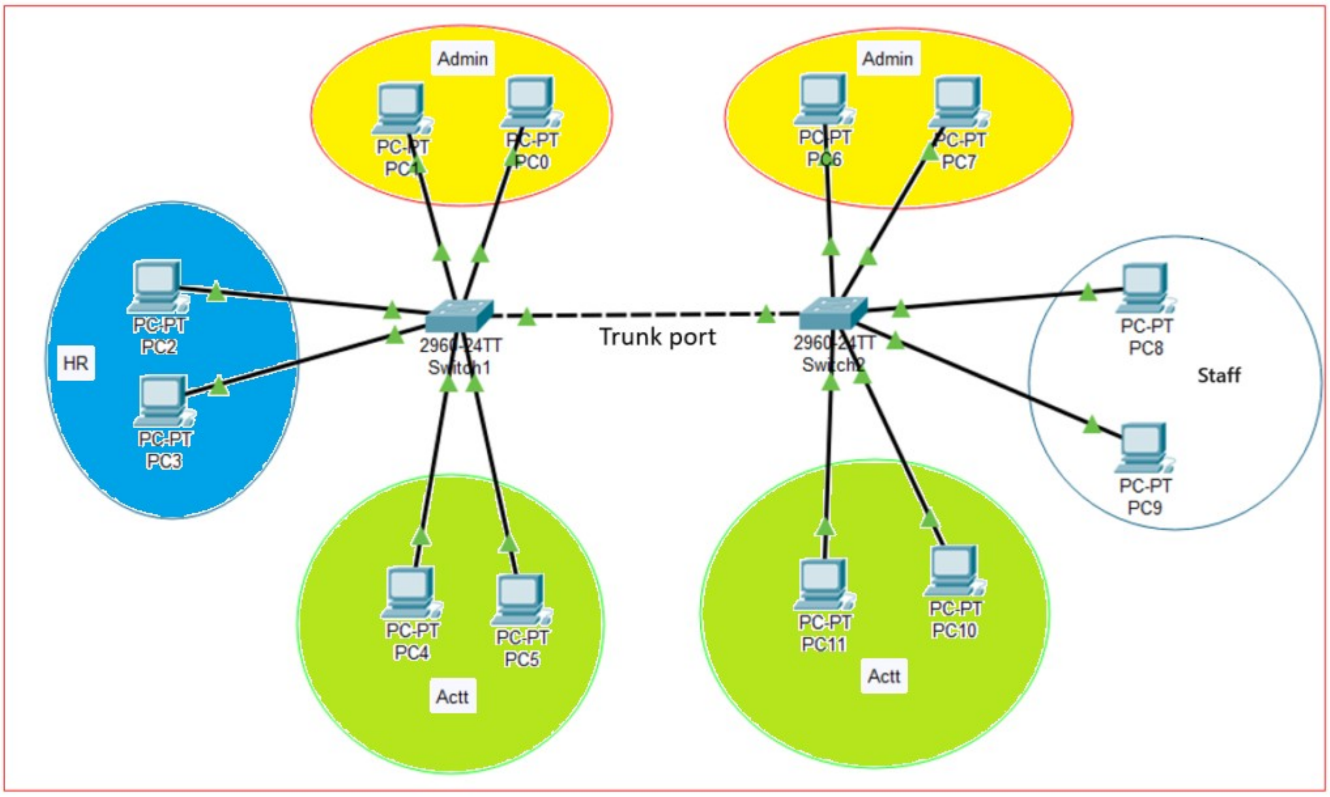
- Assign the switch port connect to the router as trunk
- Create subinterfaces on the router's physical interface connected to the switch for each VLAN.
- Assign IP addresses to each subinterface matching the respective VLANs:
  - Subinterface G0/0.10 (VLAN 10): IP Address - 192.168.1.1, Subnet Mask - 255.255.255.0
  - Subinterface G0/0.20 (VLAN 20): IP Address - 192.168.2.1, Subnet Mask - 255.255.255.0
- Enable routing in the router.

Retry the ping from PC1 (192.168.1.10) to PC3 (192.168.2.10). This time, the ping should be successful as the router will route traffic between VLAN 10 and VLAN 20 using the configured subinterfaces.

Additional Exercise:

You are a network administrator tasked with segmenting the network into VLANs for different departments within a company. The topology diagram provided shows the

network setup, which includes two switches (Switch1 and Switch2) and four departments: Staff, Admin, HR, and Accounting (Acct).
Create VLANs on Switch1 and Switch2 and Assign the appropriate ports to each VLAN based on the VLAN assignments. Test communication within the same VLAN. Test communication between different VLANs (will fail initially)
Provide inter-VLAN communication using Router-on-a-Stick between Acct, HR and Admin only.



