ANNA UNIVERSITY

REGIONAL CAMPUS, COIMBATORE

LABORATORY RECORD

2022 – 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY-REGIONAL

CAMPUS COIMBATORE - 641 046

ANNA UNIVERSITY

REGIONAL CAMPUS, COIMBATORE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this is the bonafide record of Practical done in CS8581 – NETWORKS LABORATORY by _____ Register No._____ in Third Year - Fifth Semester during 2022 - 2023.

STAFF IN-CHARGE HEAD OF THE DEPARTMENT

University Register No: ……

Submitted for the University Practical Examination held on……...

INTERNAL EXAMINER EXTERNAL EXAMINER

INDEX

Expt No : 1	NETWORKING COMMANDS
Date :	

Aim:

To implement some Networking commands and to know their functions.

Commands:

1.ipconfig

It displays the IP address ,subnet mask and default gateway for all adapters.

EXAMPLE :

C:\Users\Administrator>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

Media State . . . . . : Media disconnected

Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 1:

Media State . . . . . : Media disconnected

Connection-specific DNS Suffix . :

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :

IPv6 Address. . . . . : 2409:4072:6e9e:f444:cc07:a529:2138:a747

Temporary IPv6 Address. . . . . : 2409:4072:6e9e:f444:9c5f:355b:ac9b:e52e

Link-local IPv6 Address . . . . . : fe80::cc07:a529:2138:a747%3

IPv4 Address. . . . . : 192.168.181.135

Subnet Mask . . . . . : 255.255.255.0

Default Gateway . . . . . : fe80::e03f:8dff:fe2d:add2%3

192.168.181.51

1.b) ipconfig /all

C:\Users\Administrator>ipconfig /all

Windows IP Configuration

Host Name . . . . . : HP-G5-Notebook
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No

Ethernet adapter Ethernet:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Description . . . . . : Realtek PCIe GbE Family Controller
Physical Address. . . . . : 48-BA-4E-91-CB-CA
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Local Area Connection* 1:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter
Physical Address. . . . . : 42-9F-38-19-DE-79
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Local Area Connection* 2:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Description . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #2
Physical Address. . . . . : 40-9F-38-19-DE-79
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :
Description . . . . . : Realtek RTL8188EE 802.11 bgn Wi-Fi Adapter
Physical Address. . . . . : 40-9F-38-19-DE-79
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
IPv6 Address. . . . . : 2409:4072:6e9e:f444:cc07:a529:2138:a747(Preferred)
Temporary IPv6 Address. . . . . : 2409:4072:6e9e:f444:9c5f:355b:ac9b:e52e(Preferred)
Link-local IPv6 Address . . . . . : fe80::cc07:a529:2138:a747%3(Preferred)
IPv4 Address. . . . . : 192.168.181.135(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Thursday, September 8, 2022 7:49:07 PM
Lease Expires . . . . . : Thursday, September 8, 2022 9:03:14 PM
Default Gateway . . . . . : fe80::e03f:8dff:fe2d:add2%
DHCP Server . . . . . : 192.168.181.51
DHCPv6 IAID . . . . . : 54566712
DHCPv6 Client DUID. . . . . : 00-01-00-01-2A-59-B1-57-00-0C-29-B3-8B-77
DNS Servers . . . . . : 192.168.181.51
NetBIOS over Tcpip. . . . . : Enabled

2.ping

Used to verify that a computer can communicate over the network with another computer or network device.

EXAMPLE :

C:\Users\Administrator>ping aurcc.ac.in
Pinging aurcc.ac.in [14.139.186.51] with 32 bytes of data:
Reply from 14.139.186.51: bytes=32 time=135ms TTL=50
Reply from 14.139.186.51: bytes=32 time=86ms TTL=50
Reply from 14.139.186.51: bytes=32 time=90ms TTL=50
Reply from 14.139.186.51: bytes=32 time=165ms TTL=50

Ping statistics for 14.139.186.51:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 86ms, Maximum = 165ms, Average = 119ms

3.tracert

Used to show several details about the path that a packet takes from the computer to whatever destination you specify.

EXAMPLE :

C:\Users\Administrator>tracert google.com

Tracing route to google.com [172.217.167.142]

over a maximum of 30 hops:

1	<1 ms	<1 ms	1 ms	10.10.48.10
2	2 ms	2 ms	2 ms	124.124.49.74
3	41 ms	41 ms	42 ms	220.224.182.250
4	42 ms	42 ms	42 ms	220.224.182.250
5	61 ms	62 ms	61 ms	115.255.234.90
6	86 ms	62 ms	62 ms	72.14.218.146
7	68 ms	63 ms	63 ms	108.170.248.171
8	61 ms	65 ms	58 ms	216.239.50.171
9	59 ms	59 ms	59 ms	108.170.253.97
10	60 ms	60 ms	59 ms	216.239.47.143
11	69 ms	69 ms	69 ms	maa03s26-in-f14.1e100.net [172.217.167.142]

Trace complete.

4.nslookup

Used for querying the name server for IP address of the given HOST optionally using a specified DNS server.

EXAMPLE :

C:\Users\Administrator>nslookup aurcc.ac.in

Server: UnKnown

Address: 192.168.181.51

Non-authoritative answer:

Name: aurcc.ac.in

Address: 14.139.186.51

5.netstat -a

Used for monitoring network connections both incoming and outgoing as well as viewing routing tables , statistics etc..

EXAMPLE :

```
C:\Users\Administrator>netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State

  TCP    0.0.0.0:135            HP-G5-Notebook:0       LISTENING
  TCP    0.0.0.0:445            HP-G5-Notebook:0       LISTENING
  TCP    0.0.0.0:5040           HP-G5-Notebook:0       LISTENING
  TCP    0.0.0.0:49664          HP-G5-Notebook:0       LISTENING
  TCP    0.0.0.0:49665          HP-G5-Notebook:0       LISTENING
  TCP    0.0.0.0:49666          HP-G5-Notebook:0       LISTENING
  TCP    0.0.0.0:49667          HP-G5-Notebook:0       LISTENING
  TCP    0.0.0.0:49668          HP-G5-Notebook:0       LISTENING
  TCP    0.0.0.0:49669          HP-G5-Notebook:0       LISTENING
  TCP    192.168.181.135:139    HP-G5-Notebook:0       LISTENING
  TCP    192.168.181.135:50237  20.198.119.84:https    ESTABLISHED
  TCP    192.168.181.135:50296  https-103-53-14-128:http  ESTABLISHED
  TCP    192.168.181.135:50299  a-0003:https           ESTABLISHED
  TCP    [::]:135               HP-G5-Notebook:0       LISTENING
  TCP    [::]:445               HP-G5-Notebook:0       LISTENING
  TCP    [::]:49664             HP-G5-Notebook:0       LISTENING
  TCP    [::]:49665             HP-G5-Notebook:0       LISTENING
  TCP    [::]:49666             HP-G5-Notebook:0       LISTENING
  TCP    [::]:49667             HP-G5-Notebook:0       LISTENING
  TCP    [::]:49668             HP-G5-Notebook:0       LISTENING
  TCP    [::]:49669             HP-G5-Notebook:0       LISTENING
  TCP    [2409:4072:6e9e:f444:9c5f:355b:ac9b:e52e]:50295  https-2402-6800-760-a000--1:http
ESTABLISHED
  UDP    0.0.0.0:5050           *:*
  UDP    0.0.0.0:5353           *:*
  UDP    0.0.0.0:5355           *:*
```

```
UDP    0.0.0.0:49425          *:*
UDP    127.0.0.1:1900         *:*
UDP    127.0.0.1:49664        127.0.0.1:49664
UDP    127.0.0.1:61174        *:*
UDP    192.168.181.135:137    *:*
UDP    192.168.181.135:138    *:*
UDP    192.168.181.135:1900   *:*
UDP    192.168.181.135:61173  *:*
UDP    [::]:5353              *:*
UDP    [::]:5355              *:*
UDP    [::]:49425             *:*
UDP    [::1]:1900             *:*
UDP    [::1]:61172            *:*
UDP    [fe80::cc07:a529:2138:a747%3]:1900  *:*
UDP    [fe80::cc07:a529:2138:a747%3]:61171  *:*
```

6.netstat -n

```
C:\Users\Administrator>netstat -n

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    192.168.181.135:50237  20.198.119.84:443      ESTABLISHED
  TCP    192.168.181.135:50296  103.53.14.128:80       ESTABLISHED
  TCP    192.168.181.135:50300  20.189.173.7:443       TIME_WAIT
  TCP    192.168.181.135:50301  52.109.8.45:443        TIME_WAIT
  TCP    192.168.181.135:50302  52.111.240.9:443       ESTABLISHED
  TCP    192.168.181.135:50303  52.109.4.36:443        ESTABLISHED
  TCP    [2409:4072:6e9e:f444:9c5f:355b:ac9b:e52e]:50295  [2402:6800:760:a000::1]:80
ESTABLISHED
```

7.netstat -r

```
C:\Users\Administrator>netstat -r
===========================================================================
Interface List
  4...48 ba 4e 91 cb ca ......Realtek PCIe GbE Family Controller
```

```
 11...42 9f 38 19 de 79 ......Microsoft Wi-Fi Direct Virtual Adapter
  8...40 9f 38 19 de 79 ......Microsoft Wi-Fi Direct Virtual Adapter #2
  3...40 9f 38 19 de 79 ......Realtek RTL8188EE 802.11 bgn Wi-Fi Adapter
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0    192.168.181.51  192.168.181.135     55
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
    192.168.181.0    255.255.255.0         On-link  192.168.181.135    311
  192.168.181.135  255.255.255.255         On-link  192.168.181.135    311
  192.168.181.255  255.255.255.255         On-link  192.168.181.135    311
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link  192.168.181.135    311
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link  192.168.181.135    311
===========================================================================
Persistent Routes:
  None

IPv6 Route Table
===========================================================================
Active Routes:
 If   Metric   Network Destination            Gateway
  3     71     ::/0                          fe80::e03f:8dff:fe2d:add2
  1    331     ::1/128                       On-link
  3     71     2409:4072:6e9e:f444::/64      On-link
      3   311   2409:4072:6e9e:f444:9c5f:355b:ac9b:e52e/128
                                            On-link
      3   311   2409:4072:6e9e:f444:cc07:a529:2138:a747/128
                                            On-link
      3   311   fe80::/64                     On-link
```

```
  3    311    fe80::cc07:a529:2138:a747/128
                                        On-link
  1    331    ff00::/8                  On-link
  3    311    ff00::/8                  On-link
===========================================================================
Persistent Routes:
  None
```

RESULT:

The commands are executed successfully.

Expt No : 2	HTTP CLIENT
Date :	

Aim:

To write a program to get the contents of a web page using http protocol.

Algorithm:

Step 1 : Import the socket package.
Step 2 : Create a socket object.
Step 3 : Construct a get request URL to a particular wepage.
Step 4 : Connect to the webpage.
Step 5 : Get the contents of the webpage using recv()
Step 6 : Display results.
Step 7 : Stop.

Program:

```
import socket
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("aurcc.ac.in",80))
s.sendall(b"GET / HTTP/1.1\r\nHost:www.aurcc.ac.in \r\nAccept:text/html\r\n\r\n")
print(str(s.recv(4096),'utf-8'))
```

Output:

```
HTTP/1.1 200 OK
Date: Sat, 27 Aug 2022 12:42:55 GMT
Server: Apache/2.4.41 (Win64) OpenSSL/1.1.1c PHP/7.2.27
X-Powered-By: PHP/7.2.27
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

2b65

<!DOCTYPE html>
<html>
<head>
<meta name="google-site-verification" content="UstoC7Vx3Cifyhh6RjEdxdKUAoNxIGsp-
54nZUz4kdw"/>
<title>Anna University,Coimbatore</title>

  <link rel="short icon" href="images/tittle.ico">
  <meta charset="utf-8">
  <meta name="description" content="Anna University Regional Campus, Coimbatore">
  <meta name="keywords" content="AURO,AURC,Anna University,Coimbatore,Educational
```

```
Institution,about,the institution,regional office">
<link href="css/layout/styles/layout.css" rel="stylesheet" type="text/css" media="all">

<link rel="stylesheet" type="text/css" href="css/layout/styles/default.css" />
      <link rel="stylesheet" type="text/css" href="css/layout/styles/component.css" />
      <script src="css/layout/styles/modernizr.custom.js"></script>

<style type="text/css">
<!--
.bild:hover {
border-radius:50%;
box-shadow: 0 10px 6px -6px grey;
}
.bild {
-webkit-transition: all 0.7s ease;
transition: all 0.7s ease;
}
-->

a.tooltip {outline:none; }
a.tooltip strong {line-height:30px;}
a.tooltip:hover {text-decoration:none;}
a.tooltip span {
    z-index:10;display:none; padding:14px 10px;
    margin-top:10px; margin-right:-160px;
    width:250px; line-height:16px;
}
a.tooltip:hover span{
    display:inline; position:absolute;
    border:2px solid #FFF;  color:#EEE;
    background:#333 url(cssttp/css-tooltip-gradient-bg.png) repeat-x 0 0;
}
.callout {z-index:20;position:absolute;border:0;top:-14px;left:120px;}

/*CSS3 extras*/
a.tooltip span
{
    border-radius:2px;
    box-shadow: 0px 0px 8px 4px #666;
    /*opacity: 0.8;*/
}
.boxed {
  border: 1px solid green ;
}
.div1 {
   width: 270px;
   height: 72px;
```

```
border: 2px solid orange;

}

.div2 {
    width: 270px;
     height: 25px;
   margin: 10;


}
.div3 {
    width: 100px;
    height: 80px;
   margin: 10;

}

  .div4 {
    width: 220px;
    height: 55px;
   margin: 10;

}
.style8 {color: #000000}
.box {
    padding: 20px;
    background-color:#FFFFFF;
    color: #666;
    overflow: hidden;
    box-shadow: 1px 1px 4px 1px #00b3b3;
}
</style>
<script>
    $(document).ready(function(){
        if (Modernizr.touch) {
            // show the close overlay button
            $(".close-overlay").removeClass("hidden");
            // handle the adding of hover class when clicked
            $(".img").click(function(e){
                if (!$(this).hasClass("hover")) {
                    $(this).addClass("hover");
                }
            });

            // handle the closing of the overlay
            $(".close-overlay").click(function(e){
```

```
            e.preventDefault();
            e.stopPropagation();
            if ($(this).closest(".img").hasClass("hover")) {
                $(this).closest(".img").removeClass("hover");
            }
        });
    } else {
        // handle the mouseenter functionality
        $(".img").mouseenter(function(){
            $(this).addClass("hover");
        })
        // handle the mouseleave functionality
        .mouseleave(function(){

            $(this).removeClass("hover");
        });
    }
  });
</script>
</head>

<body id="top">

  <!DOCTYPE html>
<html>
<head><title>Anna University,Coimbatore</title>
<link rel="stylesheet" href="css/bootstrap.css" media="screen">
<!--<link rel="stylesheet" href="css/bootswatch.min.css">-->
<link rel="stylesheet" href="css/datepicker.css">
<link rel="stylesheet" href="css/simple-sidebar.css">
<link href="css/layout/styles/layout.css" rel="stylesheet" type="text/css" media="all">
<link href="css/layout/styles/framework.css" rel="stylesheet" type="text/css" media="all">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4. >
```

RESULT :

The given program was successfully executed and its output verified.

Expt No : 3 a	ECHO CLIENT AND ECHO SERVER
Date :	

Aim:

To implement TCP Echo Client Server using python programming.

Algorithm:

Server:

Step 1: Create a server socket.
Step 2: Wait for client to be connected.
Step 3: Read text from the client.
Step 4: Echo the text back to the client.
Step 5: Close the server socket.
Step 6: Stop.

Client:

Step 1: Create a socket and establish connection with the server.
Step 2: Get input from user.
Step 3: Send text to the server.
Step 4: Display the text echoed by the server.
Step 5: Close the client socket.
Step 6: Stop.

Source Code:

Server:

```
import socket
server=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
server_address=('localhost',10000)
server.bind(server_address)
server.listen(1)
connection, client_address=server.accept()
print("Connection established with: ",client_address)
data=connection.recv(1000)
print("Received:",data)
connection.sendall(data)
connection.close()
server.close()
```

Client:

```
import socket
client=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
server_address=('localhost',10000)
client.connect(server_address)
print("Enter the message")
message=input()
print("Sending",message)
client.sendall(message.encode())
print("ORIGINAL: ",message)
data=client.recv(1000).decode()
print("ECHO: ",data)
client.close()
```

Output:

Server:

```
Connection established with:  ('127.0.0.1', 59373)
Received: b' Hello world'
```

Client:

```
Enter the message
 Hello world
Sending  Hello world
ORIGINAL:   Hello world
ECHO:   Hello world
```

Result:

Thus TCP Echo Client Server using python programming is implemented.

Expt No : 3 b	IMPLEMENTING CHAT APPLICATION USING TCP SOCKETS
Date :	

Aim:

To perform the full duplex chat by sending and receiving the message from the client to server
and vice versa using TCP sockets.

Algorithm:

Server :

Step 1: Create a server socket.
Step 2: Wait for client to be connected.
Step 3: Read Client's message and display it.
Step 4: Get a message from user and send it to client.
Step 5: Repeat Steps 3-4 until the client sends "end".
Step 6: Close the server and client socket.
Step 7: Stop.

Client :

Step 1: Create a client socket and establish connection with the server.
Step 2: Get a message from user and send it to server.
Step 3: Read server's response and display it.
Step 4: Repeat Steps 2-3 until chat is terminated with "end" message.
Step 5: Close the client socket.
Step 6: Stop.

Source Code:

Server:

```
import socket
server=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
server_address=('localhost',10000)
server.bind(server_address)
server.listen(1)
print("waiting for connection")
connection, client_address=server.accept()
message=" "
```

```
while message!="end":
    data=connection.recv(1000).decode()
    if data:
        print(data)
        message=input()
        connection.sendall(message.encode())
    else:
        break
connection.close()
server.close()
```

Client:

```
import socket
client=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
server_address=('localhost',10000)
client.connect(server_address)
message=input()
client.sendall(message.encode())
while message!="end":
    data=client.recv(1000).decode()
    if data:
        print(data)
        message=input()
        client.sendall(message.encode())
    else:
        break
client.close()
```

Output:

Server:

Hi
Hello. How are you?
I am fine. How about you?
I am good. Ok. Take care
Bye
end
end

Client:

Hi
Hello. How are you?
I am fine. How about you?
I am good. Ok. Take care
Bye
end
end

Result:

The full duplex chat by sending and receiving the message from the client to server and vice versa using TCP sockets is performed.

