Lab Assignments List

Subject Name: Computer Network LAB
Subject Code: PCC-CST-692
Semester: 6th

List of Experiments:

S.No	Experiment
1	Study of different types of Network cables and Practically implement the cross-wired cable and straight through cable using clamping tool.
2	Study of Network Devices in Detail.
3	Study of network IP.
4	Connect the computers in Local Area Network.
5	Implementation of client side and server side program with command line arguments for connection less socket using C.
6	Implementation of the client side and server side program for the connection oriented socket using C to the students. Set of two program of TCP/IP socket program
7	Implementation of the multi-client and single server program for the connection oriented socket programming using C. Set of two programs of TCP/IP socket programs listed below.
8	Connecting a Switch using cisco packet tracer
9	Implementing an IP Addressing Scheme
10	Configuring a Default Route
11	Configuring a Cisco Router as a DHCP Server

Experiment-1

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	

Experiment - 2

Aim: Study of following Network Devices in Detail

- Repeater
- Hub
- Switch
- Bridge
- Router
- Gate Way

Apparatus (Software): No software or hardware needed.

Procedure: Following should be done to understand this practical.

1. **Repeater:**

Functioning at Physical Layer.A**repeater** is an electronic device that receives a signal and retransmits it at a higher level and/or higher power, or onto the other side of an obstruction, so that the signal can cover longer distances. Repeater have two ports ,so cannot be use to connect for more than two devices

2. **Hub:**

An **Ethernet hub**, **active hub**, **network hub**, **repeater hub**, **hub** or **concentrator** is a device for connecting multiple twisted pair or fiber optic Ethernet devices together and making them act as a single network segment. Hubs work at the physical layer (layer 1) of the OSI model. The device is a form of multiport repeater. Repeater hubs also participate in collision detection, forwarding a jam signal to all ports if it detects a collision.

3. **Switch:**

N**etwork switch** or **switching hub** is a computer networking device that connects network segments. The term commonly refers to a network bridge that processes and routes data at the data link layer (layer 2) of the OSI model. Switches that additionally process data at the network layer (layer 3 and above) are often referred to as Layer 3 switches or multilayer switches.

4. **Bridge:**

A **network bridge** connects multiple network segments at the data link layer (Layer 2) of the OSI model. In Ethernet networks, the term *bridge* formally means a device that behaves according to the IEEE 802.1D standard. A bridge and switch are very much alike; a switch being a bridge with numerous ports. *Switch* or *Layer 2 switch* is often used interchangeably with *bridge. Bridges* can analyze incoming data packets to determine if the bridge is able to send the given packet to another segment of the network.

5. **Router:**

A **router** is an electronic device that interconnects two or more computer networks, and selectively interchanges packets of data between them. Each data packet contains address information that a router can use to determine if the source and destination are on the same network, or if the data packet must be transferred from one network to another. Where multiple routers are used in a large collection of interconnected networks, the routers exchange information about target system addresses, so that each router can build up a table showing the preferred paths between any two systems on the interconnected networks.

6. **Gate Way:** In a communications network, a network node equipped for interfacing with

another network that uses different protocols.

- A gateway may contain devices such as protocol translators, impedance matching devices, rate converters, fault isolators, or signal translators as necessary to provide system interoperability. It also requires the establishment of mutually acceptable administrative procedures between both networks.
- A protocol translation/mapping gateway interconnects networks with different network protocol technologies by performing the required protocol conversions.

Experiment – 3

Aim: Study of network IP

- Classification of IP address
- Sub netting
- Super netting

Apparatus (Software): NA

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

- Classification of IP address

As show in figure we teach how the ip addresses are classified and when they are used.

Class	Address Range	Supports
Class A	1.0.0.1 to 126.255.255.254	Supports 16 million hosts on each of 127 networks.
Class B	128.1.0.1 to 191.255.255.254	Supports 65,000 hosts on each of 16,000 networks.
Class C	192.0.1.1 to 223.255.254.254	Supports 254 hosts on each of 2 million networks.
Class D	224.0.0.0 to 239.255.255.255	Reserved for multicast groups.
Class E	240.0.0.0 to 254.255.255.254	Reserved.

- **Sub netting**

Why we Develop sub netting and How to calculate subnet mask and how to identify subnet address.

- **Super netting**

Why we develop super netting and How to calculate supernet mask and how to identify supernet address.

Experiment-4

Aim: Connect the computers in Local Area Network.