Week 10: Dynamic Routing with RIP and Router configuration with DHCP / DNS

Implement dynamic routing using RIP and configure DHCP and DNS on a router to provide automatic IP configuration and name resolution services to network clients.

DNS Configuration on a Cisco Router

Configuring **DNS** on a router can also involve setting up the router to forward DNS requests to an external DNS server, caching DNS entries, and managing local domain names. Routers use DNS servers to translate domain names (like www.example.com) into IP addresses (like 192.0.2.1) that devices can use to communicate over the internet.

Use the dns-server command to enable the router to provide DNS services. This command effectively turns the router into a DNS server.

```
Router(config)# ip dns server
```

Specify DNS server addresses using the ip name-server command followed by the IP address of the DNS server

```
Router(config)# ip name-server 8.8.8.8
```

Configure additional DNS parameters such as DNS domain names and DNS timeout settings.

```
Router(config)# ip domain-name example.com
```

The command ip host in Cisco IOS is used to manually configure a static mapping between a hostname and an IP address in the router's local DNS cache. Here’s how you would use it to map www.example.com to 8.8.8.8

```
Router(config)# ip host www.example.com 8.8.8.8
```

Note: Set up the DNS address in Host devices

Dynamic Host Configuration Protocol (DHCP)

A vital network management protocol used to automatically assign IP addresses and other network configuration parameters to devices on a network. By automating the IP assignment process, DHCP reduces the administrative burden and potential for human error associated with manually configuring IP addresses. DHCP servers manage and allocate IP addresses from a predefined pool, ensuring that each device on the network receives a unique IP address, along with additional configuration

details such as the subnet mask, default gateway, and DNS servers. This facilitates seamless connectivity and efficient network operations.
Before defining the DHCP pool, it is advisable to exclude specific IP addresses from the DHCP pool to reserve them for static assignments. This can be done using the ip dhcp excluded-address command, specifying the range of addresses to exclude.

```
Router(config)# ip dhcp excluded-address 192.168.1.1 192.168.1.10
```

Once the exclusions are set, create a DHCP pool with a unique name using the ip dhcp pool command. Within the DHCP pool configuration mode, define the network and subnet mask using the network command. Specify the default gateway that DHCP clients should use with the default-router command. Optionally, you can set additional parameters such as DNS servers with the dns-server command and the domain name using the domain-name command.

```
Router(config)# ip dhcp pool pool-name
Router(config-dhcp)# network network-address subnet-mask
Router(config-dhcp)# default-router default-gateway
Router(config-dhcp)# dns-server dns-address
```

- Replace pool-name with a name for your DHCP pool.
- network-address and subnet-mask define the range of IP addresses available for DHCP assignment.
- default-gateway is the router's IP address that clients use to reach devices outside their own subnet.
- dns-address is the IP address of the DNS server(s) to be provided to clients.

DHCP clients use broadcast messages to send DHCP requests. Routers do not forward broadcast messages. If the DHCP server and clients are in different network segments and connected through a router, the clients will not receive IP configurations from the DHCP server. In such a situation, you need to configure the router's interface connected to the DHCP server as a DHCP relay agent using the ip helper command Configuring a relay agent, also known as a DHCP relay agent, on a Cisco router is necessary when DHCP requests and servers are located on different network segments or VLANs. The relay agent forwards DHCP messages between clients and DHCP servers, ensuring that clients can receive IP address assignments even if the DHCP server is not on the same subnet.

Configure the interface or VLAN where DHCP requests are expected to arrive and need forwarding to the DHCP server. Use the ip helper-address command to specify the IP address of the DHCP server.

```
Router(config)# interface interface-id
Router(config-if)# ip helper-address dhcp-server-ip-address
```

- interface-id is the interface or VLAN where DHCP requests will be received.
- dhcp-server-ip-address is the IP address of the DHCP server to which DHCP requests should be forwarded.

This configuration allows the Cisco router to act as a DHCP relay agent, forwarding DHCP messages (discover, request, etc.) from clients on one subnet to DHCP servers located on another subnet, enabling centralized management of IP address allocation in larger networks.

Dynamic routing

The process by which routers automatically learn about and update their routing tables based on information exchanged with other routers. Unlike static routing, where routes are manually configured and do not change unless updated manually, dynamic routing protocols allow routers to dynamically adjust to changes in network topology, such as link failures or network expansions. This adaptive nature makes dynamic routing essential for large, complex networks where manual management of routing tables would be impractical.

