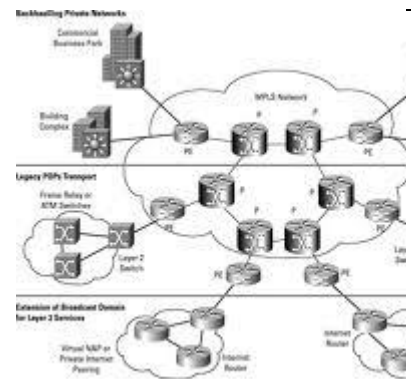


# DAYANAND SAGAR COLLEGE OF ENGINEERING

Department Of Computer Science & Engineering



## Computer Networks Laboratory Manual



1. Analyze VLAN communication using CPT
    - a) Sketch and Simulate three VLANS
    - b) Setup an extended VLAN using Trunk Interface
    - c) Inter VLAN Routing
- 

#### a) Sketch and Simulate three VLANS

**VLANs (Virtual LANs)** are logical grouping of devices in the same broadcast domain. VLANs are usually configured on switches by placing some interfaces into one broadcast domain and some interfaces into another. VLANs can be spread across multiple switches, with each VLAN being treated as its own subnet or broadcast domain. This means that frames broadcasted onto the network will be switched only between the ports within the same VLAN.

A VLAN acts like a physical LAN, but it allows hosts to be grouped together in the same broadcast domain even if they are not connected to the same switch. Here are the main reasons why you should use VLANs in your network:

Divides one single Broadcast domain into Multiple Broadcast domains.

VLANs increase the number of broadcast domains while decreasing their size.

- VLANs reduce security risks by reducing the number of hosts that receive
- copies of frames that the switches flood (Layer 2 security)
- you can keep hosts that hold sensitive data on a separate VLAN to improve security.
- Vlan 1 is the default VLAN
- We can create VLANS from 2 to 1001
- Can be configured on a manageable switches only

Configuration at layer2 switch:

```
Switch>enable
```

```
Switch#config t
```

```
Switch#config terminal
```

Enter configuration commands, one per line. End with CNTL/Z.

```
Switch(config)#vlan 10
```

```
Switch(config-vlan)#name green
```

```
Switch(config-vlan)#exit
```

```
Switch(config)#vlan 20
```

```
Switch(config-vlan)#name Red
```

```
Switch(config-vlan)#exit
```

```
Switch(config)#vlan 30
```

```
Switch(config-vlan)#name blue
```

```
Switch(config-vlan)#end
```

## Assign port -VLAN

Switch(config)# interface <interface type> <interface no>

Switch(config-if)# switchport mode access

Switch(config-if)# switchport access vlan <VLAN>

## Blue vlan

Switch(config)# interface range f0/7-8

Switch(config-if-range)# switchport mode access

Switch(config-if-range)# switchport access vlan 30

Switch(config-if-range)#

## Green Vlan(10)

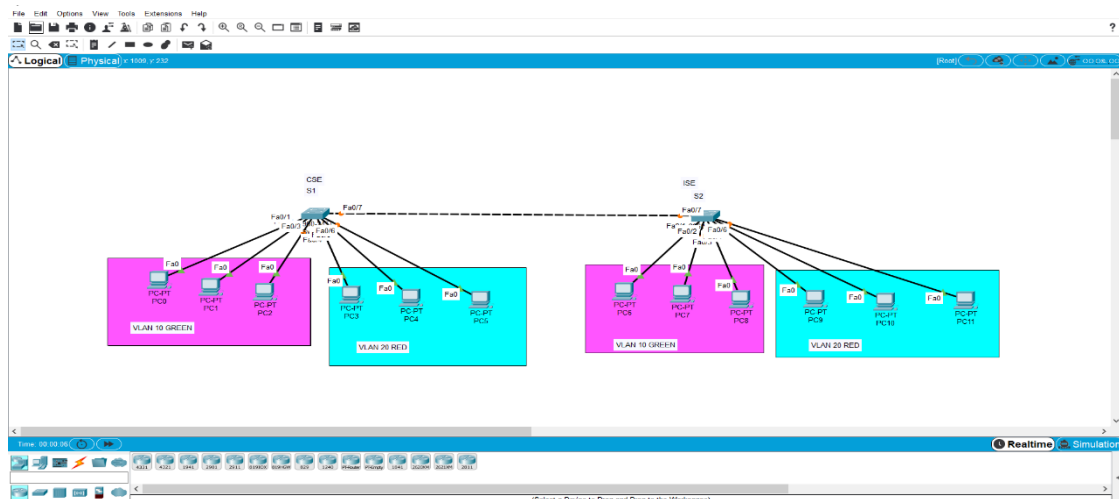
Switch(config)# interface range f0/1-4,f0/9,f0/12

Switch(config-if-range)# switchport mode access

Switch(config-if-range)# switchport access

Switch(config-if-range)# switchport access vlan 10

Switch(config-if-range)#



1b) Set up an extended VLAN using TRUNK interface

### TRUNKING

- A trunk interface is an **interface** that is **connected** to **another switch**. This type of interface can carry traffic of multiple VLANs.
- A single **VLAN** can span over **Multiple Switches**.
- User of the same VLAN may connect on two or more switches in the LAN . •  
Passing the same Vlan Traffic between switches using single link

### FRAME TAGGING

- In order to make sure that **same Vlan** Uses on **different switches** communicate with each other there is a method of tagging happen on trunk links.
- **Tag** is added **before** a **frame** is send and **removed** once it is on **trunk link**.
- Frame includes **source** and **Destination MAC entries**
- Tag Includes the **Vlan-ID**

### Trunk Configuration

```
Switch(config)#interface<interface type><interface no>  
Switch(config-if)#switchport mode trunk  
Switch(config-if)#switchport trunk encapsulation dotq
```

### Trunk interface at S1:

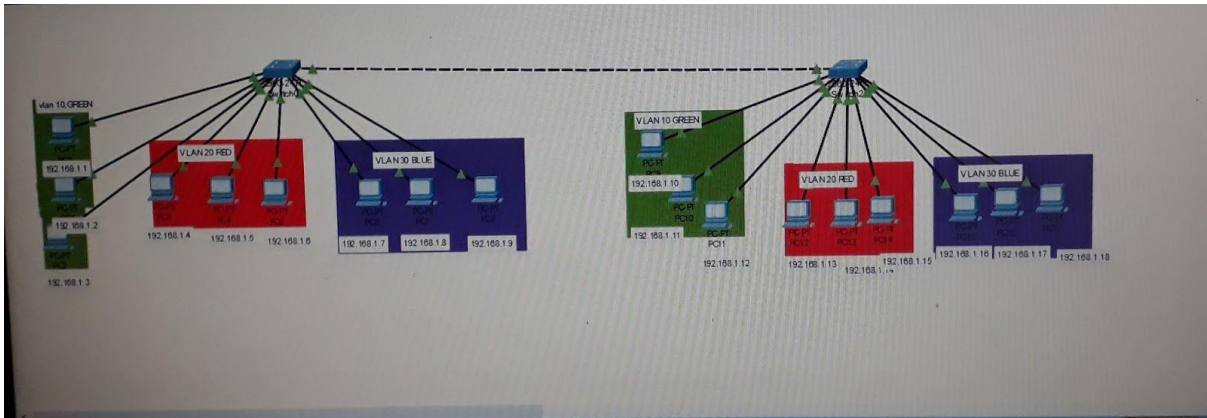
```
switch>enable  
Switch#config t  
Switch(config)#interface fa0/7  
Switch(config-if)#switchport mode trunk  
Switch(config-if)#exit
```

### Trunk interface at S2:

```
switch>enable  
Switch#config t  
Switch(config)#interface fa0/7  
Switch(config-if)#switchport mode trunk  
Switch(config-if)#exit
```