Expt No : 3 c	IMPLEMENTING FILE TRANSFER USING TCP SOCKETS
Date :	

Aim:

To implement File Transfer Application using TCP socket.

Algorithm:

Server:

Step 1: Create a server socket.
Step 2: Wait for client to be connected.
Step 3: Specify the file name and send it to client.
Step 4: Find the length of the file and read the contents of the file into a byte array.
Step 5: Transfer the file in the socket to the client.
Step 6: Close the file.
Step 7: Close the server and client socket.
Step 8: Stop.

Client:

Step 1: Create a client socket and establish connection with the server.
Step 2: Create a new file name in preferred directory.
Step 3: Read the file contents from server socket and write into the new file.
Step 4: Close the file.
Step 5: Close the client socket.
Step 6: Stop.

Source Code:

Server:

```
import socket
import sys
server=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
server_address=('localhost',10000)
server.bind(server_address)
server.listen(1)
connection, client_address=server.accept()
file_name="Sample.txt"
connection.sendall(file_name.encode())
file=open("Sample.txt","rb")
data=file.read()
connection.sendall(data)
file.close()
connection.close()
server.close()
```

Client:

```
import socket
import sys
client=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
server_address=('localhost',10000)
client.connect(server_address)
file_name=client.recv(1000).decode()
file_name="D:\Anna University\Sem 5\\"+file_name
file=open(file_name,"wb")
data=client.recv(4097)
file.write(data)
file.close()
client.close()
```

Output:

File read by server:

Hello
This file is a sample of expt. 3c
Thank you

File written by client:

Hello
This file is a sample of expt. 3c
Thank you

Result:

File Transfer Application using TCP socket is implemented.

Expt No : 4 a	SIMULATION OF DNS USING UDP PROTOCOLS
Date :	

Aim:

To write python code to simulate DNS using UDP sockets.

Algorithm :

Step 1. Start the process.
Step 2. Establish UDP socket connection using socket package in both client and server side.
Step 3. In the client, give input as the domain name for which the IP is required.
Step 4. Send the domain name to the server.
Step 5. Based on the dictionary makeup in the server side with DNS table, the corresponding IP is sent back to the client. me system.
Step 6. Receive the IP from the server and print the address.
Step 7. End the process.

Source Code:

Client :

```
import socket
hostname = socket.gethostname()
ipaddr = "127.0.0.1"
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
addr = (ipaddr, 1234)
c = "Y"
while c.upper() == "Y":
        req_domain = input("Enter domain name for which the ip is needed: ")
        send = s.sendto(req_domain.encode(), addr)
        data, address = s.recvfrom(1024)
        reply_ip = data.decode().strip()
        print(f"The ip for the domain name {req_domain}: {reply_ip}")
        c = (input("Continue? (y/n)"))
s.close()
```

Server:

```
import socket
dns_table = {"www.google.com": "192.168.0.1", "www.wikipedia.org": "192.168.0.2",
"www.python.org": "192.168.0.3", "www.aurcc.ac.in": "192.168.0.4", "www.amazon.in":
"192.168.0.5"," www.tamilrockers.ta": "192.168.1.3"}
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
print("Starting Server...")
s.bind(("127.0.0.1", 1234))
while True:
      data, address = s.recvfrom(1024)
```

```
        print(f"{address} wants to fetch
        data!") data = data.decode()
        ip = dns_table.get(data, "Not Found!").encode()
        send = s.sendto(ip, address)
    s.close()
```

OUTPUT:

Client:

Enter domain name for which the ip is needed: www.google.com
The ip for the domain name www.google.com: 192.168.0.1
Continue? (y/n)y
Enter domain name for which the ip is needed: www.aurcc.ac.in The
ip for the domain name www.aurcc.ac.in: 192.168.0.4 Continue?
(y/n)y
Enter domain name for which the ip is needed: www.wikipediaa.com
The ip for the domain name www.wikipediaa.com: Not Found! Continue?
(y/n)n

Server:

Starting Server...
('127.0.0.1', 53862) wants to fetch data!
('127.0.0.1', 53862) wants to fetch data!
('127.0.0.1', 53862) wants to fetch data!

RESULT:

Thus, the DNS is simulated using UDP sockets in python.

Expt No : 4 b	SIMULATION OF DNS USING UDP SOCKETS IN CISCO PACKET TRACER
Date :	

Aim:

To simulate DNS using cisco packet tracer.

Algorithm:

Step 1: Configure the IP address and the DNS server for the PC and the Web Server.

Step 2: In the webserver switch on http under the services section.

Step 3: In the DNS server, switch on DNS under the services section and add the domain name and the IP address of the webserver.

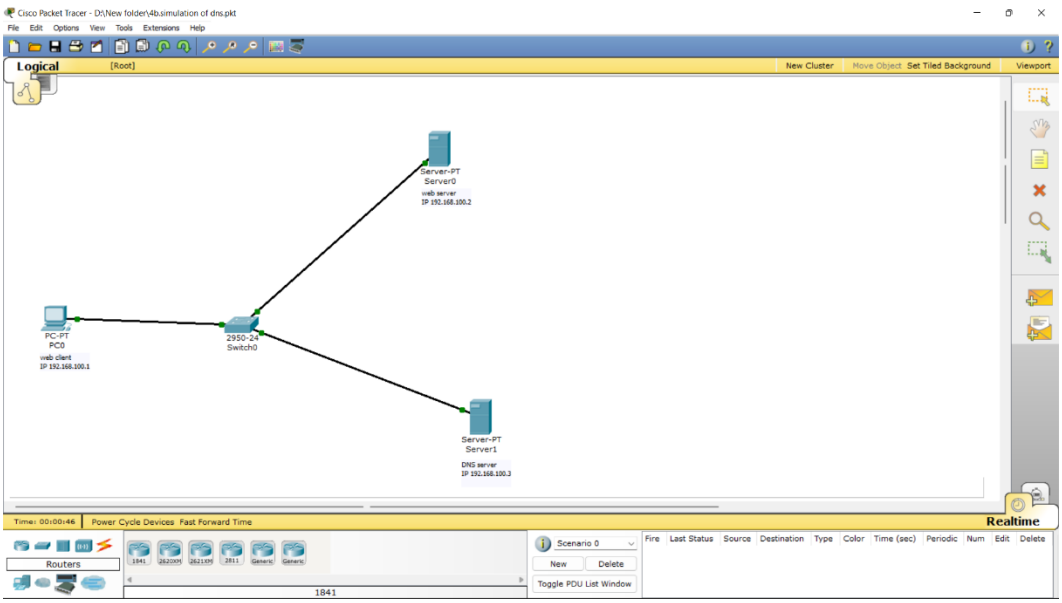
Step 4: Using the web browser in the PC, go to the domain name corresponding to the webserver

Step 5: Or in the command prompt, use the nslookup command to view the address corresponding to the domain name.

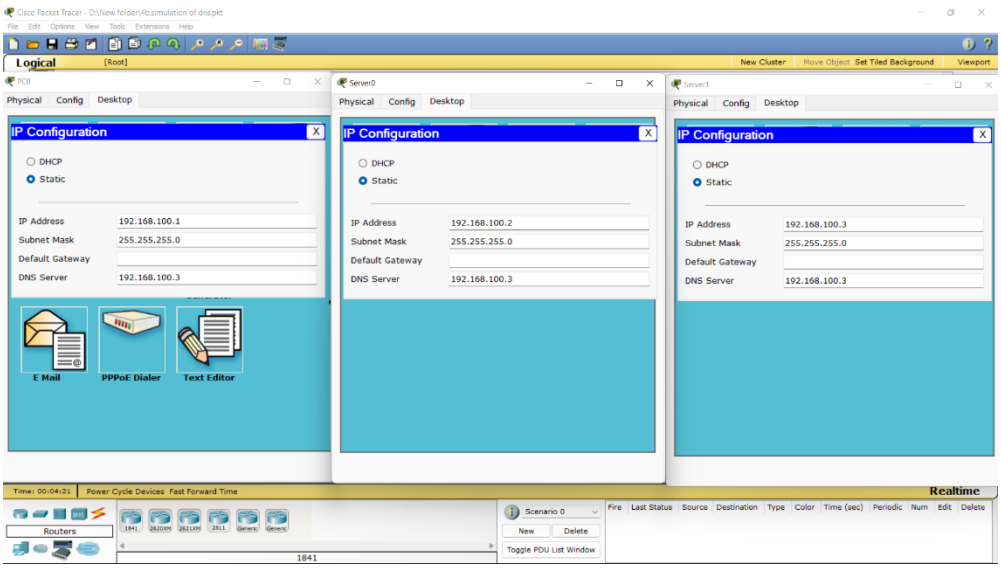
Step 6: End the process.

Output:

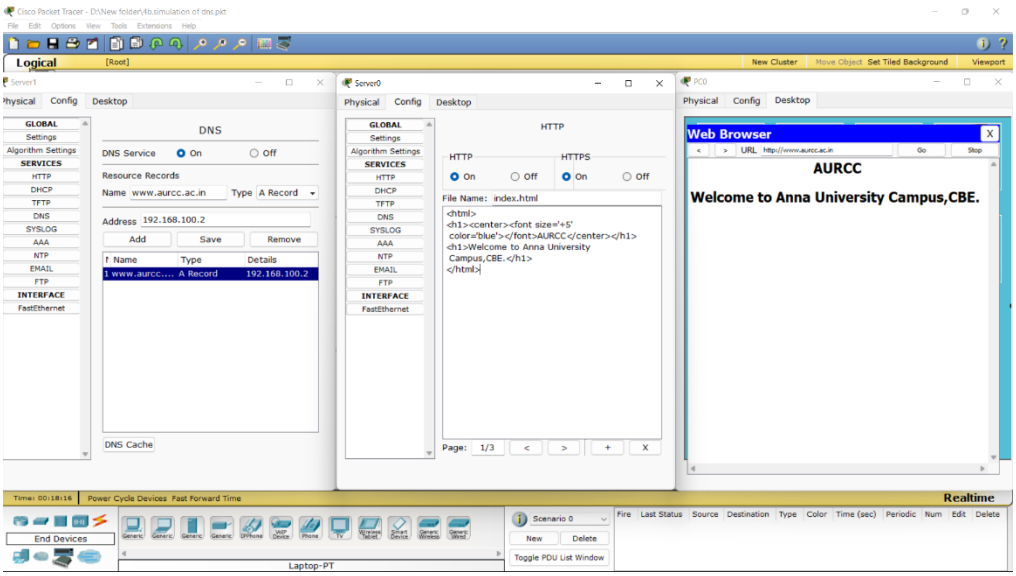
Initial Configuration



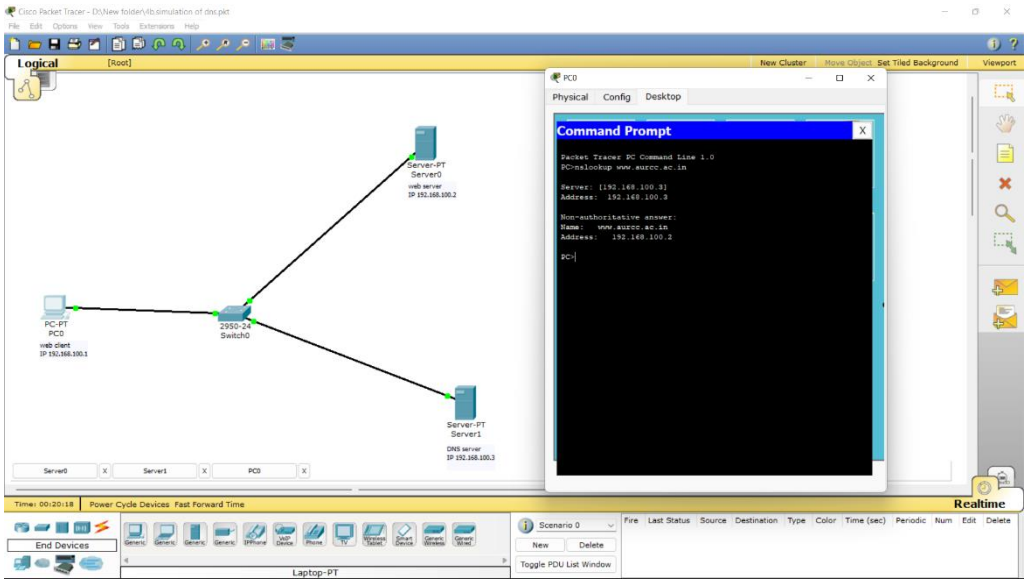
IP Configuration of DNS server, web server and web client respectively



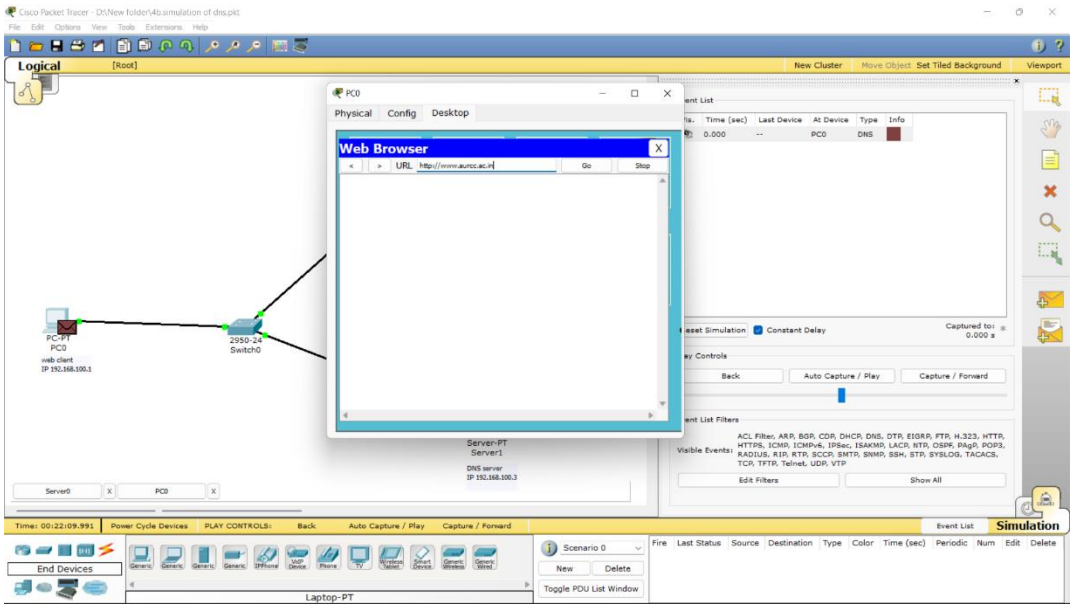
Switching on DNS services in DNS server, switching on HTTP services in web server and visiting the web in web client respectively

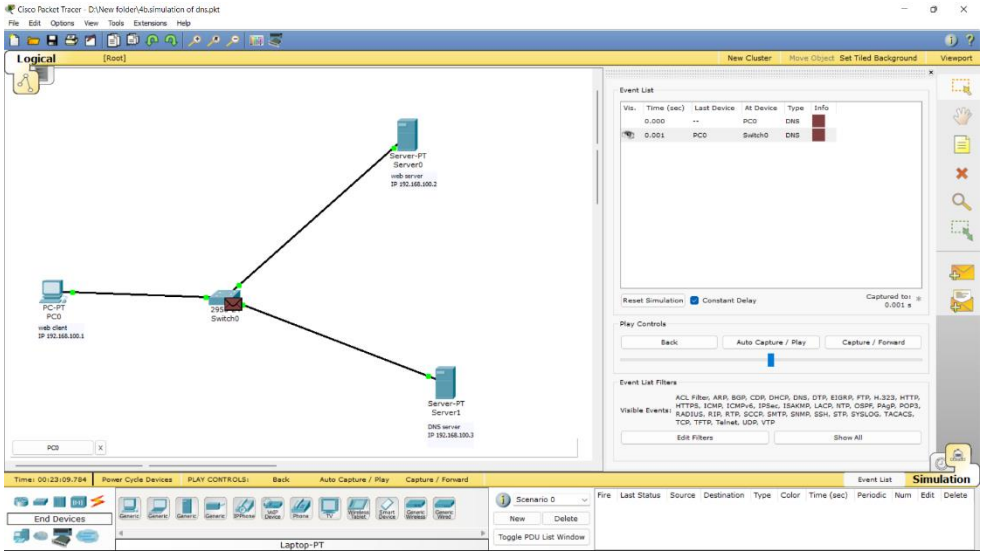


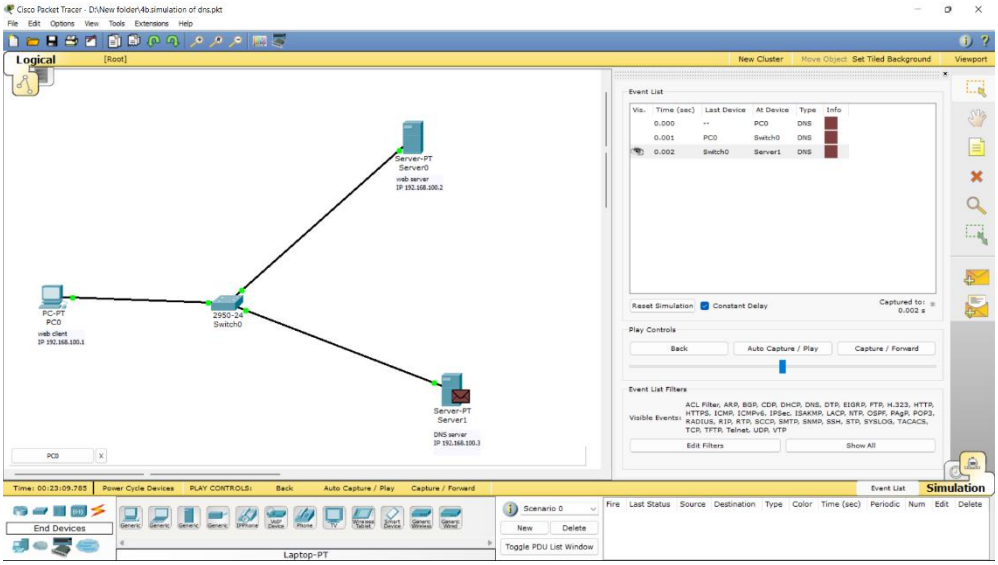
nslookup in the command prompt of web client

