JSS MAHAVIDYAPEETHA
JSS SCIENCE AND TECHNOLOGY UNIVERSITY
SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING
• Constituent College of JSS Science and Technology University
• Approved by A.I.C.T.E
• Governed by the Grant-in-Aid Rules of Government of Karnataka
• Identified as lead institution for World Bank Assistance under TEQIP Scheme

DEPARTMENT OF ELECRONICS AND COMMUNICATION ENGINEERING

COURSE CODE: EC620 (3:0:1)

NETWORKS LAB MANUAL

Lab Location: Room No CS201

Prepared by
Dr. Shankaraiah
Prof. Vinayprasad M S

Vision statement of the JSS Science and Technology University

- **Advancing JSS S&T University as a leader in education, research and technology on the International arena.**
- **To provide the students a universal platform to launch their careers, vesting the industry and research community with skilled and professional workforce.**
- **Accomplishing JSS S&T University as an epicenter for innovation, center of excellence for research with state of the art lab facilities.**
- **Fostering an erudite, professional forum for researchers and industrialist to coexist and to work cohesively for the growth and development of science and technology for betterment of society.**

Mission statement of the JSS Science and Technology University

1. **Education, research and social outreach are the core doctrines of JSS S&T University that are responsible for accomplishment of in-depth knowledge base, professional skill and innovative technologies required to improve the socio economic conditions of the country.**
2. **Our mission is to develop JSS S&T University as a global destination for cohesive learning of engineering, science and management which are strongly supported with interdisciplinary research and academia.**
3. **JSS S&T University is committed to provide world class amenities, infrastructural and technical support to the students, staff, researchers and industrial partners to promote and protect innovations and technologies through patents and to enrich entrepreneurial endeavors.**
4. **JSS S&T University core mission is to create knowledge led economy through appropriate technologies, and to resolve societal problems by educational empowerment and ethics for better living.**

Vision statement of the department of E&CE

Be a leader in providing globally acceptable education in electronics and communication engineering with emphasis on fundamentals-to-applications, creative thinking, research and career- building.

Mission statement of the department of E&CE

1. **To provide best infrastructure and up-to-date curriculum with a conducive learning environment.**
2. **To enable students to keep pace with emerging trends in Electronics and Communication Engineering.**
3. **To establish strong industry participation and encourage student entrepreneurship.**
4. **To promote socially relevant eco-friendly technologies and inculcate inclusive innovation activities.**

Program Outcomes (POs)

1. **Engineering Knowledge**: Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

2. **Problem Analysis**: Identify, formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences

3. **Design/ Development of Solutions**: Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.

4. **Conduct investigations of complex problems**: Using research based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.

5. **Modern Tool Usage**: Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations

6. **The Engineer and Society**: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to professional engineering practice.

7. **Environment and Sustainability**: Understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate knowledge of and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

9. **Individual and Team Work**: Function effectively as an individual, and as a member or leader in diverse teams and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.

11. **Lifelong Learning**: Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

12. **Project Management and Finance**: Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

Program Specific Outcomes (PSOs)

1. Analyze, design and provide engineering solutions in the areas of electronic circuits and systems.

2. Demonstrate the mathematical modeling techniques, nurture analytical and computational skills to provide engineering solutions in the areas of electronics and communication.

3. Ability to address multidisciplinary research challenges and nurture entrepreneurship

Program Educational Objectives (PEOs)

1. To enable the graduates to have strong Engineering fundamentals in Electronics & Communication, with adequate orientation to mathematics and basic sciences.
2. To empower graduates to formulate, analyze, design and provide innovative solutions in Electronics & Communication, for real life problems.
3. To ensure that graduates have adequate exposure to research and emerging technologies through industry interaction and to inculcate professional and ethical values.
4. To nurture required skill sets to enable graduates to pursue successful professional career in industry, higher education, competitive exams and entrepreneurship.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

RECORD OF CIE FOR PERFORMANCE IN THE LAB CLASSES

Evaluation Sheet

Section		**Batch**		**Group No**	
Staff in Charge		**Day**		**Timings**	

Sl. No.	USN	Name of the Students	AC1: Preparedness (8M) AC2: Conduction (8M) AC3: Viva (8M) AC4: Report Writing (8M) AC5: Result Interpretation (8M) T: Total (40M)	SUBJECT: Networking Lab
1.				
2.				
3.				
4.				

Sl. No	Date	Experiments	Student-1						Student-2						Student-3						Student-4					
			AC1	AC2	AC3	AC4	AC5	T	AC1	AC2	AC3	AC4	AC5	T	AC1	AC2	AC3	AC4	AC5	T	AC1	AC2	AC3	AC4	AC5	T
1.																										
2.																										
3.																										
4.																										
5.																										

Sl. No.	Date	Experiments	Student-1						Student-2						Student-3						Student-4					
			AC1	AC2	AC3	AC4	AC5	T	AC1	AC2	AC3	AC4	AC5	T	AC1	AC2	AC3	AC4	AC5	T	AC1	AC2	AC3	AC4	AC5	T
6.																										
7.																										
8.																										
9.																										
10.																										
11.																										
12.																										
Average marks from experiments 1 to 10 = (40 Marks)																										
Lab Test =(10 Marks)																										
Total CIE (50 MARKS)																										
Percentage of Attendance																										

Signature of the Staff in Charge: **Signature of the Lab in Charge:**

Prof. VINAYPRASAD M S

1.

2.

Course outcome: At the end of the course, the student should be able to

1. Use the network simulator for learning and practice of networking algorithms.
2. Illustrate the operation of network protocols and algorithms using C programming.
3. Analyze various Routing protocols and addressing schemes by creating various network configurations.

EC 620L: Computer Networks Lab

List of Experiments:

PART-A: Simulation experiments using CISCO Packet Tracer/ GNS3 Tool.

1. Study of different types of Network cables and practically implement the cross-wired cable and straight through cable using clamping tool.
2. Using CISCO packet Tracer, perform the following experiments
   a. Configure a basic Network topology.
   b. Subnetting (FLSM and VLSM).
   c. Investigate Unicast, Broadcast and Multicast Traffic.
3. Using CISCO Packet Tracer, Perform the following experiments
   a. Skills Integration challenge-planning subnets and configuring IP addresses.
   b. Observing the effects of collision in a shared media environment.
   c. Static routing and default routing.
4. Configure a Network topology using Distance Vector Routing protocol (IPv4, Ipv6).
5. Configure a Network topology using Link State Routing protocol (IPv4, Ipv6).
6. Using CISCO Packet Tracer, Perform the followings
   a. Network Address Translation (NAT)
   b. Access Control List (ACLs)
7. Using packet Tracer, perform the following experiments
   a. Basic switching configuration.
   b. Configure VLAN and Inter-VLAN routing for a Network.

1. **Study of different types of Network cables and practically implement the cross-wired cable and straight through cable using clamping tool.**

Aim: Study of different types of Network cables and Practically implement the cross-wired cable and straight through cable using clamping tool.

Apparatus (Components): RJ-45 connector, Climping Tool, Twisted pair Cable

Procedure: To do these practical following steps should be done:

1. Start by stripping off about 2 inches of the plastic jacket off the end of the cable. Be very careful at this point, as to not nick or cut into the wires, which are inside. Doing so could alter the characteristics of your cable, or even worse render is useless. Check the wires, one more time for nicks or cuts. If there are any, just whack the whole end off, and start over.

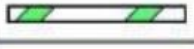
2. Spread the wires apart, but be sure to hold onto the base of the jacket with your other hand. You do not want the wires to become untwisted down inside the jacket. Category 5 cable must only have 1/2 of an inch of 'untwisted' wire at the end; otherwise it will be 'out of spec'. At this point, you obviously have ALOT more than 1/2 of an inch of un-twisted wire.

3. You have 2 end jacks, which must be installed on your cable. If you are using a pre-made cable, with one of the ends whacked off, you only have one end to install - the crossed over end. Below are two diagrams, which show how you need to arrange the cables for each type of cable end. Decide at this point which end you are making and examine the associated picture below.

Diagram shows you how to prepare Cross wired connection

RJ45 Pin # (END 1)	Wire Color	Diagram End #1	RJ45 Pin # (END 2)	Wire Color	Diagram End #2
1	White/Orange		1	White/Green	
2	Orange		2	Green	
3	White/Green		3	White/Orange	
4	Blue		4	White/Brown	
5	White/Blue		5	Brown	
6	Green		6	Orange	
7	White/Brown		7	Blue	
8	Brown		8	White/Blue	

Diagram shows you how to prepare straight through wired connection

RJ45 Pin # (END 1)	Wire Color	Diagram End #1	RJ45 Pin # (END 2)	Wire Color	Diagram End #2
1	White/Orange		1	White/Green	
2	Orange		2	Green	
3	White/Green		3	White/Orange	
4	Blue		4	White/Brown	
5	White/Blue		5	Brown	
6	Green		6	Orange	
7	White/Brown		7	Blue	
8	Brown		8	White/Blue	

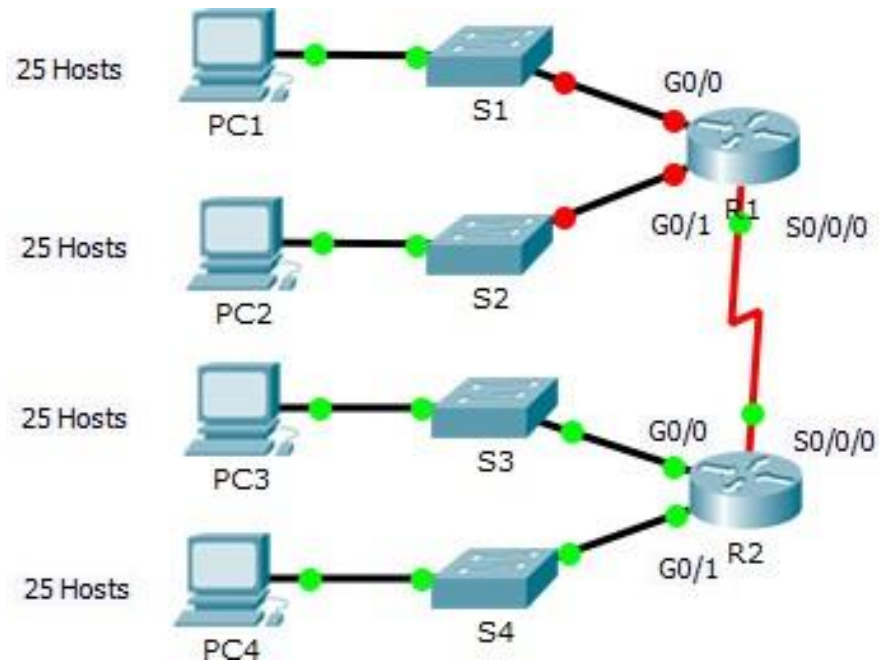
2. Using CISCO packet Tracer, perform the following experiments

a. Configure a basic Network topology.

b. Subnetting (FLSM and VLSM).

c. Investigate Unicast, Broadcast and Multicast Traffic.

Packet Tracer - Subnetting Scenario



Objectives

1. Design an IP Addressing Scheme
2. Assign IP Addresses to Network Devices and Verify Connectivity

Scenario

In this activity, you are given the network address of 193.162.98.0/24 to subnet and provide the IP addressing for the network shown in the topology. Each LAN in the network requires enough space for, at least, 25 addresses for end devices, the switch and the router. The connection between R1 to R2 will require an IP address for each end of the link.

Design an IP Addressing Scheme

Step 1: Subnet the 193.162.98.0/24 network into the appropriate number of subnets.

a) Based on the topology, how many subnets are needed?

b) How many bits must be borrowed to support the number of subnets in the topology table?

c) How many subnets does this create?

d) How many usable hosts does this create per subnet?

Note: If your answer is less than the 25 hosts required, then you borrowed too many bits.

a) Calculate the binary value for the first five subnets.

Net 0: 193. 162. 98 .0 0 0 0 0 0 0 0

Net 1: 193. 162. 98. _____

Net 2: 193. 162. 98. _____

Net 3: 193. 162. 98. _____

Net 4: 193. 162. 98. _____

b) Calculate the binary and decimal value of the new subnet mask.

11111111.11111111.11111111. _____

255.255.255. ____

c) Fill in the Subnet Table, listing the decimal value of all available subnets, the first and last usable
host address, and the broadcast address. Repeat until all addresses are listed.

Note: You may not need to use all rows.

Subnet Table

Subnet Number	Subnet Address	First Usable Host Address	Last Usable Host Address	Broadcast Address
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

Step 02: Assign the subnets to the network shown in the topology.

a. Assign Subnet 0 to the LAN connected to the GigabitEthernet 0/0 interface of R1.

b. Assign Subnet 1 to the LAN connected to the GigabitEthernet 0/1 interface of R1.

c. Assign Subnet 2 to the LAN connected to the GigabitEthernet 0/0 interface of R2

d. Assign Subnet 3 to the LAN connected to the GigabitEthernet 0/1 interface of R2

e. Assign Subnet 4 to the WAN link between R1 to R2

Step 3:Document the addressing scheme.

Fill in the Addressing Table using the following guidelines:

a) Assign the first usable IP addresses to R1 for the two LAN links and the WAN link.

b) Assign the first usable IP addresses to R2 for the LANs links. Assign the last usable IP address for the WAN link.

c) Assign the second usable IP addresses to the hosts.

d) Assign the last usable IP addresses to the hosts.

Addressing Table

Device	Interface	IP Address	Subnet Mask	Default Gateway
R1	Go/o			N/A
	Go/1			N/A
	So/o/o			N/A
R2	Go/o			N/A
	Go/1			N/A
	So/o/o			N/A
PC1	NIC			
PC2	NIC			
PC3	NIC			
PC4	NIC			

Assign IP Addresses to Network Devices and Verify Connectivity

Most of the IP addressing is already configured on this network. Implement the following steps to complete the addressing configuration.

Step 1: Configure IP addressing on R1 LAN interfaces.

Step 2: Configure IP addressing on S3, including the default gateway

Step 3: Configure IP addressing on PC4, including the default gateway.

Step 4: Verify connectivity.

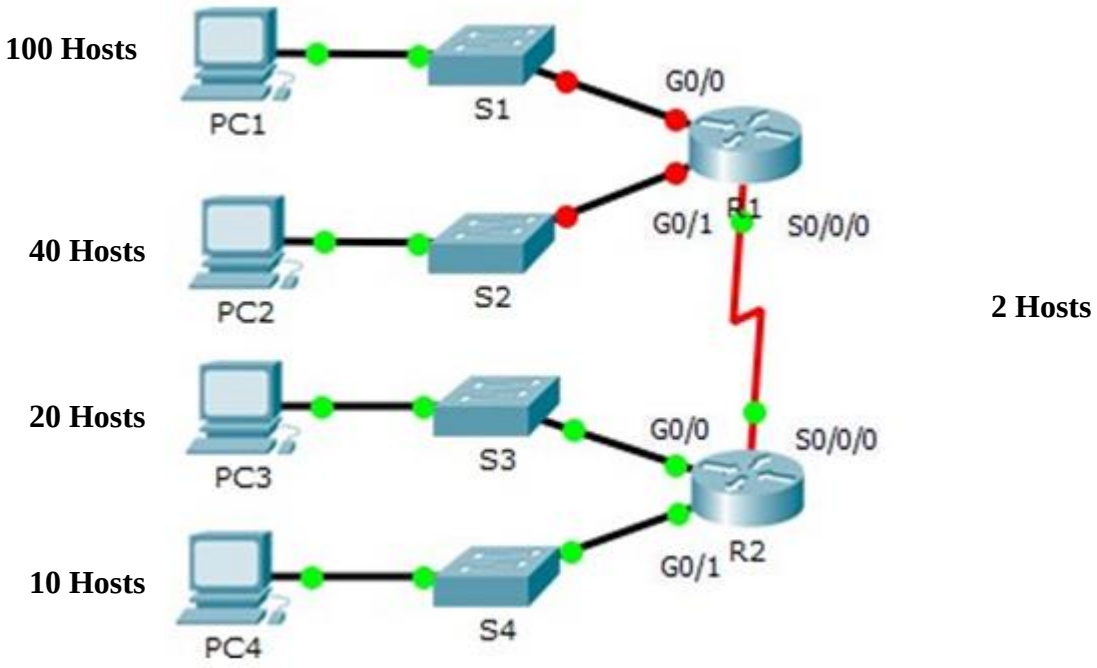
You can only verify connectivity from R1, S3, and PC4. However, you should be able to ping every IP address listed in the Addressing Table.

Note:

1. Take the screen shot of the results which is verified in Simulation Mode /Real mode and write the inference.
2. After the completion of the experiment, answer all the viva question given below in the lab itself.

Packet Tracer - Subnetting Scenario - VLSM

Topology



Objectives

3. Design an IP Addressing Scheme (VLSM)
4. Assign IP Addresses to Network Devices and Verify Connectivity

Scenario

In this activity, you are given the network address of 193.162.98.0/24 to subnet and provide the IP addressing for the network shown in the topology. Each LAN in the network requires enough space for, at least, 25 addresses for end devices, the switch and the router. The connection between R1 to R2 will require an IP address for each end of the link.

Design an IP Addressing Scheme

Step 1: Subnet the 193.162.98.0/24 network into the appropriate number of subnets.

e) Based on the topology, how many subnets are needed?

f) How many bits must be borrowed to support the number of subnets in the topology table?

g) How many subnets does this create?

h) How many usable hosts does this create per subnet?