Verify the Vlan on both the switch

```
SW-1# show vlan  
SW-2#show vlan
```



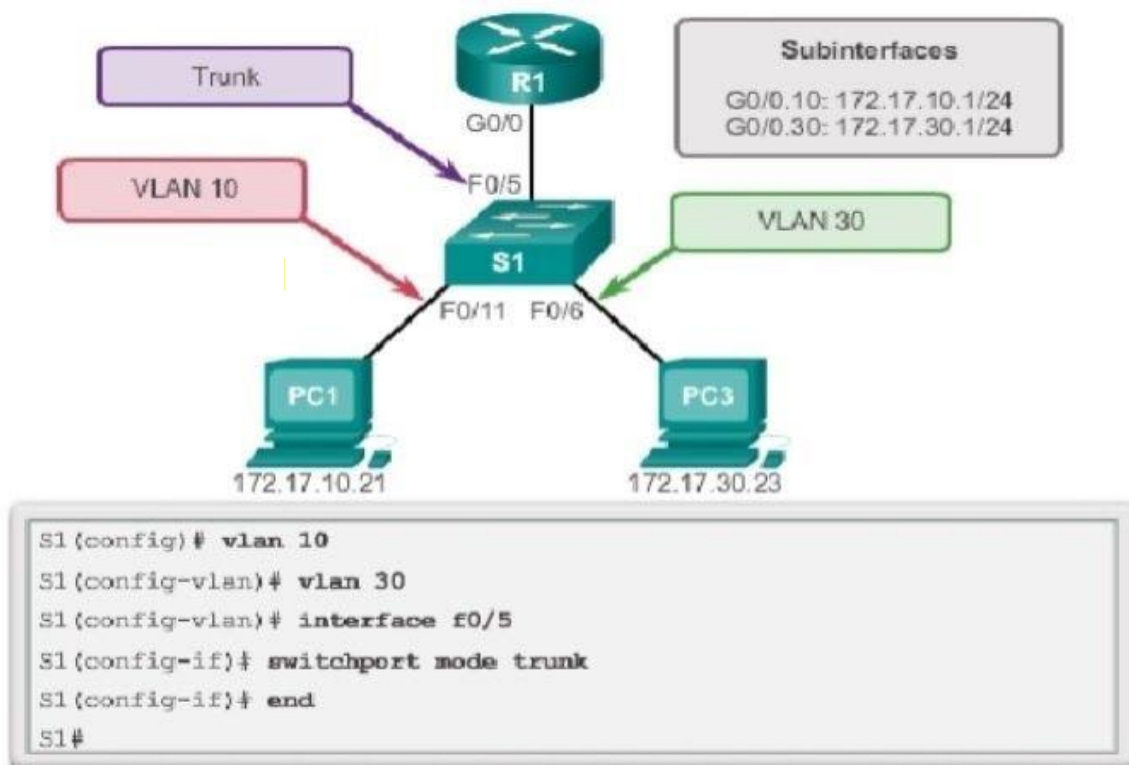
### 1c) INTER VLAN ROUTING

Allowing the users of one VLAN to access resources of other VLAN

There are three methods inter vlan routing

- A. Separate physical Gateway om Router
- B. Suing Sub-interfaces(ROUTER ON STICK)
- C. Using Layer 3 switch

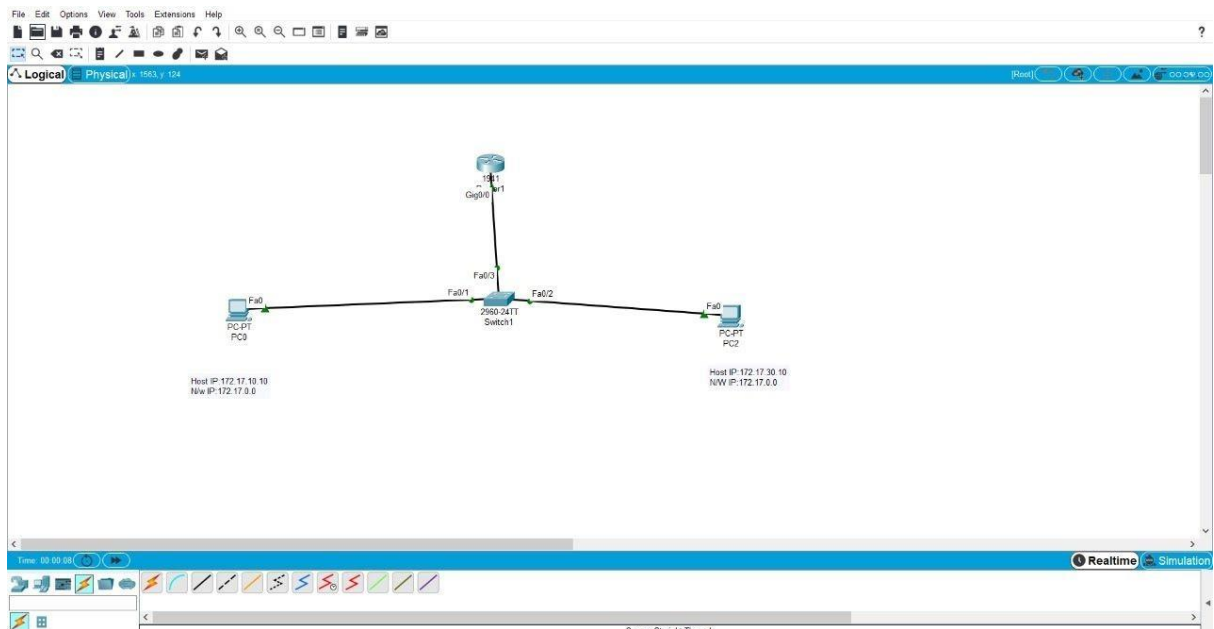
#### Suing Sub-interfaces(ROUTER ON STICK)



## Configure Router-on-a-StickRouter Subinterface Configuration

```
R1(config)# interface g0/0.10
R1(config-subif)# encapsulation dot1q 10
R1(config-subif)# ip address 172.17.10.1 255.255.255.0
R1(config-subif)# interface g0/0.30
R1(config-subif)# encapsulation dot1q 30
R1(config-subif)# ip address 172.17.30.1 255.255.255.0
R1(config)# interface g0/0
R1(config-if)# no shutdown

*Mar 20 00:20:59.299: %LINK-3-UPDOWN: Interface GigabitEthernet0/0,
changed state to down
*Mar 20 00:21:02.919: %LINK-3-UPDOWN: Interface GigabitEthernet0/0,
changed state to up
*Mar 20 00:21:03.919: %LINEPROTO-5-UPDOWN: Line protocol on
changed state to down
*Mar 20 00:21:02.919: %LINK-3-UPDOWN: Interface GigabitEthernet0/0,
changed state to up
*Mar 20 00:21:03.919: %LINEPROTO-5-UPDOWN: Line protocol on
Interface GigabitEthernet0/0, changed state to up
```





## LAB 2

### Implement STP

- a) Setup a network using multiplayer switch.
- b) Inter Vlan communications using multilayer switch.

### SPANNING TREE PROTOCOL(STP)

Spanning tree protocol is a layer 2 network protocol used to prevent looping within a network topology. STP was created to avoid the problems that arises when computers compete for the ability to use the shared telecommunications path on local area network(LAN).when too many computers try to send at the same time overall network performance is affected and can bring all traffic to a near halt.

STP prevents the condition known as bridge looping. It uses the spanning tree algorithm to find the shortest path between source and destination end devices.

### Configuration

In networks with redundancy STP helps prevent:

Switching Loops

Layer 2 Broadcast Storms

BID (Bridge ID) = Used to determine the Root Bridge:

1. Bridge Priority Number = 32768 (default) Lowest
2. Extended System ID = ID for VLANs
3. MAC Address = Lowest MAC Address

```
switch#show spanning-tree
```

```
switch#debug spanning-tree events
```

```
switch(config)#spanning-tree vlan 1 priority <num>
```

```
switch(config)#spanning-tree vlan 1 root primary
```

```
switch(config)#spanning-tree vlan 1 root secondary
```

```
switch(config)#interface fa0/1
```

```
switch(config-if)#spanning-tree cost <num>
```

```
switch(config-if)#spanning-tree port-priority <num, 128 default>
```

Port Costs:

10 Gig = 2

1 Gig = 4

100 Mb = 19

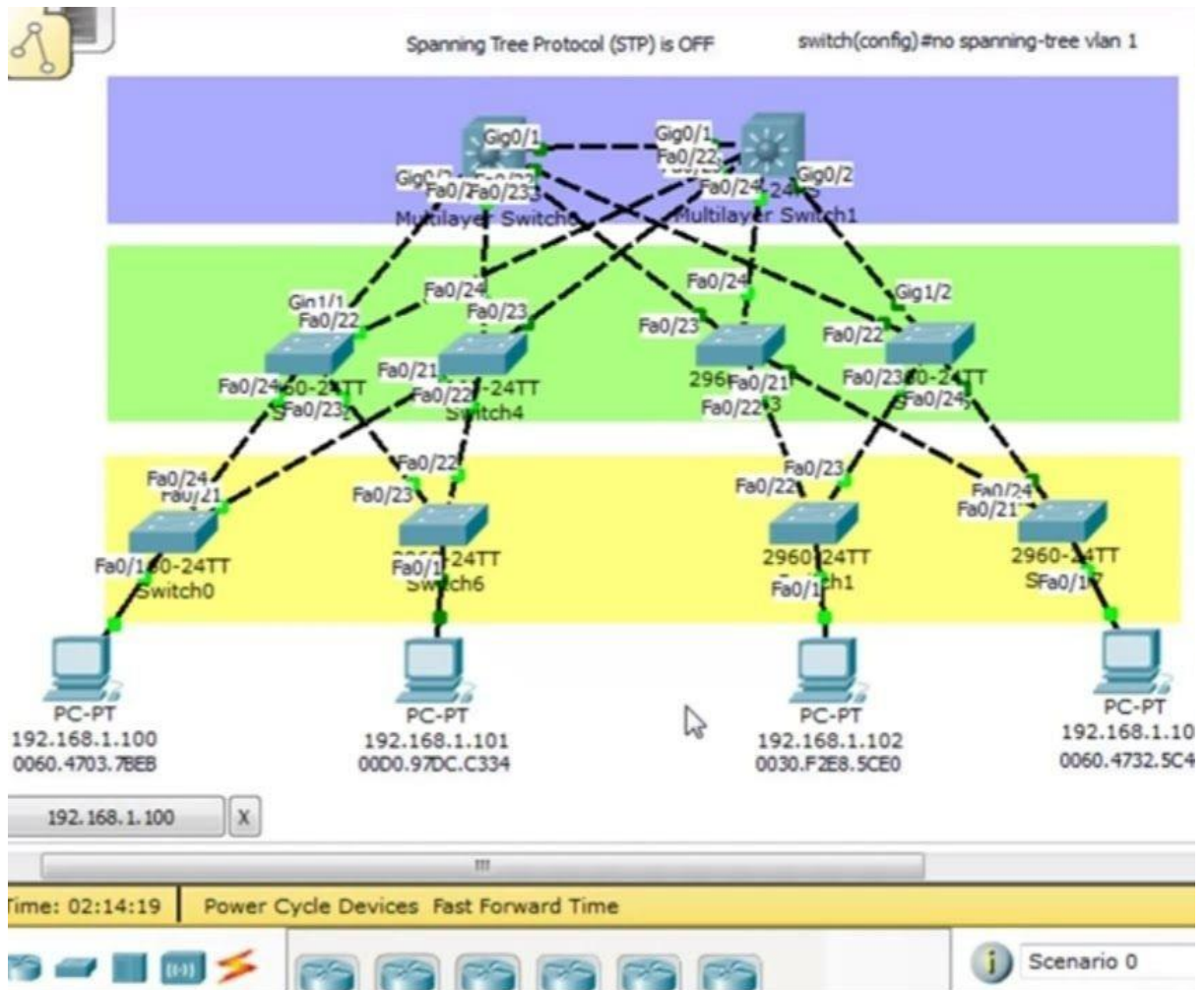
10 Mb = 100

Port Designations:

Root Port = Closest to the root bridge

Designated Port = Forwarding

Non-Designated Port = Blocking



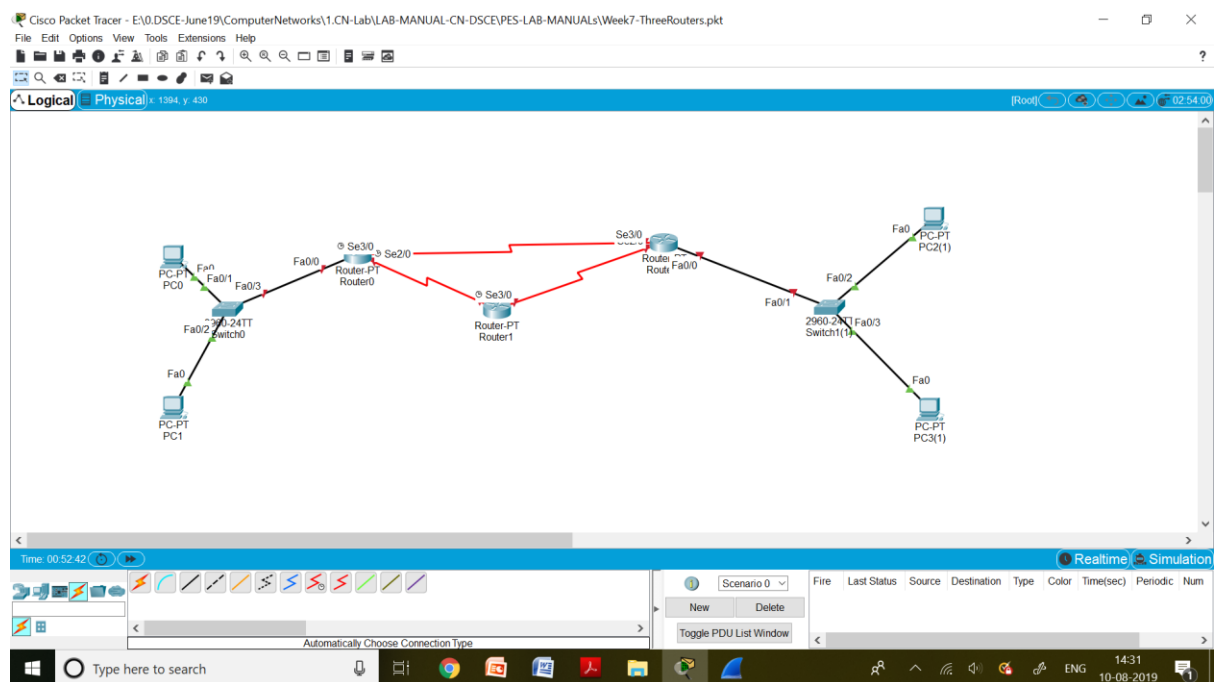


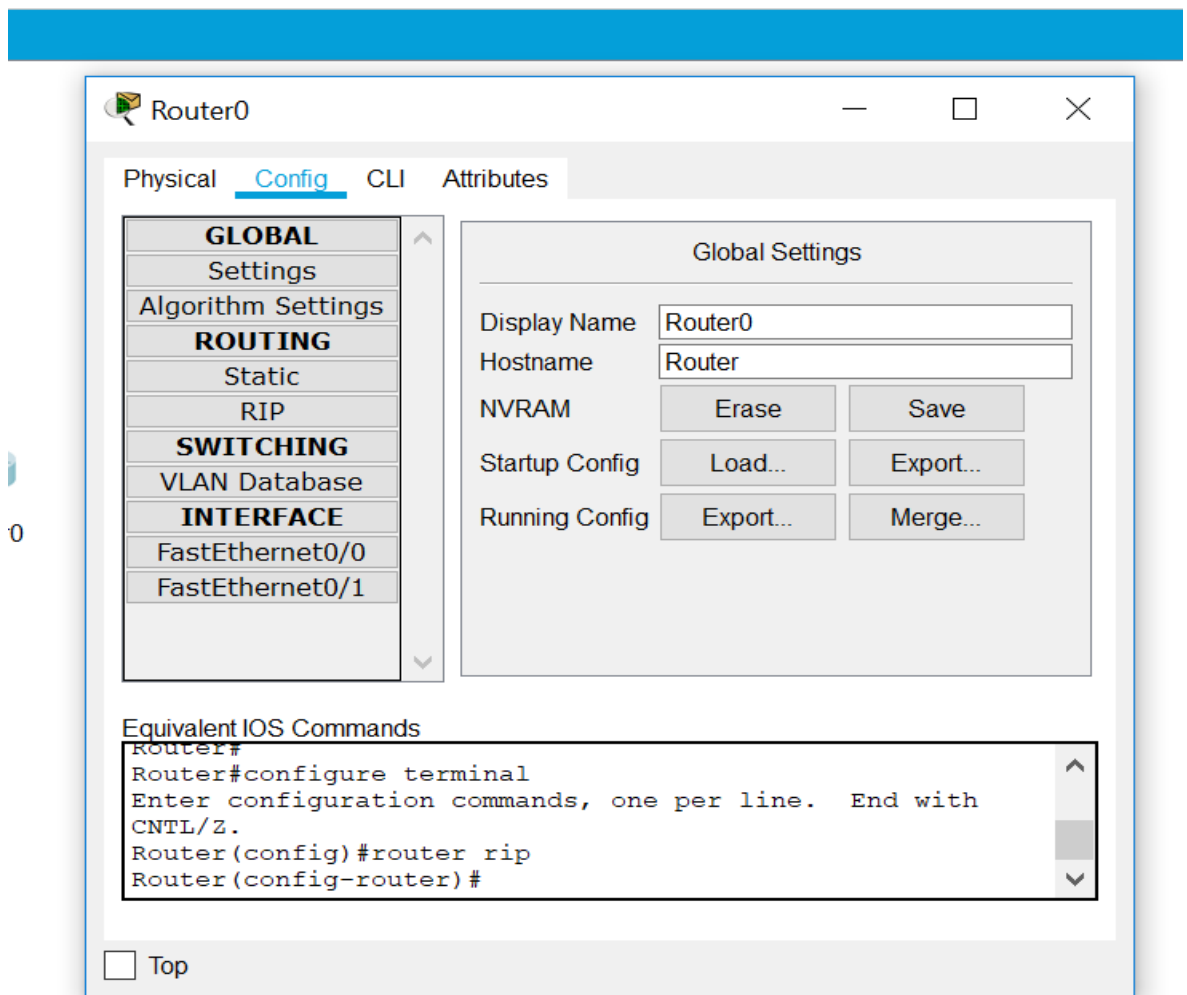
3. Setup a Router based wide area network using Dynamic routing (RIP, EIGRP, OSPF).
  - a) Set up a network of 3 and 4 routers. Configure routing in each router and test the network.