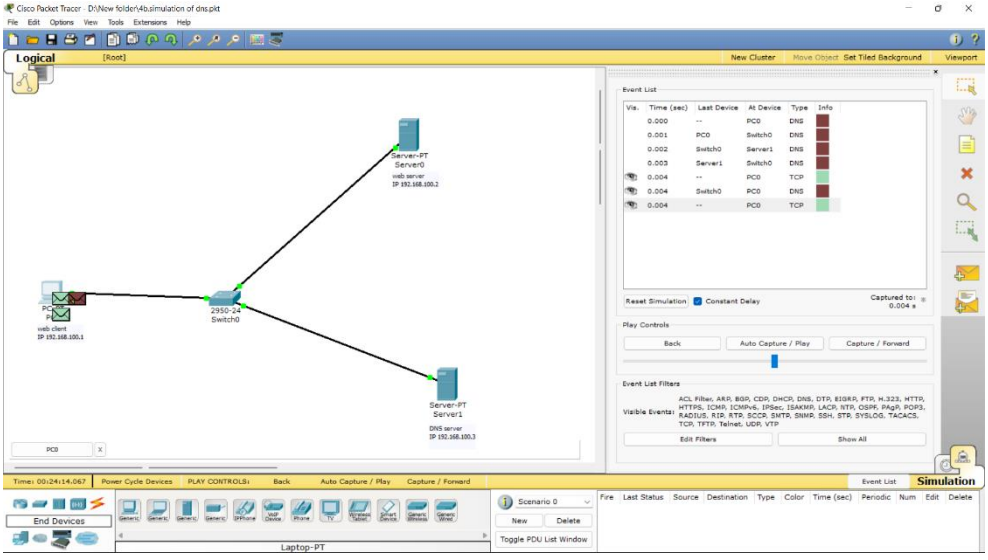


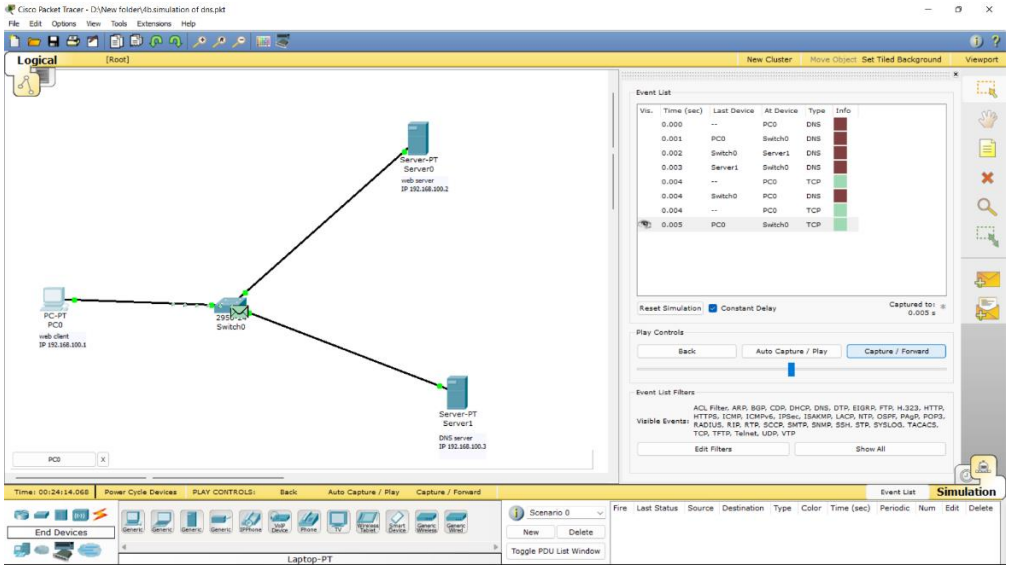
The following snaps trace the route of the packets

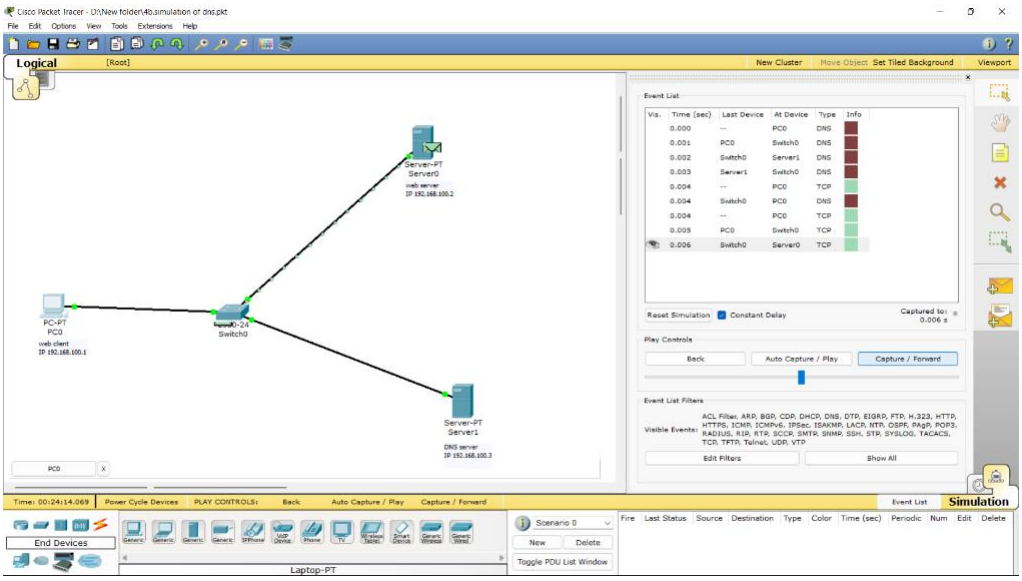


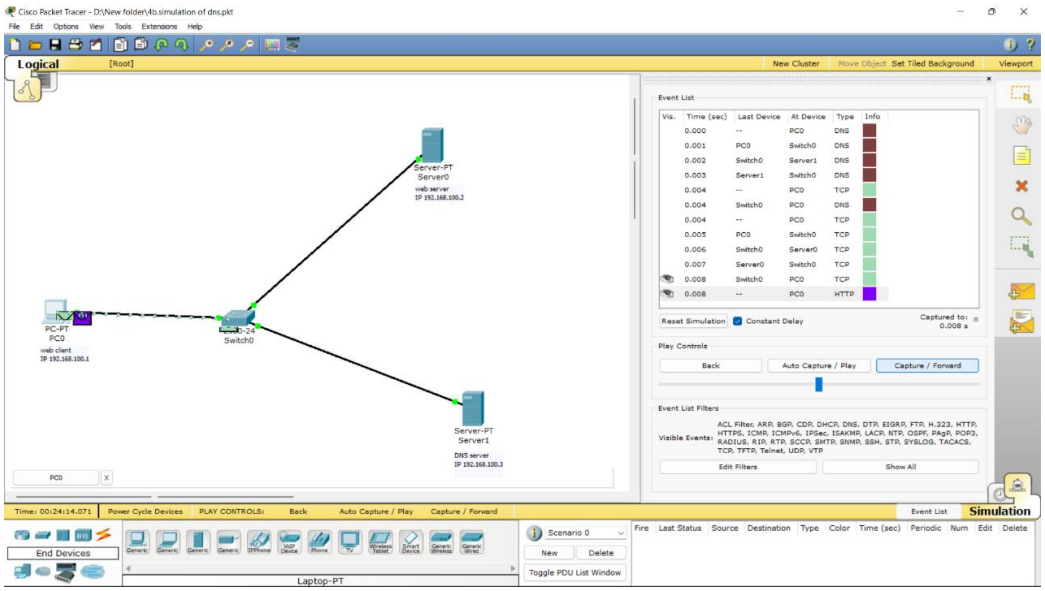

Cisco Packet Tracer - D:\New folder\4b.simulation of dns.pkt
Logical
[Root]
Server-PT Server0
web server
IP 192.168.100.2
PC-PT PC0
web client
IP 192.168.100.1
2950-24 Switch0
Server-PT Server1
DNS server
IP 192.168.100.3
Event List
Simulation
End Devices
Laptop-PT

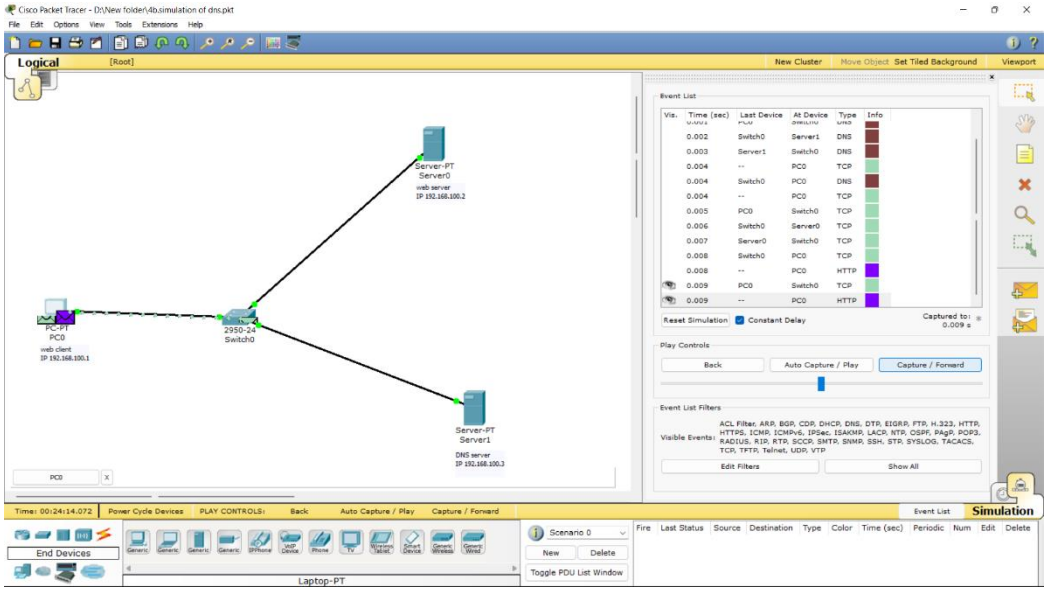

Cisco Packet Tracer - D:\New folder\4b.simulation of dns.pkt
Logical
[Root]
Server-PT Server0
web server
IP 192.168.100.2
PC-PT PC0
web client
IP 192.168.100.1
2950-24 Switch0
Server-PT Server1
DNS server
IP 192.168.100.3
Event List
Simulation
End Devices
Laptop-PT

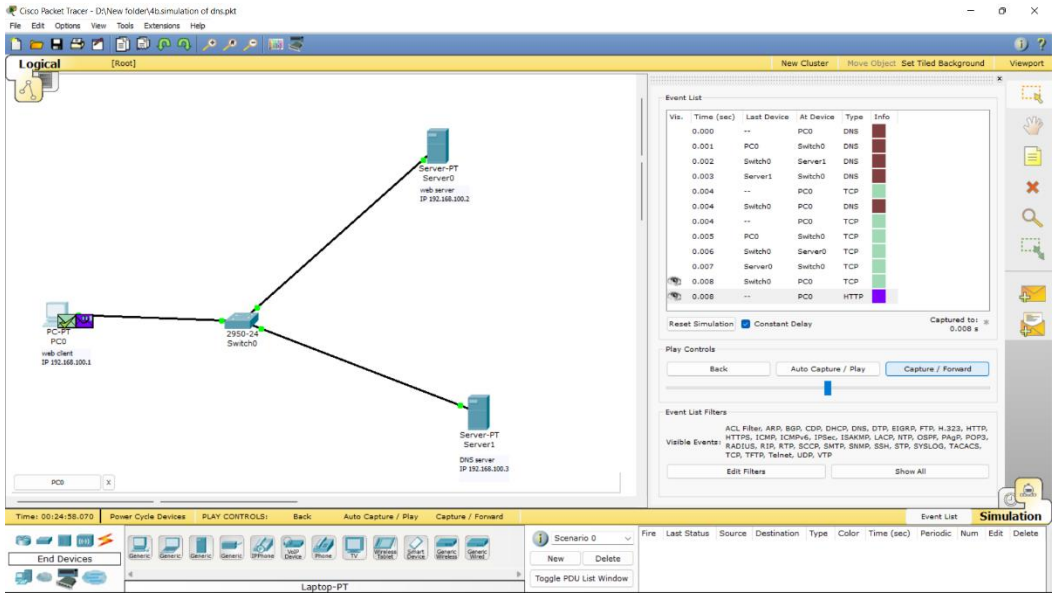

Cisco Packet Tracer - D:\New folder\4b.simulation of dns.pkt
Logical
[Root]
Server-PT Server0
web server
IP 192.168.100.2
PC-PT PC0
web client
IP 192.168.100.1
2950-24 Switch0
Server-PT Server1
DNS server
IP 192.168.100.3
Event List
Simulation
End Devices
Laptop-PT

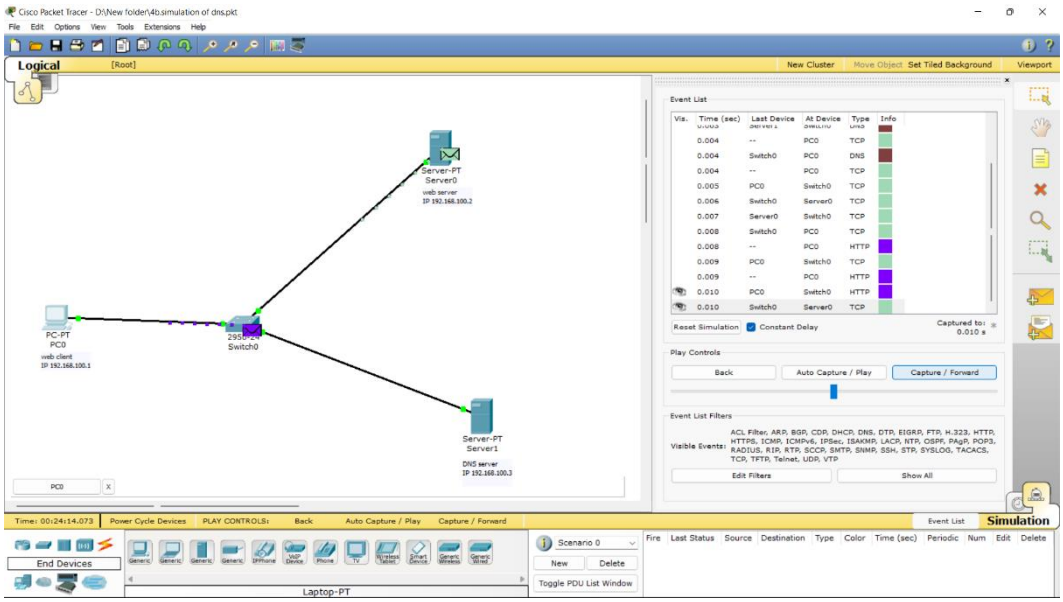

Cisco Packet Tracer - D:\New folder\4b.simulation of dns.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Server-PT Server0
web server
IP 192.168.100.2
PC-PT PC0
web client
IP 192.168.100.1
2950-24 Switch0
Server-PT Server1
DNS server
IP 192.168.100.3
Event List
Vis. Time (sec) Last Device At Device Type Info
0.000 -- PC0 DNS
0.001 PC0 Switch0 DNS
0.002 Switch0 Server1 DNS
0.003 Server1 Switch0 DNS
0.004 -- PC0 TCP
0.004 Switch0 PC0 DNS
0.004 -- PC0 TCP
0.005 PC0 Switch0 TCP
Reset Simulation
Constant Delay
Captured to: 0.005 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters
Visible Events:
ACL Filter, ARP, BGP, CDP, DHCP, DNS, DTP, EIGRP, FTP, H.323, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NTP, OSPF, PAgP, POP3, RADIUS, RIP, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All
Time: 00:24:14.068
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Event List
Simulation
End Devices
Laptop-PT
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time (sec) Periodic Num Edit Delete

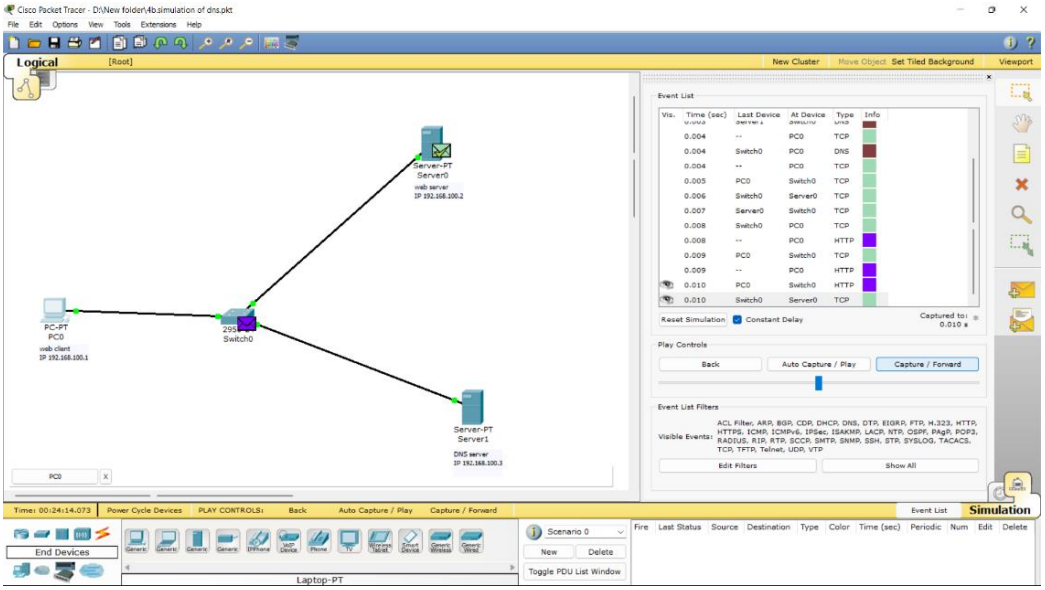

Cisco Packet Tracer - D:\New folder\4b.simulation of dns.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Server-PT Server0
web server
IP 192.168.100.2
PC-PT PC0
web client
IP 192.168.100.1
2950-24 Switch0
Server-PT Server1
DNS server
IP 192.168.100.3
Event List
Vis. Time (sec) Last Device At Device Type Info
0.000 -- PC0 DNS
0.001 PC0 Switch0 DNS
0.002 Switch0 Server1 DNS
0.003 Server1 Switch0 DNS
0.004 -- PC0 TCP
0.004 Switch0 PC0 DNS
0.004 -- PC0 TCP
0.005 PC0 Switch0 TCP
0.006 Switch0 Server0 TCP
Reset Simulation
Constant Delay
Captured to: 0.006 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters
Visible Events:
ACL Filter, ARP, BGP, CDP, DHCP, DNS, DTP, EIGRP, FTP, H.323, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NTP, OSPF, PAgP, POP3, RADIUS, RIP, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All
Time: 00:24:14.069
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Event List
Simulation
End Devices
Laptop-PT
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time (sec) Periodic Num Edit Delete

