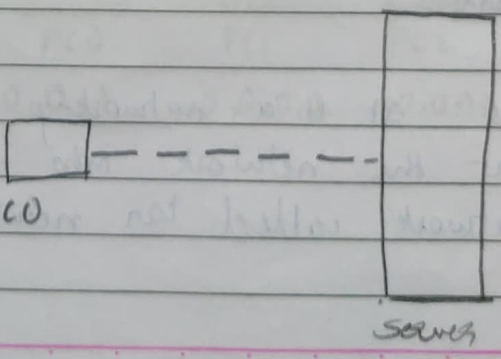
CISCO PACKET TRACER

Date 13/11/22
Page

1> Interface overview:

When we open packet tracer we are presented by:
-> menu bar
-> main tool bar - consists of shortcut to file and edit commands
-> workspace => two types
    -> logical : not relative to position
    -> ~~Real~~ physical : relative to position
-> Simulation bar => has real time simulation as well
-> network component box : device type selection box and device specific box
-> Device type box : consists of types of devices routers, hubs, switches etc
-> Device specific box : consists of specific devices for a type of device
-> user created packet window : This window manages the packets you put in the network during simulation mode.

IP address V4 : 32 bits - 4 octets
subnet mask = network address + host address



Date / /
Page

Creating first packet tracer

1. Open packet tracer
2. In device selective select a end device and servers
3. In connection select copper cross over cable and add to PC-PT PC0 and select fast ethernet and connect to server-PT server0 fast ethernet. The lights turn green indicating a working connection.
4. Click on server and in desktop top select ipconfig set the ipv4 to 10.0.0.1 and the subnet mask is generated for the entered ipv4
5. Repeat the same process for PC0 and set ipv4 to 10.0.0.2
6. From common tool bar select data packet and place it on generic pc and then on generic server select realtime or simulation if in realtime it shows transferred successfully the connections are proper. In simulation play controls → auto capture → play controls → auto capture / play then you can see realtime transfer of packet.

Router: A router is a networking device used to transfer data packet b/w computer network

Switches: It is a networking device used to segment the network into different subnetwork called lan networks.