## Scenario #1

- Set up a network of 3 routers as shown in the following topology. Configure **dynamic routing** in each router. Add redundancy ( Even if a link fails, network should work )

Hint : Add stand by route information also in every routers table .





\*Select RIP in Routing

\*Enter Network address of directly connected subnets.

#### 4. Practice IP addressing principles.

a) Set up a Subnet (N1) comprising 4 nodes. Change the subnet masks in some of the nodes and test the network; Set up another Subnet (N2) of 4 nodes; Connect these two Subnets using a router.

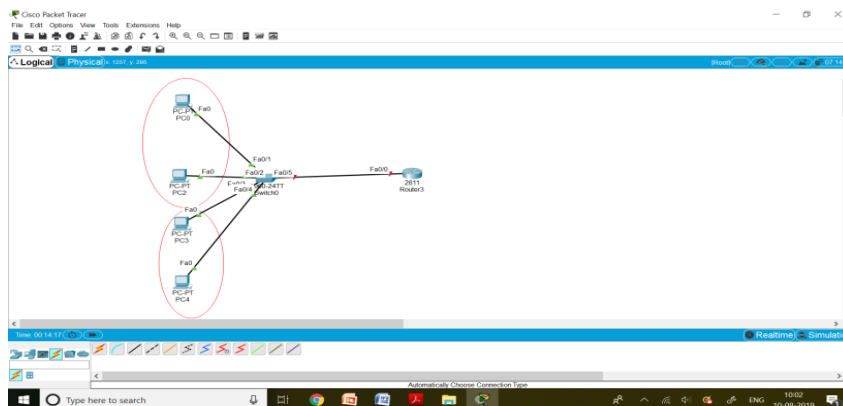
b) Create 4 equal sized subnets in the subnet N1 and test the network.

## Common principles of configuration

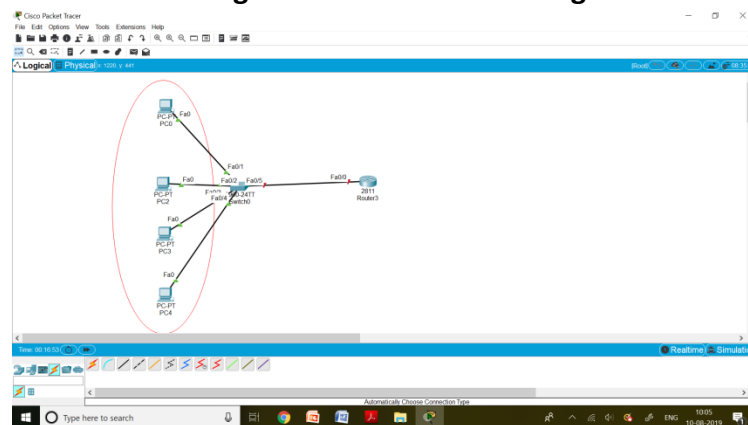
- Plan Subnet ID
- Decide the IP addresses for each Subnet
- Configure IP addresses ( along with the subnet mask ) for the end PCs
- Configure IP address of the Gateway in every PC
- Configure IP addresses for the Router interfaces

