DHCPrequest
successful

Connect to wireless network:

In the desktop tab, under PC wireless setting, Under connect select "Home network" from the list of network & connect

Step(3): Configure the PC:

(1) Enable DHCP

(2) Verify IP address

open command prompt on the PC run IPconfig/all to confirm the PC received an IP address in the 192.168.0.x range

Physical	Config	Desktop	Programming	Attributes
IP Configuration				
Interface	FastEthernet0			
DHCP	Static		DHCP request successful	
IP address	192.168.0.101			
Subnet mask	255.255.255.0			
Default Gateway	192.168.0.1			
DNS server	208.67.220.220			

Step(4)- Configure the Internet cloud

① Install network modules (if needed)

→ click the internet cloud icon, go to the physical tab

→ If missing power off the device and install PT-Cloud-NM-1CX and PT-CLOUD-NM-1CFE modules.

→ power the device back on

Set up and to ports

- In the config tab, select cable under connection
- set from port as coaxial and to port as ethernet, then click add to establish the connection

step(5)- configure the cisco.com server

select DHCP from the services in the left pane

→ click on the turn the DHCP service On
→ Poolname: DHCP pool
→ Default Gateway: 208.67.220.220
→ DNS server: 208.67.220.220
→ starting IP address: 208.67.220
→ subnetmask: 255.255.255.0
→ Max.no.of.users.50

(a) click add to add the pool

(b) configure the cisco.com server as a DNS server to provide domain name to IPV4 address.

While still on the servicestab select DNS from the services listed in the left pane.

→ click on to turn the DNS service on

Name: cisco.com
Type: A Record
Address: 208.67.220.220

click add to add the DNS service setting

config cisco.com server

Global settings:

Go the config tab, select settings

config as follows:

Select: static

Gateway: 208.67.220.1

DNS server: 208.67.220.220

Fast Ethernet0 Interface settings:

select: static

Ip address: 208.67.220.220

Subnet mask: 255.255.255.0

Verify connectivity:-

Refresh IPV4 settings on PC:-

Open Desktop> Command prompt

Run ipconfig/release and ipconfig/renew to conform the IP is in the 192.168.0.* range.

From the PC's configure prompt issue ping cisco.com to verify connection

Result:-

Thus all the connections are given and executed successfully.

Student observation:

1) Write down the key features of configuring wireless routers & DHCP server

Configuring wireless Router:

- SSID configuration: Set the SSID for wireless network
- Security settings:- configure wireless security protocol (WPA2, WPA3)
- DHCP server configuration:- ensure DHCP is enabled to assign IP address.
- Internet connection setup:- configure WAN settings for internet connectivity.

» Configuring DHCP server:

- IP Address Range: Define the range of IP addresses
- Lease time: set the duration for which an IP address is valid before renewal.
- Static IP Assignments:- optionally configure static IP address for specific devices.
- DNS configuration:- Specify DNS server addresses for devices on the network.

(2) What is the significance of DHCP server in internetworking.

Significance:

- → Dynamic IP assignment
- → Efficient IP management
- → Centralised configuration
- → Network scalability

(2) Design & configure and intel network in your lab using switch routes & ethernet cables

→ Intel network design:

Switch: connects multiple devices within the local network.

Router: connects the local network to the Internet, other networks.

E.g. configuration:

Switch IP address:- 192.168.1.1

Router WAN IP address: Obtain from ISP

LAN IP address: 192.168.1.254

PC1:- IP address: 192.168.1.10

Subnet mask: 255.255.255.0

Default Gateway:- 192.168.1.254

PC 2:- IP address: 192.168.1.11

Subnet mask: 255.255.255.0

Default Gateway: 192.168.1.254

Wireless Device(Laptop)

Dynamic IP from DHCP: 192.168.1.x

Subnet mask: 255.255.255.0

Default Gateway: 192.168.1.254

Ex.no-11 Simulate static Routing configuration using cisco packet tracer

Date: 15/10/24

Aim: simulate static routing configuration using cisco packet tracer. In this static routing configuration using cisco packet tracer, static routes are manually added to a router's routing table for networks not directly connected

Steps:

1) Lab setup: create a network with routers for each network (main & backup)

2) main & Backup Routes: set the route with fewer routers as lower administrative distance (AD) as the main route.