Note: If your answer is less than the 100 hosts required, then you borrowed too many bits.

d) Calculate the binary value for the first five subnets.

Net 0: 193. 162. 98 .0 0 0 0 0 0 0 0 0

Net 1: 193. 162. 98. _____

Net 2: 193. 162. 98. _____

Net 3: 193. 162. 98. _____

Net 4: 193. 162. 98. _____

e) Calculate the binary and decimal value of the new subnet mask.
Subnet 0 Mask: **255.255.255**. Subnet 3 Mask: **255.255.255.** _____
Subnet 1 Mask: **255.255.255.** _____ subnet 4 Mask: **255.255.255.** _____

Subnet 2 Mask: **255.255.255.** _____

f) Fill in the Subnet Table, listing the decimal value of all available subnets, the first and last usable
host address, and the broadcast address. Repeat until all addresses are listed.

Note: You may not need to use all rows.

Subnet Table

Subnet Number	Subnet Address	First Usable Host Address	Last Usable Host Address	Broadcast Address
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

Step 02: Assign the subnets to the network shown in the topology.

f. Assign Subnet 0 to the LAN connected to the GigabitEthernet 0/0 interface of R1.

g. Assign Subnet 1 to the LAN connected to the GigabitEthernet 0/1 interface of R1.

h. Assign Subnet 2 to the LAN connected to the GigabitEthernet 0/0 interface of R2

i. Assign Subnet 3 to the LAN connected to the GigabitEthernet 0/1 interface of R2

j. Assign Subnet 4 to the WAN link between R1 to R2

Step 3: Document the addressing scheme.

Fill in the Addressing Table using the following guidelines:

e) Assign the first usable IP addresses to R1 for the two LAN links and the WAN link.

f) Assign the first usable IP addresses to R2 for the LANs links. Assign the last usable IP address for the WAN link.

g) Assign the second usable IP addresses to the hosts.

h) Assign the last usable IP addresses to the hosts.

Addressing Table

Device	Interface	IP Address	Subnet Mask	Default Gateway
R1	Go/o			N/A
	Go/1			N/A
	So/o/o			N/A
R2	Go/o			N/A
	Go/1			N/A
	So/o/o			N/A
PC1	NIC			
PC2	NIC			
PC3	NIC			
PC4	NIC			

Assign IP Addresses to Network Devices and Verify Connectivity

Most of the IP addressing is already configured on this network. Implement the following steps to complete the addressing configuration.

Step 1: Configure IP addressing on R1 LAN interfaces.

Step 2: Configure IP addressing on S3, including the default gateway

Step 3: Configure IP addressing on PC4, including the default gateway.

Step 4: Verify connectivity.

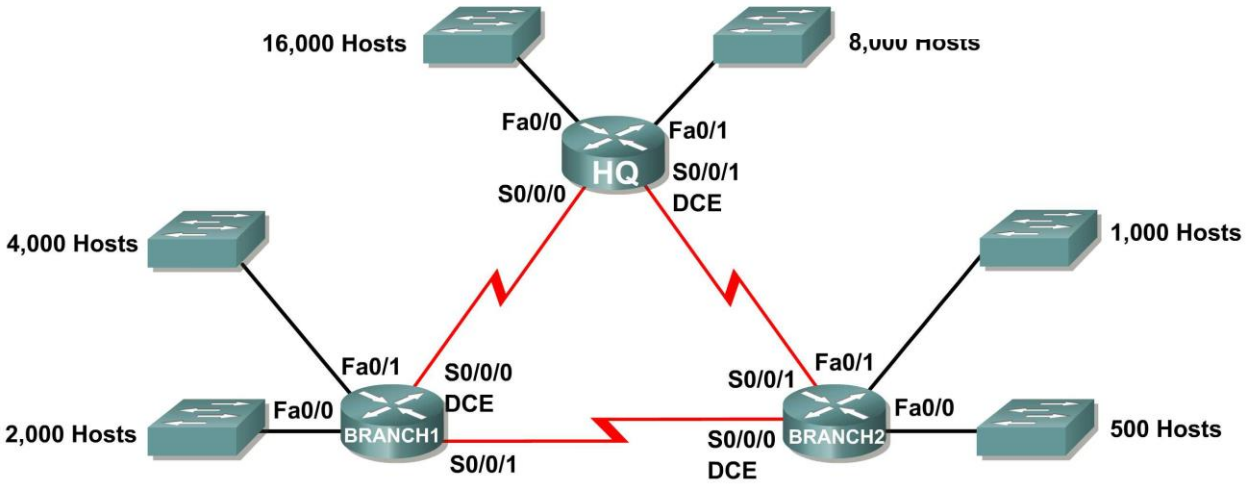
Verify connectivity from LAN to every other LAN. However, you should be able to ping every IP address listed in the Addressing Table.

Note:

1.Take the screen shot of the results which is verified in Simulation Mode /Real mode and write the inference.

2.After the completion of the experiment, answer all the viva question given below in the lab itself.

A VLSM Addressing Design / Topology Diagram



Addressing Table

Subnet	Number of IP Addresses Needed	Network Address
HQ LAN1	16,000	172.16.128.0/19
HQ LAN2	8,000	172.16.192.0/18
Branch1 LAN1	4,000	172.16.224.0/20
Branch1 LAN2	2,000	172.16.240.0/21
Branch2 LAN1	1,000	172.16.244.0/24
Branch2 LAN2	500	172.16.252.0/23
Link from HQ to Branch1	2	172.16.254.0/28
Link from HQ to Branch2	2	172.16.254.6/30
Link from Branch1 to Branch2	2	172.16.254.8/30

Learning Objectives

Upon completion of this activity, you will be able to:

- Discover errors in a VLSM design.
- Propose solutions for VLSM design errors.
- Document the corrected VLSM assignments.

Scenario

In this activity, the network address 172.16.128.0/17 has been used to provide the IP addressing for the network shown in the Topology Diagram. VLSM has been used to subnet the address space incorrectly. You will need to troubleshoot the addressing that has been assigned for each subnet to determine where errors are present and then determine the correct addressing assignments, where needed.

Task 1: Examine the Addressing for the HQ LANs.

Step 1: Examine the addressing assignment for the HQ LAN1 subnet and answer the questions below:

1. How many IP addresses are needed for the HQ LAN1 subnet? _____
2. How many IP addresses are available in the currently assigned subnet? _____
3. Will the currently assigned subnet fulfill the size requirement for the HQ LAN1 subnet? _____
4. If the answer to the previous question is **no**, propose a new subnet mask that will allow for the correct number of IP addresses. _____
5. Does the subnet overlap with any of the other currently assigned networks? _____
6. If the answer to the previous question is **yes**, propose a new subnet mask that will allow for the correct number of IP addresses without overlapping into any other subnets. _____

Step 2: Examine the addressing assignment for the HQ LAN2 subnet and answer the questions below.

1. How many IP addresses are needed for the HQ LAN2 subnet? _____
2. How many IP addresses are available in the currently assigned subnet? _____
3. Will the currently assigned subnet fulfill the size requirement for the HQ LAN2 subnet? _____
4. If the answer to the previous question is **no**, propose a new subnet mask that will allow for the correct number of IP addresses. _____
5. Does the subnet overlap with any of the other currently assigned networks?_____
6. If the answer to the previous question is **yes**, propose a new subnet mask that will allow for the correct number of IP addresses without overlapping into any other subnets. _____

Task 2: Examine the Addressing for the Branch1 LANs.

Step 1: Examine the addressing assignment for the Branch1 LAN1 subnet and answer the questions below.

1. How many IP addresses are needed for the Branch1 LAN1 subnet? _____
2. How many IP addresses are available in the currently assigned subnet? _____
3. Will the currently assigned subnet fulfill the size requirement for the Branch1 LAN1 subnet? _____
4. If the answer to the previous question is **no**, propose a new subnet mask that will allow for the correct number of IP addresses. _____
5. Does the subnet overlap with any of the other currently assigned networks?_____
6. If the answer to the previous question is **yes**, propose a new subnet mask that will allow for the correct number of IP addresses without overlapping into any other subnets. _____

Step 2: Examine the addressing assignment for the Branch1 LAN2 and answer the questions below.

1. How many IP addresses are needed for the Branch1 LAN2 subnet? _____
2. How many IP addresses are available in the currently assigned subnet? _____
3. Will the currently assigned subnet fulfill the size requirement for the Branch1 LAN2 subnet? _____
4. If the answer to the previous question is **no**, propose a new subnet mask that will allow for the correct number of IP addresses. _____
5. Does the subnet overlap with any of the other currently assigned networks?_____

6. If the answer to the previous question is **yes**, propose a new network address that will allow for the correct number of IP addresses without overlapping into any other subnets. _____

Task 3: Examine the Addressing for the Branch2 LANs.

Step 1: Examine the addressing assignment for the Branch2 LAN1 subnet and answer the questions below.

1. How many IP addresses are needed for the Branch2 LAN1 subnet? _____
2. How many IP addresses are available in the currently assigned subnet? _____
3. Will the currently assigned subnet fulfill the size requirement for the Branch2 LAN1 subnet? _____
4. If the answer to the previous question is **no**, propose a new subnet mask that will allow for the correct number of IP addresses. _____
5. Does the subnet overlap with any of the other currently assigned networks?_____
6. If the answer to the previous question is **yes**, propose a new subnet mask that will allow for the correct number of IP addresses without overlapping into any other subnets. _____

Step 2: Examine the addressing assignment for the Branch2 LAN2 and answer the questions below.

1. How many IP addresses are needed for the Branch2 LAN2 subnet? _____
2. How many IP addresses are available in the currently assigned subnet? _____
3. Will the currently assigned subnet fulfill the size requirement for the Branch2 LAN2 subnet? _____
4. If the answer to the previous question is **no**, propose a new subnet mask that will allow for the correct number of IP addresses. _____
5. Does the subnet overlap with any of the other currently assigned networks?_____
6. If the answer to the previous question is **yes**, propose a new network address that will allow for the correct number of IP addresses without overlapping into any other subnets. _____

Task 4: Examine the Addressing for the Links between Routers.

Step 1: Examine the addressing assignment for the link between the HQ and Branch1 routers and answer the questions below.

1. How many IP addresses are needed for the link between the HQ and Branch1 routers? _____
2. How many IP addresses are available in the currently assigned subnet? _____
3. Will the currently assigned subnet fulfill the size requirement for the link between the HQ and Branch1 routers? _____
4. If the answer to the previous question is **no**, propose a new subnet mask that will allow for the correct number of IP addresses. _____
5. Does the subnet overlap with any of the other currently assigned networks?_____
6. If the answer to the previous question is **yes**, propose a new subnet mask that will allow for the correct number of IP addresses without overlapping into any other subnets. _____

Step 2: Examine the addressing assignment for the link between the HQ and Branch2 routers and answer the questions below.

1. How many IP addresses are needed for the link between the HQ and Branch2 routers? _____
2. How many IP addresses are available in the currently assigned subnet? _____
3. Will the currently assigned subnet fulfill the size requirement for the link between the HQ and Branch2 routers? _____
4. If the answer to the previous question is **no**, propose a new subnet mask that will allow for the correct number of IP addresses. _____
5. Does the subnet overlap with any of the other currently assigned networks?_____
6. If the answer to the previous question is **yes**, propose a new network address that will allow for the correct number of IP addresses without overlapping into any other subnets. _____

Step 3: Examine the addressing assignment for the link between the Branch1 and Branch2 routers and answer the questions below.

1. How many IP addresses are needed for the link between the Branch1 and Branch2 routers? _____
2. How many IP addresses are available in the currently assigned subnet? _____
3. Will the currently assigned subnet fulfill the size requirement for the link between the Branch1 and Branch2 routers? _____
4. If the answer to the previous question is **no**, propose a new subnet mask that will allow for the correct number of IP addresses. _____
5. Does the subnet overlap with any of the other currently assigned networks?_____
6. If the answer to the previous question is **yes**, propose a new subnet mask that will allow for the correct number of IP addresses without overlapping into any other subnets. _____

Task 5: Document the Corrected Addressing Information.

Record the corrected addressing information in the Addressing Table below.

Subnet	Number of IP Addresses Needed	Network Address
HQ LAN1	16,000	
HQ LAN2	8,000	
Branch1 LAN1	4,000	
Branch1 LAN2	2,000	
Branch2 LAN1	1,000	
Branch2 LAN2	500	
Link from HQ to Branch1	2	
Link from HQ to Branch2	2	
Link from Branch1 to Branch2	2	

3. Using CISCO Packet Tracer, Perform the following experiments

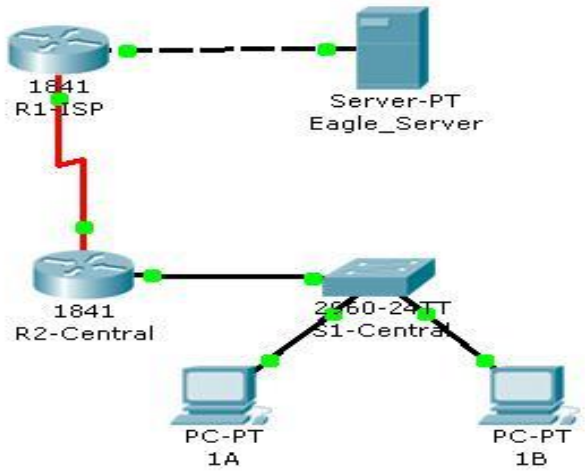
a. Skills Integration challenge-planning subnets and configuring IP addresses.

b. Observing the effects of collision in a shared media environment.

c. Static routing and default routing.

Aim: skills integration challenge-planning subnets and configuring IP addresses

- IP subnet planning
- Build and configure the network -Apply Subnetting scheme to server, Pcs, and router interfaces.
- configure services and static routing
- Test the network – using ping, trace, web traffic, inspect tool



Device	Interface	IP Address	Subnet Mask	Default Gateway	DNS Server
R1-ISP	**Fa0/0**	**192.168.23.110**	**255.255.255.240**	**N/A**	**N/A**
R1-ISP	**S0/0/0**	**192.168.23.122**	**255.255.255.252**	**N/A**	**N/A**
R2-Central	**Fa0/0**	**192.168.23.62**	**255.255.255.192**	**N/A**	**N/A**
R2-Central	**S0/0/0**	**192.168.23.121**	**255.255.255.252**	**N/A**	**N/A**
PC1A	**NIC**	**192.168.23.1**	**255.255.255.192**	**192.168.23.62**	**192.168.23.109**
PC1B	**NIC**	**192.168.23.2**	**255.255.255.192**	**192.168.23.62**	**192.168.23.109**
Eagle-Server	**NIC**	**192.168.23.109**	**255.255.255.240**	**192.168.23.110**	**N/A**

Procedure: Following is required to be study under this practical.

Task 1: IP Subnet Planning

You have been given an IP address block of 192.168.23.0 /24. You must provide for existing

networks as well as future growth.

Subnet assignments are:

- 1st subnet, existing student LAN (off of router R2-Central), up to 60 hosts;
- 2nd subnet, future student LAN, up to 28 hosts;
- 3rd subnet, existing ISP LAN, up to 12 hosts;
- 4th subnet, future ISP LAN, up to 8 hosts;
- 5th subnet, existing WAN, point-to-point link;
- 6th subnet, future WAN, point-to-point link;
- 7th subnet, future WAN, point-to-point link.

Interface IP addresses:

- For the server, configure the second highest usable IP address on the existing ISP LAN subnet.
- For R1-ISP's Fa0/0 interface, configure the highest usable IP address on the existing ISP LAN subnet.
- For R1-ISP's S0/0/0 interface, configure the highest usable address on the existing WAN subnet.
- For R2-Central's S0/0/0 interface, use the lowest usable address on the existing WAN subnet.
- For R2-Central's Fa0/0 interface, use the highest usable address on the existing student LAN subnet.
- For hosts 1A and 1B, use the first 2 IP addresses (two lowest usable addresses) on the existing
- student LAN subnet.

Additional configurations:

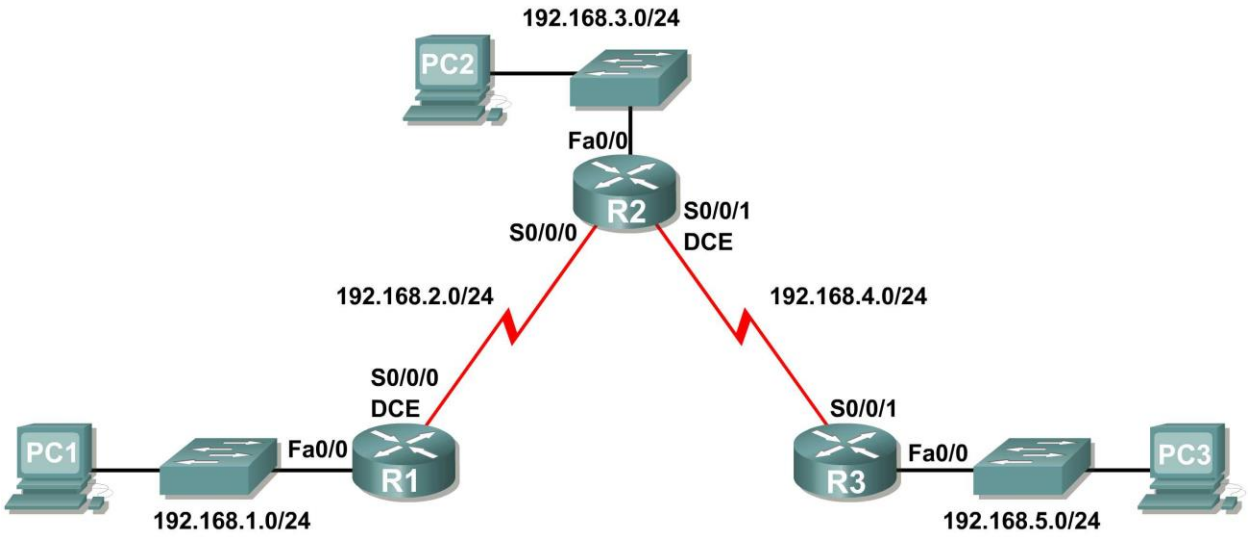
- For PCs 1A and 1B, in addition to IP configuration, configure them to use DNS services.
- For the server, enable DNS services, use the domain name eagle-server.example.com, and enable HTTP services.
- For R1-ISP router serial interface, you will need to set the clock rate (a timing mechanism required on the DCE end of serial links) to 64000.
- No clock rate is needed on the DTE side, in this case R2-Central's serial interface.

