
# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## Computer Networks Laboratory Manual

Name :

Adm.No.:

**Session 2016-17**

# TABLE OF CONTENTS

| SL. NO. | PROGRAM NAME | DATE OF EXECUTION | SIGNATURE | REMARKS |
|---|---|---|---|---|
| | **SECTION A<br>LAN TRAINER** | | | |
| 1 | Implement the ALOHA protocol for packet communication between a number of nodes connected to a common bus. | | | |
| 2 | Implement the CSMA protocol for packet communication between a number of nodes connected to a common bus. | | | |
| 3 | Implement the CSMA/CD protocol for packet communication between a number of nodes connected to a common bus. | | | |
| 4 | Implement stop-and wait protocol to provide reliable data transfer between two nodes over an unreliable network. | | | |
| 5 | Implement sliding window go back-N protocol to provide reliable data transfer between two nodes over an unreliable network. | | | |
| 6 | Provide the configuration for the transmission of packets between two nodes connected to a common bus. | | | |
| 7 | Implement the protocol FTP for file transfer between two nodes connected to a common bus. | | | |
| | **SECTION B<br>SOCKET PROGRAMMING** | | | |
| 1 | Socket Programming introduction. | | | |
| 2 | Write a program in C for Day Time Client & Day Time Server. | | | |
| 3 | Write a program in C for Chat Client & Chat Server | | | |
| | **SECTION C<br>CISCO PACKET TRACER** | | | |
| 1 | Introduction | | | |

| | | | | |
|---|---|---|---|---|
| 2 | Simple Network configuration | | | |
| 3 | Building and configuring wired topology with two different networks. | | | |
| 4 | Configuration of Wireless Infrastructure using Access Point and Wireless Router. | | | |
| **SECTION D<br>NETWORK SIMULATOR-2** | | | | |
| 1 | Write TCL Script for Connecting two nodes and sending packets in wired network. | | | |
| 2 | Write TCL Script for given STAR topology using SFQ on queue at intermediate node & use different colors for packet originated from different nodes. | | | |
| 3 | Write TCL Script for given RING topology in wired network using For loop & making topology dynamic. | | | |
| 4 | Make an X-GRAPH for given data. | | | |
| 5 | Write TCL Script in wired network for the given topology using TCP connection and sending data through node. | | | |
| 6 | Write TCL Script in wired network for the given topology using UDP connection and sending data through node | | | |
| 7 | Write TCL Script for wireless network using DSDV protocol with the given data & send packets between them. | | | |
| 8 | Write TCL Script for Linked Cluster Algorithm for wireless network | | | |

## SECTION-A

# (Lan Trainer)

**1.Implement the ALOHA protocol for packet communication between a number of nodes connected to a common bus.**

**2.Implement the CSMA protocol for packet communication between a number of nodes connected to a common bus.**

**3. Implement the CSMA\CD protocol for packet communication between a number of nodes connected to a common bus.**

**4. Implement stop-and wait protocol to provide reliable data transfer between two nodes over an unreliable network.**

**5. Implement sliding window go back-N protocol to provide reliable data transfer between two nodes over an unreliable network.**