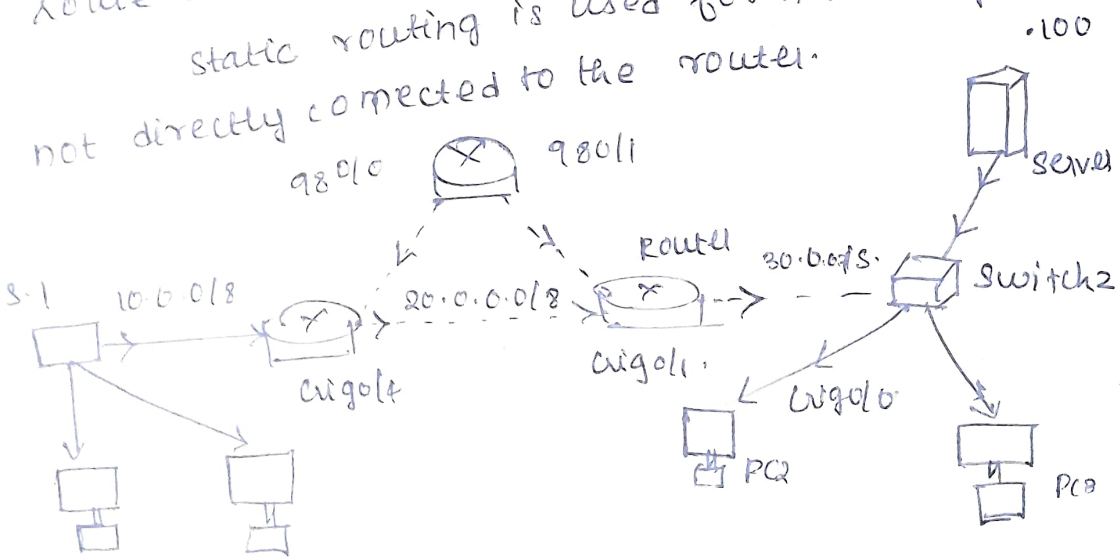
Example commands:

ip route 30.0.0.0 255.0.0.0 20.0.0.210

ip route 30.0.0.0 255.0.0.0 40.0.0.220

(3) Routing Table:- Use show ip route to verify that the main route is added, and the backup route activates if the main one fails.

static routing is used for networking not directly connected to the routers.



The following table lists the connected networks of each router:

Router	Available networks on local interface	Network available on other router interfaces
Router0	10.0.0.018 20.0.0.018 40.0.0.018	30.0.0.[illegible] 30.0.0.[illegible]
Router1	20.0.0.018 80.0.0.018 50.0.0.018	10.0.0.0.[illegible] 40.0.0.0.[illegible]
Router-2	40.0.0.018 50.0.0.018	10.0.0.018 20.0.0.018 30.0.0.0.18

Lets create static routes on each router for network that are not available on the router

Router 0 requirements:- Create two routes for network 30.0.0.18 and configure the first route (via-router1) as the main route & the second route (via-router2) as a backup route.

Create two routes for the host 30.0.0.10018 & configure the first route (via router-2) as the backup route.

Verify the routes add my main routes to the routing tables

Router-0- configuration → Access the CLI prompt of router 0 and run the following commands

Router >

Router # configure terminal

Enter configuration commands, one per line

Router (config) # ip route 30.0.0.0 255.0.0.0 20.0.0.0

Router (config) # ip route 30.0.0.100 255.255.255.255 40.0.0.210

Router (config) # ip route 50.0.0.0 255.0.0.0 40.0.0.210

Router (config) # ip route 50.0.0.0 255.0.0.0 20.0.0.20

Router (config) # exit

Router # show ip route static

- 30.0.0.0/8 is variable submitted, 2 subnets, 2 mask
- 30.0.0.0/8 [0/0] via 20.0.0.2
- 30.0.0.100/32 [110/10] via 40.0.0.2
- 50.0.0.0/8 [10/10] 10.0.0.2

Router #

Router-1 requirements:

Create 2 routes for network 10.0.0.0/8 & configure the first route (via-router-0) as the main route and the second route (via-route-1) as a backup route.

Create 2 routes for network 40.0.0.0/8 & configure the first route (via-route-0) as the main route and the second route (via-route-2) as a backup route.

Verify the router adds only main routes to the routing nodes.

Router-1 - configuration:

Router > enable

Router # configure terminal

Enter configuration commands

Router(config)# ip route 10.0.0.0 255.0.0.0
20.0.0.110

Router(config)# ip route 10.0.0.0
255.0.00.50.0.0.1

Router(config)# ip route 40.0.0.0 255.0.0.0 20.0.0.