Procedure: On the host computer

On the host computer, follow these steps to share the Internet connection:

1. Log on to the host computer as Administrator or as Owner.
2. Click **Start**, and then click **Control Panel**.
3. Click **Network and Internet Connections**.
4. Click **Network Connections**.
5. Right-click the connection that you use to connect to the Internet. For example, if you connect to the Internet by using a modem, right-click the connection that you want under Dial-up / other network available.
6. Click **Properties**.
7. Click the **Advanced** tab.

8. Under **Internet Connection Sharing**, select the **Allow other network users to connect through this computer's Internet connection** check box.

9. If you are sharing a dial-up Internet connection, select the **Establish a dial-up connection whenever a computer on my network attempts to access the Internet** check box if you want to permit your computer to automatically connect to the Internet.

10. Click **OK**. You receive the following message:

When Internet Connection Sharing is enabled, your LAN adapter will be set to use IP address 192.168.0. 1. Your computer may lose connectivity with other computers on your network. If these other computers have static IP addresses, it is a good idea to set them to obtain their IP addresses automatically. Are you sure you want to enable Internet Connection Sharing?

11. Click **Yes**.

The connection to the Internet is shared to other computers on the local area network (LAN).

The network adapter that is connected to the LAN is configured with a static IP address of 192.168.0. 1 and a subnet mask of 255.255.255.0

On the client computer

To connect to the Internet by using the shared connection, you must confirm the LAN adapter IP configuration, and then configure the client computer. To confirm the LAN adapter IP configuration, follow these steps:

1. Log on to the client computer as Administrator or as Owner.
2. Click **Start**, and then click **Control Panel**.

3. Click **Network and Internet Connections**.
4. Click **Network Connections**.
5. Right-click **Local Area Connection** and then click **Properties**.
6. Click the **General** tab, click **Internet Protocol (TCP/IP)** in the **connection uses the following items** list, and then click **Properties**.

7. In the **Internet Protocol (TCP/IP) Properties** dialog box, click **Obtain an IP address automatically** (if it is not already selected), and then click **OK**.

Note: You can also assign a unique static IP address in the range of 192.168.0.2 to 254. For example, you can assign the following static IP address, subnet mask, and default gateway:

8. IP Address 192.168.31.202
9. Subnet mask 255.255.255.0
10. Default gateway 192.168.31.1

11. In the **Local Area Connection Properties** dialog box, click **OK**.

12. Quit Control Panel

Experiment-5

/*
Implementation of client side and server side program with command line arguments for connection less socket using C
*/

Client Side:

```
// udp client driver program
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>

#define PORT 5000
#define MAXLINE 1000

// Driver code
int main()
{
   char buffer[100];
   char *message = "Hello Server";
   int sockfd, n;
   struct sockaddr_in servaddr;

   // clear servaddr
   bzero(&servaddr, sizeof(servaddr));
   servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
   servaddr.sin_port = htons(PORT);
   servaddr.sin_family = AF_INET;

   // create datagram socket
   sockfd = socket(AF_INET, SOCK_DGRAM, 0);

   // connect to server
   if(connect(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0)
   {
      printf("\n Error : Connect Failed \n");
      exit(0);
   }
```

```
    // request to send datagram
    // no need to specify server address in sendto
    // connect stores the peers IP and port
    sendto(sockfd, message, MAXLINE, 0, (struct sockaddr*)NULL, sizeof(servaddr));

    // waiting for response
    recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr*)NULL, NULL);
    puts(buffer);

    // close the descriptor
    close(sockfd);
}
```

Server side:

```
// server program for udp connection
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#define PORT 5000
#define MAXLINE 1000

// Driver code
int main()
{
    char buffer[100];
    char *message = "Hello Client";
    int listenfd, len;
    struct sockaddr_in servaddr, cliaddr;
    bzero(&servaddr, sizeof(servaddr));

    // Create a UDP Socket
    listenfd = socket(AF_INET, SOCK_DGRAM, 0);
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);
    servaddr.sin_family = AF_INET;

    // bind server address to socket descriptor
    bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr));

    //receive the datagram
    len = sizeof(cliaddr);
    int n = recvfrom(listenfd, buffer, sizeof(buffer),
```

```
        0, (struct sockaddr*)&cliaddr,&len); //receive message from server
    buffer[n] = '\0';
    puts(buffer);

    // send the response
    sendto(listenfd, message, MAXLINE, 0,
        (struct sockaddr*)&cliaddr, sizeof(cliaddr));
}
```