**6.Provide the configuration for the transmission of packets between two nodes connected to a common bus.**

**7.Implement the protocol FTP for file transfer between two nodes connected to a common bus.**

# SECTION-B
# (Socket Programming)

1. **Socket programming: Introduction**

**2. Program in C for Day Time Client & Day Time Server**

### 3. Code for chat client & chat server

```
/*Server Side*/

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

main()
{
        intsd,i,len,bi,nsd,port;
        char content[30];
        structsockaddr_inser,cli;

        if((sd=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP))==-1)
        {
                printf("\nSocket problem");
                return 0;
         }

        printf("\nSocket created\n");
        bzero((char*)&cli,sizeof(ser));
        printf("ENTER PORT NO:\n");
        scanf("%d",&port);
        printf("\nPort Address is %d\n:",port);
        ser.sin_family=AF_INET;
        ser.sin_port=htons(port);
        ser.sin_addr.s_addr=htonl(INADDR_ANY);
        bi=bind(sd,(structsockaddr *)&ser,sizeof(ser));

        if(bi==-1)
        {
              printf("\nBind error, Port busy, Plz change port in client and server");
              return 0;
         }

        i=sizeof(cli);
        listen(sd,5);
        nsd = accept(sd,((structsockaddr *)&cli),&i);

        if(nsd==-1)
           {
                printf("\nCheck the description parameter\n");
                return 0;
           }

        printf("\nConnection accepted!");
```

```
        if(fork())
          {
                printf("\nEnter the data to be send type exit for stop:\n");