Task 2: Finish Building the Network in Packet Tracer. Add cables where missing.

4. Configure a Network topology using Distance Vector Routing protocol (IPv4, Ipv6).

Basic RIP Configuration

Topology Diagram



Learning Objectives

Upon completion of this lab, you will be able to:

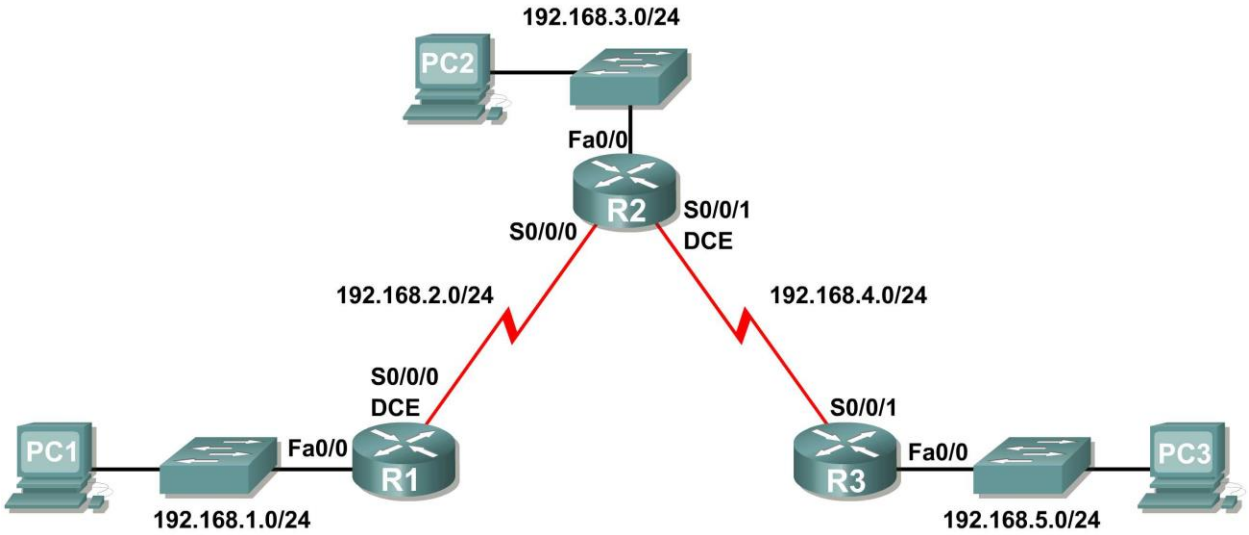
- Cable a network according to the Topology Diagram.
- Erase the startup configuration and reload a router to the default state.
- Perform basic configuration tasks on a router.
- Configure and activate interfaces.
- Configure RIP routing on all routers.
- Verify RIP routing using **`show`** and **`debug`** commands.
- Reconfigure the network to make it contiguous.
- Observe automatic summarization at boundary router.
- Gather information about RIP processing using the **`debug ip rip`** command.
- Configure a static default route.
- Propagate default routes to RIP neighbors.
- Document the RIP configuration.

Scenarios

- Scenario A: Running RIPv1 on Classful Networks
- Scenario B: Running RIPv1 with Subnets and Between Classful Networks
- Scenario C: Running RIPv1 on a Stub Network.

Scenario A: Running RIPv1 on Classful Networks

Topology Diagram



Addressing Table

Device	Interface	IP Address	Subnet Mask	Default Gateway
R1	**Fa0/0**	192.168.1.1	255.255.255.0	N/A
	S0/0/0	192.168.2.1	255.255.255.0	N/A
R2	**Fa0/0**	192.168.3.1	255.255.255.0	N/A
	S0/0/0	192.168.2.2	255.255.255.0	N/A
	S0/0/1	192.168.4.2	255.255.255.0	N/A
R3	**Fa0/0**	192.168.5.1	255.255.255.0	N/A
	S0/0/1	192.168.4.1	255.255.255.0	N/A
PC1	**NIC**	192.168.1.10	255.255.255.0	192.168.1.1
PC2	**NIC**	192.168.3.10	255.255.255.0	192.168.3.1
PC3	**NIC**	192.168.5.10	255.255.255.0	192.168.5.1

Task 1: Prepare the Network.

Step 1: Cable a network that is similar to the one in the Topology Diagram.

You can use any current router in your lab as long as it has the required interfaces shown in the topology.

Note: If you use 1700, 2500, or 2600 routers, the router outputs and interface descriptions will appear different.

Step 2: Clear any existing configurations on the routers.

Task 2: Perform Basic Router Configurations.

Perform basic configuration of the R1, R2, and R3 routers according to the following guidelines:

1. Configure the router hostname.
2. Disable DNS lookup.
3. Configure an EXEC mode password.
4. Configure a message-of-the-day banner.
5. Configure a password for console connections.
6. Configure a password for VTY connections.

Task 3: Configure and Activate Serial and Ethernet Addresses.

Step 1: Configure interfaces on R1, R2, and R3.

Configure the interfaces on the R1, R2, and R3 routers with the IP addresses from the table under the Topology Diagram.

Step 2: Verify IP addressing and interfaces.

Use the **`show ip interface brief`** command to verify that the IP addressing is correct and that the interfaces are active.

When you have finished, be sure to save the running configuration to the NVRAM of the router.

Step 3: Configure Ethernet interfaces of PC1, PC2, and PC3.

Configure the Ethernet interfaces of PC1, PC2, and PC3 with the IP addresses and default gateways from the table under the Topology Diagram.

Step 4: Test the PC configuration by pinging the default gateway from the PC.

Task 4: Configure RIP.

Step 1: Enable dynamic routing.

To enable a dynamic routing protocol, enter global configuration mode and use the **`router`** command.

Enter **`router ?`** at the global configuration prompt to a see a list of available routing protocols on your router.

To enable RIP, enter the command **`router rip`** in global configuration mode.

```
R1(config)#router rip
R1(config-router)#
```

Step 2: Enter classful network addresses.

Once you are in routing configuration mode, enter the classful network address for each directly connected network, using the **`network`** command.

```
R1(config-router)#network 192.168.1.0
R1(config-router)#network 192.168.2.0
R1(config-router)#
```

The **network** command:

- Enables RIP on all interfaces that belong to this network. These interfaces will now both send and receive RIP updates.
- Advertises this network in RIP routing updates sent to other routers every 30 seconds.

When you are finished with the RIP configuration, return to privileged EXEC mode and save the current configuration to NVRAM.

```
R1(config-router)#end
%SYS-5-CONFIG_I: Configured from console by console
R1#copy run start
```

Step 3: Configure RIP on the R2 router using the `router rip` and `network` commands.

```
R2(config)#router rip
R2(config-router)#network 192.168.2.0
R2(config-router)#network 192.168.3.0
R2(config-router)#network 192.168.4.0
R2(config-router)#end
%SYS-5-CONFIG_I: Configured from console by console
R2#copy run start
```

When you are finished with the RIP configuration, return to privileged EXEC mode and save the current configuration to NVRAM.

Step 4: Configure RIP on the R3 router using the `router rip` and `network` commands.

```
R3(config)#router rip
R3(config-router)#network 192.168.4.0
R3(config-router)#network 192.168.5.0
R3(config-router)#end
%SYS-5-CONFIG_I: Configured from console by console
R3# copy run start
```

When you are finished with the RIP configuration, return to privileged EXEC mode and save the current configuration to NVRAM.

Task 5: Verify RIP Routing.

Step 1: Use the `show ip route` command to verify that each router has all of the networks in the topology entered in the routing table.

Routes learned through RIP are coded with an **R** in the routing table. If the tables are not converged as shown here, troubleshoot your configuration. Did you verify that the configured interfaces are active? Did you configure RIP correctly? Return to Task 3 and Task 4 to review the steps necessary to achieve convergence.

```
R1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.1.0/24 is directly connected, FastEthernet0/0
C    192.168.2.0/24 is directly connected, Serial0/0/0
R    192.168.3.0/24 [120/1] via 192.168.2.2, 00:00:04, Serial0/0/0
R    192.168.4.0/24 [120/1] via 192.168.2.2, 00:00:04, Serial0/0/0
R    192.168.5.0/24 [120/2] via 192.168.2.2, 00:00:04, Serial0/0/0
R1#


R2#show ip route

<Output omitted>

R    192.168.1.0/24 [120/1] via 192.168.2.1, 00:00:22, Serial0/0/0
C    192.168.2.0/24 is directly connected, Serial0/0/0
C    192.168.3.0/24 is directly connected, FastEthernet0/0
C    192.168.4.0/24 is directly connected, Serial0/0/1
R    192.168.5.0/24 [120/1] via 192.168.4.1, 00:00:23, Serial0/0/1
R2#

R3#show ip route

<Output omitted>

R    192.168.1.0/24 [120/2] via 192.168.4.2, 00:00:18, Serial0/0/1
R    192.168.2.0/24 [120/1] via 192.168.4.2, 00:00:18, Serial0/0/1
R    192.168.3.0/24 [120/1] via 192.168.4.2, 00:00:18, Serial0/0/1
C    192.168.4.0/24 is directly connected, Serial0/0/1
C    192.168.5.0/24 is directly connected, FastEthernet0/0
R3#
```

Step 2: Use the `show ip protocols` command to view information about the routing processes.

The **`show ip protocols`** command can be used to view information about the routing processes that are occurring on the router. This output can be used to verify most RIP parameters to confirm that:

- RIP routing is configured
- The correct interfaces send and receive RIP updates
- The router advertises the correct networks
- RIP neighbors are sending updates

```
R1#show ip protocols
Routing Protocol is "rip"
Sending updates every 30 seconds, next due in 16 seconds
Invalid after 180 seconds, hold down 180, flushed after 240
Outgoing update filter list for all interfaces is not set
Incoming update filter list for all interfaces is not set
Redistributing: rip
Default version control: send version 1, receive any version
  Interface             Send  Recv  Triggered RIP  Key-chain
  FastEthernet0/0       1     2 1
  Serial0/0/0           1     2 1
Automatic network summarization is in effect
Maximum path: 4
Routing for Networks:
      192.168.1.0
      192.168.2.0
Passive Interface(s):
Routing Information Sources:
      Gateway         Distance      Last Update
      192.168.2.2          120
Distance: (default is 120)
R1#
```

R1 is indeed configured with RIP. R1 is sending and receiving RIP updates on FastEthernet0/0 and Serial0/0/0. R1 is advertising networks 192.168.1.0 and 192.168.2.0. R1 has one routing information source. R2 is sending R1 updates.

Step 3: Use the `debug ip rip` command to view the RIP messages being sent and received.

Rip updates are sent every 30 seconds so you may have to wait for debug information to be displayed.

```
R1#debug ip rip
R1#RIP: received v1 update from 192.168.2.2 on Serial0/0/0
      192.168.3.0 in 1 hops
      192.168.4.0 in 1 hops
      192.168.5.0 in 2 hops
RIP: sending  v1 update to 255.255.255.255 via FastEthernet0/0 (192.168.1.1)
RIP: build update entries
      network 192.168.2.0 metric 1
      network 192.168.3.0 metric 2
      network 192.168.4.0 metric 2
      network 192.168.5.0 metric 3
RIP: sending  v1 update to 255.255.255.255 via Serial0/0/0 (192.168.2.1)
RIP: build update entries
      network 192.168.1.0 metric 1
```

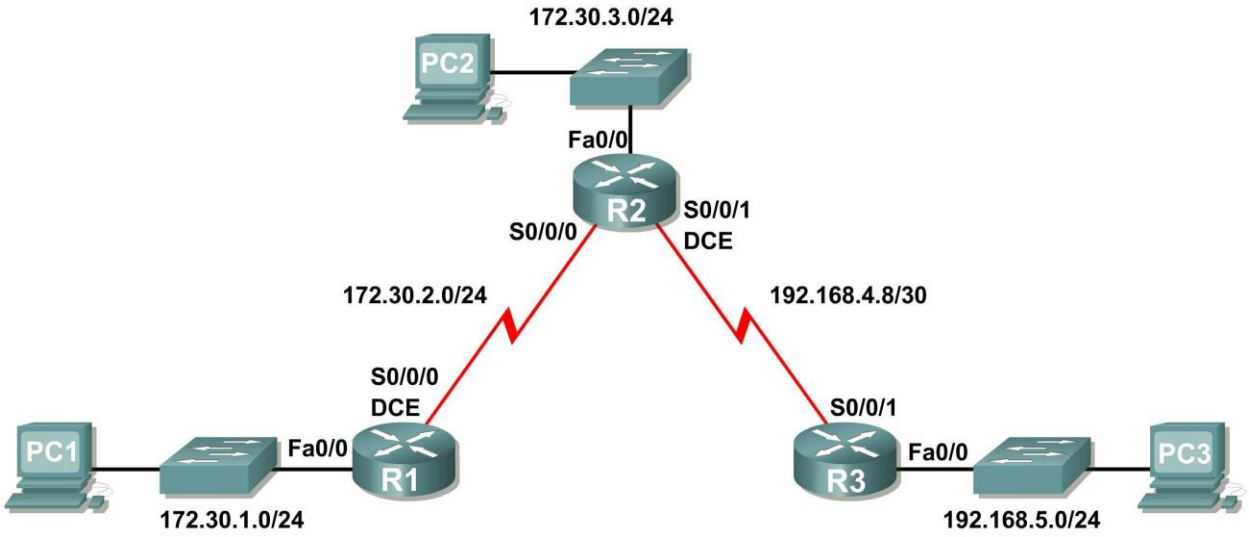
The debug output shows that R1 receives an update from R2. Notice how this update includes all the networks that R1 does not already have in its routing table. Because the FastEthernet0/0 interface belongs to the 192.168.1.0 network configured under RIP, R1 builds an update to send out that interface. The update includes all networks known to R1 except the network of the interface. Finally, R1 builds an update to send to R2. Because of split horizon, R1 only includes the 192.168.1.0 network in the update.

Step 4: Discontinue the debug output with the `undebug all` command.

```
R1#undebug all
All possible debugging has been turned off
```

Scenario B: Running RIPv1 with Subnets and Between Classfull Networks

Topology Diagram



Addressing Table

Device	Interface	IP Address	Subnet Mask	Default Gateway
R1	**Fa0/0**	172.30.1.1	255.255.255.0	N/A
	S0/0/0	172.30.2.1	255.255.255.0	N/A
R2	**Fa0/0**	172.30.3.1	255.255.255.0	N/A
	S0/0/0	172.30.2.2	255.255.255.0	N/A
	S0/0/1	192.168.4.9	255.255.255.252	N/A
R3	**Fa0/0**	192.168.5.1	255.255.255.0	N/A
	S0/0/1	192.168.4.10	255.255.255.252	N/A
PC1	**NIC**	172.30.1.10	255.255.255.0	172.30.1.1
PC2	**NIC**	172.30.3.10	255.255.255.0	172.30.3.1
PC3	**NIC**	192.168.5.10	255.255.255.0	192.168.5.1

Task 1: Make Changes between Scenario A and Scenario B

Step 1: Change the IP addressing on the interfaces as shown in the Topology Diagram and the Addressing Table.

Sometimes when changing the IP address on a serial interface, you may need to reset that interface by using the **`shutdown`** command, waiting for the `LINK-5-CHANGED` message, and then using the **`no shutdown`** command. This process will force the IOS to starting using the new IP address.

```
R1(config)#int s0/0/0
R1(config-if)#ip add 172.30.2.1 255.255.255.0
R1(config-if)#shutdown

%LINK-5-CHANGED: Interface Serial0/0/0, changed state to administratively down
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0/0, changed state to down
R1(config-if)#no shutdown

%LINK-5-CHANGED: Interface Serial0/0/0, changed state to up
R1(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0/0, changed state to up
```

Step 2: Verify that routers are active.

After reconfiguring all the interfaces on all three routers, verify that all necessary interfaces are active with the **show ip interface brief** command.

Step 3: Remove the RIP configurations from each router.

Although you can remove the old **network** commands with the **no** version of the command, it is more efficient to simply remove RIP and start over. Remove the RIP configurations from each router with the **no router rip** global configuration command. This will remove all the RIP configuration commands including the **network** commands.

```
R1(config)#no router rip

R2(config)#no router rip

R3(config)#no router rip
```

Task 2: Configure RIP

Step 1: Configure RIP routing on R1 as shown below.

```
R1(config)#router rip
R1(config-router)#network 172.30.0.0
```

Notice that only a single network statement is needed for R1. This statement includes both interfaces on different subnets of the 172.30.0.0 major network.