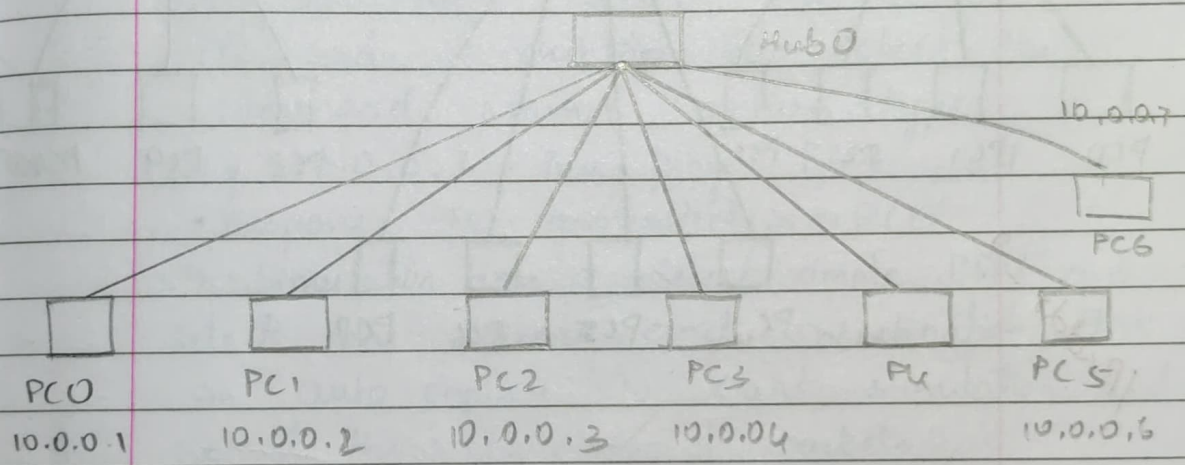
10/11/22

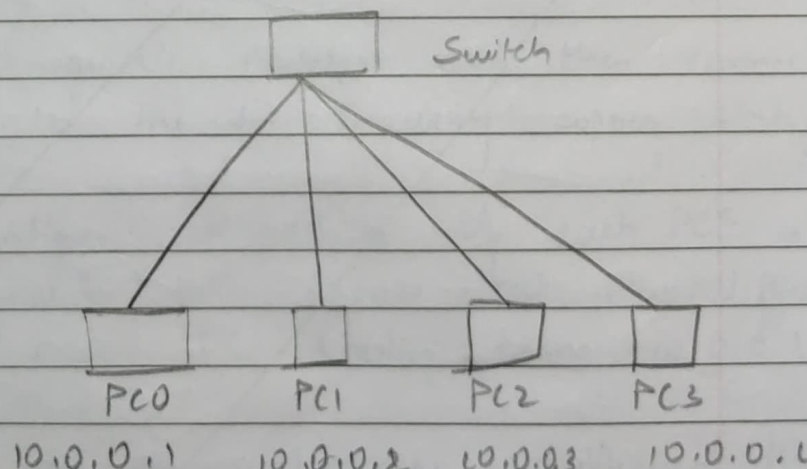
Date 10/11/22
Page

Experiment - 1

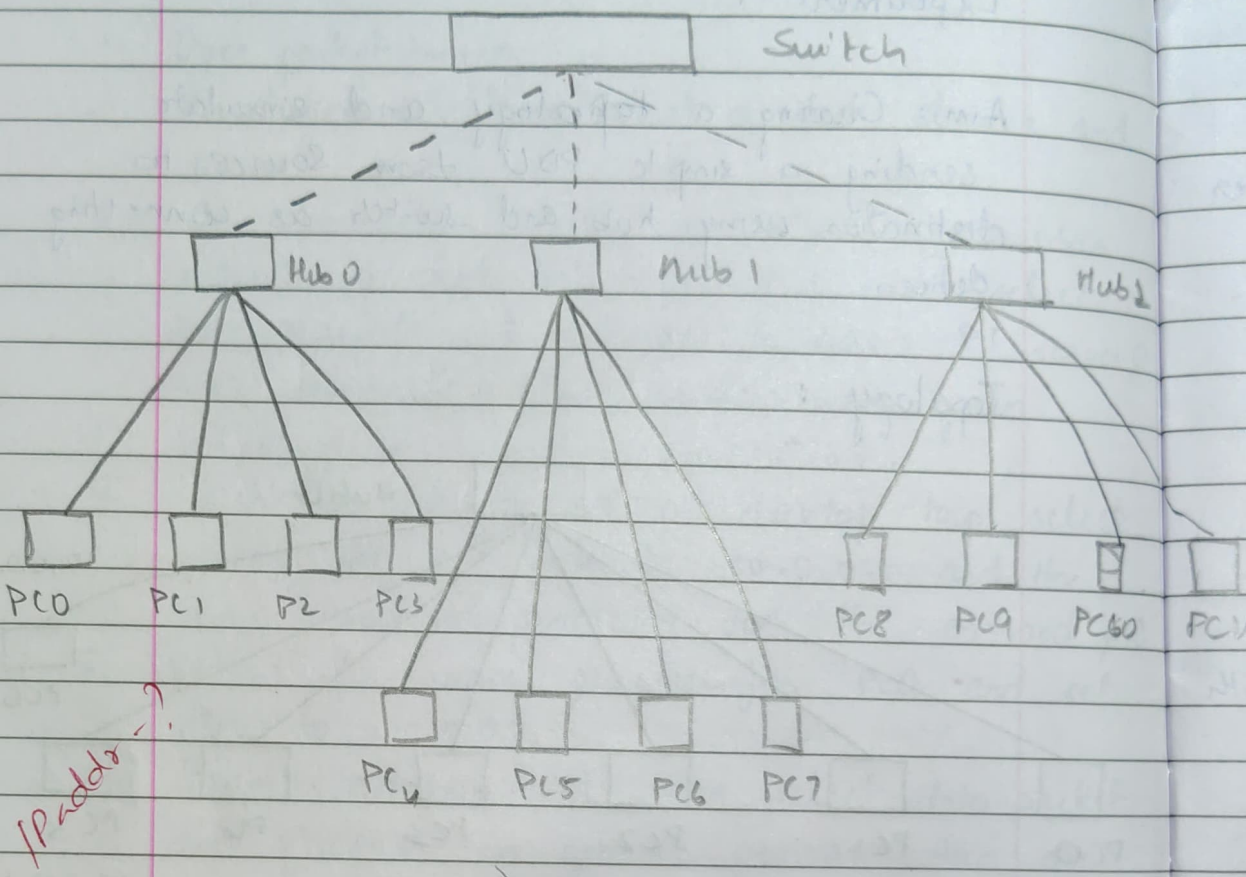
Aim: Creating a topology and simulate sending a simple PDU from source to destination using hub and switch as connecting device.

Topology:





Date __/__/__
Page _____



IP addr - ?

Date / /
Page

Procedure:

using Hub: 1) Add generic hub and seven PC's to the logical workspace
2) Configure the IP address of the PC's from 10.0.0.1 to 10.0.0.7
3) Connect all PC's to hub using copper straight cable (because different layers)
4) Real time: Select the source PC and in desktop tab select the command prompt option type ping 10.0.0.3. This pings PC2 and response is generated in PC0
5) Simulation time: Select simple PDU and select the source and destination. Click on auto capture to start simulation and start the transfer of packets

Using switch:

1) Add generic switches and then connect PC's to the switch using copper straight cable.
2) configure IP address of each PCs in the configuration tab. Ensure that IP is different for every PC starting from 10.0.0.1 to 10.0.0.7
3) If no of switch are insufficient then add parts by clicking on device

Real time sim: Select source PC and in the desktop select command prompt option. In

command line ping destination PC by specifying its IP.

simulation: Select the simple PDU and select the source and destination computer clicking on auto capture allows us to see packet transfer.

Hybrid mode:

i) Add a switch and three hubs and 12 pc's

ii) connect switch and three hubs with copper cross over wire because it is in the same layer

iii) Connect the 12 pcs in group of 4 to the three hubs respectively using copper straight wire.

iv) Configure the IP of each of the PC and add a note below each PC displaying the IP addresses

Realtime mode: Select PC you want to send packet from and open its command prompt specify the destination PC by specifying its IP address. A response is sent by destination PC to source PC.

Date __/__/__
Page _____

Simulation mode: Add a simple PDU by selecting the pair of PC and click on auto capture from right panel.

Observation:

-> Hub:

Learning outcome: i) ~~token~~ Connection b/w hub and devices is established through copper straight through wire as they belong to different layers

ii) No. of ports can be added if needed by clicking on the device and adding the necessary port.

iii) Hub broadcasts the packets to all the connected devices and the packet is only accepted by matching with IP address.

Result: PC > ping 10.0.0.7

pinging 10.0.0.7 with 32 bytes of data

Reply from 10.0.0.7 : byte = 32 time 0ms

Reply from 10.0.0.7 byte = 32 time 0ms

ping statistic for 10.0.0.7

packet set = 2 recieve 2 lost

Switches :

-> Learning outcome :

i) When source device sends a message to switch it only sends the message to the destination unlike hub which broadcasts over the entire network.

ii) Connection b/w the switch and end devices is through copper straight-cable as they belong to different network layers.

iii) No of ports can be added if needed by clicking on device and adding the necessary ports.