                scanf("%s",content);

                while(strcmp(content,"exit")!=0)
                      {
                         send(nsd,content,30,0);
                         scanf("%s",content);
                      }

                send(nsd,"exit",5,0);
          }
          else
                i = recv(nsd,content,30,0);

      while(strcmp(content,"exit")!=0)
         {
                printf("\nClient: %s\n",content);
                i=recv(nsd,content,30,0);
         }

      printf("\nBye");
      send(nsd,"Offline",10,0);
      close(sd);
      close(nsd);
      return 0;
}

/*Client Side*/

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

int main()
{
        intsd,con,port,i,Res;
        char content[30];
        structsockaddr_in cli;

        if((sd=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP))==-1)
  {
                printf("\nSocket problem");
                return 0;
  }

          bzero((char*)&cli,sizeof(cli));
```

```
        cli.sin_family = AF_INET;
        printf("ENTER PORT NO:\n");
        scanf("%d",&port);
        cli.sin_port=htons(port);
        cli.sin_addr.s_addr=htonl(INADDR_ANY);

        con=connect(sd,(structsockaddr*)&cli,sizeof(cli));

        if(con==-1)
          {
                printf("\nConnection error");
                return 0;
           }

        if(fork())
          {
                printf("\nEnter the data to be send type exit for stop:\n");
                scanf("%s",content);

                while(strcmp(content,"exit")!=0)
                      {
                         send(sd,content,30,0);