Step 2: Configure R1 to stop sending updates out the FastEthernet0/0 interface.

Sending updates out this interface wastes the bandwidth and processing resources of all devices on the LAN. In addition, advertising updates on a broadcast network is a security risk. RIP updates can be intercepted with packet sniffing software. Routing updates can be modified and sent back to the router, corrupting the router table with false metrics that misdirects traffic.

The **passive-interface fastethernet 0/0** command is used to disable sending RIPv1 updates out that interface. When you are finished with the RIP configuration, return to privileged EXEC mode and save the current configuration to NVRAM.

```
R1(config-router)#passive-interface fastethernet 0/0
R1(config-router)#end
%SYS-5-CONFIG_I: Configured from console by console
R1#copy run start
```

Step 3: Configure RIP routing on R2 as shown below.

```
R2(config)#router rip
R2(config-router)#network 172.30.0.0
R2(config-router)#network 192.168.4.0
R2(config-router)#passive-interface fastethernet 0/0
R2(config-router)#end
%SYS-5-CONFIG_I: Configured from console by console
R2#copy run start
```

Again notice that only a single network statement is needed for the two subnets of 172.30.0.0. This statement includes both interfaces, on different subnets, of the 172.30.0.0 major network. The network for the WAN link between R2 and R3 is also configured.

When you are finished with the RIP configuration, return to privileged EXEC mode and save the current configuration to NVRAM.

Step 4: Configure RIP routing on R3 as shown below.

```
R3(config)#router rip
R3(config-router)#network 192.168.4.0
R3(config-router)#network 192.168.5.0
R3(config-router)#passive-interface fastethernet 0/0
R3(config-router)#end
%SYS-5-CONFIG_I: Configured from console by console
R3#copy run start
```

When you are finished with the RIP configuration, return to privileged EXEC mode and save the current configuration to NVRAM.

Task 3: Verify RIP Routing

Step 1: Use the `show ip route` command to verify that each router has all of the networks in the topology in the routing table.

```
R1#show ip route

<Output omitted>

     172.30.0.0/24 is subnetted, 3 subnets
C       172.30.1.0 is directly connected, FastEthernet0/0
C       172.30.2.0 is directly connected, Serial0/0/0
R       172.30.3.0 [120/1] via 172.30.2.2, 00:00:22, Serial0/0/0
R    192.168.4.0/24 [120/1] via 172.30.2.2, 00:00:22, Serial0/0/0
R    192.168.5.0/24 [120/2] via 172.30.2.2, 00:00:22, Serial0/0/0
R1#
```

Note: RIPv1 is a classful routing protocol. Classful routing protocols do not send the subnet mask with network in routing updates. For example, 172.30.1.0 is sent by R2 to R1 without any subnet mask information.

```
R2#show ip route

<Output omitted>

     172.30.0.0/24 is subnetted, 3 subnets
R       172.30.1.0 [120/1] via 172.30.2.1, 00:00:04, Serial0/0/0
C       172.30.2.0 is directly connected, Serial0/0/0
C       172.30.3.0 is directly connected, FastEthernet0/0
     192.168.4.0/30 is subnetted, 1 subnets
C       192.168.4.8 is directly connected, Serial0/0/1
R    192.168.5.0/24 [120/1] via 192.168.4.10, 00:00:19, Serial0/0/1
R2#

R3#show ip route

<Output omitted>

R    172.30.0.0/16 [120/1] via 192.168.4.9, 00:00:22, Serial0/0/1
     192.168.4.0/30 is subnetted, 1 subnets
C       192.168.4.8 is directly connected, Serial0/0/1
C    192.168.5.0/24 is directly connected, FastEthernet0/0
```

Step 2: Verify that all necessary interfaces are active.

If one or more routing tables does not have a converged routing table, first make sure that all necessary interfaces are active with **`show ip interface brief`**.

Then use **`show ip protocols`** to verify the RIP configuration. Notice in the output from this command that the FastEthernet0/0 interface is no longer listed under **Interface** but is now listed under a new section of the output: **Passive Interface(s)**.

```
R1#show ip protocols
Routing Protocol is "rip"
  Sending updates every 30 seconds, next due in 20 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Redistributing: rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    Serial0/1/0           2     2
  Automatic network summarization is in effect
  Maximum path: 4
  Routing for Networks:
    172.30.0.0
    209.165.200.0
Passive Interface(s):
      FastEthernet0/0
  Routing Information Sources:
    Gateway         Distance      Last Update
    209.165.200.229      120      00:00:15
  Distance: (default is 120)
```

Step 3: View the RIP messages being sent and received.

To view the RIP messages being sent and received use the **debug ip rip** command. Notice that RIP updates are not sent out of the fa0/0 interface because of the **passive-interface fastethernet 0/0** command.

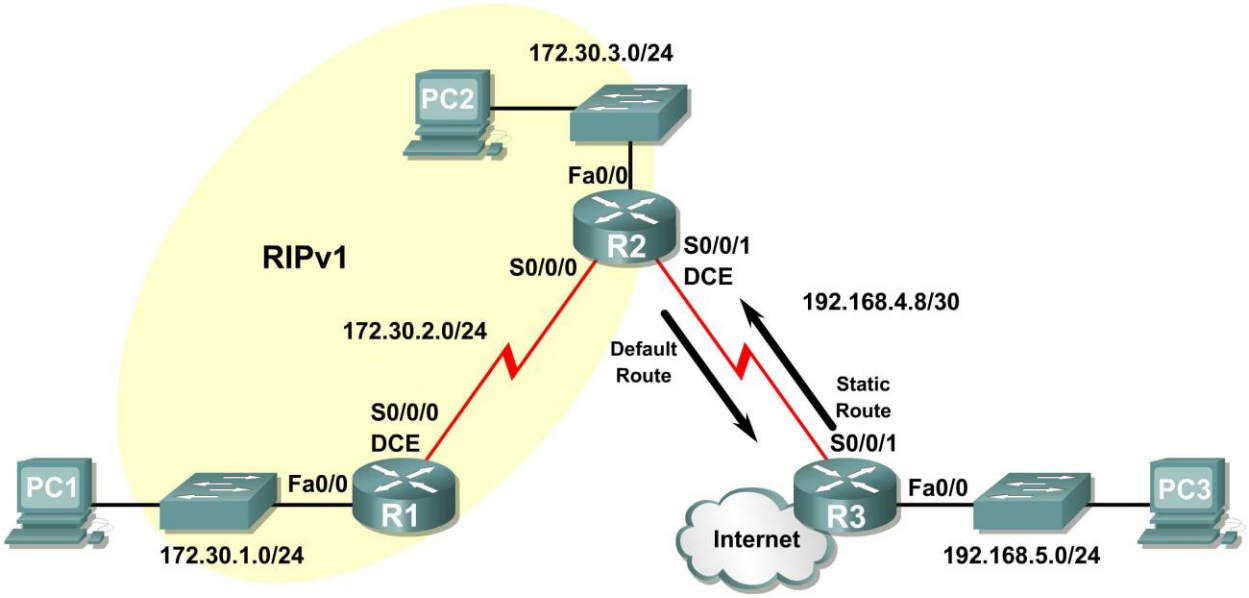
```
R1#debug ip rip
R1#RIP: sending  v1 update to 255.255.255.255 via Serial0/0/0 (172.30.2.1)
RIP: build update entries
      network 172.30.1.0 metric 1
RIP: received v1 update from 172.30.2.2 on Serial0/0/0
      172.30.3.0 in 1 hops
```

Step 4: Discontinue the debug output with the undebug all command.

```
R1#undebug all
All possible debugging has been turned off
```

Scenario C: Running RIPv1 on a Stub Network

Topology Diagram



Background

In this scenario we will modify Scenario B to only run RIP between R1 and R2. Scenario C is a typical configuration for most companies connecting a stub network to a central headquarters router or an ISP. Typically, a company runs a dynamic routing protocol (RIPv1 in our case) within the local network but finds it unnecessary to run a dynamic routing protocol between the company's gateway router and the ISP. For example, colleges with multiple campuses often run a dynamic routing protocol between campuses but use default routing to the ISP for access to the Internet. In some cases, remote campuses may even use default routing to the main campus, choosing to use dynamic routing only locally.

To keep our example simple, for Scenario C, we left the addressing intact from Scenario B. Let's assume that R3 is the ISP for our Company XYZ, which consists of the R1 and R2 routers using the 172.30.0.0/16 major network, subnetted with a /24 mask. Company XYZ is a stub network, meaning that there is only one way in and one way out of the 172.30.0.0/16 network—in via R2 (the gateway router) and out via R3 (the ISP). It doesn't make sense for R2 to send R3 RIP updates for the 172.30.0.0 network every 30 seconds, because R3 has no other way to get to 172.30.0.0 except through R2. It makes more sense for R3 to have a static route configured for the 172.30.0.0/16 network pointing to R2.

How about traffic from Company XYZ toward the Internet? It makes no sense for R3 to send over 120,000 summarized Internet routes to R2. All R2 needs to know is that if a packet is not destined for a host on the 172.30.0.0 network, then it should send the packet to the ISP, R3. This is the same for all other Company XYZ routers (only R1 in our case). They should send all traffic not destined for the 172.30.0.0 network to R2. R2 would then forward the traffic to R3.

Task 1: Make Changes between Scenario B and Scenario C.

Step 1: Remove network 192.168.4.0 from the RIP configuration for R2.

Remove network 192.168.4.0 from the RIP configuration for R2, because no updates will be sent between R2 and R3 and we don't want to advertise the 192.168.4.0 network to R1.

```
R2(config)#router rip
R2(config-router)#no network 192.168.4.0
```

Step 2: Completely remove RIP routing from R3.

```
R3(config)#no router rip
```

Task 2: Configure the Static Route on R3 for the 172.30.0.0/16 network.

Because R3 and R2 are not exchanging RIP updates, we need to configure a static route on R3 for the 172.30.0.0/16 network. This will send all 172.30.0.0/16 traffic to R2.

```
R3(config)#ip route 172.30.0.0 255.255.252.0 serial0/0/1
```

Task 3: Configure a Default Static Route on R2.

Step 1: Configure R2 to send default traffic to R3.

Configure a default static route on R2 that will send all default traffic—packets with destination IP addresses that do not match a specific route in the routing table—to R3.

```
R2(config)# ip route 0.0.0.0 0.0.0.0 serial 0/0/1
```

Step 2: Configure R2 to send default static route information to R1.

The **default-information originate** command is used to configure R2 to include the default static route with its RIP updates. Configure this command on R2 so that the default static route information is sent to R1.

```
R2(config)#router rip
R2(config-router)#default-information originate
R2(config-router)#
```

Note: Sometimes it is necessary to clear the RIP routing process before the **default-information originate** command will work. First, try the command **clear ip route *** on both R1 and R2. This command will cause the routers to immediately flush routes in the routing table and request updates from each other. Sometimes this does not work with RIP. If the default route information is still not sent to R1, save the configuration on R1 and R2 and then reload both routers. Doing this will reset the hardware and both routers will restart the RIP routing process.

Task 4: Verify RIP Routing.

Step 1: Use the show ip route command to view the routing table on R2 and R1.

```
R2#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 0.0.0.0 to network 0.0.0.0

     172.30.0.0/24 is subnetted, 3 subnets
C       172.30.2.0 is directly connected, Serial0/0/0
```

```
C       172.30.3.0 is directly connected, FastEthernet0/0
R       172.30.1.0 [120/1] via 172.30.2.1, 00:00:16, Serial0/0/0
     192.168.4.0/30 is subnetted, 1 subnets
C       192.168.4.8 is directly connected, Serial0/0/1
S*   0.0.0.0/0 is directly connected, Serial0/0/1
```

Notice that R2 now has a static route tagged as a **candidate default**.

```
R1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route
Gateway of last resort is 172.30.2.2 to network 0.0.0.0

     172.30.0.0/24 is subnetted, 3 subnets
C       172.30.2.0 is directly connected, Serial0/0/0
R       172.30.3.0 [120/1] via 172.30.2.2, 00:00:05, Serial0/0/0
C       172.30.1.0 is directly connected, FastEthernet0/0
R*   0.0.0.0/0 [120/1] via 172.30.2.2, 00:00:19, Serial0/0/0
```

Notice that R1 now has a RIP route tagged as a **candidate default** route. The route is the “quad-zero” default route sent by R2. R1 will now send default traffic to the **Gateway of last resort** at 172.30.2.2, which is the IP address of R2.

Step 2: View the RIP updates that are sent and received on R1 with the `debug ip rip` command.

```
R1#debug ip rip
RIP protocol debugging is on
R1#RIP: sending  v1 update to 255.255.255.255 via Serial0/0/0 (172.30.2.1)
RIP: build update entries
      network 172.30.1.0 metric 1
RIP: received v1 update from 172.30.2.2 on Serial0/0/0
      0.0.0.0 in 1 hops
      172.30.3.0 in 1 hops
```

Notice that R1 is receiving the default route from R2.

Step 3: Discontinue the debug output with the `undebug all` command.

```
R1#undebug all
All possible debugging has been turned off
```

Step 4: Use the `show ip route` command to view the routing table on R3.

```
R3#show ip route

<Output omitted>
S    172.30.0.0/16 is directly connected, Serial0/0/1
     192.168.4.0/30 is subnetted, 1 subnets
C       192.168.4.8 is directly connected, Serial0/0/1
C    192.168.5.0/24 is directly connected, FastEthernet0/0
```

Notice that RIP is not being used on R3. The only route that is not directly connected is the static route.

Task 5: Document the Router Configurations

On each router, capture the following command output to a text file and save for future reference:

- Running configuration
- Routing table
- Interface summarization
- Output from **show ip protocols**

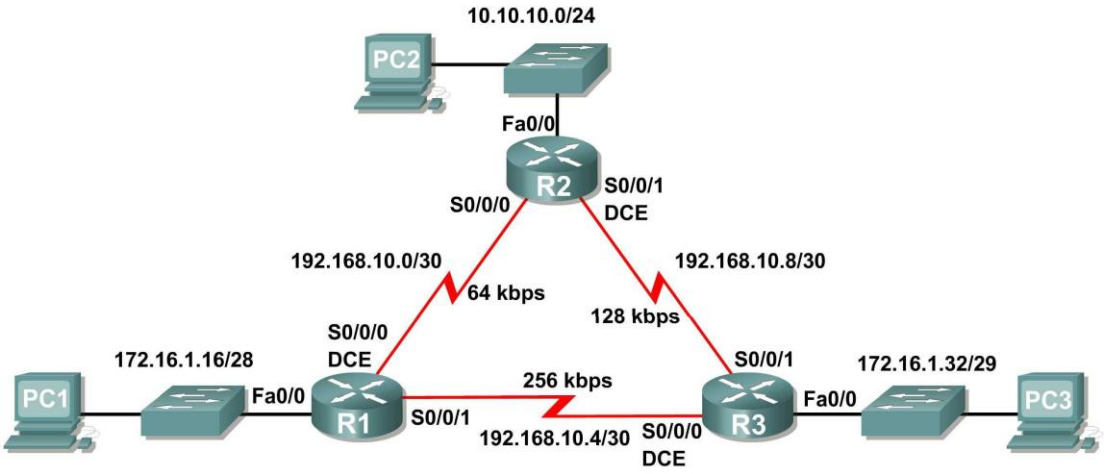
Task 6: Clean Up

Erase the configurations and reload the routers. Disconnect and store the cabling. For PC hosts that are normally connected to other networks (such as the school LAN or to the Internet), reconnect the appropriate cabling and restore the TCP/IP settings.

5.Configure a Network topology using Link State Routing protocol

Basic OSPF Configuration Lab

Topology Diagram



Addressing Table

Device	Interface	IP Address	Subnet Mask	Default Gateway
R1	Fa0/0	172.16.1.17	255.255.255.240	N/A
	S0/0/0	192.168.10.1	255.255.255.252	N/A
	S0/0/1	192.168.10.5	255.255.255.252	N/A
R2	Fa0/0	10.10.10.1	255.255.255.0	N/A
	S0/0/0	192.168.10.2	255.255.255.252	N/A
	S0/0/1	192.168.10.9	255.255.255.252	N/A
R3	Fa0/0	172.16.1.33	255.255.255.248	N/A
	S0/0/0	192.168.10.6	255.255.255.252	N/A
	S0/0/1	192.168.10.10	255.255.255.252	N/A
PC1	NIC	172.16.1.20	255.255.255.240	172.16.1.17
PC2	NIC	10.10.10.10	255.255.255.0	10.10.10.1
PC3	NIC	172.16.1.35	255.255.255.248	172.16.1.33

Step 1: Configure the routers
On the routers, enter global configuration mode and configure the hostname as shown on the chart. Then configure the console, virtual terminal lines password (both "cisco") and privileged EXEC password ("class"):

Step 2: Disable DNS lookup
Router(config)#no ip domain-lookup

Step 3: Configure the interfaces on R1, R2, and R3

Configure the interfaces on the R1, R2, and R3 routers with the IP addresses from the table under the Topology Diagram.

Step 4: Verify IP addressing and interfaces
Use the show ip interface brief command to verify that the IP addressing is correct and that the interfaces are active.

Step 5: Configure Ethernet interfaces of PC1, PC2, and PC3
Configure the Ethernet interfaces of PC1, PC2, and PC3 with the IP addresses and default gateways from the table under the Topology Diagram.