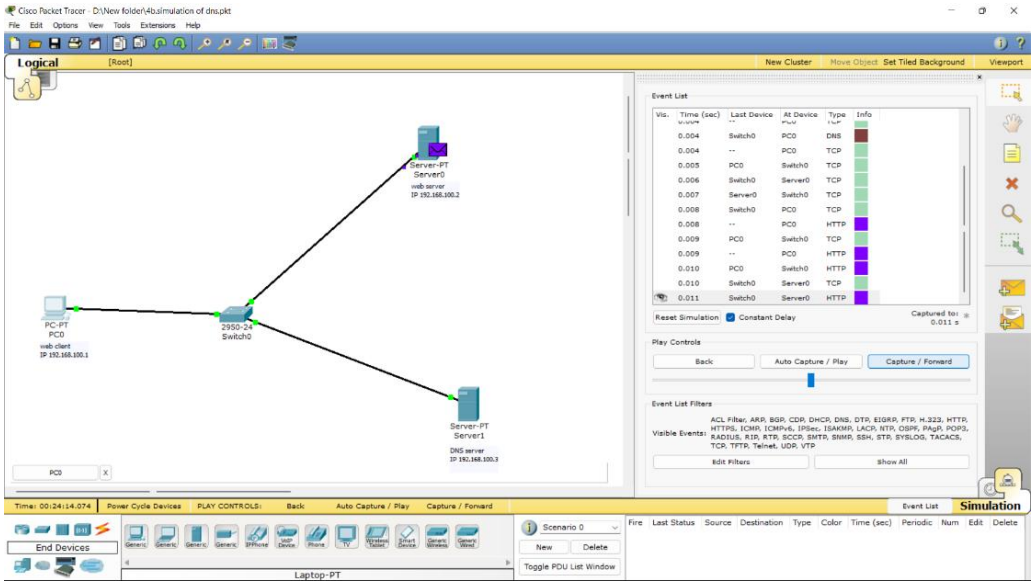

Cisco Packet Tracer - D:\New folder\4b.simulation of dns.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Server-PT Server0
web server
IP 192.168.100.2
PC-PT PC0
web client
IP 192.168.100.1
2950-24 Switch0
Server-PT Server1
DNS server
IP 192.168.100.3
Event List
Vis. Time (sec) Last Device At Device Type Info
0.000 -- PC0 DNS
0.001 PC0 Switch0 DNS
0.002 Switch0 Server1 DNS
0.003 Server1 Switch0 DNS
0.004 -- PC0 TCP
0.004 Switch0 PC0 DNS
0.004 -- PC0 TCP
0.005 PC0 Switch0 TCP
0.006 Switch0 Server0 TCP
0.007 Server0 Switch0 TCP
0.008 Switch0 PC0 TCP
0.008 -- PC0 HTTP
Reset Simulation
Constant Delay
Captured to: 0.008 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters
Visible Events:
ACL Filter, ARP, BGP, CDP, DHCP, DNS, DTP, EIGRP, FTP, H.323, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NTP, OSPF, PAgP, POP3, RADIUS, RIP, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All
Time: 00:24:14.071
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Event List
Simulation
End Devices
Laptop-PT
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time (sec) Periodic Num Edit Delete

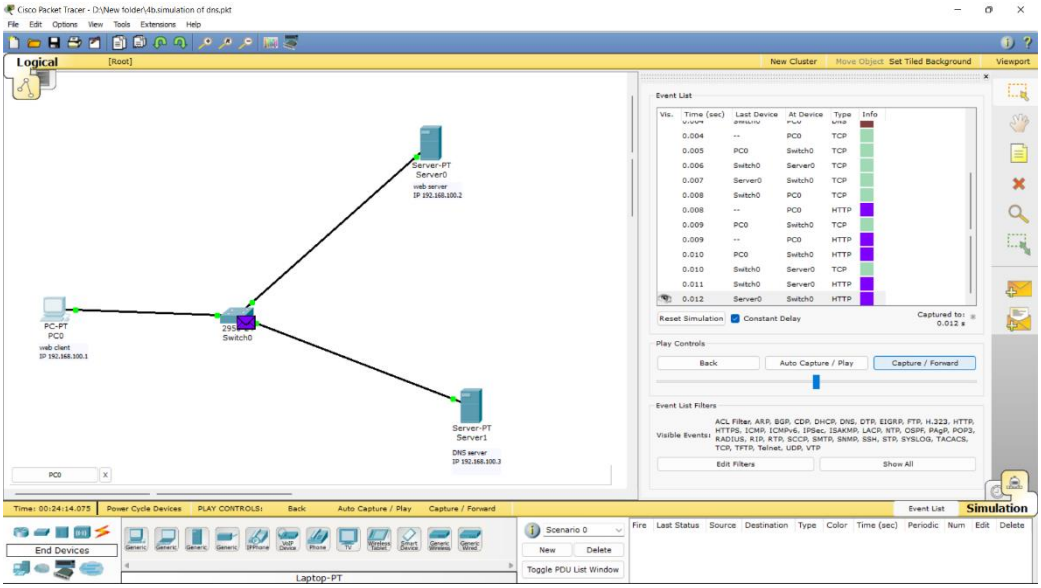

Cisco Packet Tracer - D:\New folder\4b.simulation of dns.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Server-PT Server0
web server
IP 192.168.100.2
PC-PT PC0
web client
IP 192.168.100.1
2950-24 Switch0
Server-PT Server1
DNS server
IP 192.168.100.3
Event List
Vis. Time (sec) Last Device At Device Type Info
0.002 Switch0 Server1 DNS
0.003 Server1 Switch0 DNS
0.004 -- PC0 TCP
0.004 Switch0 PC0 DNS
0.004 -- PC0 TCP
0.005 PC0 Switch0 TCP
0.006 Switch0 Server0 TCP
0.007 Server0 Switch0 TCP
0.008 Switch0 PC0 TCP
0.008 -- PC0 HTTP
0.009 PC0 Switch0 TCP
0.009 -- PC0 HTTP
Reset Simulation
Constant Delay
Captured to: 0.009 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters
Visible Events:
ACL Filter, ARP, BGP, CDP, DHCP, DNS, DTP, EIGRP, FTP, H.323, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NTP, OSPF, PAgP, POP3, RADIUS, RIP, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All
Time: 00:24:14.072
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Event List
Simulation
End Devices
Laptop-PT
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time (sec) Periodic Num Edit Delete

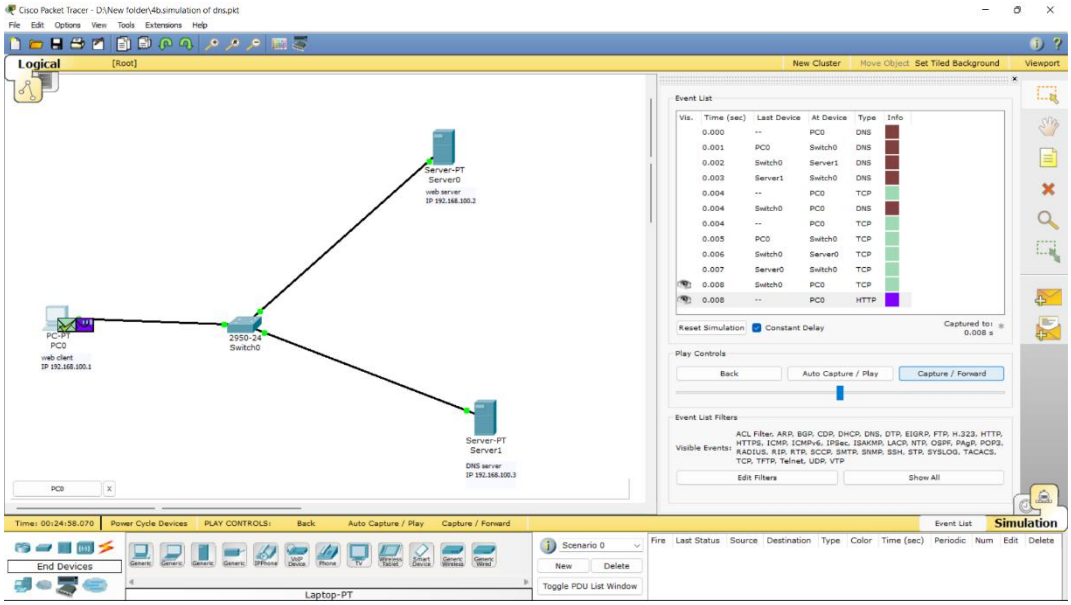

Cisco Packet Tracer - D:\New folder\4b.simulation of dns.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Server-PT Server0
web server
IP 192.168.100.2
PC-PT PC0
web client
IP 192.168.100.1
2950-24 Switch0
Server-PT Server1
DNS server
IP 192.168.100.3
Event List
Vis. Time (sec) Last Device At Device Type Info
0.000 -- PC0 DNS
0.001 PC0 Switch0 DNS
0.002 Switch0 Server1 DNS
0.003 Server1 Switch0 DNS
0.004 -- PC0 TCP
0.004 Switch0 PC0 DNS
0.004 -- PC0 TCP
0.005 PC0 Switch0 TCP
0.006 Switch0 Server0 TCP
0.007 Server0 Switch0 TCP
0.008 Switch0 PC0 TCP
0.008 -- PC0 HTTP
Reset Simulation
Constant Delay
Captured to: 0.008 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters
Visible Events:
ACL Filter, ARP, BGP, CDP, DHCP, DNS, DTP, EIGRP, FTP, H.323, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NTP, OSPF, PAgP, POP3, RADIUS, RIP, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All
Time: 00:24:58.070
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Event List
Simulation
End Devices
Laptop-PT
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time (sec) Periodic Num Edit Delete

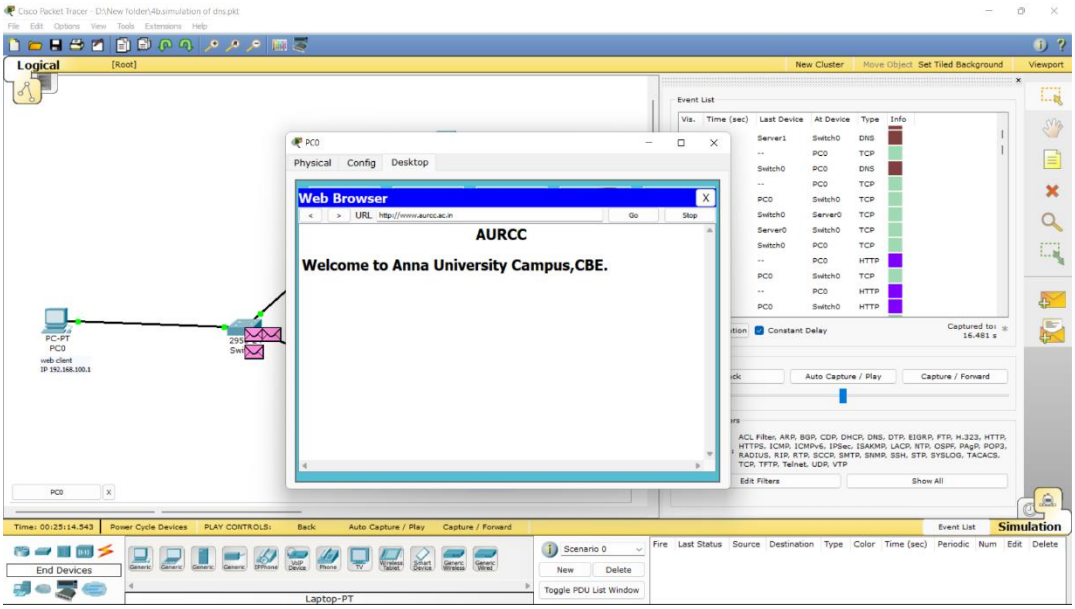

Cisco Packet Tracer - D:\New folder\4b.simulation of dns.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Server-PT Server0
web server
IP 192.168.100.2
PC-PT PC0
web client
IP 192.168.100.1
2950-24 Switch0
Server-PT Server1
DNS server
IP 192.168.100.3
Event List
Vis. Time (sec) Last Device At Device Type Info
0.004 -- PC0 TCP
0.004 Switch0 PC0 DNS
0.004 -- PC0 TCP
0.005 PC0 Switch0 TCP
0.006 Switch0 Server0 TCP
0.007 Server0 Switch0 TCP
0.008 Switch0 PC0 TCP
0.008 -- PC0 HTTP
0.009 PC0 Switch0 TCP
0.009 -- PC0 HTTP
0.010 PC0 Switch0 HTTP
0.010 Switch0 Server0 TCP
Reset Simulation
Constant Delay
Captured to: 0.010 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters
Visible Events:
ACL Filter, ARP, BGP, CDP, DHCP, DNS, DTP, EIGRP, FTP, H.323, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NTP, OSPF, PAgP, POP3, RADIUS, RIP, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All
Time: 00:24:14.073
Power Cycle Devices
PLAY CONTROLS:
Back
Auto Capture / Play
Capture / Forward
Event List
Simulation
End Devices
Laptop-PT
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time (sec) Periodic Num Edit Delete


Cisco Packet Tracer - D:\New folder\4b.simulation of dns.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Server-PT Server0
web server
IP 192.168.100.2
PC-PT PC0
web client
IP 192.168.100.1
2950-24 Switch0
Server-PT Server1
DNS server
IP 192.168.100.3
Event List
Vis. Time (sec) Last Device At Device Type Info
0.004 -- PC0 TCP
0.004 Switch0 PC0 DNS
0.004 -- PC0 TCP
0.005 PC0 Switch0 TCP
0.006 Switch0 Server0 TCP
0.007 Server0 Switch0 TCP
0.008 Switch0 PC0 TCP
0.008 -- PC0 HTTP
0.009 PC0 Switch0 TCP
0.009 -- PC0 HTTP
0.010 PC0 Switch0 HTTP
0.010 Switch0 Server0 TCP
Reset Simulation
Constant Delay
Captured to: 0.010 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters
Visible Events:
ACL Filter, ARP, BGP, CDP, DHCP, DNS, DTP, EIGRP, FTP, H.323, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NTP, OSPF, PAgP, POP3, RADIUS, RIP, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All
Time: 00:24:14.073
Power Cycle Devices
PLAY CONTROLS:
Simulation
End Devices
Laptop-PT
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time (sec) Periodic Num Edit Delete


Cisco Packet Tracer - D:\New folder\4b.simulation of dns.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Server-PT Server0
web server
IP 192.168.100.2
PC-PT PC0
web client
IP 192.168.100.1
2950-24 Switch0
Server-PT Server1
DNS server
IP 192.168.100.3
Event List
Vis. Time (sec) Last Device At Device Type Info
0.004 Switch0 PC0 DNS
0.004 -- PC0 TCP
0.005 PC0 Switch0 TCP
0.006 Switch0 Server0 TCP
0.007 Server0 Switch0 TCP
0.008 Switch0 PC0 TCP
0.008 -- PC0 HTTP
0.009 PC0 Switch0 TCP
0.009 -- PC0 HTTP
0.010 PC0 Switch0 HTTP
0.010 Switch0 Server0 TCP
0.011 Switch0 Server0 HTTP
Reset Simulation
Constant Delay
Captured to: 0.011 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters
Visible Events:
ACL Filter, ARP, BGP, CDP, DHCP, DNS, DTP, EIGRP, FTP, H.323, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NTP, OSPF, PAgP, POP3, RADIUS, RIP, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All
Time: 00:24:14.074
Power Cycle Devices
PLAY CONTROLS:
Simulation
End Devices
Laptop-PT
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time (sec) Periodic Num Edit Delete


Cisco Packet Tracer - D:\New folder\4b.simulation of dns.pkt
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
Server-PT Server0
web server
IP 192.168.100.2
PC-PT PC0
web client
IP 192.168.100.1
2950-24 Switch0
Server-PT Server1
DNS server
IP 192.168.100.3
Event List
Vis. Time (sec) Last Device At Device Type Info
0.004 -- PC0 TCP
0.005 PC0 Switch0 TCP
0.006 Switch0 Server0 TCP
0.007 Server0 Switch0 TCP
0.008 Switch0 PC0 TCP
0.008 -- PC0 HTTP
0.009 PC0 Switch0 TCP
0.009 -- PC0 HTTP
0.010 PC0 Switch0 HTTP
0.010 Switch0 Server0 TCP
0.011 Switch0 Server0 HTTP
0.012 Server0 Switch0 HTTP
Reset Simulation
Constant Delay
Captured to: 0.012 s
Play Controls
Back
Auto Capture / Play
Capture / Forward
Event List Filters
Visible Events:
ACL Filter, ARP, BGP, CDP, DHCP, DNS, DTP, EIGRP, FTP, H.323, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NTP, OSPF, PAgP, POP3, RADIUS, RIP, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters
Show All
Time: 00:24:14.075
Power Cycle Devices
PLAY CONTROLS:
Simulation
End Devices
Laptop-PT
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time (sec) Periodic Num Edit Delete





RESULT:

Thus, the DNS is simulated using cisco packet tracer.

Expt No : 5 a	ARP AND RARP
Date :	

Aim:

To implement Address Resolution Protocol and Reverse Address Resolution Protocol program using python.

Algorithm:

Step 1: Start the process.

Step 2: Establish socket connection using socket package in both client and server side.

Step 3: Send IP or MAC based on the given condition.

Step 4: Based on the dictionary makeup in the server side with IP and MAC table, the corresponding IP for MAC or MAC for IP is sent.

Step 5: Receive the IP or MAC from the server and print the address.

Step 6: End the process.

Program:

Server code:

```
import socket
table={
'192.168.1.1':'1E.4A.4A.11', '192.168.2.1':'5E.51.4B.01',
'192.168.1.3':'4B.35.CD.32', '192.168.4.1':'AF.4D.1F.FF',
'192.168.3.2':'C3.C5.EE.C2', '1E.4A.4A.11':'192.168.1.1',
'5E.51.4B.01':'192.168.2.1', '4B.35.CD.32':'192.168.1.3',
'AF.4D.1F.FF':'192.168.4.1',
'C3.C5.EE.C2':'192.168.3.2'
}
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM) s.bind(('',1234))
s.listen()
clientsocket, address = s.accept()
print("Connection From ", address, " Has Established") ip = clientsocket.recv(1024)
ip = ip.decode("utf-8")

mac = table.get(ip,'No entry for given address') clientsocket.send(mac.encode())
```

Client code:

```
import socket
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(('localhost',1234))
a=input("ARP or RARP:")
if(a=="ARP"):
    add = input('Enter IP : ')
elif(a=="RARP"):
    add = input('Enter MAC : ')
s.send(add.encode())
mac = s.recv(1024)

mac = mac.decode("utf-8")
if(a=="ARP"):
    print('MAC of',add,' is : ',mac)
else:
    print('IP of',add,' is : ',mac)
```

Input and Output:

```
ARP or RARP: ARP
Enter IP : 192.168.1.1
MAC of 192.168.1.1 is : 1E.4A.4A.11


ARP or RARP: RARP
Enter MAC : 1E.4A.4A.11
IP of 1E.4A.4A.11 is : 192.168.1.1
```

Result:

The IP or MAC address was sent and the corresponding MAC or IP address was received using the ARP or RARP implementation successfully.