                         scanf("%s",content);
                      }

          send(sd,"exit",5,0);
          }
          else
          {
                i=recv(sd,content,30,0);

                while(strcmp(content,"exit")!=0)
                      {
                         printf("\nServer: %s\n",content);
                         i=recv(sd,content,30,0);
                      }
                send(sd,"exit",5,0);
}

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

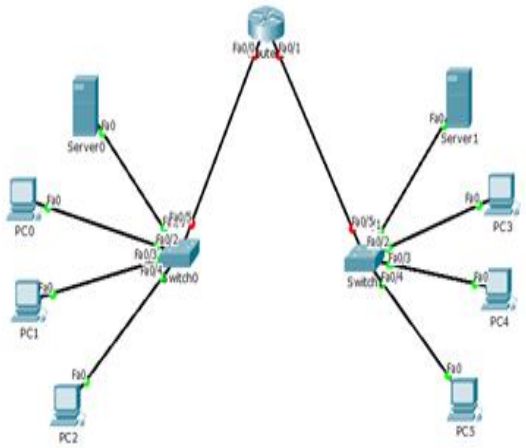
# SECTION-C

**(CISCO PACKET TRACER)**

**1.Simple Network configuration → Connect two PC and a Router.**


**Objective**→ PC0 should be able to PING Router0 and PC1 and vice-versa.

**2. Building and configuring wired topology with two different networks [ using DHCP ].**




**Configuration command**

**At Router**

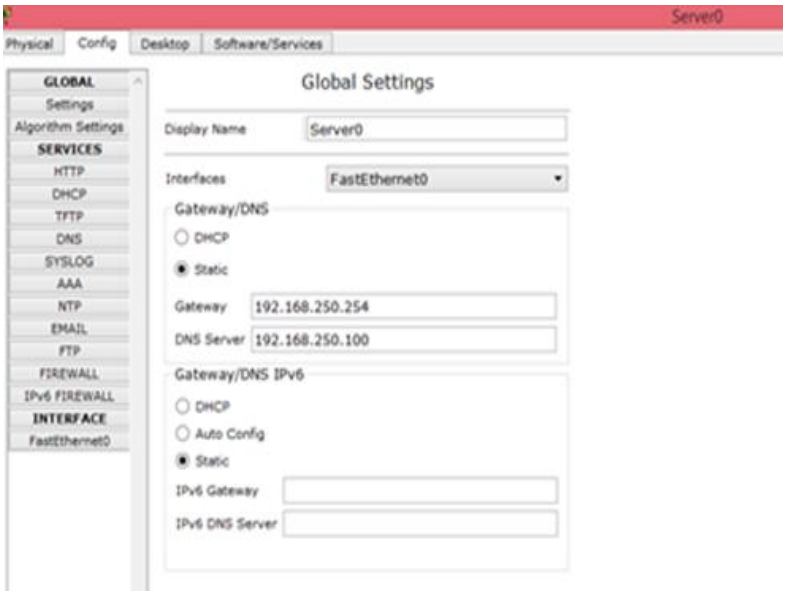
Click on router and go to CLI Tab.

Router>
Router>enable
Router# config terminal
Router(config)# int fa0/0
Router(config-if)#  ip address 192.168.250.254  255.255.255.0
Router(config-if)# no shut
Router(config-if)#exit
Router(config)# int fa0/1
Router(config-if)# ip address 192.168.251.254 255.255.255.0
Router(config-if)# no shut
Router(config-if)# exit
Router(config)#exit
Router#exit
Close the window

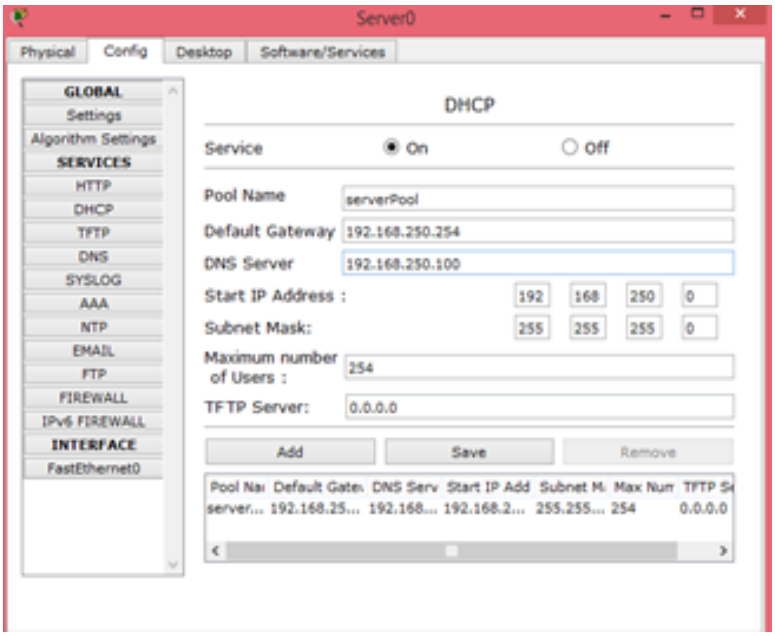
**At Server 0**

1) **Select  config Tab :**

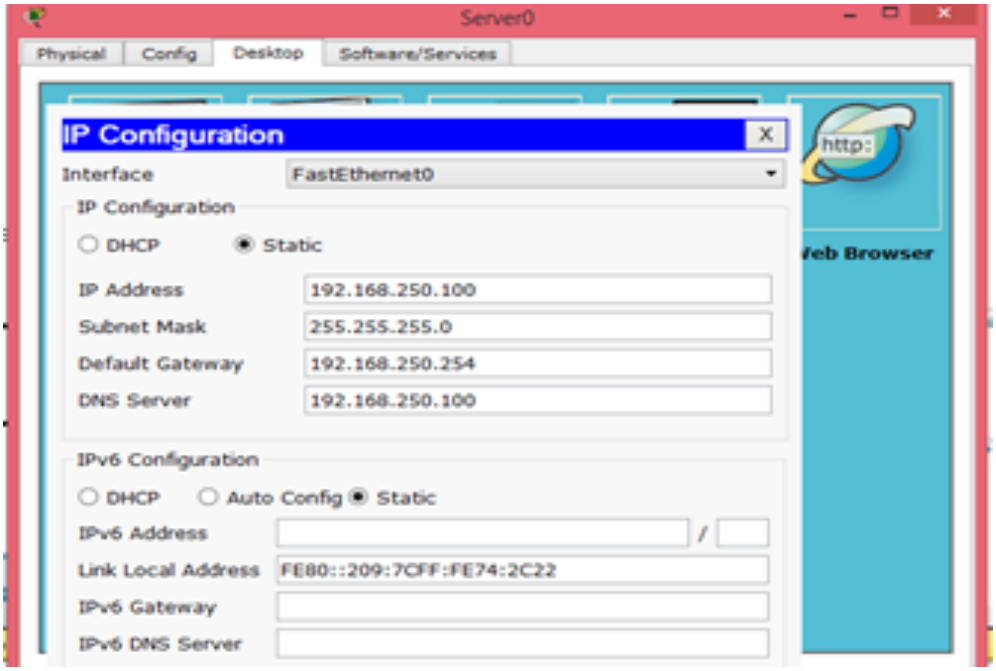
i) Assign gateway IP address as 192.168.250.254
ii) Assign DNS Server as 192.168.250.100




**2) In Config/Services Tab select DHCP and configure as shown :**

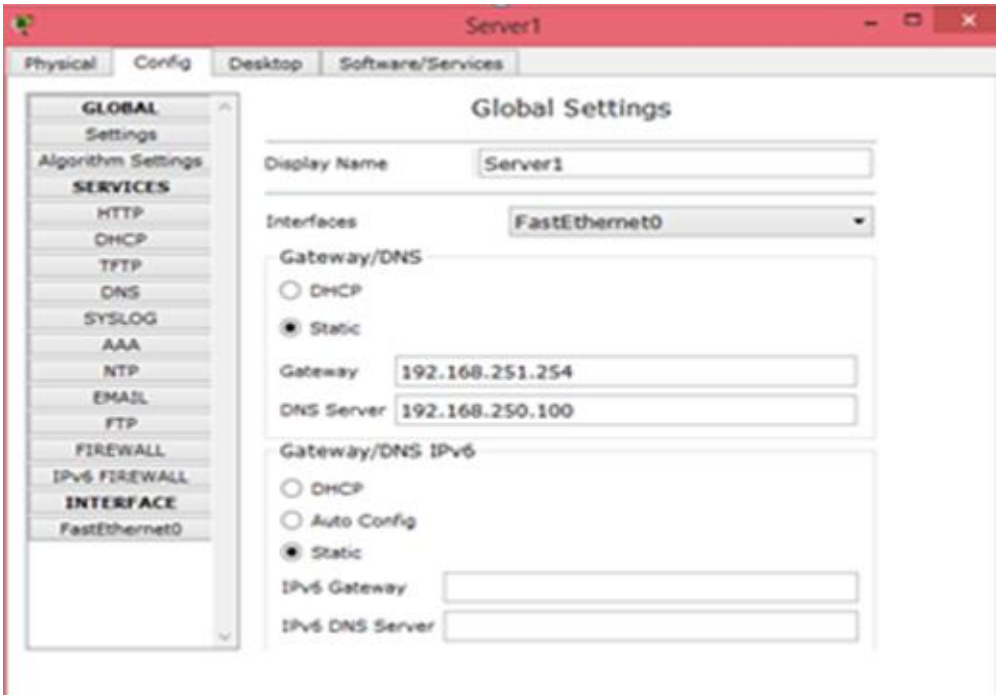