Task: Configure OSPF on the R1 Router
Step 1: Use the router ospf command in global configuration mode to enable OSPF on the R1 router. Enter a process ID of 1 for the *process-ID* parameter.

```
R1(config)#router ospf 1
R1(config-router)#
```

Step 2: Configure the `network` statement for the LAN network.
Once you are in the Router OSPF configuration sub-mode, configure the LAN network 172.16.1.16/28 to be included in the OSPF updates that are sent out of R1.

The OSPF network command uses a combination of *network-address* and *wildcard-mask* similar to that which can be used by EIGRP. Unlike EIGRP, the wildcard mask in OSPF is required.

Use an area ID of 0 for the OSPF *area-id* parameter. 0 will be used for the OSPF area ID in all of the network statements in this topology.

```
R1(config-router)#network 172.16.1.16 0.0.0.15 area 0
R1(config-router)#
```

Step 3: Configure the router to advertise the 192.168.10.0/30 network attached to the Serial0/0/0 interface.

```
R1(config-router)#network 192.168.10.0 0.0.0.3 area 0
R1(config-router)#
```

Step 4: Configure the router to advertise the 192.168.10.4/30 network attached to the Serial0/0/1 interface.

```
R1(config-router)#network 192.168.10.4 0.0.0.3 area 0
R1(config-router)#
```

Step 5: When you are finished with the OSPF configuration for R1, return to privileged EXEC mode.

```
R1(config-router)#end
%SYS-5-CONFIG_I: Configured from console by consoleR1#
```

Task: Configure OSPF on the R2 and R3 Routers

Step 1: Enable OSPF routing on the R2 router using the router ospf command.
Use a process ID of 1.

```
R2(config)#router ospf 1
R2(config-router)#
```

Step 2: Configure the router to advertise the LAN network 10.10.10.0/24 in the OSPFupdates.

```
R2(config-router)#network 10.10.10.0 0.0.0.255 area 0
R2(config-router)#
```

Step 3: Configure the router to advertise the 192.168.10.0/30 network attached to theSerial0/0/0 interface.

```
R2(config-router)#network 192.168.10.0 0.0.0.3 area 0
R2(config-router)#
00:07:27: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.10.5 on Serial0/0/0
from EXCHANGE to FULL, Exchange Done
```

Notice that when the network for the serial link from R1 to R2 is added to the OSPF configuration, the router sends a notification message to the console stating that a neighbor relationship with another OSPF router has been established.

Step 4: Configure the router to advertise the 192.168.10.8/30 network attached to theSerial0/0/1 interface. When you are finished, return to privileged EXEC mode.

```
R2(config-router)#network 192.168.10.8 0.0.0.3 area 0
R2(config-router)#end
%SYS-5-CONFIG_I: Configured from console by consoleR2#
```

Step 5: Configure OSPF on the R3 router using the router ospfand network commands.
Use a process ID of 1. Configure the router to advertise the three directly connected networks. When you are finished, return to privileged EXEC mode.

```
R3(config)#router ospf 1
R3(config-router)#network 172.16.1.32 0.0.0.7 area 0
R3(config-router)#network 192.168.10.4 0.0.0.3 area 0
R3(config-router)#
00:17:46: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.10.5 on Serial0/0/0
from LOADING to FULL, Loading Done
R3(config-router)#network 192.168.10.8 0.0.0.3 area 0
R3(config-router)#
00:18:01: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.10.9 on Serial0/0/1
from EXCHANGE to FULL, Exchange DoneR3(config-router)#end
%SYS-5-CONFIG_I: Configured from console by consoleR3#
```

Notice that when the networks for the serial links from R3 to R1 and R3 to R2 are added to the OSPF configuration, the router sends a notification message to the console stating that a neighbor relationship with another OSPF router has been established.

Task: Configure OSPF Router IDs
The OSPF router ID is used to uniquely identify the router in the OSPF routing domain. A router ID is an IP address. Cisco routers derive the Router ID in one of three ways and with the following precedence:

IP address configured with the OSPF router-idcommand.

Highest IP address of any of the router's loopback addresses.
Highest active IP address on any of the router's physical interfaces.

Step 1: Examine the current router IDs in the topology.

Since no router IDs or loopback interfaces have been configured on the three routers, the router ID for each router is determined by the highest IP address of any active interface.

What is the router ID for R1? ___
What is the router ID for R2? ___
What is the router ID for R3? ___

The router ID can also be seen in the output of the show ip protocols, show ip ospf, and show ip ospf interfaces commands.

```
R3#show ip protocols
Routing Protocol is "ospf 1"
Outgoing update filter list for all interfaces is not set Incoming update filter list for all interfaces is not set
Router ID 192.168.10.10
Number of areas in this router is 1. 1 normal 0 stub 0 nssa Maximum path: 4

<output omitted>

R3#show ip ospf
Routing Process "ospf 1" with ID 192.168.10.10Supports only single TOS(TOS0) routes
Supports opaque LSA
SPF schedule delay 5 secs, Hold time between two SPFs 10 secs

<output omitted>
R3#show ip ospf interface
FastEthernet0/0 is up, line protocol is up Internet address is 172.16.1.33/29, Area 0
Process ID 1, Router ID 192.168.10.10, Network Type BROADCAST, Cost:
1
Transmit Delay is 1 sec, State DR, Priority 1
Designated Router (ID) 192.168.10.10, Interface address 172.16.1.33No backup designated router on this network
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5Hello due in 00:00:00
Index 1/1, flood queue length 0Next 0x0(0)/0x0(0)
Last flood scan length is 1, maximum is 1
Last flood scan time is 0 msec, maximum is 0 msec Neighbor Count is 0, Adjacent neighbor count is 0
Suppress hello for 0 neighbor(s)
<output omitted>

R3#
```

Step 2: Use loopback addresses to change the router IDs of the routers in the topology.

```
R1(config)#interface loopback 0
R1(config-if)#ip address 10.1.1.1          255.255.255.255
```

```
R2(config)#interface loopback 0
R2(config-if)#ip address 10.2.2.2 255.255.255.255

R3(config)#interface loopback 0
R3(config-if)#ip address 10.3.3.3 255.255.255.255
```

Step 3: Reload the routers to force the new Router IDs to be used.

When a new Router ID is configured, it will not be used until the OSPF process is restarted. Make sure that the current configuration is saved to NRAM, and then use the reloadcommand to restart each of the routers.

When the router is reloaded, what is the router ID for R1? __
When the router is reloaded, what is the router ID for R2? __
When the router is reloaded, what is the router ID for R3? __

Step 4: Use the show ip ospf neighbors command to verify that the router IDs have changed.

R1#show ip ospf neighbor

Neighbor ID	Pri	State	Dead Time	Address	Interface
10.3.3.3	0	FULL/	- 00:00:30	192.168.10.6	Serial0/0/1
10.2.2.2	0	FULL/	- 00:00:33	192.168.10.2	Serial0/0/0

R2#show ip ospf neighbor

Neighbor ID	Pri	State	Dead Time	Address	Interface
10.3.3.3	0	FULL/	- 00:00:36	192.168.10.10	Serial0/0/1
10.1.1.1	0	FULL/	- 00:00:37	192.168.10.1	Serial0/0/0

R3#show ip ospf neighbor

Neighbor ID	Pri	State	Dead Time	Address	Interface
10.2.2.2	0	FULL/	- 00:00:34	192.168.10.9	Serial0/0/1
10.1.1.1	0	FULL/	- 00:00:38	192.168.10.5	Serial0/0/0

Step 5: Use the `router-id` command to change the router ID on the R1 router.

Note: Some IOS versions do not support the router-idcommand. If this command is notavailable, continue to the next Task.

```
R1(config)#router ospf 1
R1(config-router)#router-id 10.4.4.4
```
Reload or use “clear ip ospf process” command, for this to take effect

If this command is used on an OSPF router process which is already active (has neighbors), the new router-ID is used at the next reload or at a manual OSPF process restart. To manually restart the OSPF process, use the clear ip ospf processcommand.

R1#(config-router)#end R1#clear ip ospf process
Reset ALL OSPF processes? [no]:yesR1#

Step 6: Use the show ip ospf neighborcommand on router R2 to verify that the routerID of R1 has been changed.

R2#show ip ospf neighbor

Neighbor ID	Pri	State	Dead Time	Address	Interface
10.3.3.3	0	FULL/ -	00:00:36	192.168.10.10	Serial0/0/1
10.4.4.4	0	FULL/ -	00:00:37	192.168.10.1	Serial0/0/0

Step 7: Remove the configured router ID with the noform of the router-idcommand.

R1(config)#router ospf 1
R1(config-router)#router-id 10.4.4.4
Reload or use "clear ip ospf process" command, for this to take effect

Step 8: Restart the OSPF process using the clear ip ospf processcommand.

Restarting the OSPF process forces the router to use the IP address configured on the Loopback 0 interface as the Router ID.

R1(config-router)#end R1#clear ip ospf process
Reset ALL OSPF processes? [no]:yesR1#

Task: Verify OSPF Operation.

Step 1: On the R1 router, Use the show ip ospf neighborcommand to view the information about the OSPF neighbor routers R2 and R3. You should be able to see the neighbor ID and IP address of each adjacent router, and the interface that R1 uses to reach thatOSPF neighbor.

R1#show ip ospf neighbor

Neighbor ID	Pri	State	Dead Time	Address	Interface
10.2.2.2	0	FULL/-	00:00:32	192.168.10.2	Serial0/0/0
10.3.3.3	0	FULL/-	00:00:32	192.168.10.6	Serial0/0/1

R1#

Step 2: On the R1 router, use the show ip protocols command to view information about the routing protocol operation.

Notice that the information that was configured in the previous Tasks, such as protocol, process ID, neighbor ID, and networks, is shown in the output. The IP addresses of the adjacent neighbors are also shown.

R1#show ip protocols

Routing Protocol is "ospf 1"
Outgoing update filter list for all interfaces is not set Incoming update filter list for all interfaces is not set
Router ID 10.1.1.1
Number of areas in this router is 1. 1 normal 0 stub 0 nssa Maximum path: 4
Routing for Networks: 172.16.1.16 0.0.0.15 area 0
192.168.10.0 0.0.0.3 area 0
192.168.10.4 0.0.0.3 area 0
Routing Information Sources:
Gateway Distance Last Update
10.2.2.2 110 00:11:43
10.3.3.3 110 00:11:43
Distance: (default is 110)R1#

Notice that the output specifies the process ID used by OSPF. Remember, the process ID mustbe the same on all routers for OSPF to establish neighbor adjacencies and share routing information.

Task: Examine OSPF Routes in the Routing Tables

View the routing table on the R1 router. OSPF routes are denoted in the routing table with an “O”.

R1#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B -BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter areaN1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
inter area
* - candidate default, U - per-user static route, o - ODRP - periodic downloaded static route

Gateway of last resort is not set

10.0.0.0/8 is variably subnetted, 2 subnets, 2 masksC 10.1.1.1/32 is directly connected, Loopback0
O 10.10.10.0/24 [110/65] via 192.168.10.2, 00:01:02, Serial0/0/0
172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks C 172.16.1.16/28 is directly connected, FastEthernet0/0
O 172.16.1.32/29 [110/65] via 192.168.10.6, 00:01:12, Serial0/0/1
192.168.10.0/30 is subnetted, 3 subnets
C 192.168.10.0 is directly connected, Serial0/0/0
C 192.168.10.4 is directly connected, Serial0/0/1
O 192.168.10.8 [110/128] via 192.168.10.6, 00:01:12, Serial0/0/1
[110/128] via 192.168.10.2, 00:01:02, Serial0/0/0
R1#

Notice that unlike RIPv2 and EIGRP, OSPF does not automatically summarize at major networkboundaries.

Task: Configure OSPF Cost

Step 1: Use the `show ip route` command on the R1 router to view the OSPF cost toreach the 10.10.10.0/24 network.

R1#**show ip route**

<output omitted>

10.0.0.0/8 is variably subnetted, 2 subnets, 2 masksC 10.1.1.1/32 is directly connected, Loopback0
O 10.10.10.0/24 [110/65] via 192.168.10.2, 00:16:56, Serial0/0/0
172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks C 172.16.1.16/28 is directly connected, FastEthernet0/0
O 172.16.1.32/29 [110/65] via 192.168.10.6, 00:17:06, Serial0/0/1

```
        192.168.10.0/30     is subnetted, 3 subnets
C       192.168.10.0        is directly connected, Serial0/0/0
C       192.168.10.4        is directly connected, Serial0/0/1
O       192.168.10.8        [110/128] via 192.168.10.6, 00:17:06, Serial0/0/1
                            [110/128] via 192.168.10.2, 00:16:56, Serial0/0/0
R1#
```

Step 2: Use the `show interfaces serial0/0/0` command on the R1 router to view thebandwidth of the Serial 0/0/0 interface.

R1#**show interfaces serial0/0/0**
Serial0/0/0 is up, line protocol is up (connected)Hardware is HD64570
Internet address is 192.168.10.1/30
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load1/255
Encapsulation HDLC, loopback not set, keepalive set (10 sec)Last input never, output never, output hang never
Last clearing of "show interface" counters never
Input queue: 0/75/0 (size/max/drops); Total output drops: 0

<output omitted>

On most serial links, the bandwidth metric will default to 1544 Kbits. If this is not the actual bandwidth of the serial link, the bandwidth will need to be changed so that the OSPF cost can becalculated correctly.

Step 3: Use the bandwidth command to change the bandwidth of the serial interfaces ofthe R1 and R2 routers to the actual bandwidth, 64 kbps.

R1 router:
R1(config)#**interface serial0/0/0** R1(config-if)#**bandwidth 64** R1(config-if)#**interface serial0/0/1**R1(config-if)#**bandwidth 64**

R2 router:
R2(config)#interface serial0/0/0
R2(config-if)#bandwidth 64
R2(config)#interface serial0/0/1
R2(config-if)#bandwidth 64

Step 4: Use the show ip ospf interfacecommand on the R1 router to verify the costof the serial links.
The cost of each of the Serial links is now 1562, the result of the calculation: 10^8/64,000 bps.

R1#show ip ospf interface

<output omitted>

Serial0/0/0 is up, line protocol is up Internet address is 192.168.10.1/30, Area 0
Process ID 1, Router ID 10.1.1.1, Network Type POINT-TO-POINT, Cost:1562
Transmit Delay is 1 sec, State POINT-TO-POINT,
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5Hello due in 00:00:05
Index 2/2, flood queue length 0Next 0x0(0)/0x0(0)
Last flood scan length is 1, maximum is 1
Last flood scan time is 0 msec, maximum is 0 msec Neighbor Count is 1 , Adjacent neighbor count is 1
Adjacent with neighbor 10.2.2.2Suppress hello for 0 neighbor(s)
Serial0/0/1 is up, line protocol is up Internet address is 192.168.10.5/30, Area 0
Process ID 1, Router ID 10.1.1.1, Network Type POINT-TO-POINT, Cost:1562
Transmit Delay is 1 sec, State POINT-TO-POINT,

<output omitted>

Step 5: Use the ip ospf cost command to configure the OSPF cost on the R3 router. An alternative method to using the bandwidth command is to use the ip ospf cost command, which allows you to directly configure the cost. Use the ip ospf cost command to change the bandwidth of the serial interfaces of the R3 router to 1562.

```
R3(config)#interface serial0/0/0
R3(config-if)#ip ospf cost 1562
R3(config-if)#interface  serial0/0/1
R3(config-if)#ip ospf cost 1562
```

Step 6: Use the show ip ospf interfacecommand on the R3 router to verify that thecost of the link the cost of each of the Serial links is now 1562.

```
R3#show ip ospf interface
```

<output omitted>

Serial0/0/1 is up, line protocol is up Internet address is 192.168.10.10/30, Area 0
Process ID 1, Router ID 10.3.3.3, Network Type POINT-TO-POINT, Cost:1562
Transmit Delay is 1 sec, State POINT-TO-POINT,
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5Hello due in 00:00:06
Index 2/2, flood queue length 0Next 0x0(0)/0x0(0)
Last flood scan length is 1, maximum is 1
Last flood scan time is 0 msec, maximum is 0 msec Neighbor Count is 1 , Adjacent neighbor count is 1
Adjacent with neighbor 10.2.2.2Suppress hello for 0 neighbor(s)
Serial0/0/0 is up, line protocol is up Internet address is 192.168.10.6/30, Area 0
Process ID 1, Router ID 10.3.3.3, Network Type POINT-TO-POINT, Cost:1562
Transmit Delay is 1 sec, State POINT-TO-POINT,

<output omitted>

Task: Redistribute an OSPF Default Route

Step 1: Configure a loopback address on the R1 router to simulate a link to an ISP.

```
R1(config)#interface loopback1
```

%LINK-5-CHANGED: Interface Loopback1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback1, changedstate to up

R1(config-if)#**ip address 172.30.1.1 255.255.255.252**

Step 2: Configure a static default route on the R1 router.

Use the loopback address that ha been configured to simulate a link to an ISP as the exitinterface.

R1(config)#**ip route 0.0.0.0 0.0.0.0 loopback1**
R1(config)#

Step 3: Use the default-information originatecommand to include the static routein the OSPF updates that are sent from the R1 router.

R1(config)#**router ospf 1**
R1(config-router)#**default-information originate**
R1(config-router)#
Step 4: View the routing table on the R2 router to verify that the static default route isbeing redistributed via OSPF.