Router(config)# ip route 40.0.0.0 255.0.0.50.00

Router(config)# exit

Router # show ip route static

10.0.0.0/8 [10/0] via 20.0.0.1

40.0.0/2 [10/2] via 20.0.0.1

Router #

Router 2 configuration:-

Router > enable

Router # configuration method

Enter configuration commands, one per line
end with CNTL+Z

Router(config)# ip route 10.0.0.0 255.0.0.0

Router(config)# ip route 30.0.0.0 255.0.0.50

Router(config)#exit

Router # show ip route static

S 10.0.0.0/8 [1/10] via 40.0.0.1

S 30.0.0.0/8 [1/15] via 50.0.0.2

Router#

verifying static routing on Router 0:

(1) Ping and trace Route:

→ use the command:

tracert 30.0.0.100

→ This path checks to 30.0.0.0 and confirms if the main route via route 1 used

(2) check routing table

(1) Execute:

show ip route

→ This displays the routing table ensuring only the main route for 30.0.0.0/8 (via router 1) is listed.

verifying Lost Route 30.0.0.0/8.

→ Ping test:- use tracert 30.0.0.100 to check route usage.

→ Routing table:- Run show ip route to confirm the static route is present.

Verifying Backup Route for 30.0.0.0/8:-

→ simulate failure: Disconnect the link between Router-0 & Router 1

→ Ping-test:- using ping 30.0.0.0 to verify the Backup route is used.

Deleting a static route:

→ Display Routes: use show ip route static to list all static routes.

→ Note route: Identify the route to delete.

→ Delete route: use no ip route <dest> <subnet mask> <next hop> to remove the route.

→ After deletion, the backup route becomes the main route automatically.

Result: Successfully simulated static routing in Cisco packet tracer, configured main & backup routes are routing table entries.

Ex.no (11.B)

Date: 15/10/24

Aim: Simulate RIP using cisco packet tracer initial IP configuration.

Device	Interface	IP configuration	connected with.
PC0	Fast Ethernet	10.0.0.2/8	Router 0's Fa0/1
Router 0	Fa0/1	10.0.0.1/8	PC0's Fast Ethernet
Router 0	s0/0/1	192.168.1.254/30	Router 1's s0/0/1
Router 0	s0/0/0	192.168.1.209/30	Router 1's s0/0/0
Router 1	s0/0/1	192.168.1.250/30	Router 0's s0/0/0
Router 2	s0/0/1	192.167.1253/30	Router 0's s0/0/1
Router 2	s0/0/1	192.168.1253/30	Router 2's Fa0/1
PC1	Fast Ethernet	20.0.0.2/30	

Assign IP address to PCs:-

Double click the PC, go to desktop > IP configuration and assign the IP addresses according to the table.

Assign IP address to Router Interface:-

Access router CLI by double clicking the router. Enter global configuration mode

Router> enable

Router# configuration terminal.

* Assign IP addresses to fast Ethernet & serial interfaces:

Router (config) # interface fast Ethernet 0/0
Router (config-if) # ip address [ip] [subnet mask]
Router (config-if) # no shutdown
Router (config-if) # exit

For serial interfaces with DCE & also for clock rate and Bandwidth

Router (config-if) # clock rate 64000
Router (config-if) # bandwidth 64

(3) Router specific configurations:-

For router 0, router 1, router 2, assign IP addresses to each interface as shown

Router 0 - 10.0.0.1, 192.168.1.249, 192.168.1.254

do the same for all other routers.

(4) Enable RIP routing protocol:-

Enter global configuration mode:-
Router (config) # router rip
* specific networks to advertise Router (config) # network [network address]
* Repeat for all the router

(5) verify connectivity:

- use the ping command from PC1 to test connectivity to other devices.
- This process configures the IPs, interfaces & enables the RIP protocol for routing b/w networks.

command prompt (packet tracer PC command line)

① PC > IPconfig

ip address: 10.0.0.2

subnet mask: 255.0.0.0

default gateway :- 20.0.0.1

② PC > Ping 10.0.0.2

Results sent = 4. Recived = 3. = lost = 1

Round-trip times = 3ms, max = 3ms

avg = 3ms

Routing Information Protocols:-

* manages route & choose alternative path if one fails.
* Route 1 ⟶ fewer hops, default
* Route 12 ⟶ more hops, if route 1 fails
* If route 1 fails (ex. cable disconection) RIP switches to route 2.
* use tracert before & after disconnection to verify RIP rerouting.

Result:- When Route 1 fails, RP automatically reroutes traffic through route 2. ensuring continous connectivity b/w PC0 & PC1.