3) **At server 0 in Desktop Tab select IP configuration and assign static IP address to DHCP Server as shown below**

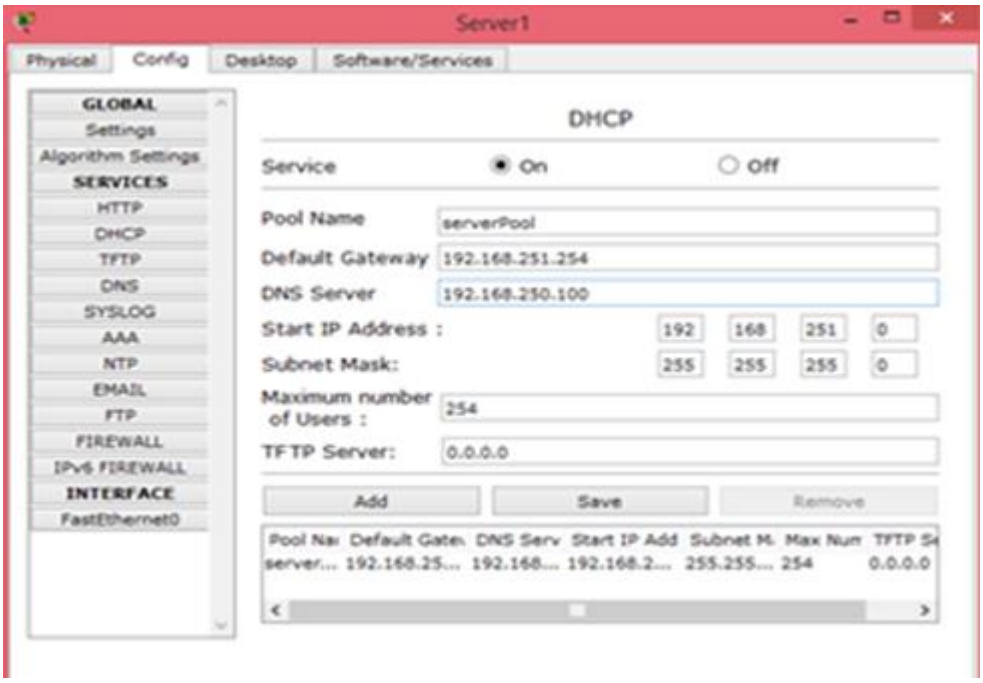



**At SERVER 1**

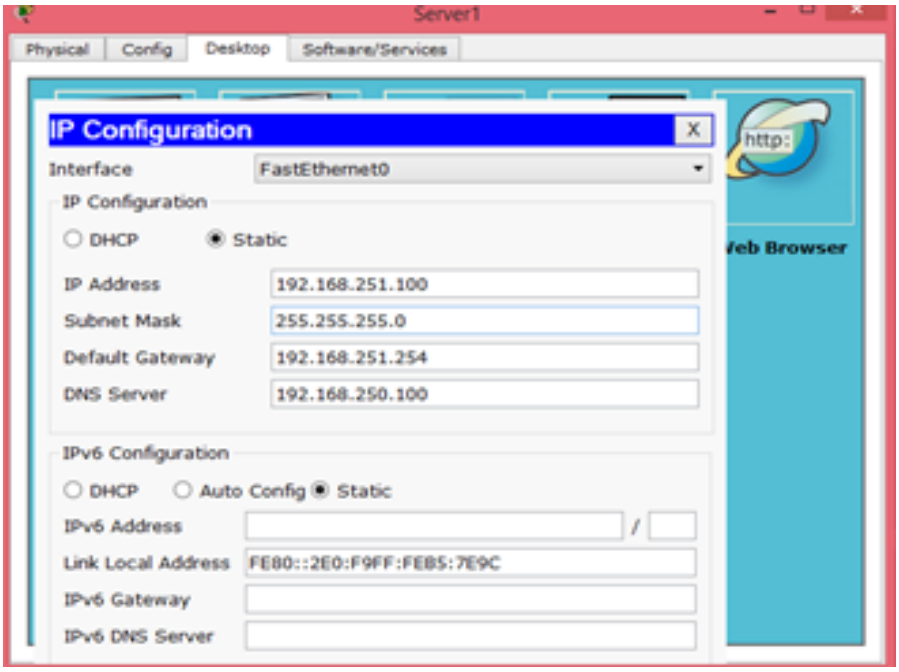
1) **Select config Tab and assign IP address as shown below-**



2) **In Config/Services Tab select DHCP and configure as shown :**



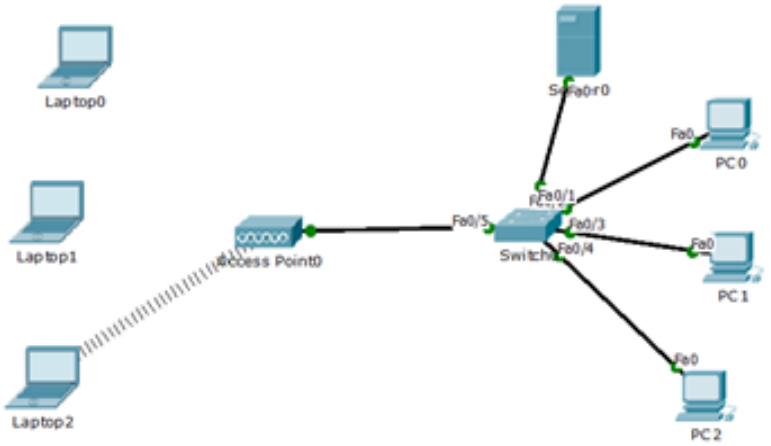

3) **At server 1 in Desktop Tab select IP configuration and assign static IP address to DHCP Server as shown below-**




To check node connectivity :

1) Ping PC3 from PC1
   At PC1 :: ping 192.168.251.254

**3.aConfiguration of Wireless Infrastructure [Using Access Point].**




## Configuring Command

**At Server0** – configure DHCP Server as -

Select as :Config/Services-> DHCP->start IP Address : 192.168.1.2

Subnet Mask : 255.255.255.0

Now check at each client whether DHCP request is successful or not.

After that **Configure Laptop** :- for wireless configuration

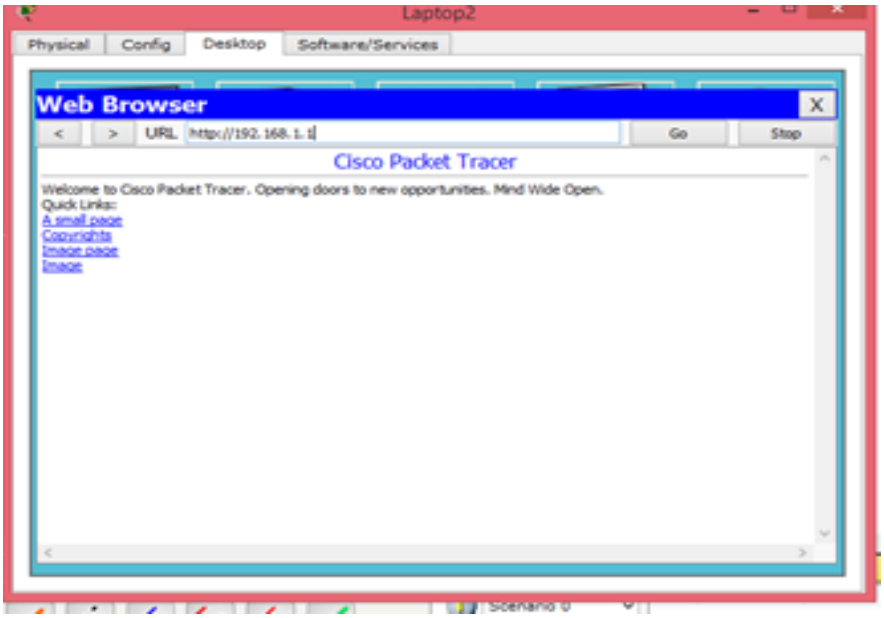
Switch off the laptop and then remove wired NIC from the laptop and then insert the wireless NIC (Linksys-WPC300N) into laptop and after that switch on the laptop.

**At Access Point** →Assign WEP Key (minimum of 10-digit) at interface.

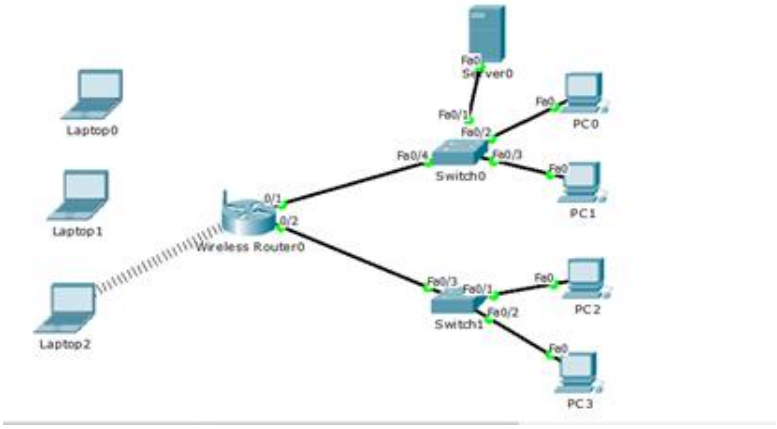
**At Laptop2**

1) **configure as shown → Connect laptop to access point.**


2) **In Desktop select Web browser and check connectivity with DHCP server.**



**3.bPerform exercise 3.a  using wireless Router.**



# SECTION-D
# (Network Simulator 2)

1.**Write TCL Script for connecting two nodes and sending packets in wired network**.