Hybrid mode :

Learning outcome : i) switch and hub are connected through copper ~~straight~~ crossover cable and the devices and hubs are connected using ~~cop~~ cable copper ~~cro~~ straight through as they are different network.

ii) Switch after reciving the packet sends it to the hub containg the destination PC.

Present legibly

17/11/22

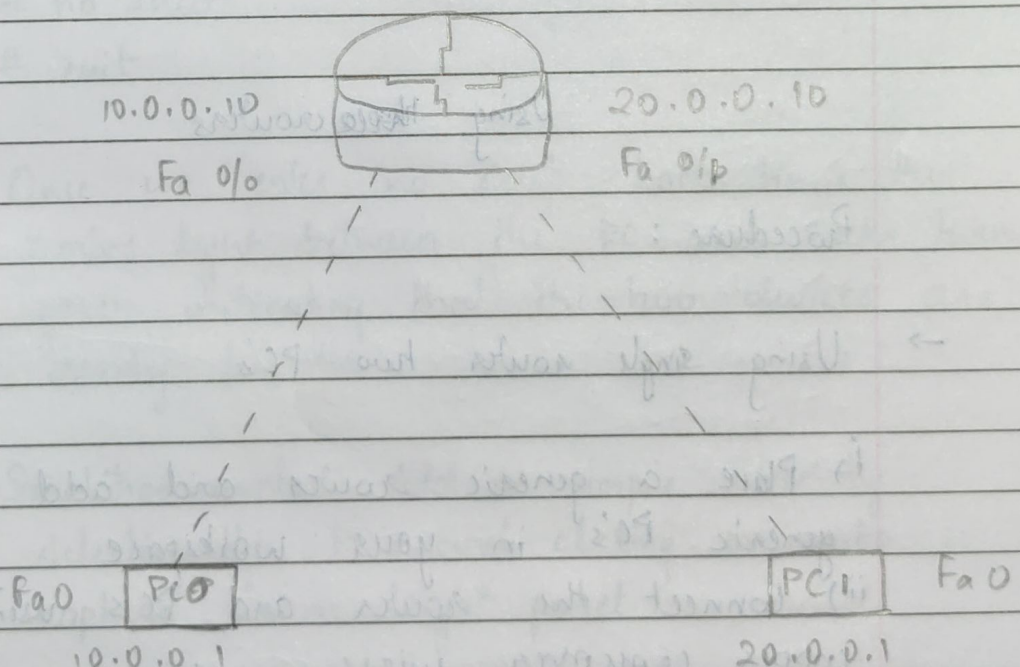
Date __/__/__
Page _____

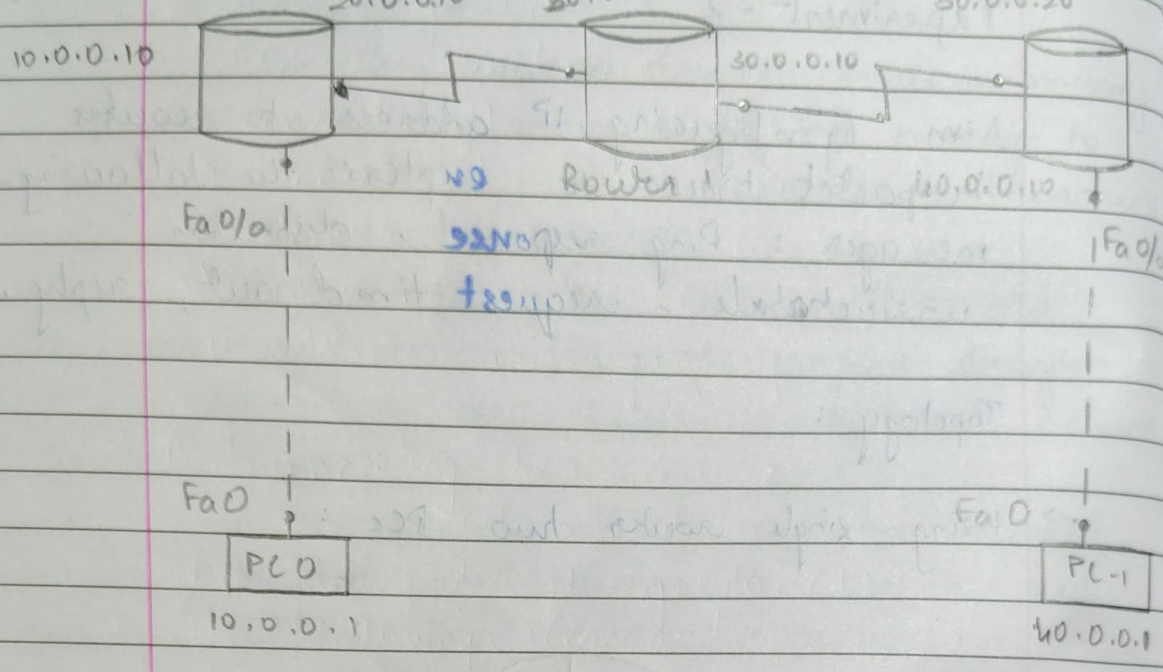
Experiment - 2

Aim: Configuring IP address to router in packet tracer to explore the following messages: Ping response, destination unreachable, request timed out, reply.

Topology:

-> Using single router two PCs:





Using three routers

Procedure :

→ Using single router two PC's

i) Place a generic router and add two generic PC's in your workspace
ii) Connect the router and PC's using copper crossover wires
iii) configure the IP address of each PC and in the configuration tab, under settings set gateway for both PCs to the router
iv) click on the generic router and go to the CLI tab and enter the following command to set up connection between PCs and generic routers through gateway 10.0.0.10

Date __/__/__
Page _____

→ enable
config t
(config) # interface fast ethernet 0/0
(config-if) # ip address 10.0.0.10 255.0.0.0
no shut
exit

No to set up connection between PC1 and the router gateway 20.0.0.10

interface fastethernet 1/0
ip address 20.0.0.10 255.0.0.0
no shut
exit

Once we enter no shut both times the amber light between the PC and router turns green indicating that the two devices are ready to use.