R2#**show ip route**

<output omitted>
Gateway of last resort is 192.168.10.1 to network 0.0.0.010.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C 10.2.2.2/32 is directly connected, Loopback0
C 10.10.10.0/24 is directly connected, FastEthernet0/0172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks
O 172.16.1.16/28 [110/1563] via 192.168.10.1, 00:29:28,
Serial0/0/0
O 172.16.1.32/29 [110/1563] via 192.168.10.10, 00:29:28,
Serial0/0/1
192.168.10.0/30 is subnetted, 3 subnets
C 192.168.10.0 is directly connected, Serial0/0/0
O 192.168.10.4 [110/3124] via 192.168.10.10, 00:25:56,
Serial0/0/1
[110/3124] via 192.168.10.1, 00:25:56, Serial0/0/0
C 192.168.10.8 is directly connected, Serial0/0/1
O*E2 0.0.0.0/0 [110/1] via 192.168.10.1, 00:01:11, Serial0/0/0R2#

Task: Configure Additional OSPF Features
Step 1: Use the auto-cost reference-bandwidthcommand to adjust the referencebandwidth value.
Increase the reference bandwidth to 10000 to simulate 10GigE speeds. Configure this command on all routers in the OSPF routing domain.

R1(config-router)#**auto-cost reference-bandwidth 10000**
% OSPF: Reference bandwidth is changed.
Please ensure reference bandwidth is consistent across allrouters.

R2(config-router)#**auto-cost reference-bandwidth 10000**
% OSPF: Reference bandwidth is changed.
Please ensure reference bandwidth is consistent across allrouters.

R3(config-router)#**auto-cost reference-bandwidth 10000**
% OSPF: Reference bandwidth is changed.
Please ensure reference bandwidth is consistent across allrouters.

Step 2: Examine the routing table on the R1 router to verify the change in the OSPF costmetric.
Notice that the values are much larger cost values for OSPF routes.

R1#show ip route

<output omitted>

Gateway of last resort is 0.0.0.0 to network 0.0.0.0 10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C 10.1.1.1/32 is directly connected, Loopback0
O 10.10.10.0/24 [110/65635] via 192.168.10.2, 00:01:01,
Serial0/0/0
172.16.0.0/16 is variably subnetted, 2 subnets, 2 masks C 172.16.1.16/28 is directly connected, FastEthernet0/0
O 172.16.1.32/29 [110/65635] via 192.168.10.6, 00:00:51,
Serial0/0/1
172.30.0.0/30 is subnetted, 1 subnets
C 172.30.1.0 is directly connected, Loopback1192.168.10.0/30 is subnetted, 3 subnets
C 192.168.10.0 is directly connected, Serial0/0/0C 192.168.10.4 is directly connected, Serial0/0/1
O 192.168.10.8 [110/67097] via 192.168.10.2, 00:01:01,
Serial0/0/0
S* 0.0.0.0/0 is directly connected, Loopback1R1#

Step 3: Use the show ip ospf neighborcommand on R1 to view the Dead Time counter.
The Dead Time counter is counting down from the default interval of 40 seconds.

R1#**show ip ospf neighbor**

Neighbor ID	Interface	Pri	State	Dead Time	Address
10.2.2.2	Serial0/0/0	0	FULL/-	00:00:34	192.168.10.2
10.3.3.3	Serial0/0/1	0	FULL/-	00:00:34	192.168.10.6

Step 4: Configure the OSPF Hello and Dead intervals.
The OSPF Hello and Dead intervals can be modified manually using the **ip ospf hello- interval**and **ip ospf dead-interval**interface commands. Use these commands to change the hello interval to 5 seconds and the dead interval to 20 seconds on the Serial 0/0/0interface of the R1 router.

R1(config)#**interface serial0/0/0**
R1(config-if)#**ip ospf hello-interval 5**
R1(config-if)#**ip ospf dead-interval 20**
R1(config-if)#
01:09:04: %OSPF-5-ADJCHG: Process 1, Nbr 10.2.2.2 on Serial0/0/0 fromFULL to DOWN, Neighbor Down: Dead timer expired
01:09:04: %OSPF-5-ADJCHG: Process 1, Nbr 10.2.2.2 on Serial0/0/0 fromFULL to Down: Interface down or detached

After 20 seconds the Dead Timer on R1 expires. R1 and R2 loose adjacency because the Dead Timer and Hello Timers must be configured identically on each side of the serial link between R1 and R2.

Step 5: Modify the Dead Timer and Hello Timer intervals.
Modify the Dead Timer and Hello Timer intervals on the Serial 0/0/0 interface in the R2 router to match the intervals configured on the Serial 0/0/0 interface of the R1 router.

```
R2(config)#interface serial0/0/0
R2(config-if)#ip ospf hello-interval 5
R2(config-if)#ip ospf dead-interval 20
R2(config-if)#
01:12:10: %OSPF-5-ADJCHG: Process 1, Nbr 10.1.1.1 on Serial0/0/0 from EXCHANGE to FULL, Exchange Done
```

Notice that the IOS displays a message when adjacency has been established with a state of Full.

Step 5: Use the show ip ospf interface serial0/0/0command to verify that theHello Timer and Dead Timer intervals have been modified.

```
R2#show ip ospf interface serial0/0/0
Serial0/0/0 is up, line protocol is up Internet address is 192.168.10.2/30, Area 0
Process ID 1, Router ID 10.2.2.2, Network Type POINT-TO-POINT, Cost:1562
Transmit Delay is 1 sec, State POINT-TO-POINT,
Timer intervals configured, Hello 5, Dead 20, Wait 20, Retransmit 5Hello due in 00:00:00
Index 3/3, flood queue length 0Next 0x0(0)/0x0(0)
Last flood scan length is 1, maximum is 1
Last flood scan time is 0 msec, maximum is 0 msec Neighbor Count is 1 , Adjacent neighbor count is 1
Adjacent with neighbor 10.1.1.1Suppress hello for 0 neighbor(s)
R2#
```

Step 6: Use the show ip ospf neighborcommand on R1 to verify that the neighboradjacency with R2 has been restored.

Notice that the Dead Time for Serial 0/0/0 is now much lower since it is counting down from 20 seconds instead of the default 40 seconds. Serial 0/0/1 is still operating with default timers.

```
R1#show ip ospf  neighbor
Neighbor        IDPri     State          Dead Time       Address
Interface
10.2.2.2          0       FULL/-         00:00:19        192.168.10.2
Serial0/0/0
10.3.3.3          0       FULL/-         00:00:34        192.168.10.6
Serial0/0/1
```

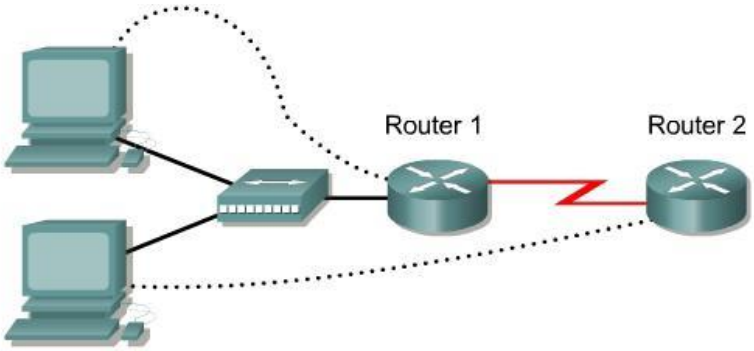
Task: Clean Up: Erase the configurations and disconnect attached cabling

6. Using CISCO Packet Tracer, Perform the followings

a. Network Address Translation (NAT)

b. Access Control List (ACLs)

NAT and PAT Configuration



Router Designation	Router Name	FastEthernet 0 Address/ Subnet Mask	Interface Type	Serial 0 Address/ Subnet Mask	Loopback 0 Address/ Subnet Mask	Enable Secret Password	Enable/VTY/ Console Passwords
Router 1	Gateway	10.10.10.1/24	DCE	200.2.2.18/30	NA	class	cisco
Router 2	ISP	NA	DTE	200.2.2.17/30	172.16.1.1/32	class	cisco

Objective

- Configure a router for Network Address Translation (NAT) and Port Address Translation (PAT)
- Test the configuration and verify NAT/PAT statistics

Step 1 Configure the routers

Configure all of the following according to the chart:

- The hostname
- The console
- The virtual terminal
- The enable passwords
- The interfaces

If problems occur during this configuration, refer to Lab 1.1.4a Configuring NAT.

Step 2 Save the configuration

At the privileged EXEC mode prompt, on both routers, type the command **`copy running-config startup-config`**.

Step 3 Configure the hosts with the proper IP address, subnet mask, and default gateway

Each workstation should be able to ping the attached router. If for some reason this is not the case, troubleshoot as necessary. Check and verify that the workstation has been assigned a specific IP

address and default gateway. If running Windows 98, check using **Start** > **Run** > **winipcfg**. If running Windows 2000 or higher, check using **ipconfig** in a DOS window.

Step 4 Verify that the network is functioning

a. From the attached hosts, ping the FastEthernet interface of the default gateway router.

b. Was the ping from the first host successful? _____

c. Was the ping from the second host successful? _____

d. If the answer is no for either question, troubleshoot the router and host configurations to find the error. Then ping again until they both are successful.

Step 5 Create a static route

a. Create a static route from the ISP to the Gateway router. Addresses 199.99.9.32/27 have been allocated for Internet access outside of the company. Use the **ip route** command to create the static route.

```
ISP(config)#ip route 199.99.9.32 255.255.224.0 200.2.2.18
```

b. Is the static route in the routing table? _____

c. What command checks the routing table contents? _____

d. If the route was not in the routing table, give one reason why this might be so?

_____

Step 6 Create a default route

a. Add a default route, using the **ip route** command, from the Gateway router to the ISP router. This will forward any unknown destination address traffic to the ISP:

```
Gateway(config)#ip route 0.0.0.0 0.0.0.0 200.2.2.17
```

b. Is the static route in the routing table? _____

c. Try to ping from one of the workstations to the ISP serial interface IP address.

d. Was the ping successful? _____

e. Why? _____

Step 7 Define the pool of usable public IP addresses

To define the pool of public addresses, use the **ip nat pool** command:

```
Gateway(config)#ip nat pool public-access 199.99.9.32 199.99.9.35
netmask 255.255.255.252
```

Step 8 Define an access list that will match the inside private IP addresses

To define the access list to match the inside private addresses, use the **access list** command:

```
Gateway(config)#access-list 1 permit 10.10.10.0 0.0.0.255
```

Step 9 Define the NAT translation from inside list to outside pool

To define the NAT translation, use the **ip nat inside source** command:

```
Gateway(config)#ip nat inside source list 1 pool public-access overload
```

Step 10 Specify the interfaces

The active interfaces on the router need to be identified as either inside or outside interfaces with respect to NAT. To do this, use the **ip nat inside** or **ip nat outside** command:

```
Gateway(config)#interface fastethernet 0
Gateway(config-if)#ip nat inside
Gateway(config-if)#interface serial 0
Gateway(config-if)#ip nat outside
```

Step 11 Testing the configuration

a. From the workstations, **ping 172.16.1.1**. Open multiple DOS windows on each workstation and Telnet to the 172.16.1.1 address. Next, view the NAT translations on the Gateway router, with the command **show ip nat translations**.

b. What is the translation of the inside local host addresses?

_____ = _____ _____ = _____

Step 12 Verify NAT and PAT Statistics

a. To view the NAT and PAT statistics type the **show ip nat statistics** command at the privileged EXEC mode prompt.

b. How many active translations have taken place? _____

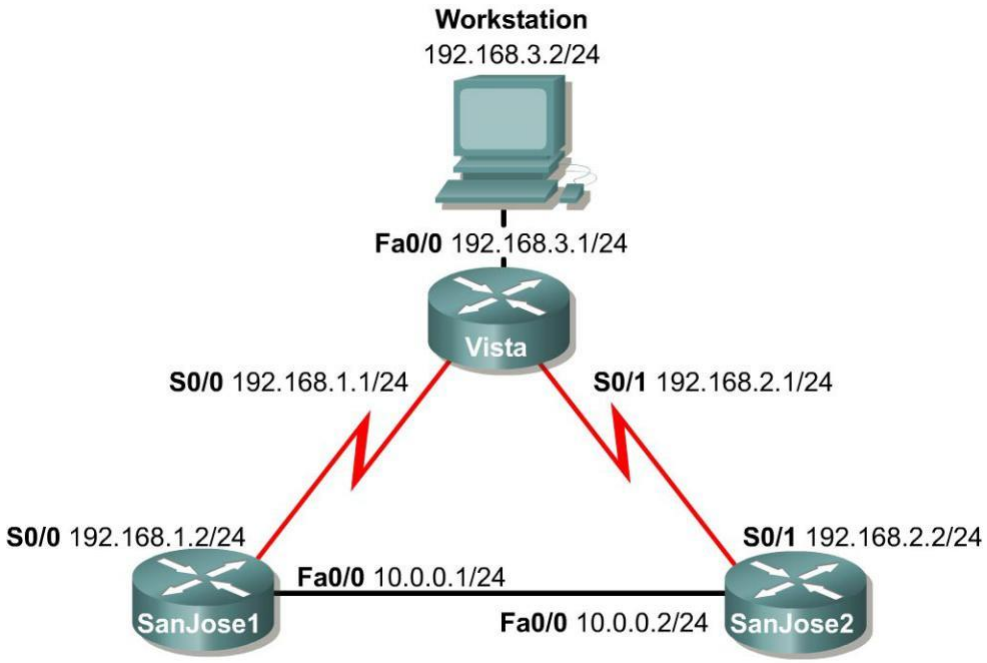
c. How many addresses are in the pool? _____

d. How many addresses have been allocated so far? _____

Upon completion of the previous steps finish the lab by doing the following:

- Logoff by typing **exit**
- Turn the router off
- Remove and store the cables and adapter

Access Control List Basics and Extended Ping



Objective

This lab activity reviews the basics of standard and extended access lists.

Scenario

The LAN users connected to the Vista router are concerned about access to the network from hosts on network 10.0.0.0. A standard access list must be used to block all access to the Vista LAN from network 10.0.0.0 /24.

Also, an extended ACL must be used to block network 192.168.3.0 host access to Web servers on the 10.0.0.0 /24 network.

Step 1

Build and configure the network according to the diagram. Use RIPv1, and enable updates on all active interfaces with the appropriate **`network`** commands. The commands necessary to configure SanJose1 are shown in the following example:

```
SanJose1(config)#router rip
SanJose1(config-router)#network 192.168.1.0
SanJose1(config-router)#network 10.0.0.0
```

Use the **`ping`** command to verify the work and test connectivity between all interfaces.

Step 2

Check the routing table on Vista using the **show ip route** command. Vista should have all four networks in the routing table. Troubleshoot, if necessary

Access Control List Basics

Access Control Lists (ACLs) are simple but powerful tools. When the access list is configured, each statement in the list is processed by the router in the order in which it was created. If an individual packet meets a statement's criteria, the permit or deny is applied to that packet, and no further list entries are checked. Each packet starts at the top of the list, every time.

It is not possible to reorder an access list, skip statements, edit statements, or delete statements from a numbered access list, while in the router configuration mode. With numbered access lists, any attempt to delete a single statement results in deletion of the entire list. Named ACLs (NACLs) do allow for the deletion of individual statements. It is suggested that ACLs, of all kinds, be created in an off-line editor and pasted into the configuration.

The following concepts apply to both standard and extended access lists:

Two step process

The access list may be created with one or more **access-list** commands while in global configuration mode. Second, the access list is applied to or referenced by other commands, such as the **ip access-group** command which applies the ACL to an interface. An example would be the following:

```
Vista#config terminal
Vista(config)#access-list 50 deny 10.0.0.0 0.0.0.255
Vista(config)#access-list 50 permit any
Vista(config)#interface fastethernet 0/0
Vista(config-if)#ip access-group 50 out
Vista(config-if)#^Z
```

Syntax and Keywords

The basic syntax for creating an access list entry is as follows:

```
router(config)#access-list # {permit | deny}ip address wildcard mask
```

The **permit** command allows packets matching the specified criteria to be accepted for whatever application the access list is being used for. The **deny** command discards packets matching the criteria on that line.

Two important keywords, **any** and **host**, can be used with IP addresses and the access list. The keyword **any** matches all hosts on all networks, equivalent to **0.0.0.0 255.255.255.255**. The keyword **host** can be used with an IP address to indicate a single host address. The syntax is **host** *ip address* (**host 192.168.1.10**). This is the same as entering 192.168.1.10 0.0.0.0.

Implicit deny statement

Every access list contains a final "deny" statement that matches all packets. This is called the implicit deny. Because the implicit deny statement is not visible in **show** command output, it is often overlooked, with serious consequences. As an example, consider the following single-line access list:

```
Router(config)#access-list 75 deny host 192.168.1.10
```

Access- list 75 clearly denies all traffic sourced from the host, 192.168.1.10. What might not be obvious is that all other traffic will be discarded as well. This is because the implicit **deny any** is the final statement in any access list.

At least one permit statement is required

There is no requirement that an ACL contain a **deny** statement. If nothing else, the implicit **deny any** statement takes care of that. But if there are no **permit** statements, the effect will be the same as if there were only a single **deny any** statement.

Wildcard mask

In identifying IP addresses, ACLs use a wildcard mask instead of a subnet mask. Initially, the masks might look the same, but closer observation reveals that they are very different. Remember that a binary 0 in a wildcard mask instructs the router to match the corresponding bit in the IP address.