Experiment-6

Implementation of the client side and server side program for the connection oriented socket using C to the students. Set of two program of TCP/IP socket program

Client side:

```
#include <arpa/inet.h> // inet_addr()
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h> // bzero()
#include <sys/socket.h>
#include <unistd.h> // read(), write(), close()
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
   char buff[MAX];
   int n;
   for (;;) {
      bzero(buff, sizeof(buff));
      printf("Enter the string : ");
      n = 0;
      while ((buff[n++] = getchar()) != '\n')
         ;
      write(sockfd, buff, sizeof(buff));
      bzero(buff, sizeof(buff));
      read(sockfd, buff, sizeof(buff));
      printf("From Server : %s", buff);
      if ((strncmp(buff, "exit", 4)) == 0) {
         printf("Client Exit...\n");
         break;
      }
   }
}

int main()
{
   int sockfd, connfd;
   struct sockaddr_in servaddr, cli;

   // socket create and verification
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

```
    if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr, sizeof(servaddr));

    // assign IP, PORT
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);

    // connect the client socket to server socket
    if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr))
        != 0) {
        printf("connection with the server failed...\n");
        exit(0);
    }
    else
        printf("connected to the server..\n");

    // function for chat
    func(sockfd);

    // close the socket
    close(sockfd);
}
```

Server Side:

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h> // read(), write(), close()
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

// Function designed for chat between client and server.
void func(int connfd)
{
```

```
    char buff[MAX];
    int n;
    // infinite loop for chat
    for (;;) {
        bzero(buff, MAX);

        // read the message from client and copy it in buffer
        read(connfd, buff, sizeof(buff));
        // print buffer which contains the client contents
        printf("From client: %s\t To client : ", buff);
        bzero(buff, MAX);
        n = 0;
        // copy server message in the buffer
        while ((buff[n++] = getchar()) != '\n')
            ;

        // and send that buffer to client
        write(connfd, buff, sizeof(buff));

        // if msg contains "Exit" then server exit and chat ended.
        if (strncmp("exit", buff, 4) == 0) {
            printf("Server Exit...\n");
            break;
        }
    }
}

// Driver function
int main()
{
    int sockfd, connfd, len;
    struct sockaddr_in servaddr, cli;

    // socket create and verification
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully created..\n");
    bzero(&servaddr, sizeof(servaddr));

    // assign IP, PORT
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
```

```
    servaddr.sin_port = htons(PORT);

    // Binding newly created socket to given IP and verification
    if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
        printf("socket bind failed...\n");
        exit(0);
    }
    else
        printf("Socket successfully binded..\n");

    // Now server is ready to listen and verification
    if ((listen(sockfd, 5)) != 0) {
        printf("Listen failed...\n");
        exit(0);
    }
    else
        printf("Server listening..\n");
    len = sizeof(cli);

    // Accept the data packet from client and verification
    connfd = accept(sockfd, (SA*)&cli, &len);
    if (connfd < 0) {
        printf("server accept failed...\n");
        exit(0);
    }
    else
        printf("server accept the client...\n");

    // Function for chatting between client and server
    func(connfd);

    // After chatting close the socket
    close(sockfd);
}
```

Experiment-7

Implementation of the multi-client and single server program for the connection oriented socket programming using C. Set of two programs of TCP/IP socket programs listed below.

Client side:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <errno.h>

#define BUFSIZE 1024

void send_recv(int i, int sockfd)
{
	char send_buf[BUFSIZE];
	char recv_buf[BUFSIZE];
	int nbyte_recvd;

	if (i == 0){
		fgets(send_buf, BUFSIZE, stdin);
		if (strcmp(send_buf , "quit\n") == 0) {
			exit(0);
		}else
			send(sockfd, send_buf, strlen(send_buf), 0);
	}else {
		nbyte_recvd = recv(sockfd, recv_buf, BUFSIZE, 0);
		recv_buf[nbyte_recvd] = '';
		printf("%s\n" , recv_buf);
		fflush(stdout);
	}
}


void connect_request(int *sockfd, struct sockaddr_in *server_addr)
{
	if ((*sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
		perror("Socket");
		exit(1);
	}
```