Simulation mode : Add a simple PDU by selecting the PC's and clicking on auto capture from right panel

Real time mode : Select the PC you want to send the packet from which is PC0 and open the command prompt from desktop ~~tap~~ tab. Specify destination address. A response is sent from destination PC to source PC

Date / /
Page

→ Using three routers and two PCs

i) Place 3 generic router and two generic PCs in workspace

ii) connect all routers using serial DCE

iii) connect router and PC using copper crossover wire.

iv) Click on each PC and configure the IP address and gateway of PCs.

For connecting two routers

Click on Router 0 go to CLI and enter the following command

-> no
-> enable
-> config t
-> interface serial 2/0
-> ip address 20.0.0.10 255.0.0.0
-> no shut
-> exit

Click on Router 1 and go to CLI and enter the following command

-> enable
-> config t
-> interface serial 2/0
-> ip address 20.0.0.20 255.0.0.0
-> no shut.

After this the red light between routers turn green.

Date __/__/__
Page _____

For connecting two devices (Router and PC)

i) Since IP address is configured,

ii) open CLI of Router 0 and enter the following commands

-> enable
-> config t
-> interface fastethernet 0/0
-> ip address 10.0.0.10 255.0.0.0
-> no shut

The red light turns green meaning that the router is ready for communication

Teaching Router 0 of network 30/40

-> enable
-> config t
-> interface serial 2/0 40.0.0.10
-> ip route 30.0.0.10/8 255.0.0.0 20.0.0.2
-> end
-> show ip route

Similarly do this for both the routers

Simulation mode : Add a simple PDU by selecting two PCs and clicking on add to capture / play.

Date / /
Page

Realtime :- Select the PC PC0 and go to command prompt ~~once~~ and ping the router 0 and repeat it for router 1 and router 2 finally ping PC1

Observation

→ 1 Router :

When PC0 pings PC1 for the first time we get request timed out.

Now if we ping PC1 we get all 4 packets without any packet loss.

→ 3 Routers :

Before training the network we get the result as destination host unreachable

After ~~the~~ training the router we get clear statistics as the result.

Results :

Using single routers :

Ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data

Request timed out

Reply from 20.0.0.1 bytes = 32 time<1ms

Reply from 20.0.0.1 byte = 32 time<1ms

Reply from 20.0.0.1 byte = 32 time<1ms

Reply from 20.0.0.1 byte = 32 time<1ms

ping statistics for 20.0.0.1

packets = 4 recieved = 4 lost = 0

Date / /
Page

Using three router two PCs

1> Ping 40.0.0.1

pinging 40.0.0.1 with 32 bytes of data

Reply from 10.0.0.10 : bytes = 32 time = 0ms
Reply from 10.0.0.10 : bytes = 32 time = 0ms
Reply from 10.0.0.10 : bytes = 32 time = 0
Reply from 10.0.0.10 : bytes = 32 time = 0

Ping statistic for 40.0.0.1
Packet sent = 4 recieved = 4 lost = 0

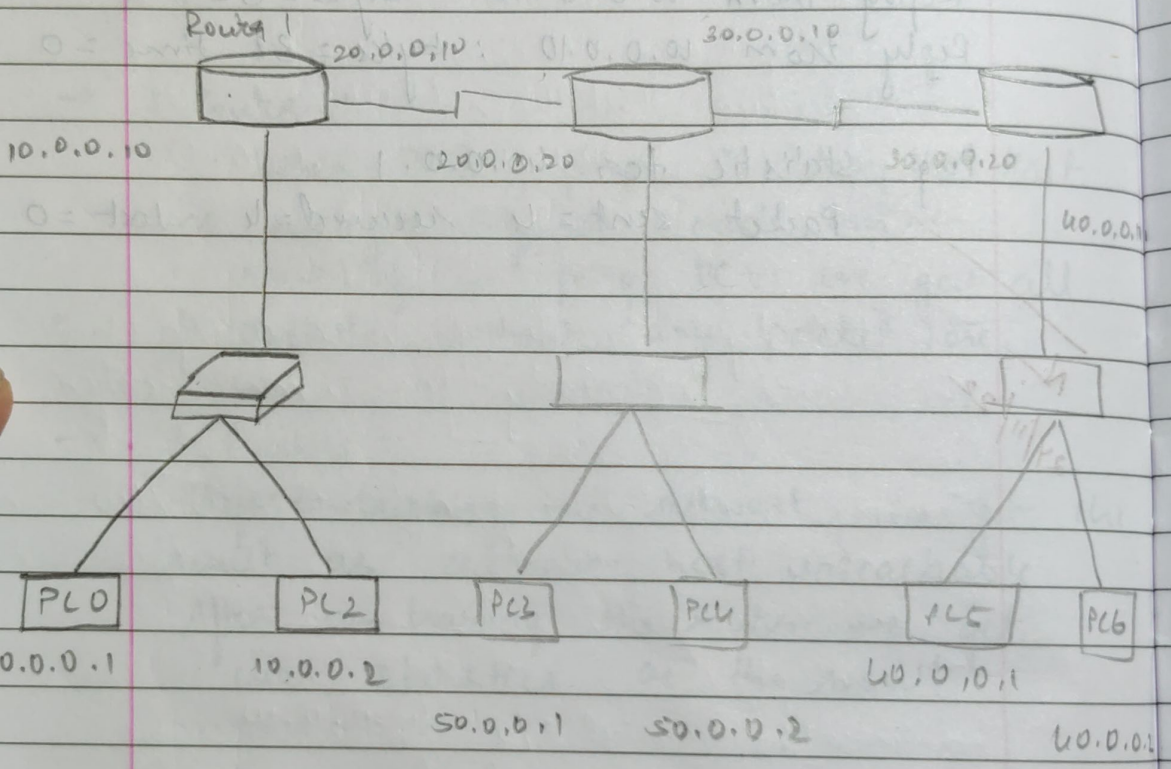
24/11/22

Date 24/11/22
Page

Experiment - 3

Aim: Configuring default route to the router.