### Scenareo #1 – Single Subnet

#### Hosts not following IP Addressing principles

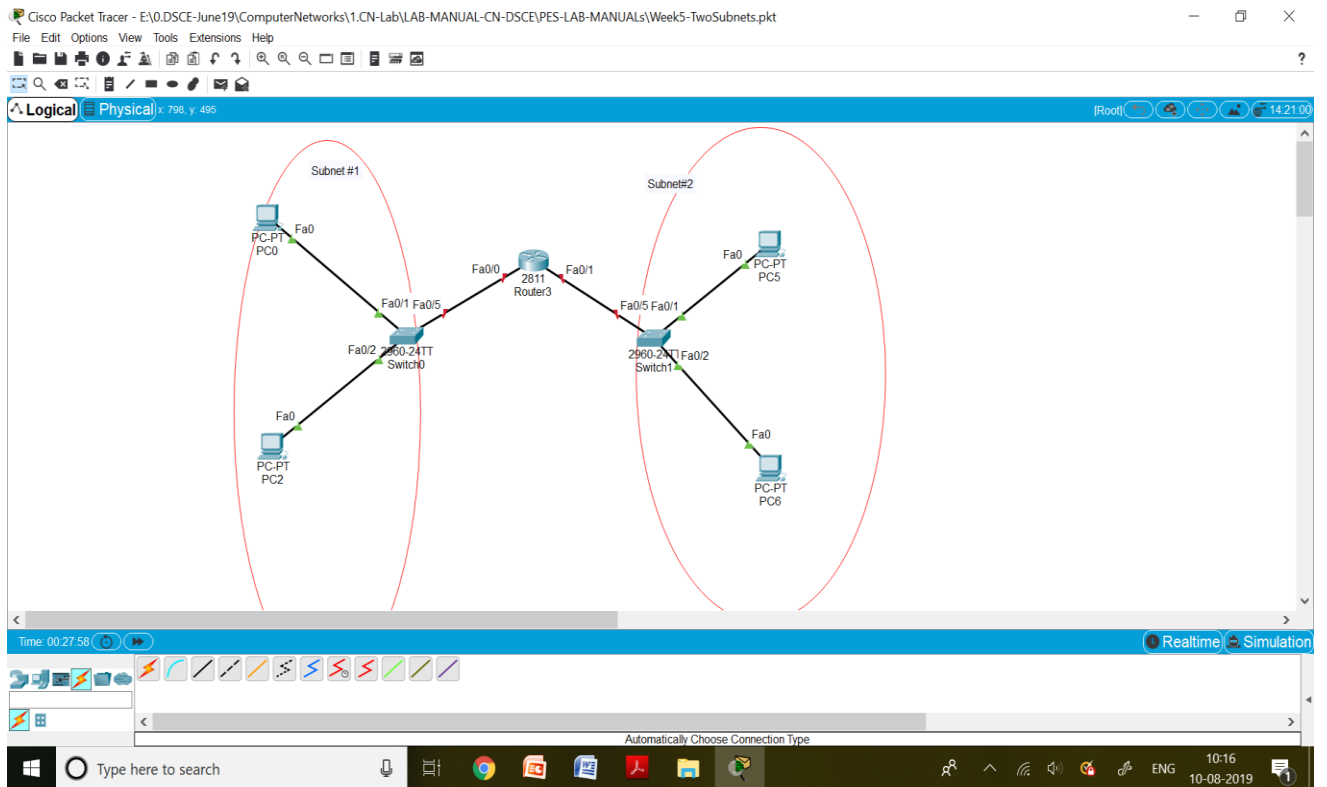


### Scenareo #2 –Single Subnet Hosts following IP Addressing principles

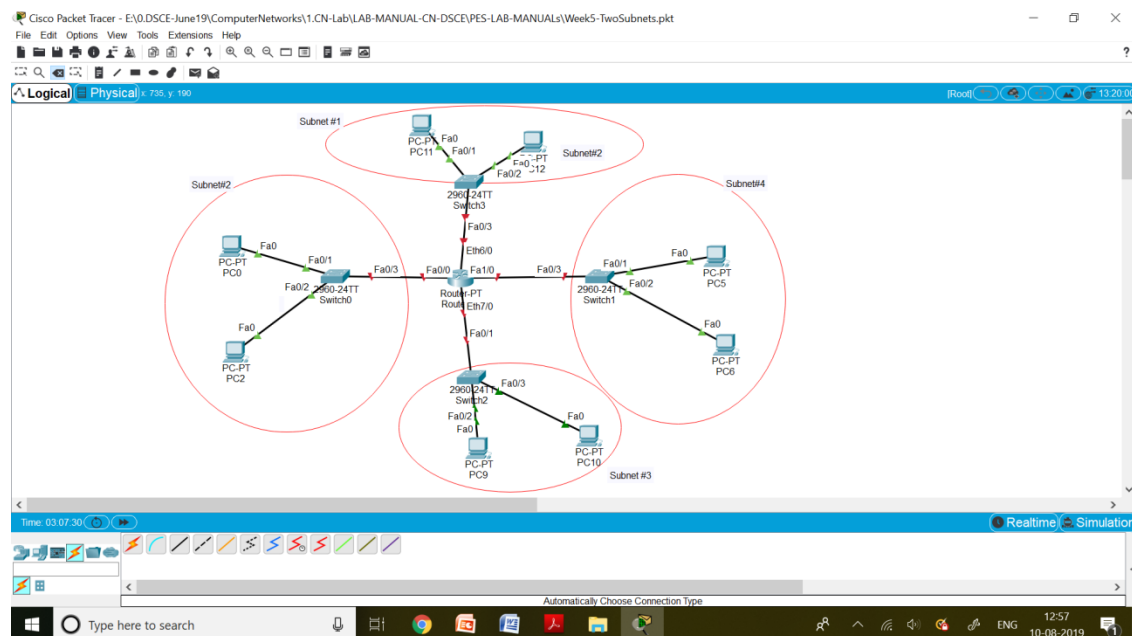


### Scenareo #3 –Creating 2 Subnets

#### Hosts following IP Addressing principles



## Scenario #4 –Creating 4 Subnets Hosts following IP Addressing principles



## 5. Implement **DHCP** and **DNS**.

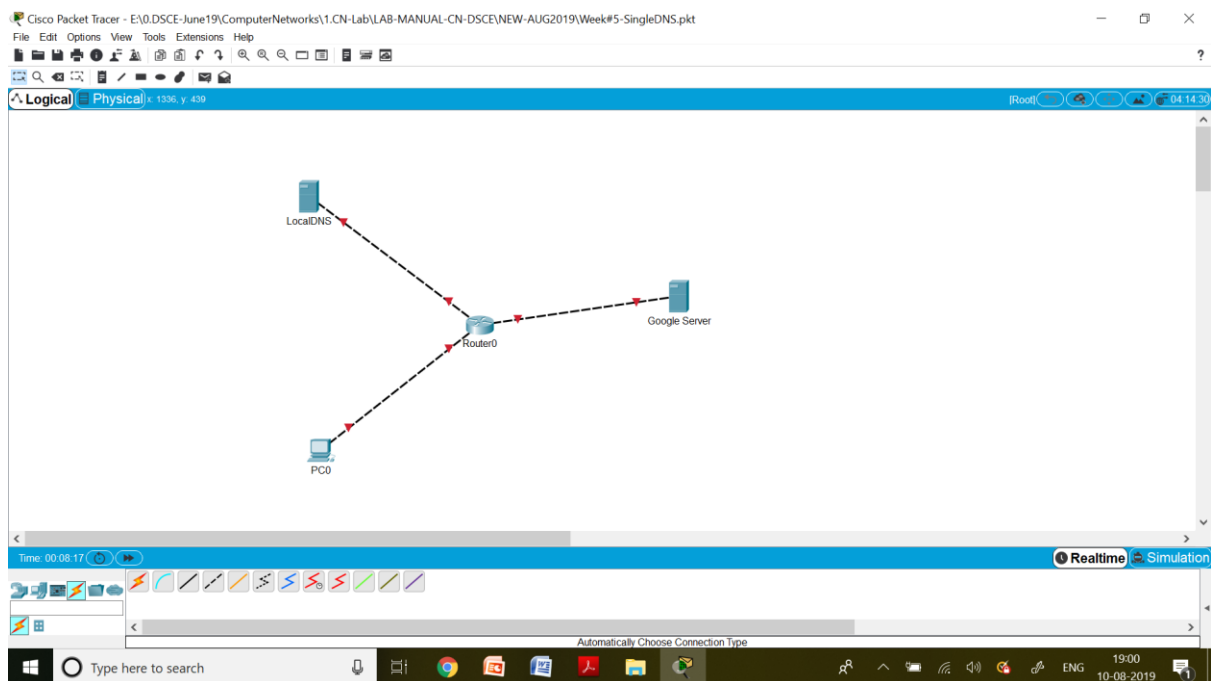
- a) A client, single DNS server and a Web Server
- b) A client, two DNS servers and a Web Server
- c) A client and a hierarchy of DNS servers and a Web Server

**Test the DNS operation in following 3 scenarios:**

### **Scenario #1 : A client, single DNS server and a Web Server**

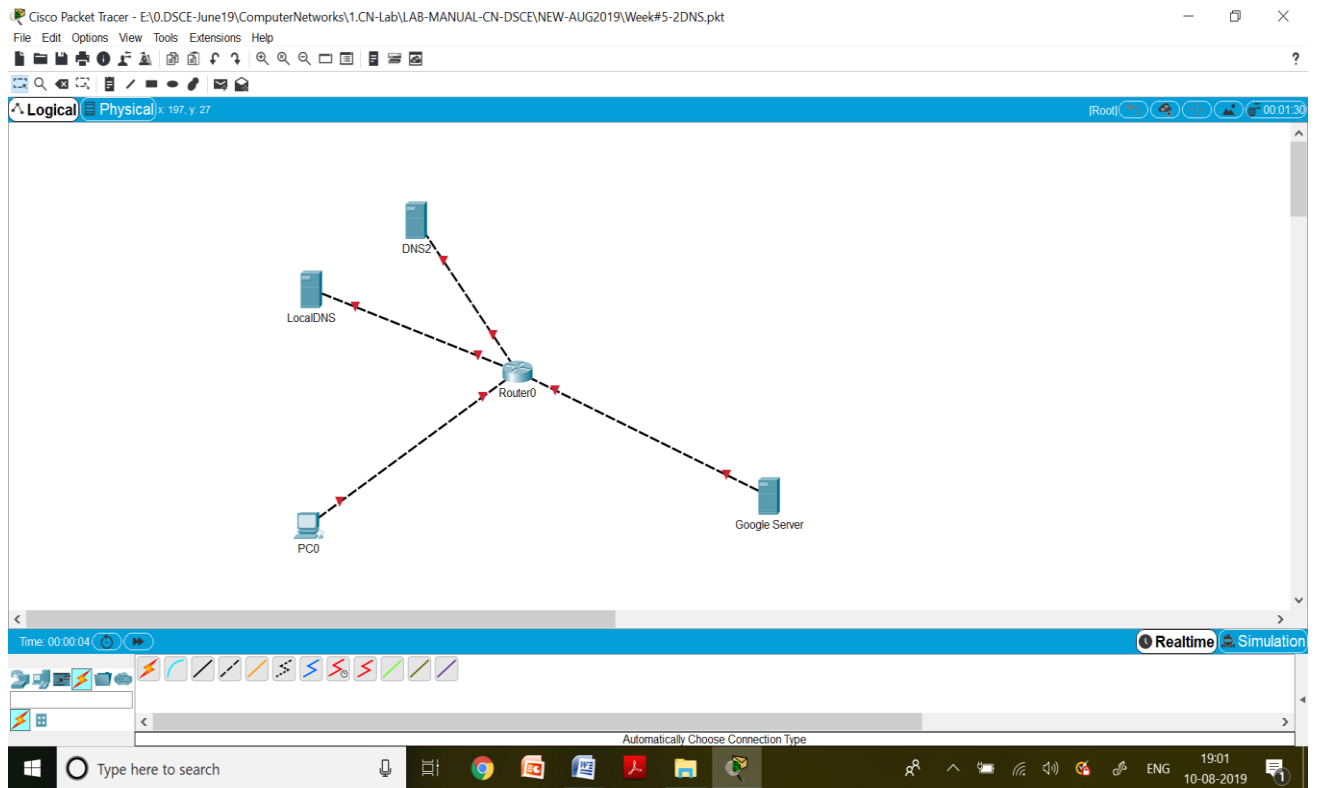
- Draw topology as per the following diagram
- Configure IP addresses, Gateway addresses
- Set up one of the servers HTTP server
- Set up another server as DNS server
- Create DNS table
  - Name and IP address of the web server
- At Client enter the IP address of the DNS Server