Expt No : 5 b	ADDRESS RESOLUTION PROTOCOL / REVERSE ARP – PACKET TRACER
Date :	

Aim:

To implement Address Resolution Protocol and Reverse Address Resolution Protocol program using packet tracer tool.

Algorithm:

Step 1: Open Packet tracer tool and enter into simulation mode.

Step 2: Select 4 PCs and assign them the required IP addresses.

Step 3: Select a Switch and connect the PCs.

Step 4: In the “events list” select ARP and ICMP protocols and deselect all other protocols.

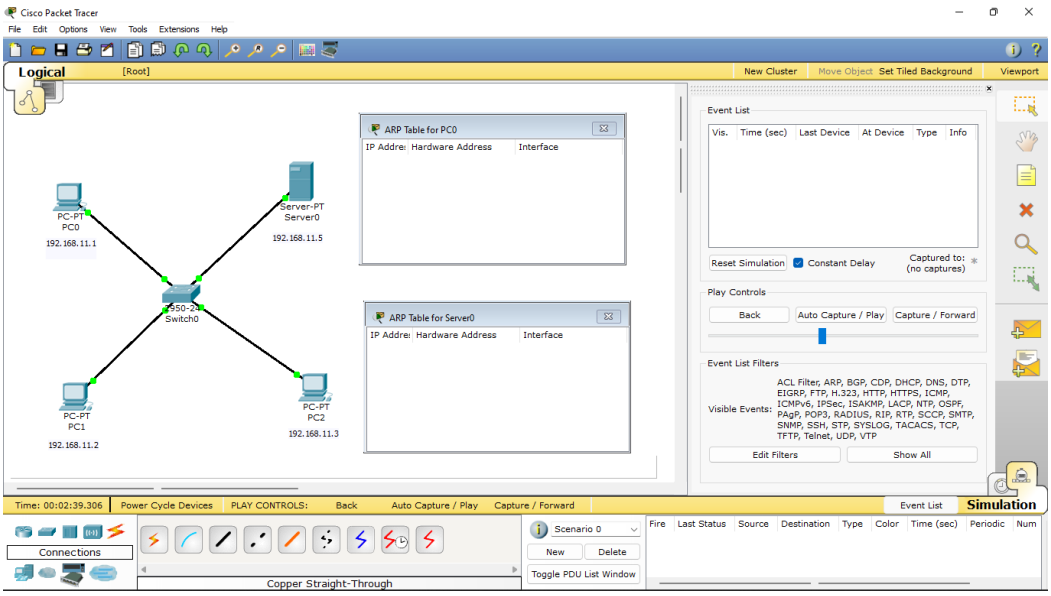
Step 5: Use the simple PDU packet to send from any selected PC at “Source” to other PC as “Destination”.

Step 6: Use capture/forward button to examine the ARP request/reply packets.

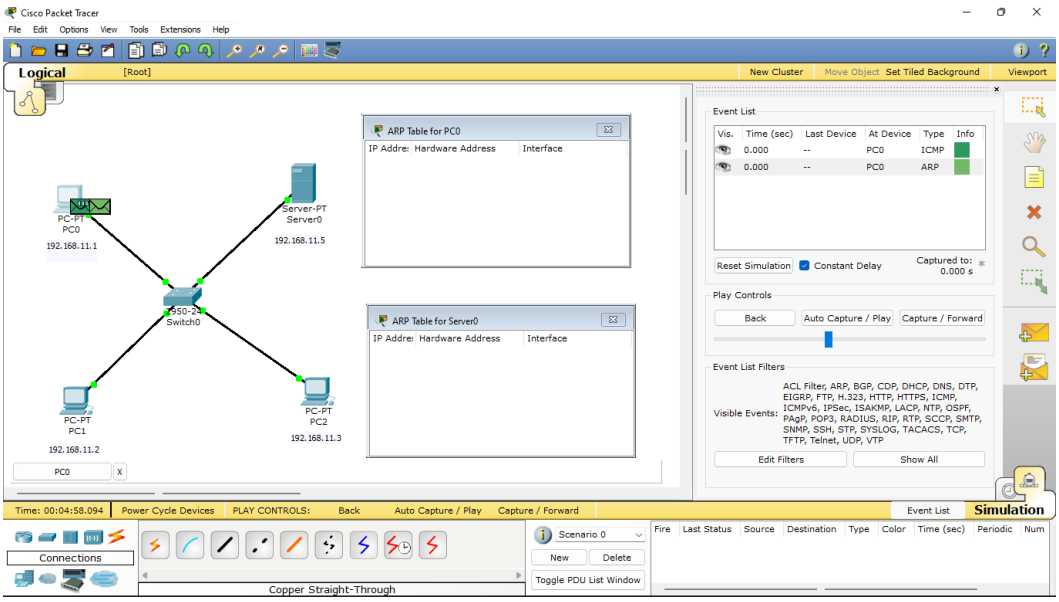
Step 7: Save the configuration.

Program & Output:

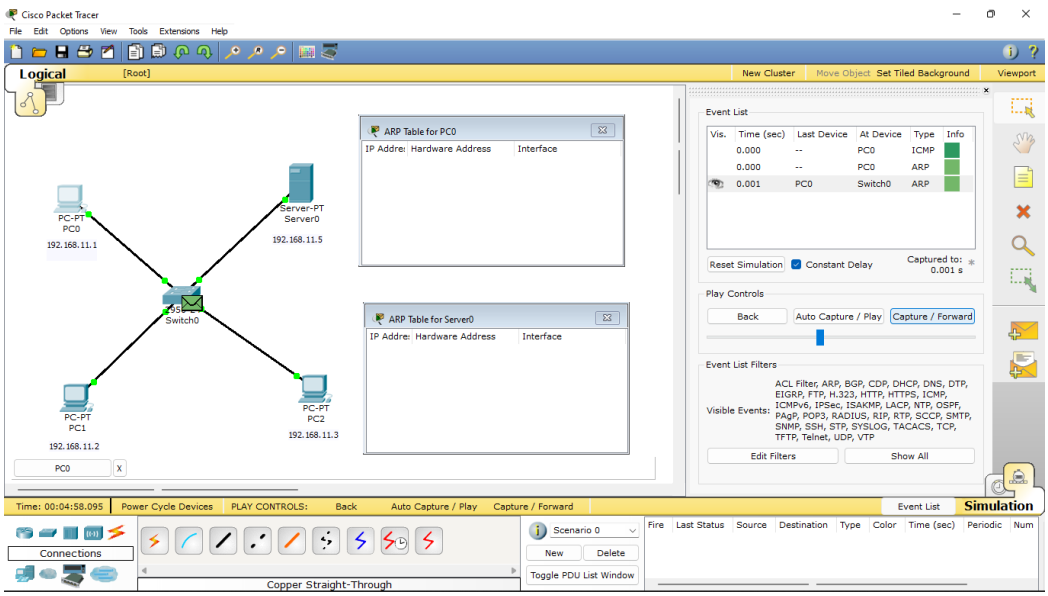
1. Initial connection before any transmission. ARP table of Server0 and PC0 is empty.



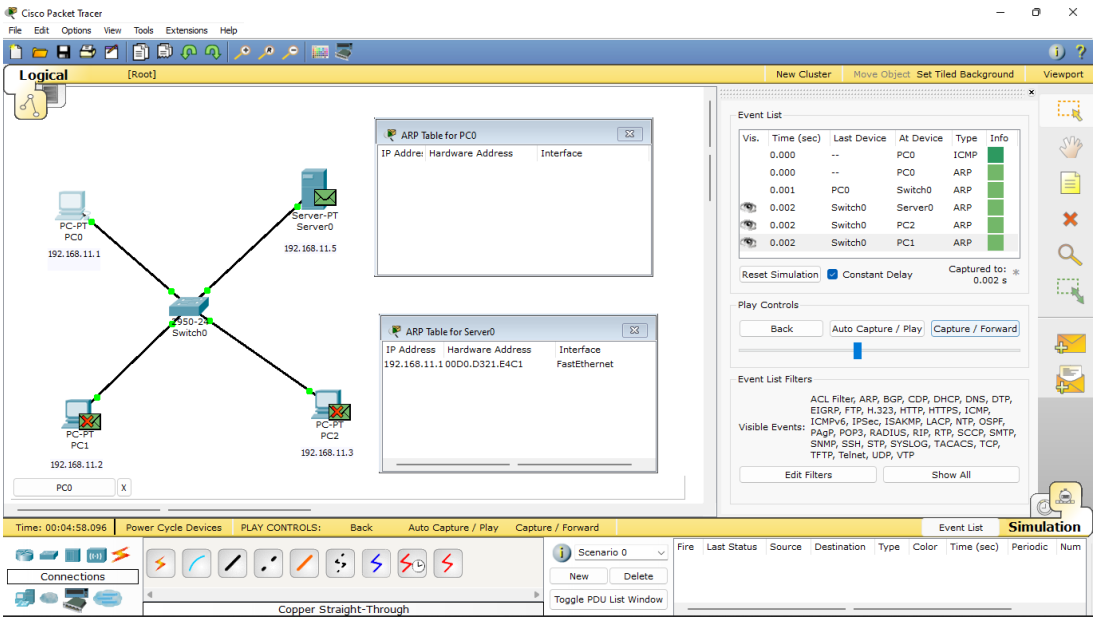
2. A ping from PC0 to Server0 is initialized.



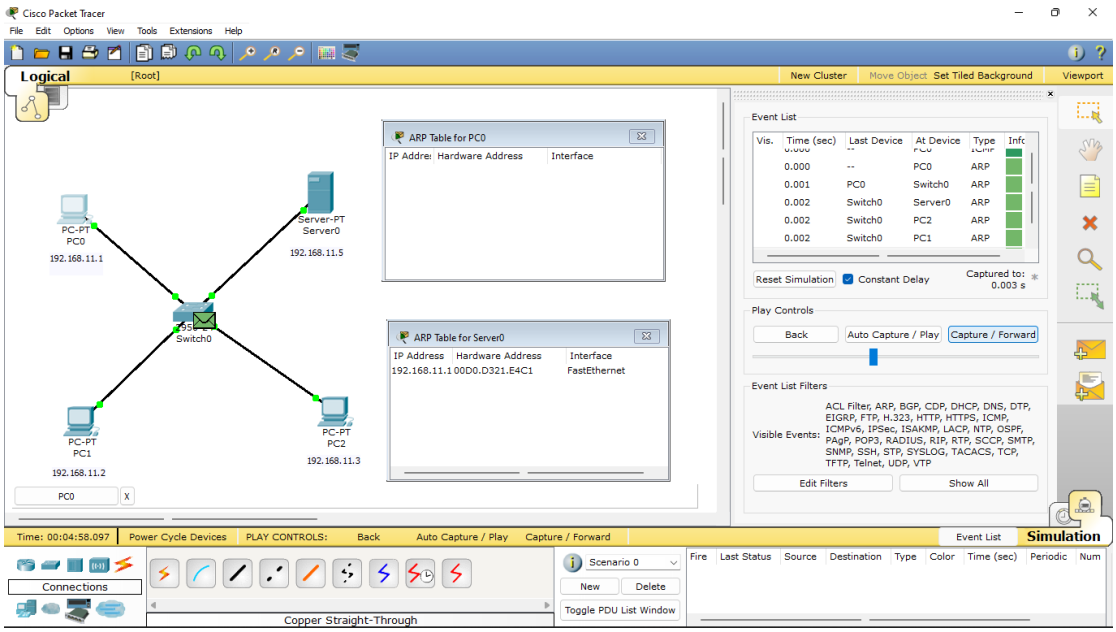
3. The ping request from PC0 is transmitted through the Switch to Server0.



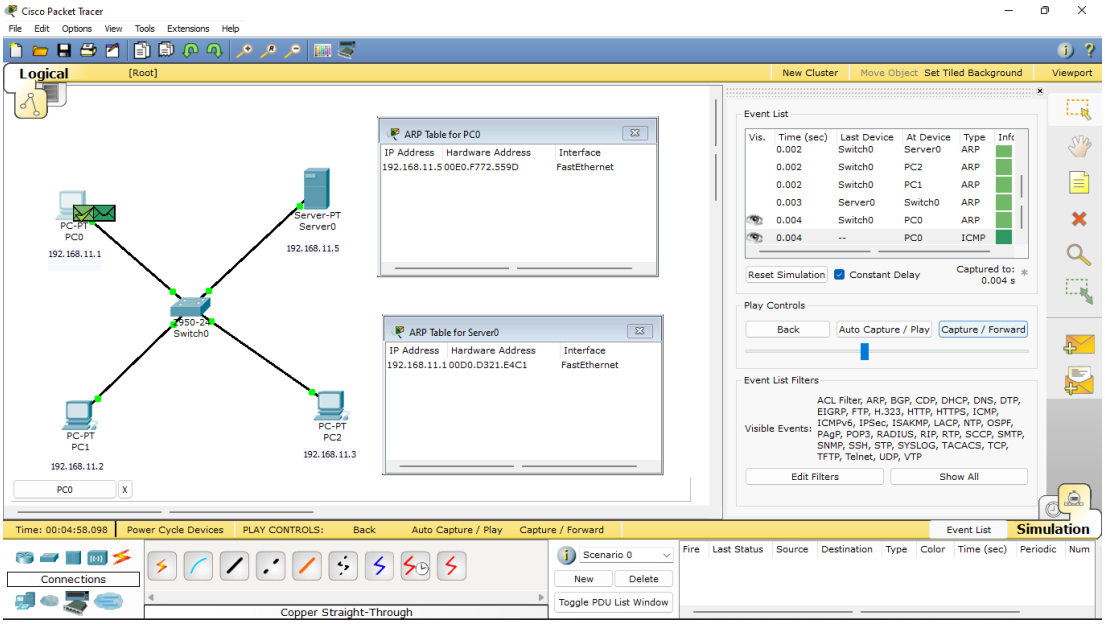
4. From the switch, the ping request is sent to all end systems except PC0 that are connected to the switch. However, the ping request is accepted only by the Server0. The ARP table of Server0 is updated.



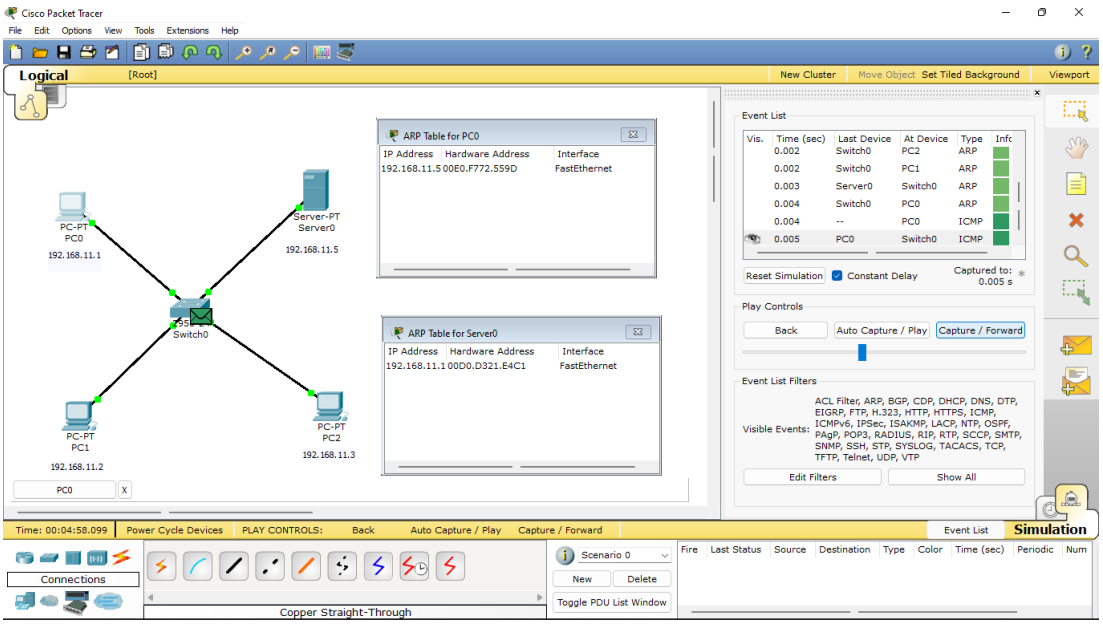
5. Ping acceptance acknowledgement is transmitted from Server0 to PC0 through switch.



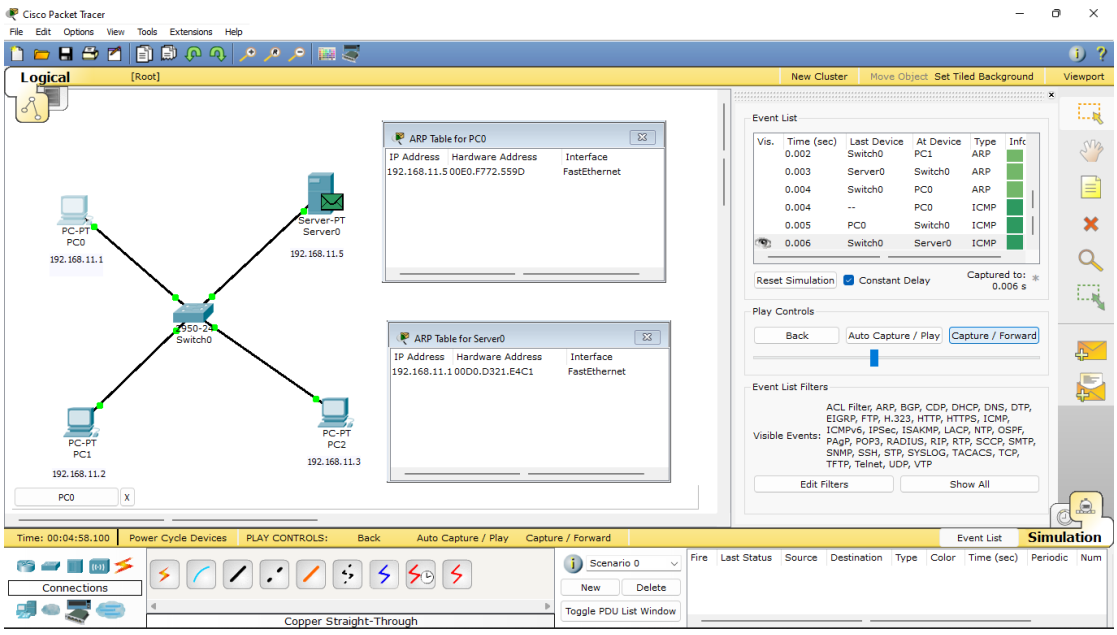
6. The acknowledgement from Server0 reaches PC0. The ARP table of PC0 is also updated.