Topology:



Procedure:

i) Place 3 generic routers and two generic PCs in workspace

ii) connect all routers using single dce cable and connect all switch, router and PC's with copper straight as shown above.

iii) A PC is clicked to set the attribute for a PC and each PC has three attributes which are the IP address.

Date
Page

subnet mask and the gateway and all the three are set according to the nodes placed. This process is done for all 6 PC's

iv) For router 1, the configuration are done in the command line interface. The IP address and subnet mask are set for both interfaces - fastethernet 0/0 as 10.0.0.10 and 255.0.0.0 and serial 2/0 as 40.0.0.1 and 255.0.0.0 Router 2 is the default router for Router 1 and this is done by the command ip route 0.0.0.0 0.0.0.0 0.0.0.0

v) For Router 2 the IP address and subnet mask are set for all three interfaces - fastethernet 0/0 as 20.0.0.3 and 255.0.0.0 and serial 2/0 as 40.0.0.20 and 255.0.0.0 and serial 3/0 as 50.0.0.10 & 255.0.0.0. Router 2 does not have any default route and the static routing is done for the network 10 & 40 by the following command

ip route 10.0.0.0 255.0.0.0 40.0.0.10

ip route 30.0.0.0 255.0.0.0 50.0.0.10

vi) Router 3 is configured in both interfaces and the route is set by ip route 0.0.0.0 0.0.0.0 50.0.0.0

vii) Ping command is executed from 10.0.0.1 to 20.0.0.1 and from 10.0.0.1 to 30.0.0.2

Date __/__/____
Page _____

Observation :

-> One router cannot have two default route
-> The default route for first router is the middle router because any packet which have to be delivered will go to the middle router
-> The default route for third router is the middle sector for the same reason
-> The middle router does not have any default route because if one of the router is made default then there is a chance that the packet which are to be sent to the switch are sent to router.

Result :

ping 20.0.0.1

pinging 20.0.0.1 with 32 bytes of data
Request timed out
Reply from 20.0.0.1 bytes = 32 time = 1ms TTL = 126
Reply from 20.0.0.1 byte = 32 time = 2ms TTL = 12
Reply from 20.0.0.1 byte = 32 time = 2ms TTL =

Date __/__/__
Page _____

ping 30.0.0.2

pinging 30.0.0.2 with 32 bytes of data
Request timed out
Reply from 30.0.0.2 bytes=32 Time=4ms TTL=125
Reply from 30.0.0.2 byte=32 Time=4ms TTL=125
Reply from 30.0.0.2 bytes=32 Time=4ms TTL=12

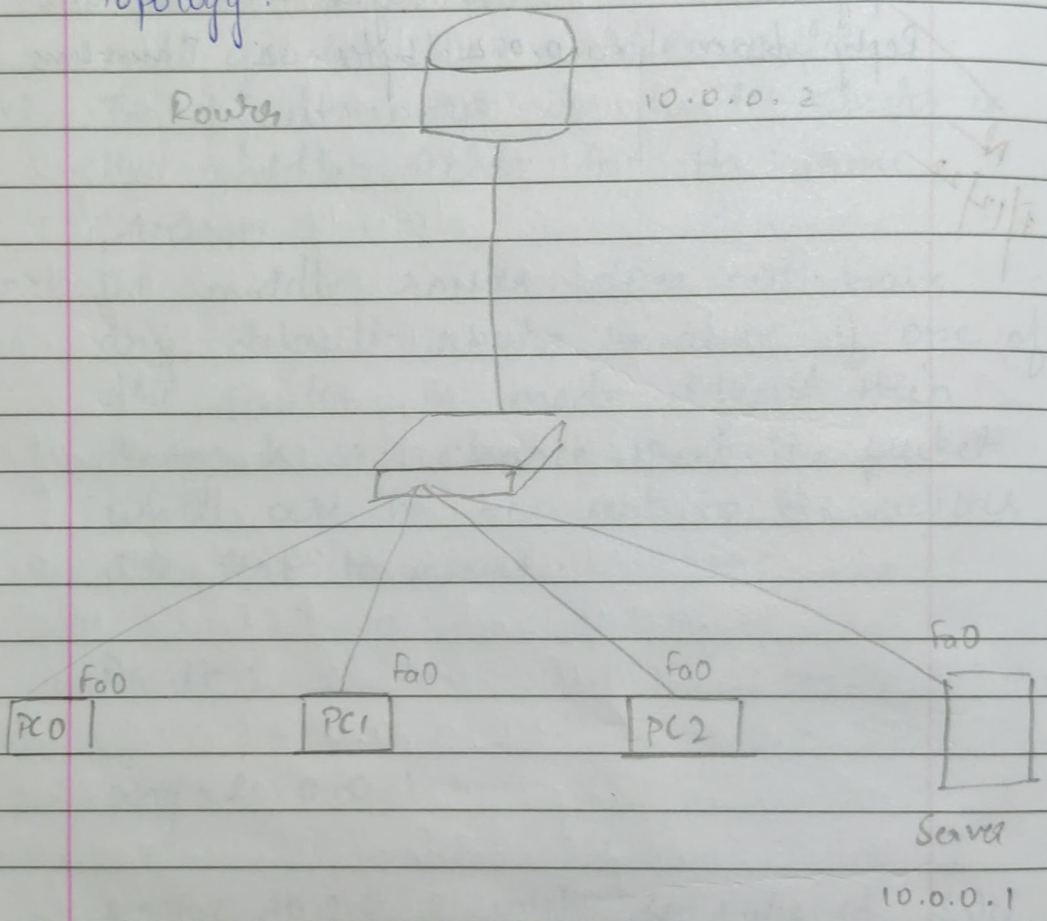
7/12/22

Date __/__/__
Page _____

Experiment - 4

Aim: Configuring DHCP within a LAN in a packet tracer

Topology:



Procedure

-> Place 3 pcs, 1 switch, 1 server and 1 router in the workspace
-> Connect 3 pcs, 1 server to switch using copper straight cable
-> Connect the ~~server~~ switch to router using fibre cable.