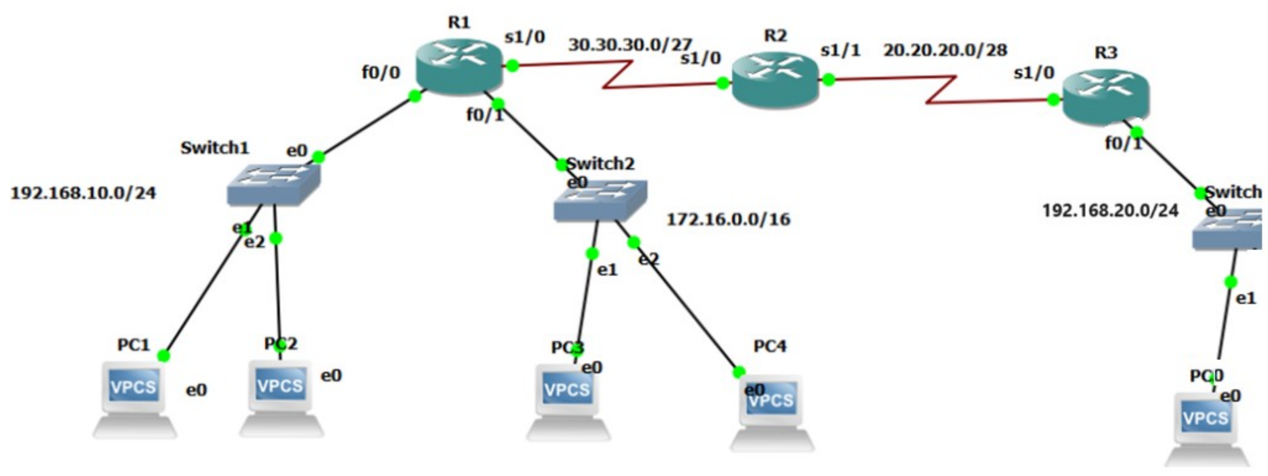
RIP is one of the oldest and simplest dynamic routing protocols. It operates based on the distance-vector algorithm, where routers exchange routing information at regular intervals (typically every 30 seconds) using broadcast packets. RIP uses hop count as its metric to determine the best path to a destination network, with a maximum hop count limit of 15 (16 hops is considered unreachable).
Configuring RIP (Routing Information Protocol) on a Cisco router involves several steps to enable dynamic routing and allow the router to exchange routing information with neighboring routers. First, you enter global configuration mode on the router's command-line interface (CLI) using the configure terminal command. Once in global configuration mode, you initiate the RIP configuration by entering router rip.
Next, you specify the version of RIP to use; for modern networks, RIP version 2 (version 2) is typically preferred due to its support for classless routing and additional

features compared to RIP version 1. You can configure this with the version 2 command under the RIP configuration mode.
After setting the RIP version, you define which networks connected to the router should participate in RIP routing. This is done using the network command followed by the network address (typically a classful network address or subnet address). This command tells RIP to advertise routes for networks that match the specified network address.

```
Router(config)# router rip
Router(config-router)# version 2 // Configures RIP version 2
Router(config-router)# network network-address // Specifies the network address for
RIP routing
```

Lab Exercise 1

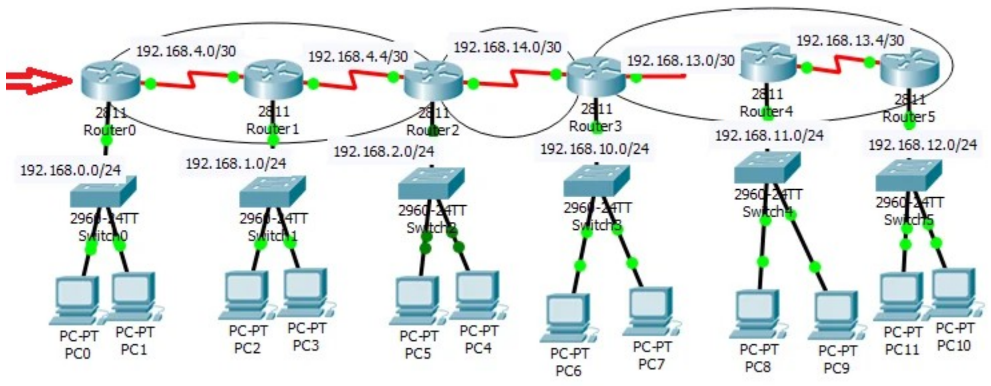
Design the Network with routing using RIP. Provide DHCP and DNS Configuration in R3



- Configure the IP Addresses to the interfaces on the router R1,R2 and R3 and PCs
- Configure RIP on the routers.
- Configure R3 as the DNS Server with the Domain Name www.NAME.com. (Test using PC2)
- Configure R3 as the DHCP router with the following restrictions:
  - Use name POOL-CLASS-C for the hosts in the 192.168.10.0/24 Network.
  - Use name POOL-CLASS-B for the hosts in the 192.168.20.0/24 Network.
  - Exclude the first and the last host addresses.
  - Assign the DNS Server address during DHCP Configuration.
  - Assign appropriate DHCP relay agents.
- Test IP Configuration in hosts and Access to the DNS Server.
- Save the configurations on all routers.