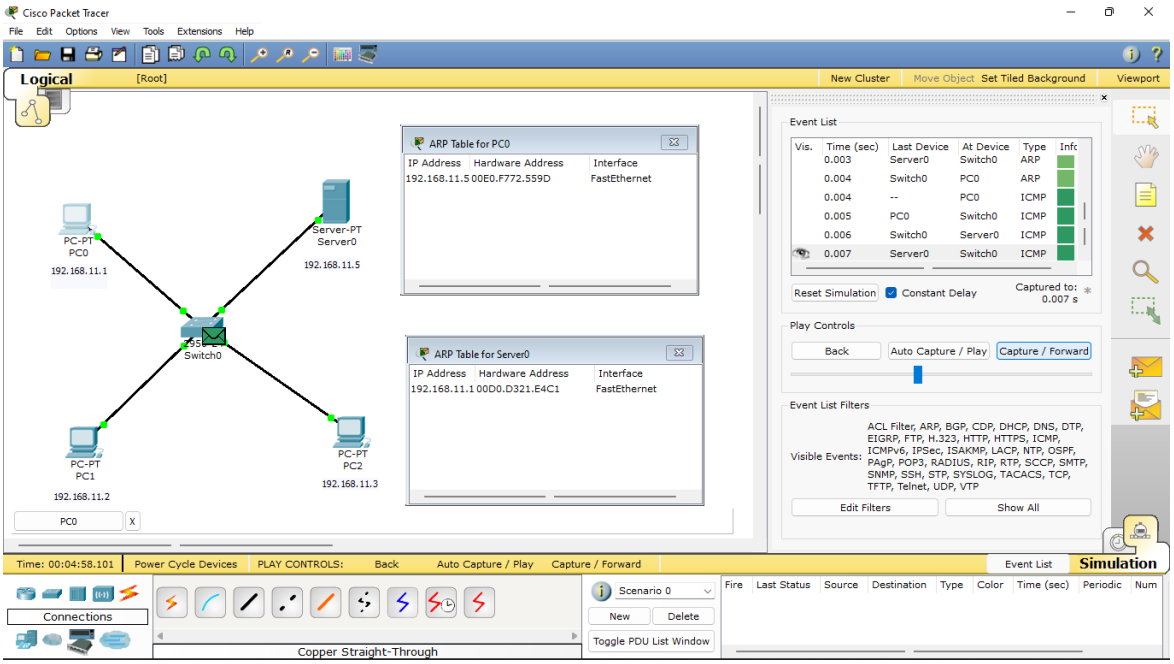
7. The data from PC0 is now transmitted to Server0 through switch.



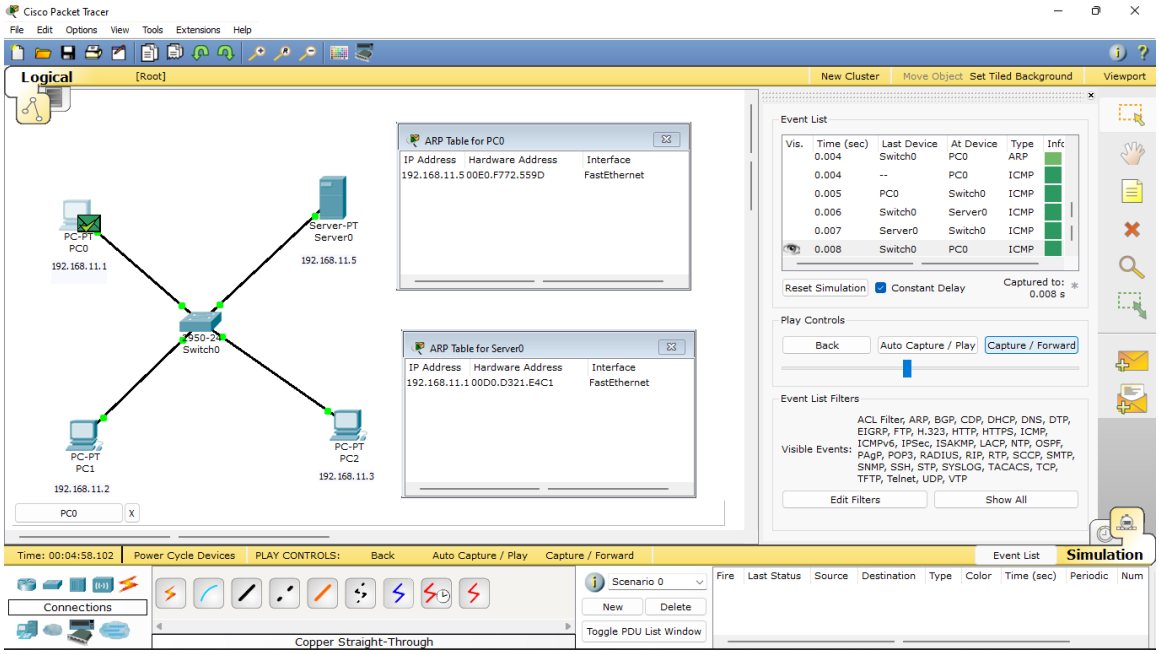
8. The data from PC0 reaches Server0.



9. Once again, the acknowledgement from Server0 is transmitted to PC0 through Switch.



10. The acknowledgement from Server0 reaches PC0.



RESULT :

Address Resolution Protocol and Reverse Address Resolution Protocol program using packet tracer tool is implemented.

Expt No : 6 a	STUDY OF NETWORK SIMULATOR
Date :	

Aim:

To study the Network Simulator in detail.

Procedure:

Step 1: Download and Extract ns2.

Step 2: Building the dependencies.

Step 3: Install ns2 using this code

sudo su cd ~/ns-allinone-2.35/./install

Step 4: Setting the environment path

sudo gedit ~/.bashrc

Step 5: Running

ns2

Prelab Learning Material

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors. Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989. Ever since, several revolutions and revisions have marked the growing maturity of the tool, thanks to substantial contributions from the players in the field. Among these are the University of California and Cornell University who developed the REAL network simulator,1 the foundation which NS is based on. Since 1995 the Defense Advanced Research Projects Agency (DARPA) supported development of NS through the Virtual Inter Network Testbed (VINT) project. Currently the National Science Foundation (NSF) has joined the ride in development. Last but not the least, the group of Researchers and developers in the community are constantly working to keep NS2 strong and versatile.

Basic architecture:

Figure 2.1 shows the basic architecture of NS2. NS2 provides users with executable command ns which takes on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns.

In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend).

The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as handles. Conceptually, a handle (e.g., n as a Node handle) is just a string (e.g.,_o10) in the OTcl domain, and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may defines its own procedures and variables to facilitate the interaction. Note that the member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively. Before proceeding further, the readers are encouraged to learn C++ and OTcl languages. We refer the readers to [14] for the detail of C++, while a brief tutorial of Tcl and OTcl tutorial are given in Appendices A.1 and A.2, respectively.

NS2 provides a large number of built-in C++ objects. It is advisable to use these C++ objects to set up a simulation using a Tcl simulation script. However, advance users may find these objects insufficient. They need to develop their own C++ objects, and use a OTcl configuration interface to put together these objects. After simulation, NS2 outputs either text-based or animation-based simulation results. To interpret these results graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used. To analyze a particular behaviour of the network, users can extract a relevant subset of text-based data and transform it to a more conceivable presentation.

Concept overview:

NS uses two languages because simulator has two different kinds of things it needs to do. On one hand,detailed simulations of protocols requires a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data

sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important. On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios.

In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of this part of the task is less important. ns meets both of these needs with two languages, C++ and OTcl.

Tcl scripting

Tcl is a general purpose scripting language. [Interpreter]

- Tcl runs on most of the platforms such as Unix, Windows, and Mac.
- The strength of Tcl is its simplicity.
- It is not necessary to declare a data type for variable prior to the usage.

Basics of TCL

Syntax: command arg1 arg2 arg3

Hello World!

puts stdout{Hello, World!} Hello, World!

Variables Command Substitution

set a 5 set len [string length foobar]

set b $a set len [expr [string length foobar] + 9]

Wired TCL Script Components

Create the event scheduler

Open new files & turn on the tracing

Create the nodes

Setup the links

Configure the traffic type (e.g., TCP, UDP, etc)

Set the time of traffic generation (e.g., CBR, FTP)

Terminate the simulation

NS Simulator Preliminaries.

1. Initialization and termination aspects of the ns simulator.
2. Definition of network nodes, links, queues and topology.
3. Definition of agents and of applications.
4. The nam visualization tool.
5. Tracing and random variables.

Initialization and Termination of TCL Script in NS-2

An ns simulation starts with the command

set ns [new Simulator]

Which is thus the first line in the tcl script. This line declares a new variable as using the set command, you can call this variable as you wish, In general people declares it as ns because it is an instance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the reserved word new.

In order to have output files with data on the simulation (trace files) or files used for visualization (nam files), we need to create the files using —open command:

```
#Open the Trace file
set tracefile1 [open out.tr w]
$ns trace-all $tracefile1
#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
```

The above creates a dta trace file called out.tr and a nam visualization trace file called out.nam. Within the tcl script, these files are not called explicitly by their names, but instead by pointers that are declared above and called —tracefile1 and —namfile respectively. Remark that they begins with a # symbol. The second line open the file —out.tr to be used for writing, declared with the letter —w. The third line uses a simulator method called trace-all that have as parameter the name of the file where the traces will go.

Define a “finish' procedure

Proc finish { } {

global ns tracefile1 namfile

$ns flush-trace

Close $tracefile1

Close $namfile

Exec nam out.nam &

Exit 0

}

Definition of a network of links and nodes

The way to define a node is

set n0 [$ns node]

Once we define several nodes, we can define the links that connect them. An example of a definition of a link is:

$ns duplex-link $n0 $n2 10Mb 10ms DropTail

Which means that $n0 and $n2 are connected using a bi-directional link that has 10ms of propagation delay and a capacity of 10Mb per sec for each direction.

To define a directional link instead of a bi-directional one, we should replace —duplex-link by —simplex-link.

In ns, an output queue of a node is implemented as a part of each link whose input is that node. We should also define the buffer capacity of the queue related to each link. An example would be:

#set Queue Size of link (n0-n2) to 20

$ns queue-limit $n0 $n2 20

FTP over TCP

TCP is a dynamic reliable congestion control protocol. It uses Acknowledgements created by the destination to know whether packets are well received.

There are number variants of the TCP protocol, such as Tahoe, Reno, NewReno, Vegas. The type of agent appears in the first line:

set tcp [new Agent/TCP]

The command $ns attach-agent $n0 $tcp defines the source node of the tcp connection.The command set sink [new Agent /TCPSink]Defines the behavior of the destination node of TCP and assigns to it a pointer called sink.

#Setup a UDP connection

set udp [new Agent/UDP]

$ns attach-agent $n1 $udp

set null [new Agent/Null]

$ns attach-agent $n5 $null

$ns connect $udp $null

$udp set fid_2

#setup a CBR over UDP connection

The below shows the definition of a CBR application using a UDP agent

The command $ns attach-agent $n4 $sink defines the destination node. The command $ns connect$tcp $sink finally makes the TCP connection between the source and destination nodes.

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set packetsize_ 100

$cbr set rate_ 0.01Mb

$cbr set random_ false

TCP has many parameters with initial fixed defaults values that can be changed if mentioned explicitly. For example, the default TCP packet size has a size of 1000bytes.This can be changed to another value, say 552bytes, using the command $tcp set packetSize_ 552.

When we have several flows, we may wish to distinguish them so that we can identify them with different colors in the visualization part. This is done by the command $tcp set fid_ 1 that assigns to the TCP connection a flow identification of —1.We shall later give the flow identification of —2‖ to the UDP connection.

RESULT:

Thus the Network Simulator is studied in detail.

Expt No : 6 b	SIMULATION OF CONGESTION CONTROL ALGORITHMS USING NS
Date :	

Aim:

To study the simulation of Congestion control Algorithm using NS.

Algorithm :

Step 1: Read The Data For Packets
Step 2: Read The Queue Size
Step 3: Divide the Data into Packets
Step 4: Assign the random Propagation delays for each packets to input into the bucket (input_packet).
Step 5: while((Clock++< total_paclets))
Step 5 a): if (clock == input_packet) i. insert into Queue
Step 5 b): if (clock % 5 == 0) i. Remove paclet from Queue
Step 6: End

Prelab Learning Material

Wireless Ad hoc routing, mobile IP, sensor-MAC Tracing, visualization and various utilities NS(Network Simulators)

Most of the commercial simulators are GUI driven, while some network simulators areCLI driven. The network model / configuration describes the state of the network (nodes,routers,switches, links) and the events (data transmissions, packet error etc.). An important output ofsimulations are the trace files. Trace files log every packet, every event that occurred in thesimulation and are used for analysis. Network simulators can also provide other tools to facilitatevisual analysis of trends and potential trouble spots.

Simulation of networks is a very complex task. For example, if congestion is high, thenestimation of the average occupancy is challenging because of high variance. To estimate thelikelihood of a buffer overflow in a network, the time required for an accurate answer can beextremely large. Specialized techniques such as "control variates" and "importance sampling"have been developed to speed simulation.

researchpapers:

1. ns (open source)
2. OPNET (proprietary software)
3. NetSim (proprietary software)

Uses of network simulators

Network simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple networked computers, routers and data links, network simulators are relatively fast and inexpensive. They allow engineers, researchers to test scenariosthat might be particularly difficult or expensive to emulate using real hardware - for instance, simulating a scenario with several nodes or experimenting with a new protocol in the network.

Network simulators are particularly useful in allowing researchers to test new networkingprotocols or changes to existing protocols in a controlled and reproducible environment. Atypical network simulator encompasses a wide range of networking technologies and can help theusers to build complex networks from basic building blocks such as a variety of nodes and links.

Packet loss

PL occurs when one or morepacketsof data travelling across a computer networkfail to reachtheirdestination. Packet loss is distinguished as one of the three main error types encountered indigital communications; the other two being bit errorand spurious packets caused due to noise. Packets can be lost in a network because they may be dropped when a queue in the network nodeoverflows. The amount of packet loss during the steady state is another important property of acongestion control scheme.

Throughput

This is the main performance measure characteristic, and most widely used. In communication networks, such as Ethernet or packet radio, throughput or network throughput is the average rate of successful message delivery over a communication channel.

Delay

Delay is the time elapsed while a packet travels from one point e.g., source premise or network ingress to destination premise or network degrees.

Queue Length

A queuing system in networks can be described as packets arriving for service, waiting for service if it is not immediate, and if having waited for service, leaving the system after being served.

Program

```
import java.util.*;
public class leaky
{
static int min(int x,int y)
{

if(x<y)
return x;
else
return y;
}

public static void main(String[] args)

{ int drop=0,mini,nsec,cap,count=0,i,process;
int inp[]=new int[25];
Scanner sc=new Scanner(System.in);
System.out.println("Enter The Bucket Size\n");
cap= sc.nextInt();
System.out.println("Enter The Operation Rate\n");
process= sc.nextInt();
System.out.println("Enter The No. Of Seconds You Want To Stimulate\n");
nsec=sc.nextInt();
for(i=0;i<nsec;i++)

{ System.out.print("Enter The Size Of The Packet Entering At "+ i+1 + "sec");
inp[i] = sc.nextInt();
}

System.out.println("\nSecond | Packet Recieved | Packet Sent | Packet Left |Packet Dropped|\n");
System.out.println("..........\n");
for(i=0;i<nsec;i++)

{ count+=inp[i];
if(count>cap)
{ drop=count-cap;
count=cap;
}

System.out.print(i+1);
System.out.print("\t\t"+inp[i]);
mini=min(count,process);
System.out.print("\t\t"+mini);
count=count-mini;
System.out.print("\t\t"+count);
```

```
 System.out.print("\t\t"+drop);
drop=0;
  System.out.println();

  }

  for(;count!=0;i++)
  {
  if(count>cap)
  {
  drop=count-cap;
  count=cap;
  }
  System.out.print(i+1);
  System.out.print("\t\t0");
  mini=min(count,process
  );
  System.out.print("\t\t"+
  mini); count=count-
  mini;
  System.out.print("\t\t"+c
  ount);
  System.out.print("\t\t"+d
  rop);
  System.out.println();
          }
    }
  }
```

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	0

RESULT:

Thus the program to implement the simulation of congestion control algorithm using NS has verified successfully.

Expt No : 7 a	STUDY OF TCP/UDP PEFORMANCE USING SIMULATION TOOL
Date :	

Aim:

To study the performance of TCP/UDP using SImulation Tool

Procedure:

UDP Performance

Step 1: Start network simulator OTCL editor.