Date __/__/__
Page _____

-> Config the server by setting gateway as 10.0.0.2 and IP address as 10.0.0.1 and subnet mask as 255.0.0.0.

-> Config the router using CLI and set the IP address as 10.0.0.2 and subnet mask as 255.0.0.0

-> After configuring all devices go to services tab and turn on the DHCP service and set default gateway as 10.0.0.1, DNS server 10.0.0.2, start IP address as 10.0.0.3 and save the free pool.

-> Click on PC0 and select desktop tab and click IP configuration and select DHCP after selecting DHCP the request is sent for the IP address and IP address is obtained and do the same to other PCs

-> Ping other PCs and see the statistics

Observation:

Procedure

-> A pool of IP address exists from which IP addresses can be dynamically This is called Dynamic Host configuration protocol.

Date __/__/__
Page _____

Result :

ping 10.0.0.5

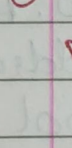
pinging 10.0.0.5 with 32 bytes of data

Replying from 10.0.0.5 byte = 32 time = 2ms

Replying from 10.0.0.5 byte = 32 time = 2ms TTL = 2

Replying from 10.0.0.5 byte = 32 time = 2ms TTL =

Replying from 10.0.0.5 byte = 32 time = 2ms TTL =

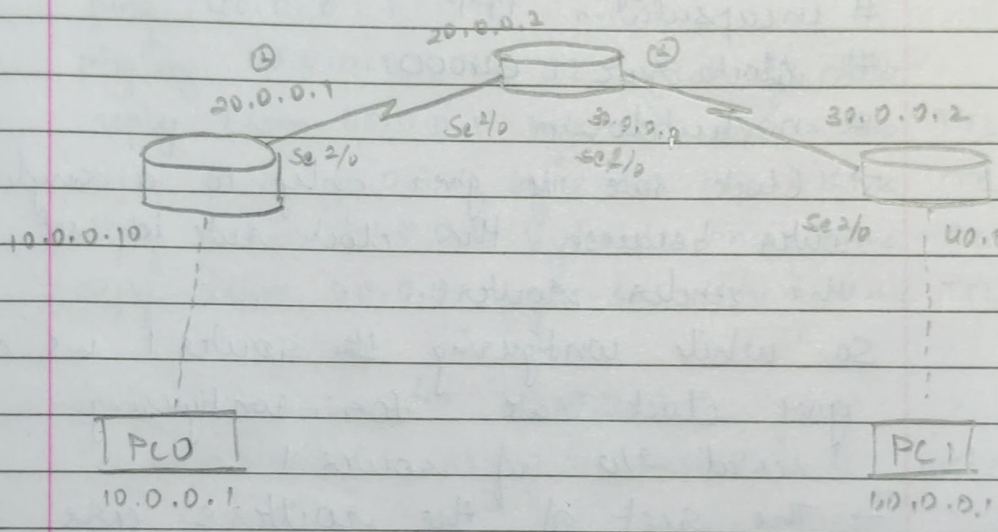


Date 8/12/22
Page

Experiment 5 :

Aim: Configuring RIP routing protocol in Router

Topology :



Procedure :

→ Place 2 PC's and 3 routers in the logical workspace
→ Connect 3 routers using serial DCE cable
→ Connect both PC's PC0 to router 0 and PC1 to router 2 using copper cross over cable
→ Config both the PC's by setting their IP addresses and gateways and subnet mask.
→ Serial connections is used for WAN and serial DCE is used to connect the routers and the sender side has a clock and the reciever is by default set

DCE — Data ~~terminal~~ carrier equipment.

Date __/__/__
Page _____

to DTE : Data Terminal equipment.

-> Configure the router 0 with serial DCE. ~~which~~ (Router 0)

interface serial 2/0

ip address 10.0.0.1 255.0.0.0

encapsulation PPP

clock rate 64000

no shutdown

-> Clock rate is given only to a single router because the clock rate is set by the sender router.

So while configuring ~~the~~ router 1 we dont give clock rate for configuring serial 2/0 of router 1.

-> The rest of the routers are configured using the commands above and changing the appropiate IP address

-> Now to configure the dynamic routing we use RIP sub-panel and type the commands for 0 configuring Router 0

router rip

network 10.0.0.0

network 20.0.0.0

exit

-> RIP - router information protocol which gives the router the ability to learn the path on its own.

-> Configure the ~~rest~~ remaining routers using appropiate network numbers.

Date __/__/__
Page _____

-> Ping the computer to check for errors.

Result:

ping 40.0.0.1
pinging 40.0.0.1 with 32 bytes of data
reply from 40.0.0.1 bytes=32 time=10ms TTL=125
reply from 40.0.0.1 bytes=32 time=15ms TTL=125
reply from 40.0.0.1 bytes=32 time=23ms TTL=125
reply from 40.0.0.1 bytes=32 time=10ms TTL=125

ping statistics for 40.0.0.1
packets: sent=4 recieved=4 lost=0

Observation:

RIP is router information protocal which is a dynamic routing protocol that uses hop count to find path between source and network. It is way better than static routing for bigger networks because the router learns the path on its own rather than teaching the path to the router.