```
# Create a simulator object
set  ns  [new Simulator]

set  f  [open out.tr w]
$ns  trace-all  $f

set  nf  [open out.nam w]
$ns  namtrace-all  $nf

 set  n0 [$ns node]
 set  n1 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

set  udp0  [new Agent/UDP]
$ns  attach-agent  $n0  $udp0
set  cbr0  [new Application/Traffic/CBR]
$cbr0  set  packetSize_  500
$cbr0  set  interval_  .005
$cbr0  attach-agent  $udp0
set  null0  [new Agent/Null]
$ns  attach-agent  $n1  $null0
$ns  connect  $udp0  $null0

proc finish {} {
global ns nf f
$ns flush-trace
close $f
close $nf
exec  nam  out.nam  &
exit 0
}

$ns  at  0.5  "$cbr0  start"
$ns  at  3.0  "$cbr0  stop"
$ns  at  5.00  "finish"

$ns  run
```

**2. Write TCL Script for given STAR topology using SFQ on queue at intermediate node & use different colors for packet originated from different nodes.**

```
set ns [new Simulator]

$ns color 1 Red
$ns color 2 Blue
set  f  [open out.tr  w]
$ns  trace-all  $f
set  nf  [open out.nam  w]
$ns  namtrace-all  $nf

set  n0 [$ns node]
set  n1 [$ns node]
set  n2 [$ns node]
set  n3 [$ns node]