```
	server_addr->sin_family = AF_INET;
	server_addr->sin_port = htons(4950);
	server_addr->sin_addr.s_addr = inet_addr("127.0.0.1");
	memset(server_addr->sin_zero, '', sizeof server_addr->sin_zero);

	if(connect(*sockfd, (struct sockaddr *)server_addr, sizeof(struct sockaddr)) == -1) {
		perror("connect");
		exit(1);
	}
}

int main()
{
	int sockfd, fdmax, i;
	struct sockaddr_in server_addr;
	fd_set master;
	fd_set read_fds;

	connect_request(&sockfd, &server_addr);
	FD_ZERO(&master);
    FD_ZERO(&read_fds);
    FD_SET(0, &master);
    FD_SET(sockfd, &master);
	fdmax = sockfd;

	while(1){
		read_fds = master;
		if(select(fdmax+1, &read_fds, NULL, NULL, NULL) == -1){
			perror("select");
			exit(4);
		}

		for(i=0; i <= fdmax; i++ )
			if(FD_ISSET(i, &read_fds))
				send_recv(i, sockfd);
	}
	printf("client-quited\n");
	close(sockfd);
	return 0;
}
```

Server side:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define PORT 4950
#define BUFSIZE 1024

void send_to_all(int j, int i, int sockfd, int nbytes_recvd, char *recv_buf, fd_set *master)
{
	if (FD_ISSET(j, master)){
		if (j != sockfd && j != i) {
			if (send(j, recv_buf, nbytes_recvd, 0) == -1) {
				perror("send");
			}
		}
	}
}

void send_recv(int i, fd_set *master, int sockfd, int fdmax)
{
	int nbytes_recvd, j;
	char recv_buf[BUFSIZE], buf[BUFSIZE];

	if ((nbytes_recvd = recv(i, recv_buf, BUFSIZE, 0)) <= 0) {
		if (nbytes_recvd == 0) {
			printf("socket %d hung up\n", i);
		}else {
			perror("recv");
		}
		close(i);
		FD_CLR(i, master);
	}else {
	//	printf("%s\n", recv_buf);
		for(j = 0; j <= fdmax; j++){
			send_to_all(j, i, sockfd, nbytes_recvd, recv_buf, master );
		}
	}
}

void connection_accept(fd_set *master, int *fdmax, int sockfd, struct sockaddr_in *client_addr)
{
	socklen_t addrlen;
	int newsockfd;
```

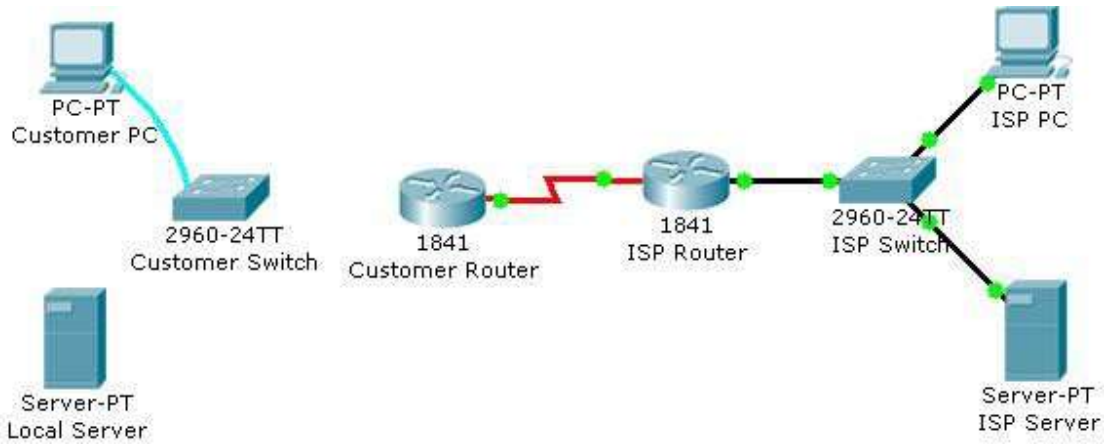
```
	addrlen = sizeof(struct sockaddr_in);
	if((newsockfd = accept(sockfd, (struct sockaddr *)client_addr, &addrlen)) == -1) {
		perror("accept");
		exit(1);
	}else {
		FD_SET(newsockfd, master);
		if(newsockfd > *fdmax){
			*fdmax = newsockfd;
		}
		printf("new connection from %s on port %d \n",inet_ntoa(client_addr->sin_addr),
ntohs(client_addr->sin_port));
	}
}

void connect_request(int *sockfd, struct sockaddr_in *my_addr)
{
	int yes = 1;

	if ((*sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
		perror("Socket");
		exit(1);
	}

	my_addr->sin_family = AF_INET;
	my_addr->sin_port = htons(4950);
	my_addr->sin_addr.s_addr = INADDR_ANY;
	memset(my_addr->sin_zero, '', sizeof my_addr->sin_zero);

	if (setsockopt(*sockfd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int)) == -1) {
		perror("setsockopt");
		exit(1);
	}

	if (bind(*sockfd, (struct sockaddr *)my_addr, sizeof(struct sockaddr)) == -1) {
		perror("Unable to bind");
		exit(1);
	}
	if (listen(*sockfd, 10) == -1) {
		perror("listen");
		exit(1);
	}
	printf("\nTCPServer Waiting for client on port 4950\n");
	fflush(stdout);
}
int main()
```

```
{
	fd_set master;
	fd_set read_fds;
	int fdmax, i;
	int sockfd= 0;
	struct sockaddr_in my_addr, client_addr;

	FD_ZERO(&master);
	FD_ZERO(&read_fds);
	connect_request(&sockfd, &my_addr);
	FD_SET(sockfd, &master);

	fdmax = sockfd;
	while(1){
		read_fds = master;
		if(select(fdmax+1, &read_fds, NULL, NULL, NULL) == -1){
			perror("select");
			exit(4);
		}

		for (i = 0; i <= fdmax; i++){
			if (FD_ISSET(i, &read_fds)){
				if (i == sockfd)
					connection_accept(&master, &fdmax, sockfd, &client_addr);
				else
					send_recv(i, &master, sockfd, fdmax);
			}
		}
	}
	return 0;
}
```