Step 2: Create new simulator using set ns [new Simulator] syntax

Step 3: Create procedure to trace all path

```
proc finish {} {
global ns nf tf
 $ns flush-trace
close $nf
close $tf
exec nam tcp.nam &
exit 0}
```

Step 4: Connect with TCP and SINK command.

```
$ns connect $tcp $sink
```

Step 5: Run and Execute the program.

```
$ns run
```

TCP Performance

Step 1: Start network simulator OTCL editor.
Step 2: Create new simulator using set ns [new Simulator] syntax
Step 3: Create Trace route to Network Animator

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

Step 4: Create procedure to trace all

```
path proc finish {} {
global ns nf
$ns flush-trace
#Close the NAM trace file
close $nf
#Execute NAM on the tracefile e
xec nam out.nam &
exit 0   }
```

Step 5: Connect with TCP and SINK command.

```
$ns connect $tcp $sink
```

Step 6: Setup a FTP over TCP

```
connection set ftp
[newApplication/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
```

Step 7: Setup a CBR over UDP

```
set connection cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
```

Step 8: Run and Execute the program.

```
$ns  run
```

Prelab learning Material

UDP performance

- Latency is the time required to transmit a packet across a network:
- Latency may be measured in many different ways: round trip, one way, etc.
- Throughput is defined as the quantity of data being sent/received by unit of time
- Packet loss reflects the number of packets lost per 100 packets sent by a host

TCP performance

- The TCP connection setup handshake
- TCP slow-start congestion control
- Nagle's algorithm for data aggregation
- TCP's delayed acknowledgment algorithm for piggybacked acknowledgments
- TIME_WAIT delays and port exhaustion

Sample Program:

UDP Performance

```
set ns [new Simulator]
set nf [open tcp.nam w]
$ns namtrace-all
$nf set tf [open out.tr w]
$ns trace-all $tf
proc finish {} {
global ns nf tf
 $ns flush-trace
close $nf
close $tf
exec nam tcp.nam & exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns duplex-link $n0 $n4 1Mb 50ms DropTail
$ns duplex-link $n1 $n4 1Mb 50ms DropTail
$ns duplex-link $n2 $n5 1Mb 1ms DropTail
```

```
$ns duplex-link $n3 $n5 1Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 50ms DropTail
$ns duplex-link-op $n4 $n5 queuePos 0.5
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n2 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0.0 "$ftp start"
$ns at 2.5 "$ftp stop"
$ns at 3 "finish"
$ns run
```

TCP Performance

```
 #Create a simulator object
set ns [new Simulator]
 #Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
 #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 NAM trace file close $nf
 #Execute NAM on the trace file exec nam out.nam &
exit 0
}
 #Create four nodes set n0 [$ns node] set n1 [$ns node] set n2 [$ns node] set n3 [$ns node]
 #Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail #Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10 #Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right #Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5 #Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
 #Setup a FTP over TCP connection set ftp [new Application/FTP]
```

```
$ftp attach-agent $tcp
$ftp set type_ FTP #Setup a UDP connection set udp [new Agent/UDP]
$ns attach-agent $n1 $udp set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2
 #Setup a CBR over UDP connection set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false
 #Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"
 #Detach tcp and sink agents (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink" #Call the finish procedure
after 5 seconds of simulation time
$ns at 5.0 "finish"
 #Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts  "CBR     interval  =[$cbr    setinterval_]"
 #Run the simulation
$ns run
```

RESULT:

The program to check the performace of TCP/UDP using simulation tool has been completed sucessfully

Expt No : 7 b	STUDY OF TCP/UDP PERFORMANCE USING CISCO PACKET TRACER
Date :	

Aim:

To study the TCP/UDP performance using Cisco packet tracer.

Algorithm:

Step 1: Configure the IP address and the DNS server for all the clients (Web, Email, FTP and DNS) and the multi-server.

Step 2: In the Email client and the web client, go to mail and configure the email id.

Step 3: In the server, under services, switch on http and edit the index.html file.

Step 4: In the server, under services, switch on DNS and add the domain name and the IP address corresponding to the server.

Step 5: In the server, under services, switch on the SMTP and POP3 services and specify the domain name of the email address. Also, configure the email username and password.

Step 6: In the server, under services, switch on FTP, create a username and password and specify the rights for that user.

Step 7: In the DNS client, using the nslookup command check for the IP address of the server.

Step 8: In the FTP client, using the ftp command, connect to the FTP server.

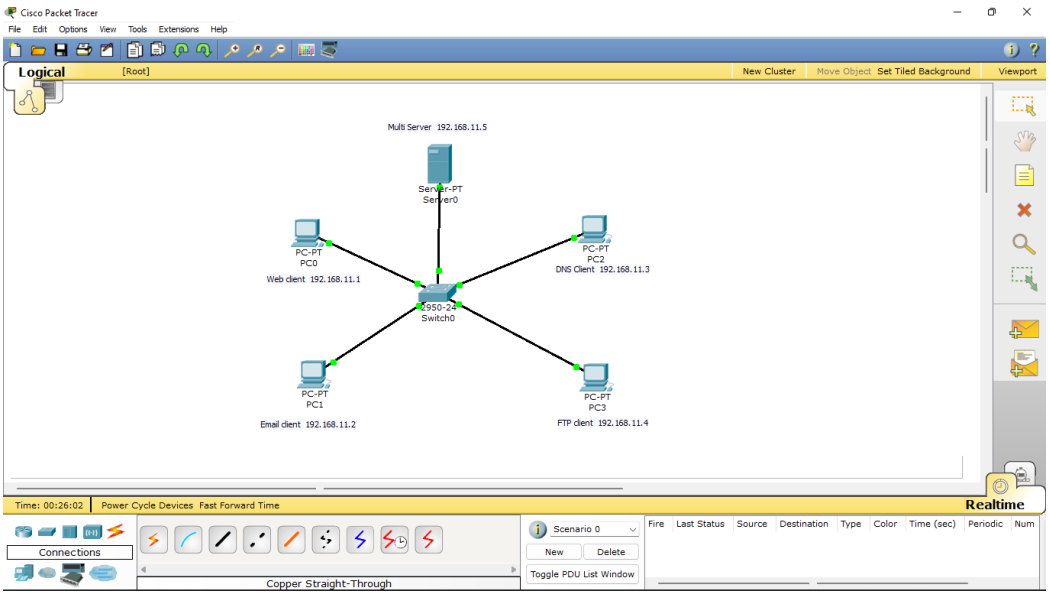
Step 9: In the email client, compose a new email to another email id that is already created and send it.

Step 10: In the web client, login using the other email id and check if the email was received.

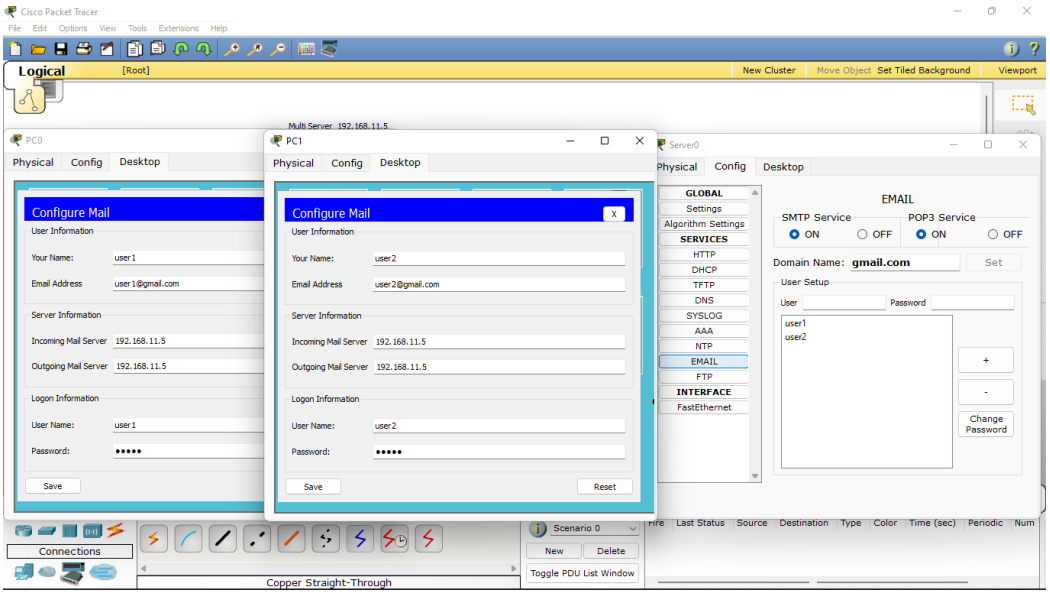
Step 11: In the web client, open the browser and enter the domain name of the webserver and load the web page.

Program &Output:

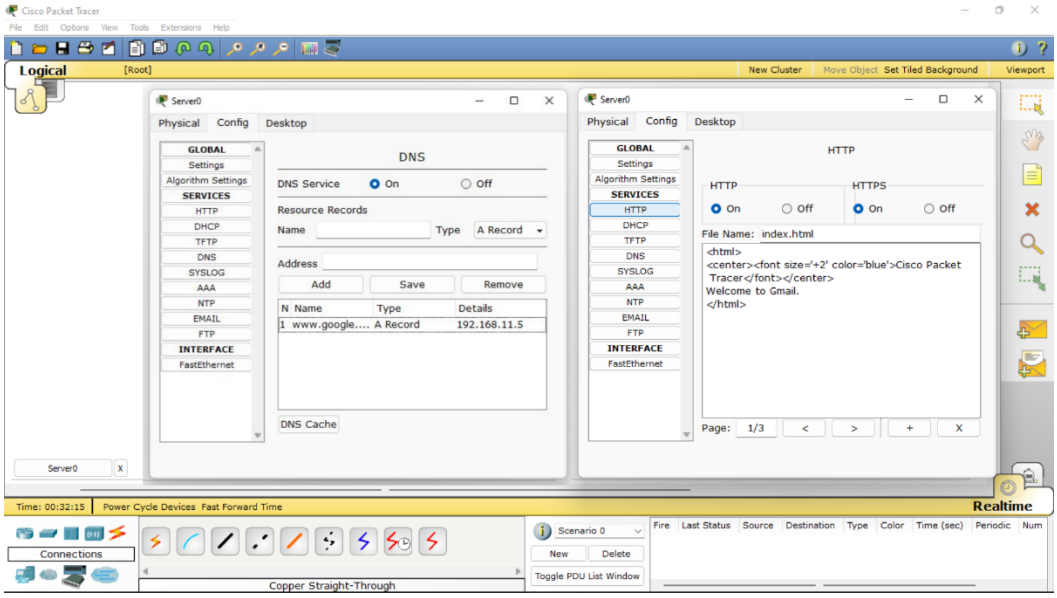
Initial Configuration



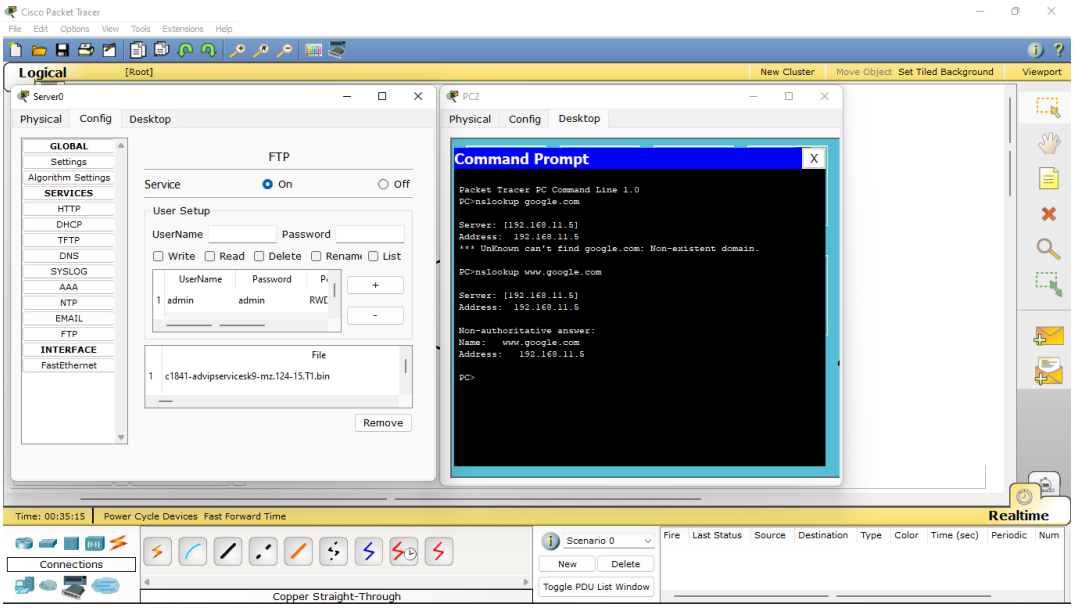
Email configuration in email client, web client and server respectively



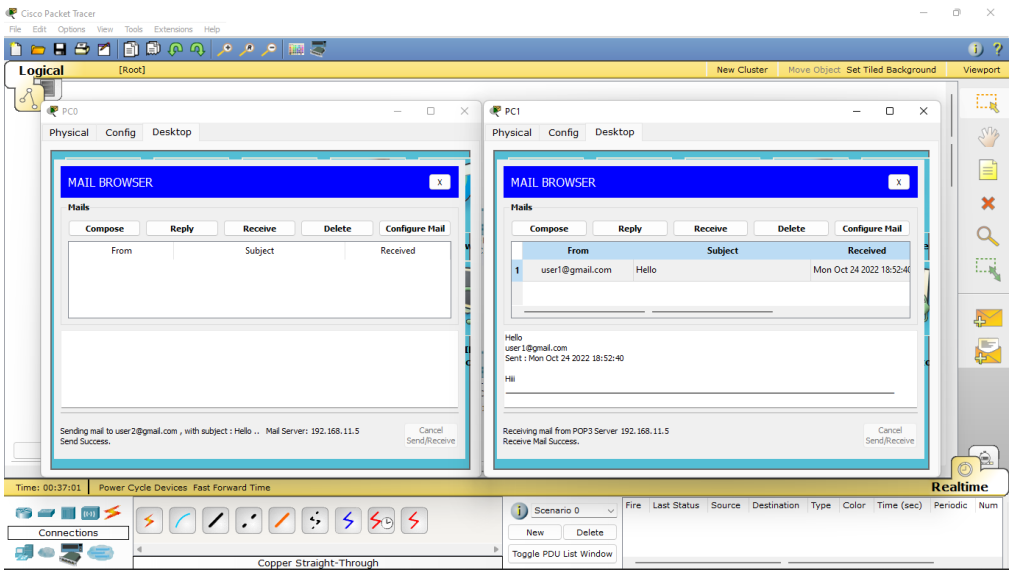
HTTP configuration and DNS configuration in server respectively



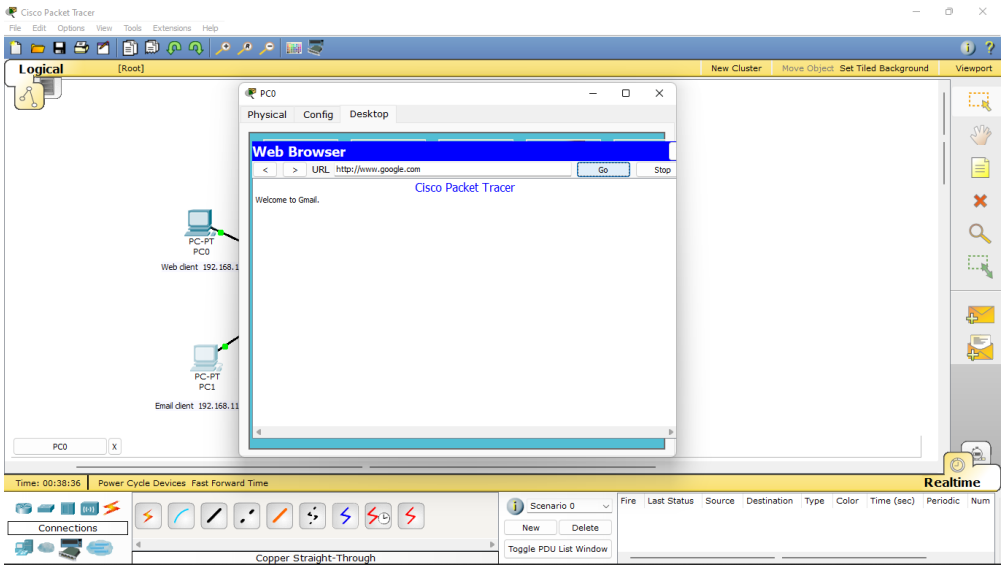
FTP Configuration in server, nslookup in DNS client and ftp login in FTP client