$ns  duplex-link  $n0  $n2  1Mb 10ms DropTail
$ns  duplex-link  $n1  $n2  1Mb 10ms DropTail
$ns  duplex-link  $n2  $n3  1Mb 10ms SFQ

set  udp0  [new Agent/UDP]
$udp0  set  class_  1
$ns  attach-agent  $n0  $udp0
set  cbr0  [new Application/Traffic/CBR]
$cbr0  set  packetSize_  500
$cbr0  set  interval_  .005
$cbr0  attach-agent  $udp0
set  udp2 [new Agent/UDP]
$ns  attach-agent  $n2  $udp2
set  cbr2  [new Application/Traffic/CBR]
$cbr2  set  packetSize_  500
$cbr2  set interval_  .005
$cbr2  attach-agent  $udp2
set  udp1  [new Agent/UDP]
$udp1  set  class_  2
$ns  attach-agent  $n1  $udp1
set  cbr1  [new Application/Traffic/CBR]
$cbr1  set  packetSize_  500
$cbr1  set  interval_  .005
$cbr1  attach-agent  $udp1
set  null0  [new Agent/Null]
$ns  attach-agent  $n3  $null0
```

```
$ns connect  $udp0  $null0
$ns connect  $udp1  $null0

proc finish {}  {
global  ns  nf  f
$ns flush-trace
close $f
close $nf
exec  nam  out.nam  &
exit 0
}

 $ns  at  0.5  "$cbr0 start"
 $ns  at  3.0  "$cbr0 stop"
 $ns  at  0.5  "$cbr1 start"
 $ns  at  3.0  "$cbr1 stop"
 $ns  at  5.00  "finish"

 $ns  run
```

**3 .Write TCL Script for given RING topology in wired network using For loop & making topology dynamic.**

```
#Create a simulator object
set  ns  [new Simulator]

#Tell the simulator to use dynamic routing
$ns  rtproto  DV

#Open the nam trace file
set   nf  [open out.nam w]
$ns  namtrace-all  $nf

#Define a 'finish' procedure
proc finish {} {
global  ns  nf
$ns  flush-trace
#Close the trace file
close  $nf
#Executenam on the trace file
exec  nam  out.nam  &
exit 0
}

#Create seven nodes
for { set i 0} {$i< 7} {incr  i} {
         set  n($i)  [$ns node]
}


#Create links between the nodes
for { set i 0 } {$i< 7} {incr  i} {
      $ns  duplex-link  $n($i)  $n([expr ($i+1)%7])  1Mb 10ms  DropTail
}

#Create a UDP agent and attach it to node n(0)
set  udp0  [new Agent/UDP]
$ns  attach-agent  $n(0)  $udp0

# Create a CBR traffic source and attach it to udp0
set  cbr0  [new Application/Traffic/CBR]
$cbr0  set  packetSize_  500
$cbr0  set interval_  0.005
$cbr0  attach-agent  $udp0

#Create a Null agent (a traffic sink) and attach it to node n(3)
set  null0  [new Agent/Null]
$ns  attach-agent  $n(3)  $null0

#Connect the traffic source with the traffic sink
```

```
$ns  connect  $udp0  $null0

#Schedule events for the CBR agent and the network dynamics

$ns  at 0.5 "$cbr0 start"
$ns  rtmodel-at  1.0  down  $n(1)  $n(2)
$ns  rtmodel-at  2.0  up  $n(1)  $n(2)
$ns  at  4.5  "$cbr0 stop"

#Call the finish procedure after 5 seconds of simulation time
$ns  at  5.0  "finish"

#Run the simulation
$ns  run
```

**4.Make an X-GRAPH for given data.**

**5.Write TCL Script in wired network for the given topology using TCP connection and sending data through node.**

```
set  ns [new Simulator]
set  f [open out.tr w]
$ns  trace-all  $f
set  nf  [open out.nam  w]
$ns  namtrace-all  $nf
set  n0 [$ns node]
set  n1 [$ns node]
set  n2 [$ns node]
set  n3 [$ns node]
$ns duplex-link  $n0 $n1 1Mb 10ms DropTail
$ns duplex-link  $n1 $n2 1Mb 10ms DropTail
$ns duplex-link  $n2 $n3 1Mb 10ms DropTail
$ns duplex-link  $n3 $n1 1Mb 10ms DropTail
$ns duplex-link  $n0 $n2 1Mb 10ms DropTail
set  tcp0  [new Agent/TCP]
$tcp0  set   maxcwnd_   16
$tcp0  set  fid_  1
$ns  attach-agent  $n0  $tcp0
set  cbr0 [new Application/Traffic/CBR]
$cbr0  set  packetSize_  500
$cbr0  set  interval_  .005
$cbr0  attach-agent  $tcp0

set  sink0  [new Agent/TCP]
$ns  attach-agent  $n3  $sink0

$ns  connect  $tcp0  $sink0
proc finish {} {
global ns nf f
 $ns flush-trace
close $f
close $nf
exec  nam  out.nam  &
exit 0
}

$ns  at  0.5  "$cbr0 start"
$ns  at  3.0  "$cbr0 stop"
$ns  at  5.0  "finish"