Experiment-8

Connecting a Switch

Topology Diagram



Objectives

- Connect a switch to the network.
- Verify the configuration on the switch.

Background / Preparation

In this activity, you will verify the configuration on the customer Cisco Catalyst 2960 switch. The switch is already configured with all the basic necessary information for connecting to the LAN at the customer site. The switch is currently not connected to the network. You will connect the switch to the customer workstation, the customer server, and customer router. You will verify that the switch has been connected and configured successfully by pinging the LAN interface of the customer router.

Step 1: Connect the switch to the LAN.

a. Using the proper cable, connect the FastEthernet0/0 on Customer Router to the FastEthernet0/1 on Customer Switch.

b. Using the proper cable, connect the Customer PC to the Customer Switch on port FastEthernet0/2.

c. Using the proper cable, connect the Local Server to the Customer Switch on port FastEthernet0/3.

Step 2: Verify the switch configuration.

a. From the Customer PC, use the terminal emulation software to connect to the console of the customer Cisco Catalyst 2960 switch.

b. Use the console connection and terminal utility on the Customer PC to verify the configurations. Use **cisco** as the console password.

c. Enter privileged EXEC mode and use the **show running-config** command to verify the following configurations. The password is **cisco123**.

   a. VLAN1 IP address = 192.168.1.5

   b. Subnet mask = 255.255.255.0

c. Password required for console access

d. Password required for vty access

e. Password enabled for privileged EXEC mode

f. Secret enabled for privileged EXEC mode

a. Verify IP connectivity between the Cisco Catalyst 2960 switch and the Cisco 1841 router by initiating a ping to 192.168.1.1 from the switch CLI.

b. Click the **Check Results** button at the bottom of this instruction window to check your work.

Reflection

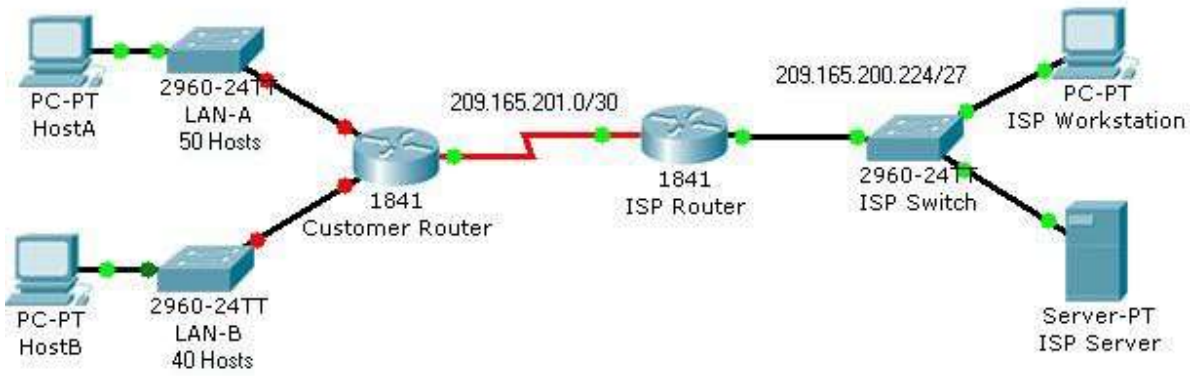
a. What is the significance of the enable secret command compared to the enable password?