Browse from the client



## Scenario#2 : A client , two DNS servers and a Web Server

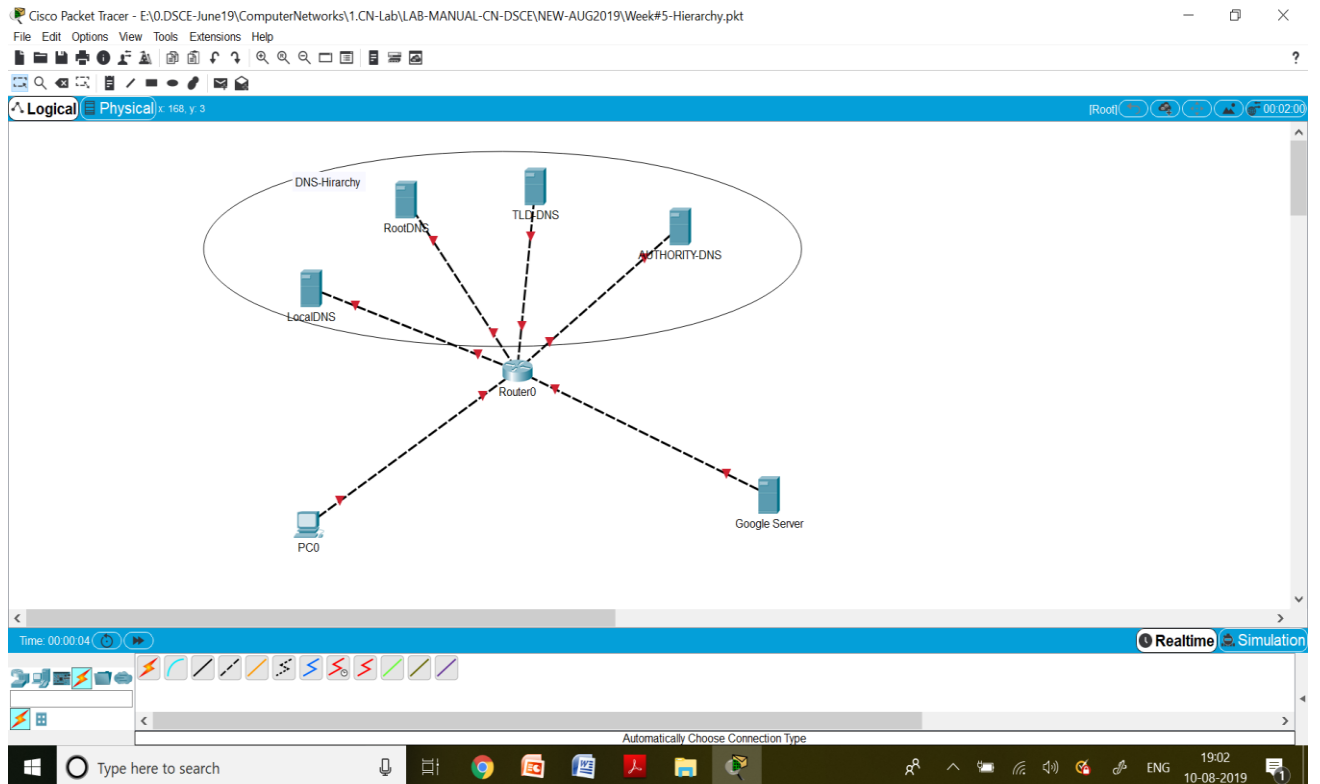
- First DNS server enter the IP address of the next DNS server
- Second DNS server, enter the Name and IP address of the Web server





### Scenario #3 : A client and a hierarchy of DNS servers and a Web Server

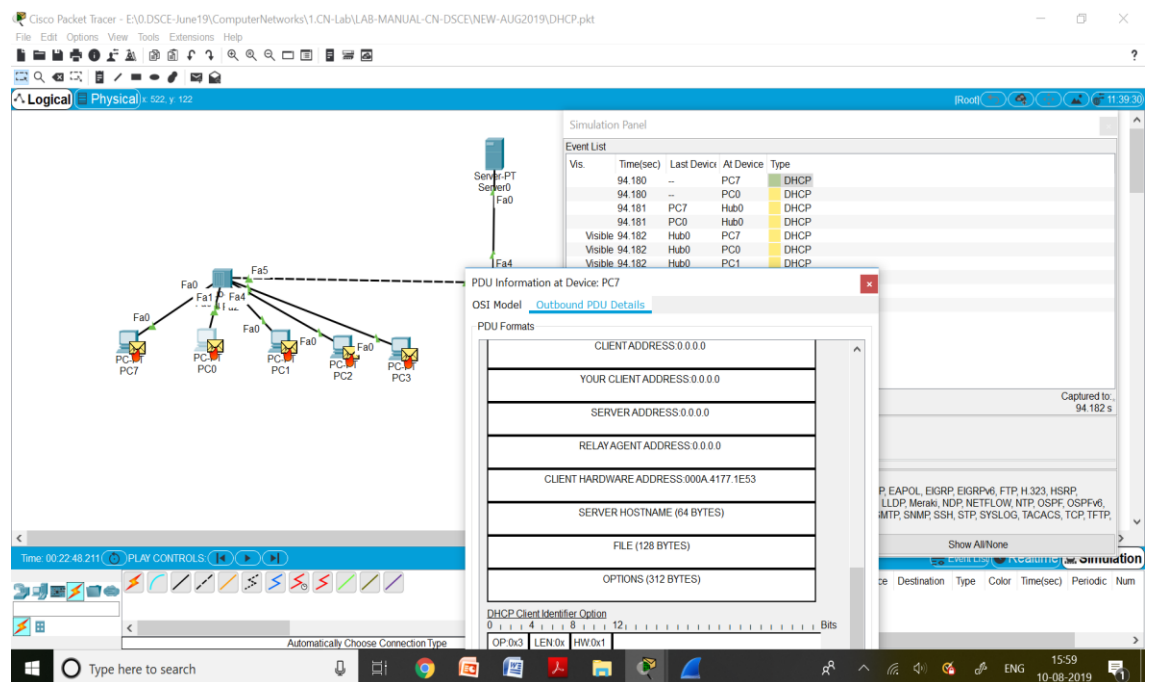
Continue the same steps as the previous scenario



## 6. Implement Static NAT, Dynamic NAT and PAT.

Set up and test a DHCP enabled LAN with 10 nodes . Use subnet ID 1.1.1.0/24.

- Select DHCP as services in the server
- Set up the pool of addresses
- Configure each client to DHCP mode
- Run simulation and analyse DHCP protocol

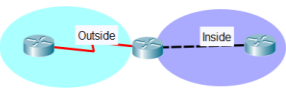


- Set up 2 different DHCP pools in the same subnet. Restrict the access to some of the nodes.
- Set up a subnet N1 with subnet ID 10.0.0.0/8. Set up another subnet N2 with subnet ID 10.0.0.0/8. Connect these two subnets using a public network using NAT.