In/out

When deciding whether an ACL should be applied to inbound or outbound traffic, always view things from the perspective of the router. In other words, determine whether traffic is coming into the router, inbound, or leaving the router, outbound.

Applying ACLs

Extended ACLs should be applied as close to the source as possible, thereby conserving network resources. Standard ACLs, by necessity, must be applied as close to the destination as possible. This is because the standard ACL can match only at the source address of a packet.

Step 3

On the Vista router, create the following standard ACL and apply it to the LAN interface:

```
Vista#config terminal
Vista(config)#access-list 50 deny 10.0.0.0 0.0.0.255
Vista(config)#access-list 50 permit any
Vista(config)#interface fastethernet 0/0
Vista(config-if)#ip access-group 50 out
Vista(config-if)#^Z
```

Try pinging 192.168.3.2 from SanJose1.

The ping should be successful. This result might be surprising, because all traffic from the 10.0.0.0/8 network was just blocked. The ping is successful because, even though it came from SanJose1, it is not sourced from the 10.0.0.0/8 network. A ping or traceroute from a router uses the closest interface to the destination as the source address. Therefore, the ping is coming from the 192.168.1.0/24, SanJose1's Serial 0/0.

```
SanJose1#ping 192.168.3.2
Sending 5, 100-byte ICMP Echos to 192.168.3.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```

Step 4

In order to test the ACL from SanJose1, the extended ping command must be used to specify a source interface as follows:

On SanJose1, issue the following commands:

Note: Remember that the extended ping works only in the privileged EXEC mode.

```
SanJose1#ping
Protocol [ip]:
Target IP address: 192.168.3.2
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 10.0.0.1
Type of service [0]:
Set DF bit in IP header? [no]:
```

```
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.3.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Step 5

Standard ACLs are numbered 1 - 99. IOS version 12.xx allows additional numbering from 1300 - 1699. Extended ACLs are numbered 100 - 199. IOS version 12.xx allows additional numbering from 2000 - 2699. Extended ACLs can be used to enforce highly specific criteria for filtering packets. In this step, configure an extended ACL to block access to a Web server. Before proceeding, issue the **`no access-list 50`** and **`no ip access-group 50`** commands on the Vista router to remove the ACL configured previously.

First, configure both SanJose1 and SanJose2 to act as Web servers, by using the **`ip http server`** command, as shown in the following:

```
SanJose1(config)#ip http server
SanJose2(config)#ip http server
```

From the workstation at 192.168.3.2, use a Web browser to view both Web servers on the router at 10.0.0.1 and 10.0.0.2. The Web login requires that the enable secret password for the router be entered as the password.

After verifying Web connectivity between the workstation and the routers, proceed to Step 6.

Step 6

On the Vista router, enter the following commands:

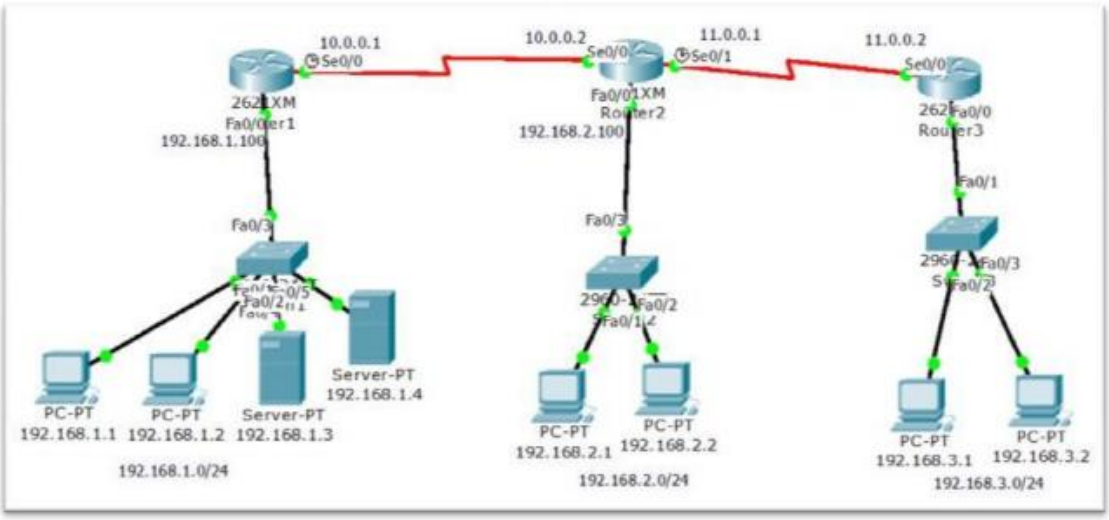
```
Vista(config)#access-list 101 deny tcp 192.168.3.0 0.0.0.255 10.0.0.0 0.0.0.255
eq www
Vista(config)#access-list 101 deny tcp 192.168.3.0 0.0.0.255 any eq ftp
Vista(config)#access-list 101 permit ip any any
Vista(config)#interface fastethernet 0/0
Vista(config-if)#ip access-group 101 in
```

From the workstation at 192.168.3.2, again attempt to view the Web servers at 10.0.0.1 and 10.0.0.2. Both attempts should fail.

Next, browse SanJose1 at 192.168.1.2. Why is this not blocked?

---

1. Implementing Numbered/ Named Standard ACL



Pre-requirement for LAB (check previous labs)

1) Design the topology (connectivity)
2) Assign the IP address according to diagram
3) Make sure that interfaces used should be in UP UP state
4) Any dynamic routing Protocol or static routing
5) Verify Routing table and reachability between the LAN's (using PING and TRACE commands)

Let's say the Requirement in this LAB is to

- Deny the host 192.168.1.1 communicating with 192.168.2.0
- Deny the host 192.168.1.2 communicating with 192.168.2.0
- Deny the network 192.168.3.0 communicating with 192.168.2.0
- Permit all the remaining traffic

***NOTE:** the Above ACL rules should not affect the other communication*

Before creating the ACL, make sure that the routing configured is correct and all the three LAN devices are able to communicate with each other using PING command

Next Extended ACL:

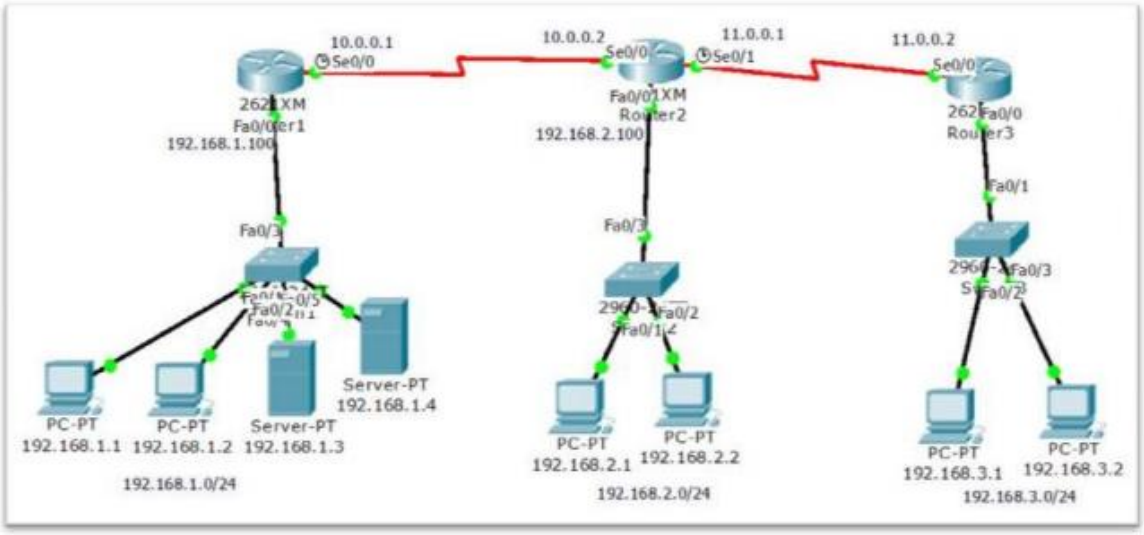
Restricting telnet access to the router to specified networks or hosts

Requirement:

- Allow only the **hosts 192.168.1.1 and 192.168.1.**2 to telnet R1. any other host should be denied of they try to telnet R1

Remove the ACL which was created the previous lab

Implementing Extended ACL



Pre-requirement for LAB (check previous labs)

1) Design the topology (connectivity)
2) Assign the IP address according to diagram
3) Make sure that interfaces used should be in UP UP state
4) Any dynamic routing Protocol or static routing

5) Verify Routing table and reachability between the LAN's (using PING and TRACE commands)

Let's say the Requirement in this LAB is to

- Deny the users on LAN 192.168.2.0 should not access 192.168.1.3 HTTP service
- Deny the users on LAN 192.168.3.0 should not access 192.168.1.4 FTP service
- Deny the users on LAN 192.168.3.1 should not access 192.168.1.3 HTTP service
- Deny the users on LAN 192.168.2.0 should not get DNS service from DNS server 192.168.1.4
- Deny ther users from the host between 192.168.3.2 and 192.168.1.2 should not be able to send ICMP (ping / trace) messages
- Remaining hosts and services should be permitted

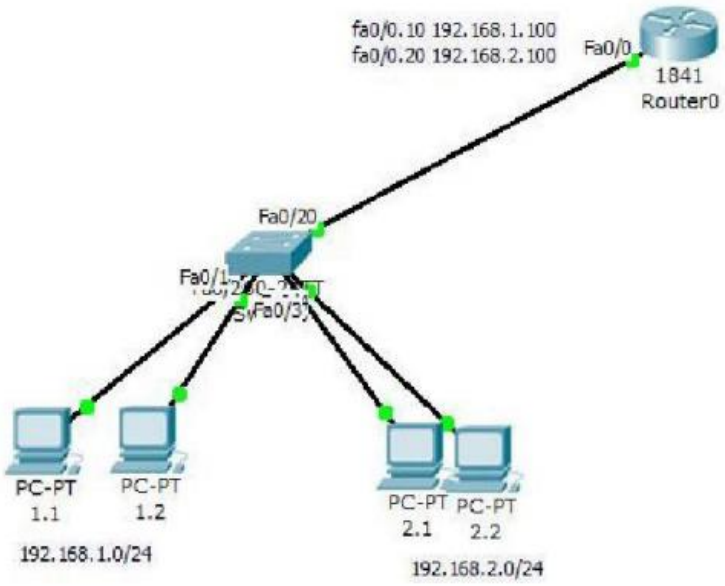
NOTE: *the Above ACL rules should not affect the other communication*

7. Using packet Tracer, perform the following experiments

a. Basic switching configuration.

b. Configure VLAN and Inter-VLAN routing for a Network.

Inter- VLAN Routing Using Router



Steps :

1) create vlan and shift the ports
2) configure on switch fa0/ 20 as trunk port
3) Create sub interfaces on router port fa0/ 0
4) Verify connectivity between vlans (ping 192.168.1.1 ---192.168.2.1)

PART-B: Implement the following in C/C++/python.

1. Write a program for a HLDC frame to perform the following.
    i) Bit stuffing and destuffing.
    ii) Character stuffing and destuffing.
2. Write a program for distance vector algorithm to find suitable path for transmission.
3. Implement Dijkstra's algorithm to compute the shortest routing path.
4. For the given data, use CRC-CCITT polynomial to obtain CRC code. Verify the program for the cases
    i) Without error
    ii) With error
5. Implementation of Stop and Wait Protocol and Sliding Window Protocol (GBN and SR Protocol).
6. Write a program for minimum spanning tree using kruskal's/Prim's algorithm.
7. Write a socket programing for client – server Model.

Part- B

1. Write a python program for a HLDC frame to perform the following.

   a. Bit stuffing and destuffing.

   b. Character stuffing and destuffing.

Bit Stuffing and destuffing

```
streak = 0
data = list(input())
i=0
while i<len(data):
    if data[i] == '1':
        streak +=1
    else:
        streak = 0
    if streak==5:
        data.insert(i+1,'0')
        streak = 0
    i+=1
print(''.join(data))
```

Bit destuffing

```
streak = 0
data = list(input())
i=0
while i<len(data):
    if data[i] == '1':
        streak +=1
    else:
        streak = 0
    if streak==5:
        data.pop(i+1)
        streak = 0
    i+=1
print(''.join(data))
```

Character Stuffing and destuffing:

```
data = "FJKSFRFFETJJEJTLDF"
data = list(data)
# F is the Flag
# E is the Escape Sequence
#Expected Output : FJKSEFREFEFEETJJEEJTLDF
i=1
while i<len(data)-1:
    if data[i] == 'F' or data[i] =='E':
        data.insert(i,'E')
        i+=1
    i+=1
print(''.join(data))
```

Destuffing:

```
data = "FTTHIS IS TTHE TTRUE DATTA TTHATT IS IN TTHE TFILEF"
# F is the Flag
# E is the Escape Sequence
# Output Expected : FTHIS IS THE TRUE DATA THAT IS IN THE FILEF
data = list(data)
i=1
while  i<len(data)-1:
    if data[i] == 'T':
        if data[i+1] == 'T' or data[i+1] == 'F':
            data.pop(i)
            i-=1
    i+=1
print(''.join(data))
```

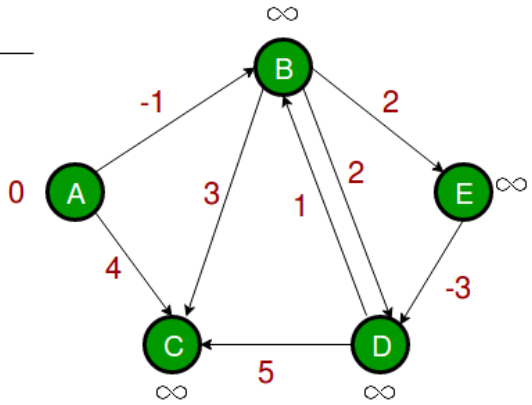
1. **Write a Python program for distance vector algorithm (Bellman ford Algorithm) to find suitable path (Shortest path) for data transmission.**

Algorithm:

Input: Graph and a source vertex **src**
Output: Shortest distance to all vertices from **src**. If there is a negative weight cycle, then shortest distances are not calculated, negative weight cycle is reported.
1) This step initializes distances from source to all vertices as infinite and distance to source itself as 0. Create an array **dist[ ]** of size |V| with all values as infinite except **dist[src]** where **src** is source vertex.
2) This step calculates shortest distances. Do following |V|-1 times where |V| is the number of vertices in given graph.
…..a) Do following for each edge u-v
……If dist[v] > dist[u] + weight of edge uv, then update dist[v]
…….dist[v] = dist[u] + weight of edge uv
3) This step reports if there is a negative weight cycle in graph. Do following for each edge u-v
……If **dist[v] > dist[u] + weight of edge uv**, then "Graph contains negative weight cycle"
The idea of step 3 is, step 2 guarantees shortest distances if graph doesn't contain negative weight cycle. If we iterate through all edges one more time and get a shorter path for any vertex, then there is a negative weight cycle

Program:

A	B	C	D	E
0	∞	∞	∞	∞




```
# Python3 program for Bellman-Ford's single source shortest
path algorithm.

from sys import maxsize

# The main function that finds shortest
```

```
# distances from src to all other vertices
# using Bellman-Ford algorithm. The function
# also detects negative weight cycle
# The row graph[i] represents i-th edge with
# three values u, v and w.
def BellmanFord(graph, V, E, src):

    # Initialize distance of all vertices as infinite.
    dis = [maxsize] * V

    # initialize distance of source as 0
    dis[src] = 0

    # Relax all edges |V| - 1 times. A simple
    # shortest path from src to any other
    # vertex can have at-most |V| - 1 edges
    for i in range(V - 1):
        for j in range(E):
            if dis[graph[j][0]] + \
                   graph[j][2] < dis[graph[j][1]]:
                dis[graph[j][1]] = dis[graph[j][0]] + \
                                   graph[j][2]

    # check for negative-weight cycles.
    # The above step guarantees shortest
    # distances if graph doesn't contain
    # negative weight cycle. If we get a
    # shorter path, then there is a cycle.
    for i in range(E):
        x = graph[i][0]
        y = graph[i][1]
        weight = graph[i][2]
        if dis[x] != maxsize and dis[x] + \
                        weight < dis[y]:
            print("Graph contains negative weight cycle")

    print("Vertex Distance from Source")
    for i in range(V):
        print("%d\t\t%d" % (i, dis[i]))

# Driver Code
if __name__ == "__main__":
    V = 5 # Number of vertices in graph
    E = 8 # Number of edges in graph

    # Every edge has three values (u, v, w) where
    # the edge is from vertex u to v. And weight
    # of the edge is w.
    graph = [[0, 1, -1], [0, 2, 4], [1, 2, 3],
             [1, 3, 2], [1, 4, 2], [3, 2, 5],
             [3, 1, 1], [4, 3, -3]]
```

```
    BellmanFord(graph, V, E, 0)
```

Excepted Output:

```
Vertex Distance from Source
0        0
1        -1
2        2
3        -2
4        1
```

2. **Implement Dijkstra's algorithm to compute the shortest routing path**.