b. If you want to remove the requirement to enter a password to access the console, what commands do you issue from your starting point in privileged EXEC mode?

Experirment-9

Implementing an IP Addressing Scheme

Topology Diagram



Objectives

- Subnet an address space based on the host requirements.
- Assign host addresses to devices.
- Configure devices with IP addressing.
- Verify the addressing configuration.

Background / Preparation

In this activity, you will subnet the private address space 192.168.1.0/24 to provide enough host addresses for the two LANs attached to the router. You will then assign valid host addresses to the appropriate devices and interfaces. Finally, you will test connectivity to verify your IP address implementation.

Step 1: Subnet an address space based on the host requirements.

a. You are given the private address space 192.168.1.0/24. Subnet this address space based on the following requirements:

- LAN-A needs enough addresses for 50 hosts.
- LAN-B needs enough addresses for 40 hosts.

How many bits must be left for host addresses? _____

How many bits can now be taken from the host portion to make a subnet? _____

How many hosts does each subnet support? _____

How many subnets are created? _____

What is the new subnet mask? _____

Step 2: Assign host addresses to devices.

What is the subnet address for subnet 0? _____

What is the subnet address for subnet 1? _____

Assign subnet 0 to LAN-A, and assign subnet 1 to LAN-B.

What is the first address in subnet 0? _____
This address is assigned the FastEthernet0/0 interface on Customer Router.

What is the first address in subnet 1? _____
This address is assigned the FastEthernet0/1 interface on Customer Router.

What is the last address in subnet 0? _____
This address is assigned to HostA.

What is the last address in subnet 1? _____
This address is assigned to HostB.

What is the default gateway for HostA? _____

What is the default gateway for HostB? _____

Step 3: Configure devices with IP addressing.

Configure HostA and HostB with IP addressing, including the subnet mask and default gateway.

a. Click HostA. On the **Desktop** tab, choose **IP Configuration**. Enter the correct addressing for HostA according to your answers in Step 1 and Step 2.

b. Click HostB. On the **Desktop** tab, choose **IP Configuration**. Enter the correct addressing for HostB according to your answers in Step 1 and Step 2.

c. Check results. On the **Assessment Items** tab, your configurations for HostA and HostB should have green checkmarks. If not, read the provided feedback for a hint on how to correct the problem.

   Note: If you cannot see all the feedback, place your mouse pointer over the right side of the **Activity Results** window. When the cursor turns into a double-headed arrow, click and drag to resize the window until you can see all the feedback text.)

Configure the LAN interfaces on Customer Router with IP addresses and a subnet mask.

a. Click Customer Router. Click the Config tab.

b. On the left side under Interface, click FastEthernet0/0. Enter the IP address and subnet mask, and then set the Port Status to On.

c. On the left side under Interface, click FastEthernet0/1. Enter the IP address and subnet mask, and then set the Port Status to On.

d. Notice in the Equivalent IOS Commands window that your actions produced actual commands. You can scroll through the command window. In the next chapter, you will learn how to enter these commands directly into the router instead of using the Config tab.

   For a better view of the commands, you can increase the size of the window. To resize the window, place your mouse pointer over the bottom border of the window. When the cursor turns into a double-headed arrow, click and drag.

Check results. On the Assessment Items tab, your configurations for Customer Router should have green checkmarks. If not, read the provided feedback for a hint on how to correct the problem.

Step 4: Verify the addressing configuration.

a. Test connectivity between HostA, HostB, ISP Workstation, and ISP Server. You can use the Add Simple PDU tool to create pings between the devices. You can also click HostA or HostB, then the Desktop tab, and then Command Prompt. Use the ping command to test connectivity to other devices. To obtain the IP address of another device, place your mouse pointer over the device.

b. Check results. On the Connectivity Tests tab, the status of each test should be successful.