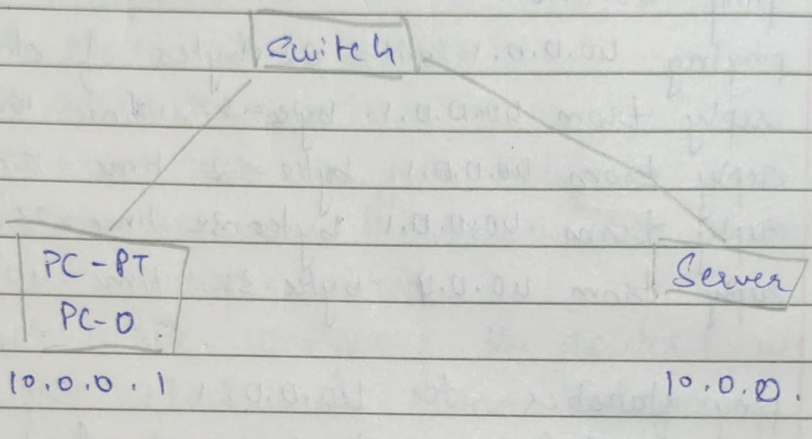
8/12/22

Date 15/02/22
Page

Experiment - 6

Aim: Demonstration of web server and DNS using Packet Tracer

Topology:



Procedure:

i) Place a PC, switch and server on the workspace and connect them

ii) set the IP address and subnet mask of the PC as 10.0.0.1 and 255.0.0.0 respectively

iii) Set the IP address and subnet mask of the server as 10.0.0.1 and 255.0.0.0 respectively.

iv) Open PC0 -> desktop -> web browser -> give IP address of server (10.0.0.2)

v) Open server -> services -> HTTP -> the HTTP windows open. Click on the edit option of index.html and change the contents -> save.

vi) Now the web browser page of PC is also modified

Domain Naming system

i) To activate DNS, open server → services → DNS → on → enter the name of resource record and IP address of the server and click on add.

ii) Now name and IP address is fixed

iii) Now create your own HTML file

iv) Now open PC0 → web browser → and specify the URL → bro

v) The content of the HTML file created is shown.

Result:

HTML file created :

```
<HTML>
<HEAD> HELLO </head>
<body>
Name : Koshal <br>
Branch : CSE
</body>
</html>
```

Date __/__/__
Page _____

Output:

HELLO
Namrata Koshal
Branch : CSE

Learning : DNS helps us map name with an IP address we are comfortable with naming conventions such as bmsce.ac.in whereas computer is comfortable with IP address. Hence DNS helps us in the mapping of the name and the IP.

29/12/22

Experiment 7 (29/12/22)

Aim: Write a program for error detecting code using CRC - CCITT (16 bits)

```
#include <iostream.h>
#include <string.h>
int crc(char *ip, char *op, char *poly, int mode)
{
  strcpy(op, ip);
  if (mode)
    for (i=1; i<strlen(poly); i++)
      strcat(op, "0");
  for (int i=0; i<strlen(ip); i++) {
    if (op[i]=='1') {
      for (int j=0; j<strlen(poly); j++) {
        if (op[i+j] == poly[j])
          op[i+j] = '0';
        else
          op[i+j] = '1';
      }
    }
  }
  for (int i=0; i<strlen(op); i++)
    if (op[i]=='1')
      return 0
  return 1;
}
int main()
{
  char ip[50], op[50], recv[50];
  char poly[] = "10001000000100001";
  cout << "Enter input in binary " << endl;
```

```
cin >> ip;
crc(ip, op, poly, 1);
cout << "Transmitted message is " << ip << op + strlen(ip)
     << endl;
cout << "Enter recieved message in binary" << endl;
cin >> recv;
if (crc(recv, op, poly, 0))
    cout << "No error" << endl;
else
    cout << "Error in data transmission" << endl
return 0
}
```

output:

1) Enter input message in binary

1011010101

Transmitted message : 10110101011111110 1101111010

Enter recieved message

10110101011111110 1101111010

No error in data

Wed 12/1/2023

Experiment 8 (5/01/23)

Aim: Write a program for distance vector algorithm to find suitable path for transmission.

```
#include <iostream>
using namespace std;
int main()
{
    cout<<" Enter bucket size and output rate "<<endl;
    int bucketSize, OutputRate, inputPacket, choice;
    int filled = 0;
    cin >> bucketSize;
    cin >> OutputSize;
    do
    {
        cout<<" Enter packet size "<< endl;
        cin >> inputPacket;
        if (inputPacket <= bucketSize)
        {
            if (filled + inputPacket -> bucketSize)
                cout<<" Packets are overflowing "<<
                                        endl;
            else
                filled = filled + inputPacket;
        }
        else
            cout<<" Packets overflowing \n";
        if (filled <= outputRate)
        {
            filled = 0;
        }
        else
            filled = filled + outputRate;
        cout<<" Amount of bucket filled " << filled <<endl
        cout<<" Do you want to enter more "<<endl;
```

```
    cin >> choice;
  } while (choice == 1);

}
```

output:

```
Enter bucket size and output rate
500
50
Enter packet size
200
Amount of bucket filled 150
Do you want to enter more
1
Enter packet size
350
Amount of bucket filled 350
Do you want to enter more
1
Enter packet size
200
Packets are overflowing
Amount of bucket filled 300
Do you want to enter more
0
```