Cisco Packet Tracer - C:\Program Files\Cisco Packet Tracer 7.2.1\saves\Router\NAT\Outside\_Nat.pkt

File Edit Options View Tools Extensions Help

Logical Physical x: 1456, y: 331 [Root] 00:49:30



The following network uses a static NAT entry to translate the outside address of Outside\_Router to an address in the local inside network.

Test Outside to Inside NAT

1. From Outside\_Router ping 171.68.1.1 using an extended ping (source address 172.16.88.1). The ping should be successful since the address is changed by the NAT router.
2. Use simulation mode to view the source address of the packets as they pass through the network.

Time: 00:01:41

Scenario 0

New Delete

Toggle PDU List Window

Automatically Choose Connection Type

Fire Last Status Source Destination Type Color Time(sec) Periodic Num

Windows taskbar: Type here to search, 16:25, 10-08-2019

## PART-B

Q7. A ) Write a C/C++ program to implement the data link layer framing methods. A) bit stuffing B) Character stuffing.

SOLUTION: #include<stdio.h>

```
#include<conio.h>
```

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

```
void main()
```

```
{
```

```
    int a[20],b[30],i,j,k,count,n;
```

```
    clrscr();
```

```
    printf("enter frame length:");
```

```
    scanf("%d",&n);
```

```
    printf("enter input frame(0's&1's only):");
```

```
    for(i=0;i<n;i++)
```

```
        scanf("%d",&a[i]);
```

```
    i=0;count=1;j=0;
```

```
    while(i<n)
```

```
    {
```

```
        if(a[i]==1)
```

```
        {
```

```
            b[j]=a[i];
```

```
            for(k=i+1;a[k]==1&& k<n&&count<5;k++)
```

```
            {
```

```
                j++; b[j]=a[k];
```

```
                count++;
```

```
                if(count==5)
```

```

        {
            j++;
            b[j]=0;
        }
        i=k;
    }
}
else {
    b[j]=a[i];
}
i++;
j++;
}

printf("After stuffing the frame is:");
for(i=0;i<j;i++)
printf("%d",b[i]);
getch();
}

```

Output:

Enter the number of bits: 10

1

0

1

0

1

1

1

1

1

1

Data after stuffing: 10101111101

Q7.B) Write a C/C++ program to implement the data link layer framing methods. B) Character stuffing

SOLUTION: //program for character stuffing

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<process.h>

void main()

{

{

int i=0,j=0,n,pos; char a[20],b[50],ch;

clrscr();

printf("enter string:\n");

scanf("%s",&a);

n=strlen(a);

printf("enter position\n");

scanf("%d",&pos);

if(pos>n)

{

printf("invalid position,Enter again:");

scanf("%d",&pos);
```



```
}  
printf("enter the character\n");  
ch=getche();  
b[0]='d';  
b[1]='l';  
b[2]='e';  
b[3]='s';  
b[4]='t';  
b[5]='x';  
j=6;  
while(i<n)  
{  
if(i==pos-1)  
{  
b[j]='d';  
b[j+1]='l';  
b[j+2]='e';  
b[j+3]=ch;  
b[j+4]='d';  
b[j+5]='l';  
b[j+6]='e';  
j=j+7;  
}  
if(a[i]=='d'&&a[i+1]=='l'&&a[i+2]=='e')  
{  
b[j]='d';  
b[j+1]='l';  
b[j+2]='e';  
j=j+3;  
}  
b[j]=a[i];
```

```
i++;  
j++;  
}  
b[j]='d';  
b[j+1]='l';  
b[j+2]='e';  
b[j+3]='e';  
b[j+4]='t';  
b[j+5]='x';  
b[j+6]='\0';  
printf("\n frame after stuffing: \n");  
printf("%s",b);  
getch();  
}
```

OUTPUT : Enter String: haiarchana

Enter position: 4

Enter the Character K

Frame after stuffing:  
dlestxhaidlekdlearchanadleetx

Q8: Write a C/C++ program to implement Distance Vector Routing algorithm.

SOLUTION: :

```
#include<stdio.h>
struct node
{
    unsigned dist[20];
    unsigned from[20];
}rt[10];
int main()
{
    int costmat[20][20];
    int nodes,i,j,k,count=0;
    printf("\nEnter the number of nodes : ");
    scanf("%d",&nodes); //Enter the nodes
    printf("\nEnter the cost matrix :\n");
    for(i=0;i<nodes;i++)
    {
        for(j=0;j<nodes;j++)
        {
            scanf("%d",&costmat[i][j]);
            costmat[i][i]=0;
            rt[i].dist[j]=costmat[i][j]; //initialise
the distance equal to cost matrix
            rt[i].from[j]=j;
        }
    }
    do
    {
        count=0;
        for(i=0;i<nodes;i++) //We choose arbitrary vertex k
and we calculate the direct distance from the node i to k
using the cost matrix
        //and add the distance from k to node j
        for(j=0;j<nodes;j++)
        for(k=0;k<nodes;k++)
            if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
            { //We calculate the minimum distance
                rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
                rt[i].from[j]=k;
                count++;
            }
    } while(count!=0);
    for(i=0;i<nodes;i++)
    {
        printf("\n\n For router %d\n",i+1);
        for(j=0;j<nodes;j++)
        {
            printf("\t\nnode %d via %d Distance %d
",j+1,rt[i].from[j]+1,rt[i].dist[j]);
        }
    }
```

```
    }  
    printf("\n\n");  
    getch();  
}
```

OUTPUT: Enter the cost matrix :

0 2 7

2 0 1

7 1 0

For router 1

node 1 via 1 Distance 0

node 2 via 2 Distance 2

node 3 via 3 Distance 3

For router 2

node 1 via 1 Distance 2

node 2 via 2 Distance 0

node 3 via 3 Distance 1

For router 3

node 1 via 1 Distance 3

node 2 via 2 Distance 1

node 3 via 3 Distance 0

## 9) Write a C/C++ Program To Implement Stop and Wait Flow Control Protocol.

SOLUTION :

```
#include<iostream>

#include <time.h>

#include <cstdlib>
#include<ctime>
#include <unistd.h>
using namespace std;
class timer {
private:
    unsigned long begTime;
public:
    void start() {
        begTime = clock();
    }
    unsigned long elapsedTime() {
        return ((unsigned long) clock() - begTime) / CLOCKS_PER_SEC;
    }
    bool isTimeout(unsigned long seconds) {
        return seconds >= elapsedTime();
    }
};
int main()
{
    int frames[] = {1,2,3,4,5,6,7,8,9,10};
    unsigned long seconds = 5;
    srand(time(NULL));
    timer t;
    cout<<"Sender has to send frames : ";
    for(int i=0;i<10;i++)
        cout<<frames[i]<<" ";
    cout<<endl;
    int count = 0;
    bool delay = false;
    cout<<endl<<"Sender\t\t\t\t\tReceiver"<<endl;
    do
    {
        bool timeout = false;
        cout<<"Sending Frame : "<<frames[count];
        cout.flush();
        cout<<"\t\t";
        t.start();
        if(rand()%2)
        {
            int to = 24600 + rand()%(64000 - 24600) + 1;
            for(int i=0;i<64000;i++)
                for(int j=0;j<to;j++) {}
        }
        if(t.elapsedTime() <= seconds)
        {
            cout<<"Received Frame : "<<frames[count]<<" ";
            if(delay)
            {
                cout<<"Duplicate";
                delay = false;
            }
        }
    }
```