Reflection

a. How many subnets are still available for future expansion?

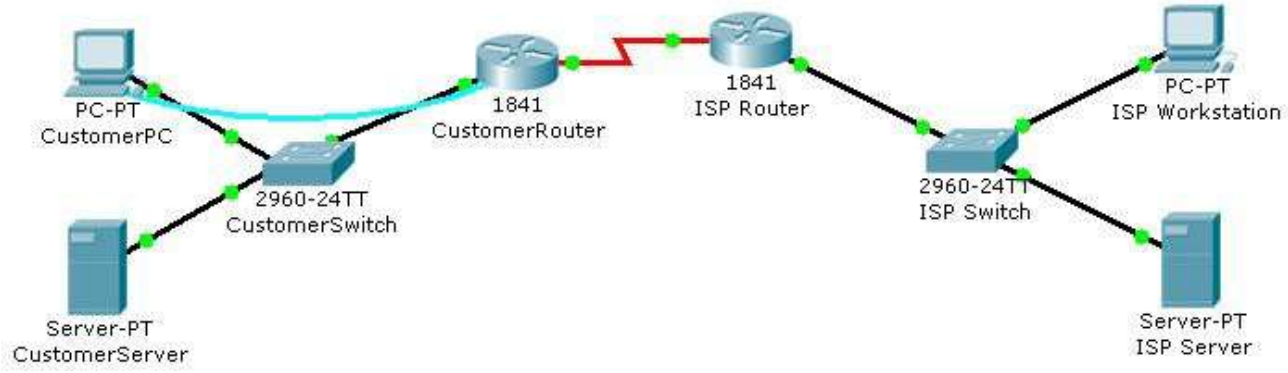
b. What would be the two subnet addresses if the host requirement was 80 hosts per LAN?

c. Challenge: Create your own Packet Tracer network using the same topology, but implement an addressing scheme based on 80 hosts per LAN. Have another student or your instructor check your work.

Experiment-10

Configuring a Default Route

Topology Diagram



Objectives

- Configure a default route on a router.

Background / Preparation

In this activity, you will configure a default route on the Cisco 1841 Customer router. The default route configuration uses the WAN IP address on the Cisco 1841 ISP router. This is the next-hop router from the Cisco 1841 Customer router.

Step 1: Verify reachability from CustomerRouter to the LAN IP address on the ISP router.

a. Use terminal emulation software on the Customer PC to connect to the customer Cisco 1841 router. Use **cisco123** for the console password.

b. Use the **ping** command to verify if the LAN IP address 209.165.201.1 on the ISP router is reachable from the CustomerRouter

CustomerRouter>**ping 209.165.201.1**

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 209.165.201.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)

Step 2: Configure the default route.

a. Enter privileged EXEC mode using the password **cisco**. The CustomerRouter# prompt indicates that you are in privileged EXEC mode.

b. Enter global configuration mode. The CustomerRouter(config)# prompt indicates that you are in global configuration mode.

c. Configure a default route using the ISP WAN IP address as the next hop IP address.

```
CustomerRouter(config)#ip route 0.0.0.0 0.0.0.0 209.165.200.226
CustomerRouter(config)#end
```

Step 3: Verify the default route configuration.

a. Use the **show ip route** command to verify the configuration of the default route. This is a partial example of the output.

```
CustomerRouter#show ip route
Codes: C - connected, S - static,...

Gateway of last resort is 209.165.200.226 to network 0.0.0.0

C    192.168.1.0/24 is directly connected, FastEthernet0/0
        209.165.200.0/27 is subnetted, 1 subnets
C       209.165.200.224 is directly connected, Serial0/1/0
S*   0.0.0.0/0 [1/0] via 209.165.200.226
```

b. Use the **ping** command to verify connectivity to the LAN IP address on the ISP router

```
CustomerRouter#ping 209.165.201.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 209.165.201.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 22/25/34 ms
```

Step 4: Save the configuration.

a. From privileged EXEC mode, save the running configuration to the startup configuration.

   i. CustomerRouter#**copy run start**

b. Click the **Check Results** button at the bottom of this instruction window to check your work.