Lucky
12/1/2023

Experiment 9 (12-01-23)

```
#include <stdio.h>
#include <stdlib.h>
int Bellman_Ford (int G[20][20], int V, int E,
    int edge [20][2])
{
    int i, u, v, k, distance [20], parent [20], s,
        flag = 1;
    for (i=0; i<V; i++)
        distance [i] = 1000, parent [i] = -1;
    printf ("Enter source");
    scanf ("%d", &s);
    distance [s-1] = 0;
    for (i=0; i<V-1; i++) {
        for (k=0; k<E; k++)
        {
            u = edge [k] [0], v = edge [k] [1];
            if (distance [u] + G[u][v] < distance[v])
                distance [v] = distance [u] + G[u][v];
                parent [v] = u;
        }
    }
    for (k=0; k<E; k++)
    {
        u = edge [k][0];
        v = edge [k][1];
        if (distance [u] + G[u][v] < distance [v])
            flag = 0;
    }
    if (flag)
        for (i=0; i<V; i++)
```

```
        print ("Vertex %d -> cost = %d parent = %d \n",
                (i+1), distance[i], parent[i]+1));
    return flag
}
int main()
{
    int V, edge[20][2], G[20][20], i, j, k=0;
    printf("Bellman Ford");
    printf("Enter no of vertices");
    scanf("%d", &V);
    printf("Enter graph in matrix form \n");
    for (i=0; i<V; i++)
        for (j=0; j<V; j++)
        {
            scanf("%d", &G[i][j]);
            if (G[i][j] != 0) {
                edge[k][0] = i;
                edge[k++][1] = j;
            }
        }
    if (Bellman_Ford(G, V, k, edge))
        printf("\n No negative weight cycle");
    else
        printf("\n Negative weight cycle exists");
    return 0;
}
```

output : Enter no of vertices

```
0   5   999   1   999
5   0   6   999   7
999   6   0   999   8
1   999   999   0   999
999   7   8   999   0
Enter source :   1
```

```
vertex 1 -> cost = 0  parent = -1
vertex 2 -> cost = 5  parent = 0
vertex 3 -> cost = 11  parent = 1
vertex 4 -> cost = 1  parent = 0
vertex 5 -> cost = 12  parent = 1

No negative weight cycle.
```

2>

```
#include <stdio.h>
#include <conio.h>
#define INFINITY 9999
#define MAX 10
void dijkstra(int G[MAX][MAX], int n, int
                start_node)
{
    int cost[MAX][MAX], distance[MAX],
        pred[MAX];
    int visited[MAX], count, mindistance,
        nextnode, i, j;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            if(G[i][j]==0)
                cost[i][j] = INFITY
            else
                cost[i][j] = G[i][j];

    for(i=0; i<n; j++){
        distance[i] = cost[startnode][i];
        pred[i] = startnode;
        visited[i] = 0;
    }
```

```
distance [start_node] = 0
visited [start_node] = 1
count = 1;
while ( count < n - 1 )
{
    mindistance = INFINTY;
    for ( i=0 ; i<n ; i++ )
        if ( distance [i] < mindistance && !visited[i] )
        {
            mindistance = distance [i];
            nextnode = i;
        }
        visited [nextnode] = 1;
        for ( i=0; i<n ; i++ )
            if ( !visited [i] ) {
                if ( mindistance + cost [nextnode][i]
                    < distance [i] )
                {
                    distance [i] = mindistance +
                                   cost [nextnode][i];
                    paird[i] = nextnode;
                }
            count++;
        }
}
for (i=0; i<n ; i++ )
    if (i != start_node) {
        printf("\n Distance of node %d = %d",
                i , distance [i] );
```

```
        printf ("\n Path = %d ", i);
        j = i;
        do {
            j = pred[j];
            printf("<- %d", j);
        } while (j != start_node);
    }
}

int main()
{
    int G[MAX][MAX], i, j, n, u;
    printf("Enter no of vertices");
    scanf("%d", &n);
    printf("\n Enter the adjacency matrix\n");
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            scanf("%d", &G[i][j]);
    printf("\n Enter the starting node");
    scanf("%d", &u);
    dijkstra(G, n, u);
    return 0;
}
```

Output-

Enter no of vertices

0	5	999	1	999
5	0	6	999	7
999	6	0	999	8
1	999	999	0	999
999	7	8	999	0

Enter starting node : 2

Distance of node 0 = 11

path = 0 ← 1 ← 2

Distance of node 1 = 6

path = 1 ← 2

Distance of node 3 = 12

path = 3 ← 0 ← 1 ← 2

Distance of node 4 = 8

path = 4 ← 2

Experiment 10 :

1) Using TCP/IP sockets write a client server program to make client sending the file name and the server to send back the contents of the requested file if present.

client TCP.py :

```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input("\n Enter File Name")
clientSocket.send(sentence.encode())
fileContents = clientSocket.recv(1024).decode()
print("\n From Server: \n")
print(fileContents)
clientSocket.close()
```

Server TCP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 1:
    print("The server is ready to recieve")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
```

```
file = open(sentence, "r")
l = file.read(1024)
connectionSocket.send(l.encode())
print('In Sent contents of ' + sentence)
file.close()
connectionSocket.close()
```

Output:

```
python server.py
The server is ready to recieve
Sent contents of file.txt
The server is ready to recieve
```

```
pytho client.py
Enter filename : file.txt
From Server
Hello, I am Koshal
```

2> Using UDP sockets write a client server program to make client sending the file name and the server to send back the content of the requested file if present

sol

ClientUDP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("\n Enter file name")
clientSocket.sendto(bytes(sentence, "UTF-8"),
                    serverName, serverPort))
fileContents, serverAddress = clientSocket.recvfrom
                              (2048)
print("\n Reply from Server : \n")
print(fileContents.decode("utf-8"))
#for i in fileContents :
    #print(str(i), end = '')
clientSocket.close()
clientSocket.close()
```

ServerUDP.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print("The server is ready to recieve")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(
                              2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence, "r")
```

```
I = file.read(2048)
serverSocket.sendto(bytes(I,"utf-8"),clientAd
print('In sent contents of ', end = '')
print(sentence)
# for i in sentence
    # print(str(i), end = ' ')
file.close()
```

Output

```
pytho serverUDP.py
The server ready to recieve
```

```
sent contents of file.txt.
```

```
python clientUDP.py
Enter file name : file.txt
Reply from server
Hello, I am Kushal
```