EX. 12

29/10/24

Aim: Implement echo client server using TCP/UDP sockets

① TCP echo client server Algorithm

TCP server algorithm:

1) creates a TCP socket
2) Bind the socket to a local add & Port
3) Listen for incoming client connections
4) Accept a client connect
5) Loop
   → Receive data from the client.
   → If data is received, send it back to client else break the loop
6) Close the connection.

TCP Client Algorithm:

* create a TCP socket
* connect to the server using specified add & port
* send a message to server
* Receive the echoed message from the server
* Display the received message
* close the socket.

tcp_server.py:

```
import socket

def tcp_server():

server_socket = socket.socket(socket.AF_INET,

socket.sock_stream)

server_socket = bind(('localhost', 12345))

server_socket.listen()

print("TCP server is waiting for connection")

connection, client_address = server_socket.accept()

print("connected to {client_address}")

try:
    while true:
        data = connection.recv(1024)

    if data:
        print("received: {data.decode}")

else
    break

finally:
    connection.close()

tcp.server()
```

tcp_client.py:

```
import socket

def tcp_client():
    client_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    client_socket.connect(("localhost", 12345))

try:
    message = input("Enter a message to send
    client_socket.sendall(message.encode())
    data = client_socket.recv(1024)
    print("Received from server: " + data.decode())
finally
        client_socket.close()
tcp_client()
```

Output:-

```
python tcp_server.py
    TCP server is waiting for connection
connected to ('07.0.0.1', 56893)
Received > Hi Im abc
>python tcp_client.py
    Enter message to send: Hi in abd
        message to receive : Hi in abd
                (msg from server)
```

2) UDP echo client server Algorithm

UDP server Algorithm:

Create a UDP socket

Bind the socket to a local Address & Port

LOOP:

Receive the data from client

echo the received data back to the client

The server keeps running & does not need to close the socket explicitly

UDP client Algorithm:

Create a UDP socket

Send a message to the server using the specified address & port.

Receive the echoed message from server

Display the recieved message

close the socket.

Udp server . py

import socket

def udp_server

server = socket.socket

(socket.AF_INET, socket.

server_socket.bind(("local host"

print("UDP server is listen

while true

data client_Address = server_socket

(10:

print(f" Received from {client_addr

: {data.decode

server_socket.send_to(data, client

UDP_server()

Udp_client.py:

import socket

def udp_client():

server_socket = socket.socket(socket.

socket.SOCK_STREAM)

Server_socket.connect(("local host",

```
Print("UDP server is listening...")

while True:
    data, client_address = server_socket.recvfrom(1234):

    Print(f"Received from {client_address}:
          {data.decode()}")

    Server_socket.sendto(data, client_address)

Udp_server()
```

Udp_client.py:

```
import socket.

def udp_client():

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    Server_address = ('local host', 12345)

    message = input("Enter message to send")

    client_socket.sendto(message.encode(),
                         server_address)

    data, _ = client_socket.recvfrom(1024)

    Print(f"Received from server: {data.decode()}")

Udp_client()
```

Output

```
> Python udp_server.py
UDP server is listening...
Received from ('127.0.0.1', 58765)
        Hi Im Agila
```

```
> Python udp_client.py
Enter message to send: Hi Im Agila
Received from server: Hi Im Agila
```

Result:

The program for using Echo client server using UDP/TCP socket has been executed successfully.

Exp No:

Date: 22/10/24

Aim:

To implement the chat client server using TCP/UDP server.

Algorithm:

Chat server:

Start the server by creating a socket bind to a specific address and port, listen for incoming connection.

When a new client connects add it to a list of connected device start a new process to talk to the clients.

For each connected client keep checking for new message

If a client disconnects Remove that client from the list & stop talking to the client

Keep running the process till the

Chat - client:

Connect to the server by creating a socket and connect it to server address & port.

Start a process to listen to message from the server

Keep asking for the new message

Chat . client . Py :

```
import socket
import threading
def receive_messages(client, socket):
while true:
  try:
    message : client-socket .recv(1024).decode('utf-8')
    if message:
      Print(f"server : {message}")
except Exception as e:
Print(f" an error occured {e}")
break
```

```
def start_client():
    client_socket = socket.socket(socket
                    AF_INET, socket.SOCK_STREAM)
    host = "127.0.0.1"
    Port = 12345
    client_socket.connect((host, port))
    Print("Connected to the chat server.")
    threading.thread(target=receive_messages,
                     args=(client_socket), daemon=True)
                     Start()
← start_client()
```

chat_server.py

```
import socket
import threading
def handle_client(client_socket):
while true:
    try:
        message = client_socket.recv(1024)
                  decode('utf-8')
    if not message:
        break:
```