Reflection

You can now access the entire ISP network. Write down some issues and considerations to discuss with your classmates about this configuration. Here are two questions to begin with:

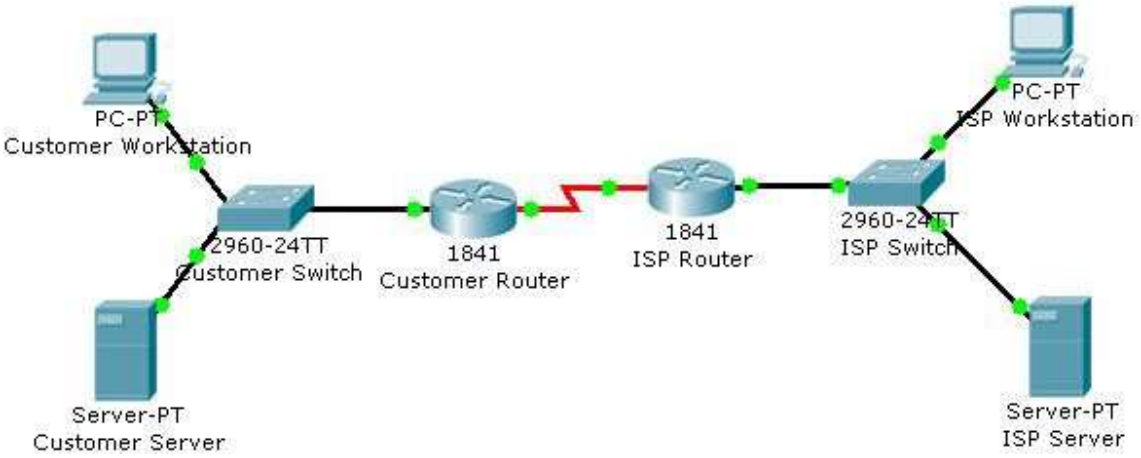
- Is this type of access to the ISP LAN likely to happen in the real world?

- Why has the student activity been configured to allow this type of access?

Experiment-11

Configuring a Cisco Router as a DHCP Server

Topology Diagram



Objectives

- Configure the customer Cisco 1841 ISR as a DHCP server.

Background / Preparation

In this activity, you will continue to configure the Cisco 1841 ISR router for the customer network by configuring the DHCP service. The customer has several workstations that need to be automatically configured with IP addresses on the local subnet and appropriate DHCP options to allow access to the Internet.

The DHCP pool will use the 192.168.1.0/24 network but the first 49 addresses are excluded. The default gateway and DNS server also need to be configured as 192.168.1.1 and 192.168.1.10.

For this activity, both the user and privileged EXEC passwords are **cisco**.

Note: Packet Tracer does not currently support the domain name and lease period options. These options are not used in this activity.

Step 1: Configure the DHCP service.

a. From the customer workstation, use a console cable and terminal emulation software to connect to the console of the customer Cisco1841 ISR.

b. Log in to the console of the Cisco 1841 ISR and enter global configuration mode.

c. Before creating a DHCP pool, configure the addresses that are excluded. The range is from 192.168.1.1 to 192.168.1.49.

CustomerRouter(config)#**ip dhcp excluded-address 192.168.1.1 192.168.1.49**

d. Create a DHCP pool called pool1.

CustomerRouter(config)#**ip dhcp pool pool1**

e. Define the network address range for the DHCP pool.

```
CustomerRouter(dhcp-config)#network 192.168.1.0 255.255.255.0
```

f. Define the DNS server as 192.168.1.10.

```
CustomerRouter(dhcp-config)#dns-server 192.168.1.10
```

g. Define the default gateway as 192.168.1.1.

```
CustomerRouter(dhcp-config)#default-router 192.168.1.1
```

h. Add an exclusion range of 192.168.1.1 to 192.168.1.49 to the DHCP pool.

```
CustomerRouter(dhcp-config)#exit
CustomerRouter(config)#ip dhcp excluded-address 192.168.1.1 192.168.1.49
```

i. Exit the terminal.

Step 2: Verify the DHCP configuration.

a. From the customer workstation, open the **Command Prompt** window.

b. Type **ipconfig /release** to release the current IP address.

c. Type **ipconfig /renew** to request a new IP address on the local network.

d. Verify that the IP address has been correctly assigned by pinging the LAN IP address of the Cisco 1841 ISR.

e. Click the **Check Results** button at the bottom of this instruction window to check your work.

Reflection

a. What is the purpose of DHCP on the customer network?

b. What IP address is assigned to the workstation after its IP address is renewed?

c. What other DHCP options can be defined on the Cisco 1841 ISR router that are not configured in this activity.