```

        cout<<endl;
        count++;
    }
    else
    {
        cout<<"---"<<endl;
        cout<<"Timeout"<<endl;
        timeout = true;
    }
    t.start();
    if(rand()%2 || !timeout)
    {
        int to = 24600 + rand()%(64000 - 24600) + 1;
        for(int i=0;i<64000;i++)
            for(int j=0;j<to;j++) {}
        if(t.elapsedTime() > seconds )
        {
            cout<<"Delayed Ack"<<endl;
            count--;
            delay = true;
        }
        else if(!timeout)
            cout<<"Acknowledgement : "<<frames[count]-1<<endl;
    }
}while(count!=10);
return 0;
}

```

## OUTPUT

**Sender has to send frames : 1 2 3 4 5 6 7 8 9 10**

**Sender      Receiver**

**Sending Frame : 1      Received Frame : 1**

**Acknowledgement : 1**

**Sending Frame : 2      ---**

**Timeout**

**Sending Frame : 2      Received Frame : 2**

**Acknowledgement : 2**

**Sending Frame : 3      Received Frame : 3**

**Acknowledgement : 3**

**Sending Frame : 4      Received Frame : 4**

**Acknowledgement : 4**

**Sending Frame : 5      Received Frame : 5**

**Acknowledgement : 5**

**Sending Frame : 6      Received Frame : 6**

**Acknowledgement : 6**

**Sending Frame : 7      Received Frame : 7**

**Delayed Ack**

**Sending Frame : 7      Received Frame : 7 Duplicate**

**Delayed Ack**

**Sending Frame : 7      ---**

**Timeout**

**Sending Frame : 7      Received Frame : 7 Duplicate**

**Acknowledgement : 7**

**Sending Frame : 8      ---**

**Timeout**

**Delayed Ack**

**Sending Frame : 7      Received Frame : 7**

**Acknowledgement : 7**

**Sending Frame : 8      Received Frame : 8**



**Acknowledgement : 8**  
**Sending Frame : 9   Received Frame : 9**  
**Delayed Ack**  
**Sending Frame : 9   ---**  
**Timeout**  
**Sending Frame : 9   Received Frame : 9 Duplicate**  
**Delayed Ack**  
**Sending Frame : 9   Received Frame : 9 Duplicate**  
**Acknowledgement : 9**  
**Sending Frame : 10   ---**  
**Timeout**  
**Sending Frame : 10   Received Frame : 10**  
**Acknowledgement : 10**

10. Write a C++ Program for ERROR detecting code using CRC-CCITT (16bit).

SOLUTION: #include<stdio.h>  
#include<conio.h>

```
int main(void)
{
int data[50],div[16],rem[16];
int datalen, divlen, i,j,k;
int ch;
clrscr();
printf("Enter the data: ");
i = 0;
while((ch = fgetc(stdin)) != '\n')
{
if(ch == '1')
data[i] = 1;
else
data[i] = 0;
i++;
}
datalen = i;
printf("\nEnter the divisor: ");
i = 0;
while((ch = fgetc(stdin)) != '\n')
{
if(ch == '1')
div[i] = 1;
else
div[i] = 0;
i++;
}
divlen = i;
for(i = datalen ; i < datalen + divlen - 1 ; i++)
data[i] = 0;
datalen = datalen + divlen - 1;
for(i = 0 ; i < divlen ; i++)
rem[i] = data[i]; k = divlen-1;
while(k < datalen)
if(rem[0] == 1)
{
for(i = 0 ; i < divlen ; i++)
rem[i] = rem[i] ^ div[i];
```

```

    }
Else
{
    if(k == datalen-1)
    break;
    for(i = 0 ; i < divlen-1 ; i++)
    {
        rem[i] = rem[i+1];
        printf("%d",rem[i]);
    }
    rem[i] = data[++k];
    printf("%d\n",rem[i]);
}
j=1;
for(i = datalen - divlen + 1 ; i < datalen ; i++)
{
    data[i] = rem[j++];
}
printf("\nThe data to be sent is\n");
for(i = 0 ; i < datalen ; i++)
    printf("%d",data[i]);
getch();
return 0;
}

```

OUTPUT: Enter the data: 10101111

Enter the divisor: 1011

0011

0111

1111

1001

0100

1000

0110

The data to be sent is 10101111110

Q5) Write a C/C++ Program for Congestion control using Leaky Bucket Algorithm..

SOLUTION : #include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

>

#define NOF\_PACKETS 10

```

int rand(int a)
{
    int rn = (random() % 10) % a;
    return rn == 0 ? 1 : rn;
}

int main()
{
    int packet_sz[NOF_PACKETS], i, clk, b_size, o_rate, p_sz_rm=0, p_sz, p_time, op;
    for(i = 0; i<NOF_PACKETS; ++i)
        packet_sz[i] = rand(6) * 10;
    for(i = 0; i<NOF_PACKETS; ++i)
        printf("\npacket[%d]:%d bytes\t", i, packet_sz[i]);
    printf("\nEnter the Output rate:");
    scanf("%d", &o_rate);
    printf("Enter the Bucket Size:");
    scanf("%d", &b_size);
    for(i = 0; i<NOF_PACKETS; ++i)
    {
        if( (packet_sz[i] + p_sz_rm) > b_size)
            if(packet_sz[i] > b_size)/*compare the packet siz with bucket size*/
                printf("\nIncoming packet size (%dbytes) is Greater than bucket capacity (%dbytes)-PACKET REJECTED", packet_sz[i], b_size);
            else
                printf("\nBucket capacity exceeded-PACKETS REJECTED!!");
        else
        {
            p_sz_rm += packet_sz[i];
            printf("\nIncoming Packet size: %d", packet_sz[i]);
            printf("\nBytes remaining to Transmit: %d", p_sz_rm);
            p_time = rand(4) * 10;
            printf("\nTime left for transmission: %d units", p_time);
            for(clk = 10; clk <= p_time; clk += 10)
            {
                sleep(1);
                if(p_sz_rm)
                {
                    if(p_sz_rm <= o_rate)/*packet size remaining comparing with output rate*/
                        op = p_sz_rm, p_sz_rm = 0;
                    else
                        op = o_rate, p_sz_rm -= o_rate;
                    printf("\nPacket of size %d Transmitted", op);
                    printf("----Bytes Remaining to Transmit: %d", p_sz_rm);
                }
            }
            else
            {
                printf("\nTime left for transmission: %d units", p_time-clk);
                printf("\nNo packets to transmit!!");
            }
        }
    }
}

```

```

    }
  }
}

```

OUTPUT: Enter The Bucket Size 5  
 Enter The Operation Rate 2  
 Enter The No. Of Seconds You Want To Stimulate 3  
 Enter The Size Of The Packet Entering At 1 sec 5  
 Enter The Size Of The Packet Entering At 1 sec 4  
 Enter The Size Of The Packet Entering At 1 sec 3  
 Second|Packet Recieved|Packet Sent|Packet Left|Packet Dropped|

```

-----
1  5  2  3  0
2  4  2  3  2
3  3  2  3  1
4  0  2  1  0
5  0  1  0  1

```

12. Write a program on a datagram socket for client/server to display the messages on client side, typed at the server side.

```
SOLUTION : #include <stdio.h>

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

#define PORT 8080
#define MAXLINE 1024

// Driver code
int main() {
    int sockfd;
    char buffer[MAXLINE];
    char *hello = "Hello from server";
    struct sockaddr_in servaddr, cliaddr;

    // Creating socket file descriptor
    if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));
    memset(&cliaddr, 0, sizeof(cliaddr));

    // Filling server information
    servaddr.sin_family = AF_INET; // IPv4
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);

    // Bind the socket with the server address
    if ( bind(sockfd, (const struct sockaddr *)&servaddr,
        sizeof(servaddr)) < 0 )
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    int len, n;

    len = sizeof(cliaddr); //len is value/result
```



```

    n = recvfrom(sockfd, (char *)buffer, MAXLINE,
        MSG_WAITALL, ( struct sockaddr *) &cliaddr,
        &len);
    buffer[n] = '\0';
    printf("Client : %s\n", buffer);
    sendto(sockfd, (const char *)hello, strlen(hello),
        MSG_CONFIRM, (const struct sockaddr *) &cliaddr,
        len);
    printf("Hello message sent.\n");

    return 0;
}
// CLIENT SIDE CODE
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define PORT    8080
#define MAXLINE 1024

// Driver code
int main() {
    int sockfd;
    char buffer[MAXLINE];
    char *hello = "Hello from client";
    struct sockaddr_in servaddr;

    // Creating socket file descriptor
    if ( (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0 ) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));

    // Filling server information
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = INADDR_ANY;

    int n, len;

    sendto(sockfd, (const char *)hello, strlen(hello),
        MSG_CONFIRM, (const struct sockaddr *) &servaddr,
        sizeof(servaddr));

```

```
printf("Hello message sent.\n");

n = recvfrom(sockfd, (char *)buffer, MAXLINE,
             MSG_WAITALL, (struct sockaddr *) &servaddr,
             &len);
buffer[n] = '\0';
printf("Server : %s\n", buffer);

close(sockfd);
return 0;
}
```

**Output :**

```
$ ./server
Client : Hello from client
Hello message sent.
$ ./client
Hello message sent.
Server : Hello from server
```