$ns  run
```

6. **Write TCL Script in wired network for the given topology using UDP connection and sending data through node**

```
set  ns [new Simulator]
set  f [open out.tr w]
$ns  trace-all $f
set  nf [open out.nam w]
$ns  namtrace-all $nf
set  n0 [$ns node]
set  n1 [$ns node]
set  n2 [$ns node]
set  n3 [$ns node]
$ns  duplex-link $n0 $n1 1Mb 10ms DropTail
$ns  duplex-link $n1 $n2 1Mb 10ms DropTail
$ns  duplex-link $n2 $n3 1Mb 10ms DropTail
$ns  duplex-link $n3 $n1 1Mb 10ms DropTail
$ns  duplex-link $n0 $n2 1Mb 10ms DropTail
set  udp0 [new Agent/UDP]
$ns  attach-agent  $n0  $udp0
set  cbr0 [new Application/Traffic/CBR]
$cbr0  set  packetSize_  500
$cbr0  set  interval_  0.005
$cbr0   attach-agent   $udp0
set  null0 [new Agent/Null]
$ns  attach-agent  $n3  $null0
$ns  connect  $udp0  $null0
proc  finish {} {
global  ns  nf  f
 $ns  flush-trace
close  $f
close  $nf
exec  nam  out.nam  &
exit  0
}
$ns  at 0.5 "$cbr0 start"
$ns  at 3.0 "$cbr0 stop"
$ns  at 5.0 "finish"
$ns  run
```

**7.Write TCL Script for wireless network using DSDV protocol with the given data & send packets between them.**

```
setval(chan)    Channel/WirelessChannel;
setval(prop)    Propagation/TwoRayGround;
setval(netif)  Phy/WirelessPhy;
setval(mac)   Mac/802_11;
setval(ifq)      Queue/DropTail/PriQueue;
setval(ll)       LL;
setval(ant)      Antenna/OmniAntenna;
setval(X)       1000;
setval(Y)       1000;
setval(ifqlen) 50;
setval(nn)      6;
setval(rp)      DSDV;

set ns [new Simulator]

set f [open out.tr w]
$ns trace-all $f

setnf [open out.nam w]
$ns namtrace-all-wireless $nf $val(X) $val(Y)

set topo [new Topography]
$topo load_flatgrid $val(X) $val(Y)

set god_ [create-god $val(nn)]

$ns node-config -adhocRouting $val(rp) \
-llType $val(ll) \
-macType $val(mac) \
-ifqType $val(ifq) \
-ifqLen $val(ifqlen) \
-antType $val(ant) \
-propType $val(prop) \
-phyType $val(netif) \
-channelType $val(chan) \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF \
-movementTrace OFF

for { set i 0 } { $i< $val(nn) } { incri } {
```

```
        set n($i) [$ns node]
        $ns initial_node_pos $n($i) 40
        $n($i) random-motion 0
}
$n(0) set x_ 100.0
$n(0) set y_ 100.0
$n(0) set z_ 000.0

$n(1) set x_ 200.0
$n(1) set y_ 200.0
$n(1) set z_ 000.0

$n(2) set x_ 300.0
$n(2) set y_ 200.0
$n(2) set z_ 000.0

$n(3) set x_ 400.0
$n(3) set y_ 300.0
$n(3) set z_ 000.0

$n(4) set x_ 500.0
$n(4) set y_ 300.0
$n(4) set z_ 000.0

$n(5) set x_ 600.0
$n(5) set y_ 400.0
$n(5) set z_ 000.0

$ns  at 0.0 "$n(0) setdest 100.0 100.0 000.0"
$ns  at 0.0 "$n(1) setdest 200.0 200.0 000.0"
$ns  at 0.0 "$n(2) setdest 300.0 200.0 000.0"
$ns  at 0.0 "$n(3) setdest 400.0 300.0 000.0"
$ns  at 0.0 "$n(4) setdest 500.0 300.0 000.0"
$ns  at 0.0 "$n(5) setdest 600.0 400.0 000.0"


set udp0 [new Agent/UDP]
$ns  attach-agent $n(0) $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0  set packetSize_ 500
$cbr0  set interval_ .005
$cbr0  attach-agent $udp0


set null0 [new Agent/Null]
```

```
$ns attach-agent $n(5) $null0

$ns connect $udp0 $null0


proc finish {} {
global ns nf f
            $ns flush-trace
          close $f
          close $nf
          exec  nam -r 5m out.nam&
          exit 0
                }

$ns at 0.5 "$cbr0 start"
$ns at 3.0 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```