Additional Exercise:

Build the nerwork by setting up the router as a DHCP server for LAN1, LAN 2 and LAN3. All other static IP addresses can be assigned. Routing should be done using RIP. Set up another router as DNS Server and verify all configurations.



192.168.4.0/30
192.168.4.4/30
192.168.14.0/30
192.168.13.0/30
192.168.13.4/30
2811 Router0
2811 Router1
2811 Router2
2811 Router3
2811 Router4
2811 Router5
192.168.0.0/24
192.168.1.0/24
192.168.2.0/24
192.168.10.0/24
192.168.11.0/24
192.168.12.0/24
2960-24TT Switch0
2960-24TT Switch1
2960-24TT Switch2
2960-24TT Switch3
2960-24TT Switch4
2960-24TT Switch5
PC-PT PC0
PC-PT PC1
PC-PT PC2
PC-PT PC3
PC-PT PC5
PC-PT PC4
PC-PT PC6
PC-PT PC7
PC-PT PC8
PC-PT PC9
PC-PT PC11
PC-PT PC10

Week 11 &12 : Build a Small Campus

You have been tasked with designing and deploying a small campus network using Packet Tracer for a university campus comprising three buildings: Building A (Administration), Building B (Engineering), and Building C (Library). Each building houses various departments spread across multiple floors, necessitating segregated VLANs for efficient traffic management while enabling inter-departmental communication.

In this scenario:

- Configure three Cisco routers (Router A, Router B, Router C) to connect each building to the campus core network. Each router should integrate with its respective building VLANs and provide connectivity to the central network for internet access.
- Implement static routing between the routers to establish connectivity across buildings. Define specific static routes to facilitate traffic routing between VLANs and ensure seamless communication between departments.
- Create VLANs tailored for each department within each building:
  - Building A: VLAN 10 (Administration), VLAN 20 (Human Resources), VLAN 30 (Finance)
  - Building B: VLAN 10 (Software Engineering), VLAN 20 (Hardware Engineering), VLAN 30 (Network Engineering)
  - Building C: VLAN 10 (General Library), VLAN 20 (Research Library), VLAN 30 (Archives)
- Configure inter-VLAN routing on the routers to enable communication among VLANs within the same building, facilitating collaboration across departments.
- Enable RIP version 2 on all routers to automate the exchange of routing information. Ensure each router advertises its connected networks to facilitate dynamic routing updates across the campus network.
- Implement DHCP services on each router to automate IP address assignment within each VLAN. Configure DHCP servers on routers to efficiently distribute IP addresses and network configuration parameters to devices within their respective VLANs.

Throughout the implementation:

- Utilize Packet Tracer simulation tools to validate connectivity between devices in different VLANs and across buildings.

- Conduct comprehensive testing of DHCP functionality to verify that devices receive correct IP addresses and network settings as configured.

Task: Set up the described campus network scenario in Packet Tracer, ensuring accurate configuration of routers, VLAN segmentation, static and dynamic routing protocols, and DHCP services. Document your configuration steps meticulously and verify network functionality through simulated tests across departments and buildings.

Deliverables:

- Network topology diagram illustrating router connections, VLAN allocations, and IP addressing details.
- Detailed configuration scripts or annotated screenshots showcasing router setups, VLAN configurations, static routing entries, RIP configuration specifics, and DHCP implementations.
- Comprehensive summary report outlining the implementation process, challenges encountered, and key insights gained during the exercise.

References:

1. W. Richard Stevens, *UNIX Network Programming*, *Volume 1: The Sockets Networking API*, (3e), Addison-Wesley Professional Computing, 2003
2. Prof. Dayanand Ambawade,Deven N.Shah & Kogent Learning Solutions Inc, *Linux Lab: Hands on Linux*, Dreamtech Press, 2009
3. GNS3 Documentation, https:///www.gns3.com
4. Todd Lammle, *CCNA Cisco Certified Network Associate Study Guide*, (7e), Wiley India Pvt. Ltd 2011.