Algorithm

1) Create a set **sptSet** (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.
2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.
3) While **sptSet** doesn't include all vertices:

- Pick a vertex u which is not there in **sptSet** and has minimum distance value.
- Include u to **sptSet**.
- Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if the sum of a distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

Program:

```
# Python program for Dijkstra's single source shortest path
algorithm.
# The program is for adjacency matrix representation of the
graph
# Library for INT_MAX

import sys

class Graph():

    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]

    def printSolution(self, dist):
        print("Vertex tDistance from Source")
```

```
        for node in range(self.V):
            print(node, "t", dist[node])

    # A utility function to find the vertex with
    # minimum distance value, from the set of vertices
    # not yet included in shortest path tree
    def minDistance(self, dist, sptSet):

        # Initialize minimum distance for next node
        min = sys.maxsize

        # Search not nearest vertex not in the
        # shortest path tree
        for v in range(self.V):
            if dist[v] < min and sptSet[v] == False:
                min = dist[v]
                min_index = v

        return min_index

    # Funtion that implements Dijkstra's single source
    # shortest path algorithm for a graph represented
    # using adjacency matrix representation
    def dijkstra(self, src):

        dist = [sys.maxsize] * self.V
        dist[src] = 0
        sptSet = [False] * self.V

        for cout in range(self.V):

            # Pick the minimum distance vertex from
            # the set of vertices not yet processed.
            # u is always equal to src in first iteration
            u = self.minDistance(dist, sptSet)

            # Put the minimum distance vertex in the
            # shortest path tree
            sptSet[u] = True

            # Update dist value of the adjacent vertices
            # of the picked vertex only if the current
            # distance is greater than new distance and
            # the vertex in not in the shortest path tree
            for v in range(self.V):
                if self.graph[u][v] > 0 and
                   sptSet[v] == False and
                   dist[v] > dist[u] + self.graph[u][v]:
                    dist[v] = dist[u] + self.graph[u][v]
```

```
        self.printSolution(dist)


# Driver program
g = Graph(9)
g.graph = [[0, 4, 0, 0, 0, 0, 0, 8, 0],
           [4, 0, 8, 0, 0, 0, 0, 11, 0],
           [0, 8, 0, 7, 0, 4, 0, 0, 2],
           [0, 0, 7, 0, 9, 14, 0, 0, 0],
           [0, 0, 0, 9, 0, 10, 0, 0, 0],
           [0, 0, 4, 14, 10, 0, 2, 0, 0],
           [0, 0, 0, 0, 0, 2, 0, 1, 6],
           [8, 11, 0, 0, 0, 0, 1, 0, 7],
           [0, 0, 2, 0, 0, 0, 6, 7, 0]]
 g.dijkstra(0)
```

Excepted Output:

```
Vertex tDistance from Source
0 t 0
1 t 4
2 t 12
3 t 19
4 t 21
5 t 11
6 t 9
7 t 8
8 t 14
```

3. **For the given data, use CRC-CCITT polynomial to obtain CRC code. Verify the python program for with and without error cases.**

Algorithm:

1. The task is to send a string data to the server/receiver side.
2. The sender sends a string let us say "**EVN**". (DATA)
3. First, this string is converted to binary string "100010110101101001110" key is known to both the side sender and receiver here key used is 1001.
4. This data is encoded using the CRC code using the key in the client/sender side.
5. This encoded data is sent to the receiver.
6. Receiver later decodes the encoded data string to verify whether there was any error or not.

Let data send is “EVN”

We convert a string to binary string data.

Example:

```
input_string = "EVN"

# CONVERT string data to binary string data
data = (''.join(format(ord(x), 'b') for x in input_string))
print (data)
```

Program:

```
# Import socket module
import socket

def xor(a, b):

    # initialize result
    result = []

    # Traverse all bits, if bits are
    # same, then XOR is 0, else 1
    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')

    return ''.join(result)


# Performs Modulo-2 division
def mod2div(divident, divisor):

    # Number of bits to be XORed at a time.
    pick = len(divisor)

    # Slicing the divident to appropriate
    # length for particular step
    tmp = divident[0 : pick]

    while pick < len(divident):

        if tmp[0] == '1':

            # replace the divident by the result
            # of XOR and pull 1 bit down
            tmp = xor(divisor, tmp) + divident[pick]
```

```
            else: # If leftmost bit is '0'

                # If the leftmost bit of the dividend (or the
                # part used in each step) is 0, the step cannot
                # use the regular divisor; we need to use an
                # all-0s divisor.
                tmp = xor('0'*pick, tmp) + divident[pick]

            # increment pick to move further
            pick += 1

        # For the last n bits, we have to carry it out
        # normally as increased value of pick will cause
        # Index Out of Bounds.
        if tmp[0] == '1':
            tmp = xor(divisor, tmp)
        else:
            tmp = xor('0'*pick, tmp)

        checkword = tmp
        return checkword

    # Function used at the sender side to encode
    # data by appending remainder of modular division
    # at the end of data.
    def encodeData(data, key):

        l_key = len(key)

        # Appends n-1 zeroes at end of data
        appended_data = data + '0'*(l_key-1)
        remainder = mod2div(appended_data, key)

        # Append remainder in the original data
        codeword = data + remainder
        return codeword

    # Create a socket object
    s = socket.socket()

    # Define the port on which you want to connect
    port = 12345

    # connect to the server on local computer
    s.connect(('127.0.0.1', port))

    # Send data to server 'Hello world'

    ## s.sendall('Hello World')
```

```
input_string = input("Enter data you want to send->")
#s.sendall(input_string)
data =(''.join(format(ord(x), 'b') for x in input_string))
print("Entered data in binary format :",data)
key = "1001"

ans = encodeData(data,key)
print("Encoded data to be sent to server in binary format
:",ans)
s.sendto(ans.encode(),('127.0.0.1', 12345))


# receive data from the server
print("Received feedback from server
:",s.recv(1024).decode())

# close the connection
s.close()
```

Receiver Side program:

```
# First of all import the socket library
import socket

def xor(a, b):

    # initialize result
    result = []

    # Traverse all bits, if bits are
    # same, then XOR is 0, else 1
    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')

    return ''.join(result)


# Performs Modulo-2 division
def mod2div(divident, divisor):

    # Number of bits to be XORed at a time.
    pick = len(divisor)

    # Slicing the divident to appropriate
    # length for particular step
```

```
    tmp = divident[0: pick]

    while pick < len(divident):

        if tmp[0] == '1':

            # replace the divident by the result
            # of XOR and pull 1 bit down
            tmp = xor(divisor, tmp) + divident[pick]

        else:  # If leftmost bit is '0'
            # If the leftmost bit of the dividend (or the
            # part used in each step) is 0, the step cannot
            # use the regular divisor; we need to use an
            # all-0s divisor.
            tmp = xor('0'*pick, tmp) + divident[pick]

        # increment pick to move further
        pick += 1

    # For the last n bits, we have to carry it out
    # normally as increased value of pick will cause
    # Index Out of Bounds.
    if tmp[0] == '1':
        tmp = xor(divisor, tmp)
    else:
        tmp = xor('0'*pick, tmp)

    checkword = tmp
    return checkword

# Function used at the receiver side to decode
# data received by sender


def decodeData(data, key):

    l_key = len(key)

    # Appends n-1 zeroes at end of data
    appended_data = data.decode() + '0'*(l_key-1)
    remainder = mod2div(appended_data, key)

    return remainder


# Creating Socket
s = socket.socket()
print("Socket successfully created")
```

```
# reserve a port on your computer in our
# case it is 12345 but it can be anything
port = 12345

s.bind(('', port))
print("socket binded to %s" % (port))
# put the socket into listening mode
s.listen(5)
print("socket is listening")


while True:
    # Establish connection with client.
    c, addr = s.accept()
    print('Got connection from', addr)

    # Get data from client
    data = c.recv(1024)

    print("Received encoded data in binary format :",
data.decode())

    if not data:
        break

    key = "1001"

    ans = decodeData(data, key)
    print("Remainder after decoding is->"+ans)

    # If remainder is all zeros then no error occured
    temp = "0" * (len(key) - 1)
    if ans == temp:
        c.sendto(("THANK you Data ->"+data.decode() +
                  " Received No error FOUND").encode(),
('127.0.0.1', 12345))
    else:
        c.sendto(("Error in data").encode(), ('127.0.0.1',
12345))

    c.close()
```

Excepted Output:

DATA: 100010110101101001110

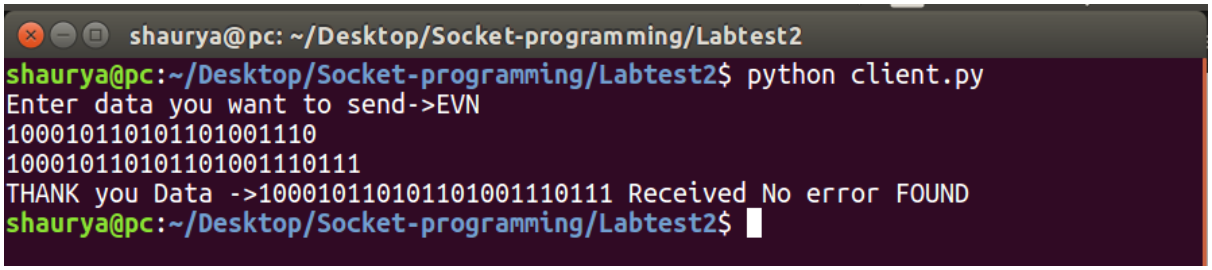
CRC KEY: 1001

Code: CRC key length -1 -> 000 appended at end of data.

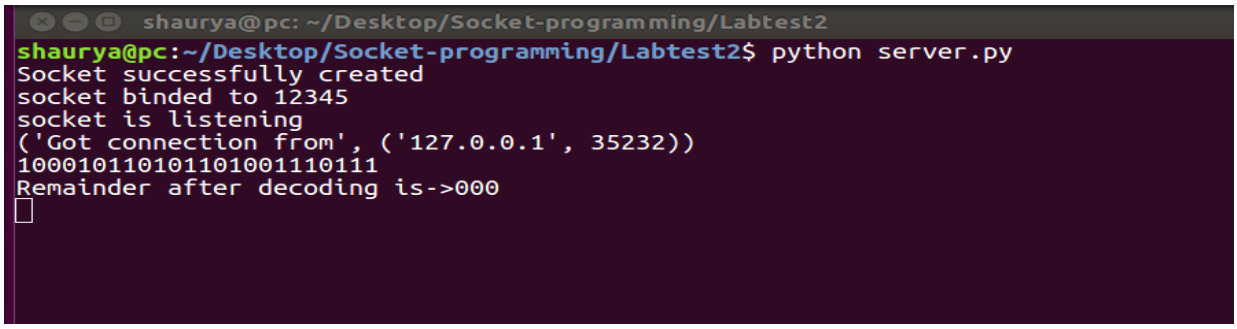
New data: 100010110101101001110000

Key:1001

Sender Side output:



Receiver Side output:



4. **Implementation of Stop and Wait Protocol and Sliding Window Protocol (GBN and SR Protocol).**

Sender Side program:

```
Import Socket
from threading import *
serversocket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
host = "localhost"
port = 8000
serversocket.bind((host, port))
class client(Thread):
    def __init__(self, socket, address):
        Thread.__init__(self)
        self.sock = socket
        self.addr = address
        self.start()


    def run(self):
        while 1:
            r=input("Send data -->")
            clientsocket.send(r.encode())
```

```
                print(clientsocket.recv(1024).decode())


    serversocket.listen(5)
    print ('Sender ready and is listening')
    while (True):
        #to accept all incoming connections
        clientsocket, address = serversocket.accept()
        print("Receiver "+str(address)+" connected")
        #create a different thread for every
        #incoming connection
        client(clientsocket, address)
```

Receiver Side Program:

```
Import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host ="localhost"
port =8000
s.connect((host,port))



while 2:
   data=s.recv(1024).decode()
   print("Received --> "+data)
   str="Acknowledgement: Message Received"
   s.send(str.encode())


s.close ()
```

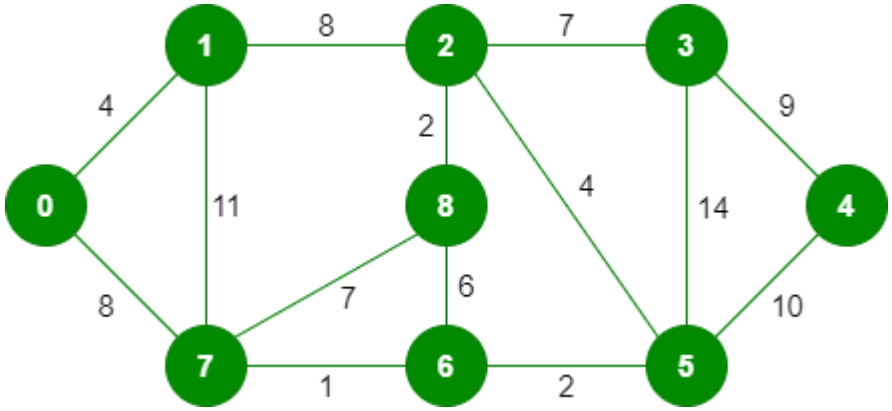
5. Write a program for minimum spanning tree using kruskal's/Prim's algorithm.

Kruskal's Algorithm:

1. Sort all the edges in non-decreasing order of their weight.

2. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If cycle is not formed, include this edge. Else, discard it.

3. Repeat step#2 until there are (V-1) edges in the spanning tree.



Program:

```
# Python program for Kruskal's algorithm to find
# Minimum Spanning Tree of a given connected,
# undirected and weighted graph

from collections import defaultdict

# Class to represent a graph


class Graph:

    def __init__(self, vertices):
        self.V = vertices  # No. of vertices
        self.graph = []  # default dictionary
        # to store graph

    # function to add an edge to graph
    def addEdge(self, u, v, w):
        self.graph.append([u, v, w])

    # A utility function to find set of an element i
    # (uses path compression technique)
    def find(self, parent, i):
        if parent[i] == i:
            return i
        return self.find(parent, parent[i])
```

```
# A function that does union of two sets of x and y
# (uses union by rank)
def union(self, parent, rank, x, y):
    xroot = self.find(parent, x)
    yroot = self.find(parent, y)

    # Attach smaller rank tree under root of
    # high rank tree (Union by Rank)
    if rank[xroot] < rank[yroot]:
        parent[xroot] = yroot
    elif rank[xroot] > rank[yroot]:
        parent[yroot] = xroot

    # If ranks are same, then make one as root
    # and increment its rank by one
    else:
        parent[yroot] = xroot
        rank[xroot] += 1

# The main function to construct MST using Kruskal's
    # algorithm
def KruskalMST(self):

    result = []  # This will store the resultant MST

    # An index variable, used for sorted edges
    i = 0

    # An index variable, used for result[]
    e = 0

    # Step 1:  Sort all the edges in
    # non-decreasing order of their
    # weight.  If we are not allowed to change the
    # given graph, we can create a copy of graph
    self.graph = sorted(self.graph,
                        key=lambda item: item[2])

    parent = []
    rank = []

    # Create V subsets with single elements
    for node in range(self.V):
        parent.append(node)
        rank.append(0)

    # Number of edges to be taken is equal to V-1
    while e < self.V - 1:
```

```
            # Step 2: Pick the smallest edge and increment
            # the index for next iteration
            u, v, w = self.graph[i]
            i = i + 1
            x = self.find(parent, u)
            y = self.find(parent, v)

            # If including this edge does't
            #  cause cycle, include it in result
            #  and increment the indexof result
            # for next edge
            if x != y:
                e = e + 1
                result.append([u, v, w])
                self.union(parent, rank, x, y)
            # Else discard the edge

        minimumCost = 0
        print ("Edges in the constructed MST")
        for u, v, weight in result:
            minimumCost += weight
            print("%d -- %d == %d" % (u, v, weight))
        print("Minimum Spanning Tree" , minimumCost)

# Driver code
g = Graph(4)
g.addEdge(0, 1, 10)
g.addEdge(0, 2, 6)
g.addEdge(0, 3, 5)
g.addEdge(1, 3, 15)
g.addEdge(2, 3, 4)

# Function call
g.KruskalMST()
```

Excepted Output:

```
Following are the edges in the constructed MST
2 -- 3 == 4
0 -- 3 == 5
0 -- 1 == 10
Minimum Cost Spanning Tree: 19
```

6. Write a socket programing for client – server Model.

Server Side python program

```
# first of all import the socket library
import socket

# next create a socket object
s = socket.socket()
print ("Socket successfully created")

# reserve a port on your computer in our
# case it is 12345 but it can be anything
port = 12345

# Next bind to the port
# we have not typed any ip in the ip field
# instead we have inputted an empty string
# this makes the server listen to requests
# coming from other computers on the network
s.bind(('', port))
print ("socket binded to %s" %(port))

# put the socket into listening mode
s.listen(5)
print ("socket is listening")

# a forever loop until we interrupt it or
# an error occurs
while True:

# Establish connection with client.
  c, addr = s.accept()
  print ('Got connection from', addr )

  # send a thank you message to the client. encoding to send
byte type.
  c.send('Thank you for connecting'.encode())

  # Close the connection with the client
  c.close()

  # Breaking once connection closed
  break
```

Client Side Python Program.

```
Import socket module
import socket
 # Create a socket object
s = socket.socket()
 # Define the port on which you want to connect
port = 12345
 # connect to the server on local computer
s.connect(('127.0.0.1', port))
 # receive data from the server and decoding to get the string.
print (s.recv(1024).decode())
# close the connection
s.close()
```

Excepted Output: server command prompt

```
# start the server:
$ python server.py
Socket successfully created
socket binded to 12345
socket is listening
Got connection from ('127.0.0.1', 52617)
```

Client Command Prompt

```
# start the client:
$ python client.py
Thank you for connecting
```