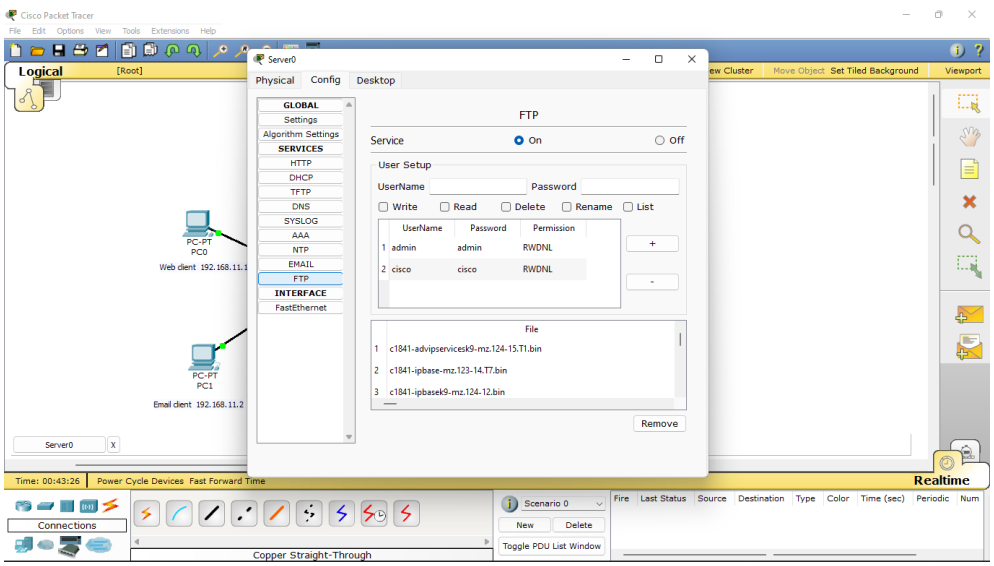


Sending email to web client and receiving email from email client respectively

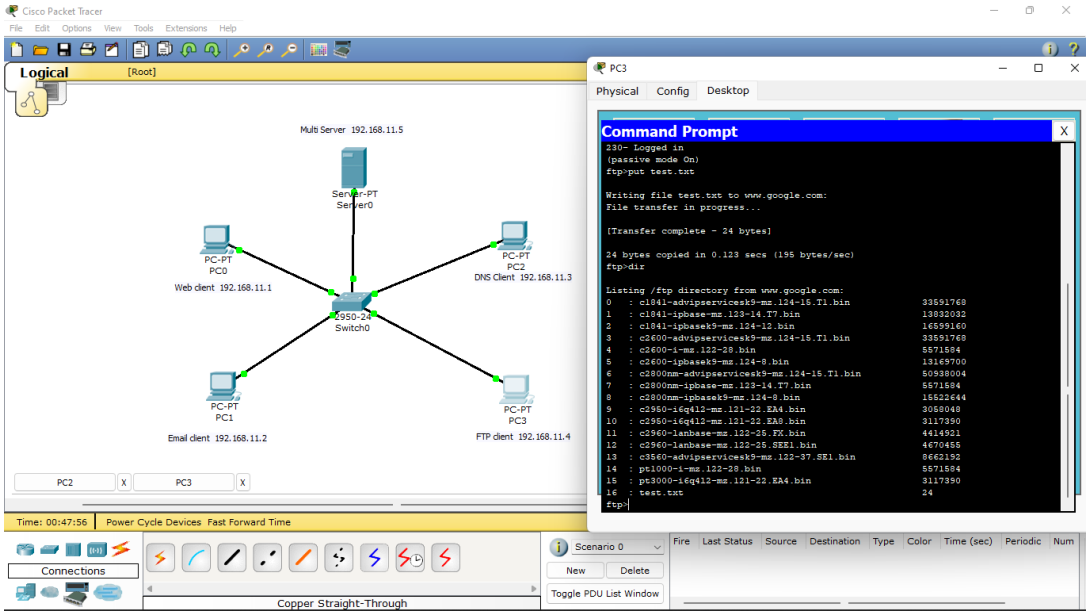


Web page loads in web client

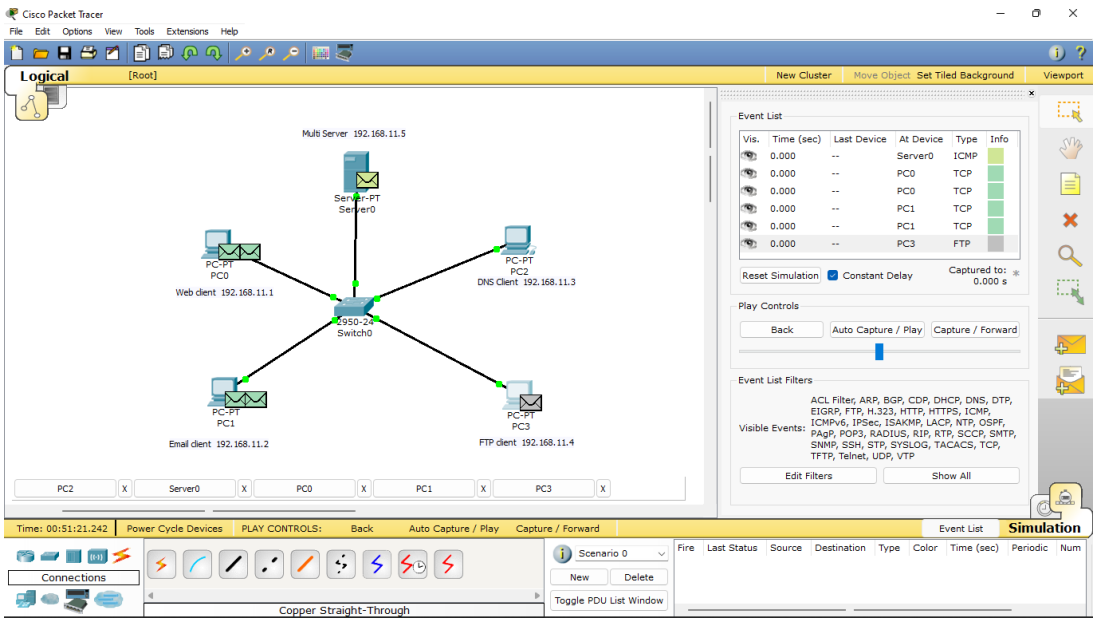


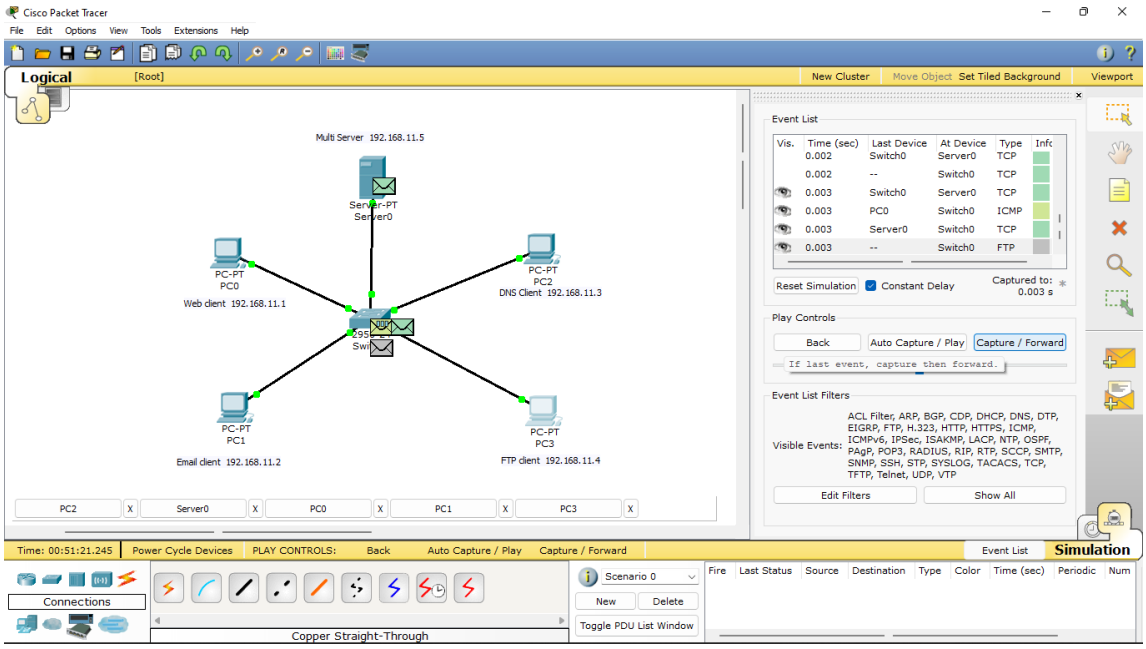


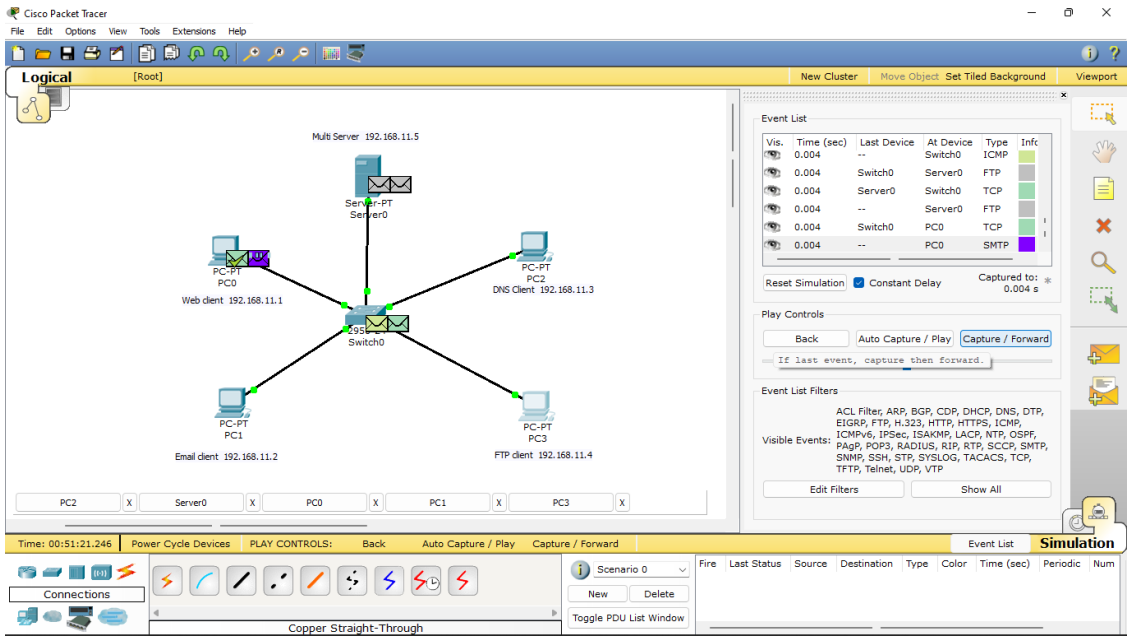
Routes of DNS request

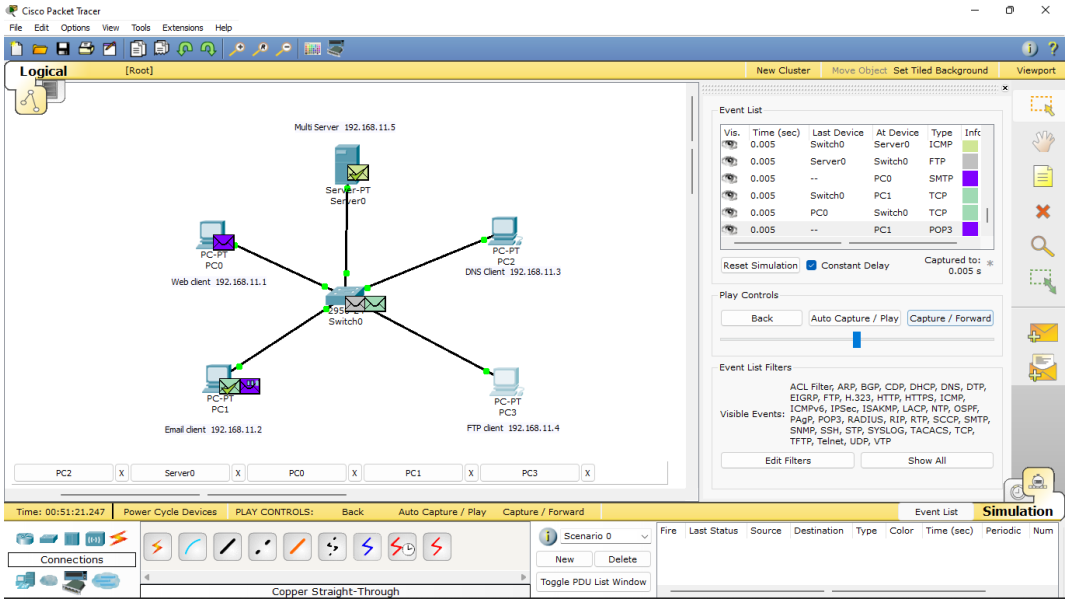


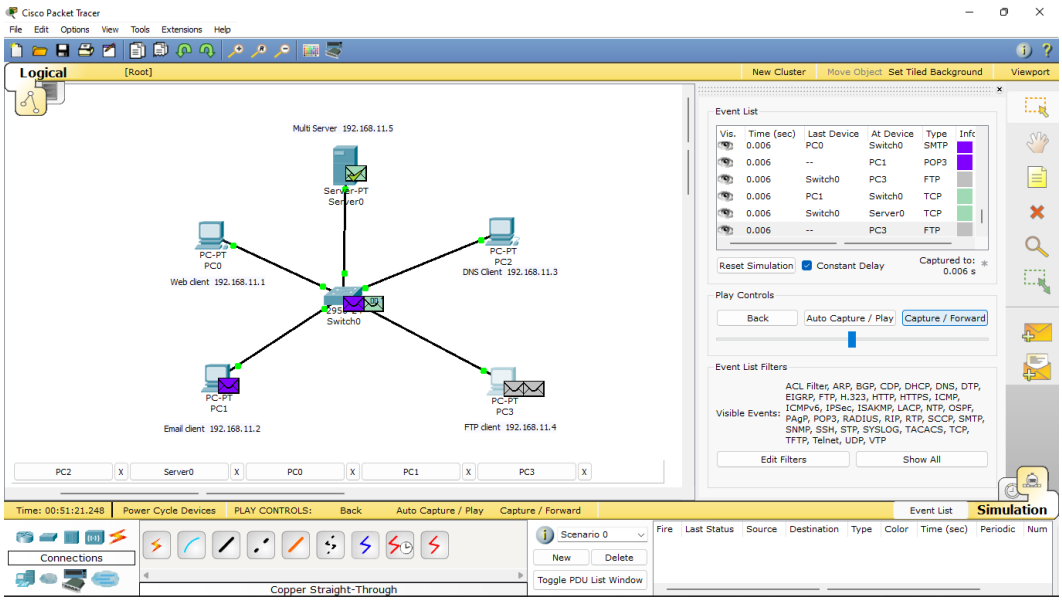
Routes of FTP and Email request




Cisco Packet Tracer
File Edit Options View Tools Extensions Help
Logical [Root]
New Cluster Move Object Set Tiled Background Viewport
Multi Server 192.168.11.5
Server-PT Server0
PC-PT PC0
Web client 192.168.11.1
PC-PT PC2
DNS Client 192.168.11.3
2950-24 Switch0
PC-PT PC1
Email client 192.168.11.2
PC-PT PC3
FTP client 192.168.11.4
PC2 Server0 PC0 PC1 PC3
Event List
Vis. Time (sec) Last Device At Device Type Info
0.002 Switch0 Server0 TCP
0.002 -- Switch0 TCP
0.003 Switch0 Server0 TCP
0.003 PC0 Switch0 ICMP
0.003 Server0 Switch0 TCP
0.003 -- Switch0 FTP
Reset Simulation Constant Delay Captured to: 0.003 s
Play Controls
Back Auto Capture / Play Capture / Forward
If last event, capture then forward.
Event List Filters
Visible Events: ACL Filter, ARP, BGP, CDP, DHCP, DNS, DTP, EIGRP, FTP, H.323, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NTP, OSPF, PAgP, POP3, RADIUS, RIP, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters Show All
Time: 00:51:21.245 Power Cycle Devices PLAY CONTROLS: Back Auto Capture / Play Capture / Forward
Event List Simulation
Connections
Copper Straight-Through
Scenario 0 New Delete Toggle PDU List Window
Fire Last Status Source Destination Type Color Time (sec) Periodic Num


Cisco Packet Tracer
File Edit Options View Tools Extensions Help
Logical [Root]
New Cluster Move Object Set Tiled Background Viewport
Multi Server 192.168.11.5
Server-PT Server0
PC-PT PC0
Web client 192.168.11.1
PC-PT PC2
DNS Client 192.168.11.3
2950-24 Switch0
PC-PT PC1
Email client 192.168.11.2
PC-PT PC3
FTP client 192.168.11.4
PC2 Server0 PC0 PC1 PC3
Event List
Vis. Time (sec) Last Device At Device Type Info
0.004 -- Switch0 ICMP
0.004 Switch0 Server0 FTP
0.004 Server0 Switch0 TCP
0.004 -- Server0 FTP
0.004 Switch0 PC0 TCP
0.004 -- PC0 SMTP
Reset Simulation Constant Delay Captured to: 0.004 s
Play Controls
Back Auto Capture / Play Capture / Forward
If last event, capture then forward.
Event List Filters
Visible Events: ACL Filter, ARP, BGP, CDP, DHCP, DNS, DTP, EIGRP, FTP, H.323, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, LACP, NTP, OSPF, PAgP, POP3, RADIUS, RIP, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, VTP
Edit Filters Show All
Time: 00:51:21.246 Power Cycle Devices PLAY CONTROLS: Back Auto Capture / Play Capture / Forward
Event List Simulation
Connections
Copper Straight-Through
Scenario 0 New Delete Toggle PDU List Window
Fire Last Status Source Destination Type Color Time (sec) Periodic Num





RESULT:

Thus, the performance of TCP and UDP is studied using the packet tracer tool successfully.

Expt No : 8 a	ROUTING ALGORITHMS – DISTANCE VECTOR ROUTING
Date :	

Aim:

To develop a code to simulate Distance Vector Routing for the network graph provided by the user.

Algorithm:

Step 1: Initialize the distance vector table with current values of each of the routers.

Step 2: All the routers reconfigure its path to other routers using the distance vector table sent to it by the other routers.

Step 3: The reconfiguration is based on minimizing the cost to each destination.

Dx(y) = Estimate of least cost from x to y
C (x, v) = Node x knows cost to each neighbor v

Dx = [Dx(y): y ∈ N] = Node x maintains distance vector
Node x also maintains its neighbors' distance vectors
– For each neighbor v, x maintains Dv = [Dv (y): y ∈N]

Step 4: From time-to-time, each node sends its own distance vector estimate to neighbors.

Step 5: When a node x receives new DV estimate from any neighbor v, it saves v's distance vector and it updates its own DV using B-F equation.

Dx(y) = min {C (x, v) + Dv(y), Dx(y)} for each node y ∈ N

Source Code:

```
def bfe(c,intermediate_node,neighbor,no_node):
    flag=0
    for i in neighbor:
        for j in range(no_node):
            if((c[i][intermediate_node]+c[intermediate_node][j])< c[i][j]):
                #Using Bellman Ford Equation
                c[i][j]=c[i][intermediate_node]+c[intermediate_node][j]
                flag=1
        #c[i][i]=0
        if(flag==1):
            bfe(c,i,neighbor_set[i],no_node)
    if(flag==0 and intermediate_node<no_node-1):
        intermediate_node+=1
```

```
        bfe(c,intermediate_node,neighbor_set[intermediate_node],no_node)
    return
no_node=int(input("Enter the number of nodes:"))
c=[[0 for j in range(no_node)] for i in range(no_node)]
for i in range(no_node):
    print("Enter initial cost for node ",i,": ")
    for j in range(no_node):
        c[i][j]=int(input())
print("Initial cost matrix :\n")
for i in range(no_node):
    for j in range(no_node):
        print(c[i][j],end=" ")
    print("\n")
neighbor_set=[[] for i in range(no_node)]
for i in range(no_node):
    for j in range(no_node):
        if(i!=j): #always reachable from its own node
            if(c[i][j]!=1000):
                neighbor_set[i].append(j)
print("Neighbors : \n",neighbor_set,"\n")
bfe(c,0,neighbor_set[0],no_node) #initial call
print("final cost matrix :\n")
for i in range(no_node):
    for j in range(no_node):
        print(c[i][j],end=" ")
    print("\n")
#no_node-number of nodes
#c-cost matrix
#nodeb-nodeeighbors
#1000 considered as infinity
```

Output:

```
4
Enter initial cost for node 0 :
3
4
1000
Enter initial cost for node 1 : 3
0
1
1000
Enter initial cost for node 2 : 4
1
0
5
Enter initial cost for node 3 : 1000
1000
5
0
Initial cost matrix : 0 3 4 1000
3 0 1 1000
4 1 0 5
1000 1000 5 0
Neighbors :
[[1, 2], [0, 2], [0, 1, 3], [2]]
final cost matrix :
0 3 4 9
3 0 1 6
4 1 0 5
9 6 5 0
```

Result:

Thus, the distance vector routing algorithm was successfully implemented for a network graph.

Expt No : 8 b	LINK STATE ROUTING
Date :	

Aim:

To develop a code to simulate Link State Routing for the network graph provided by the

user.

Algorithm:

Step 1: The node is taken and chosen as a root node of the tree, this creates the tree with a single node, and now set the total cost of each node to some value based on the information in Link State Database.

Step 2: Now the node selects one node, among all the nodes not in the tree like structure, which is nearest to the root, and adds this to the tree. The shape of the tree gets changed.

Step 3: After this node is added to the tree, the cost of all the nodes not in the tree needs to be updated because the paths may have been changed.

Step 4: The node repeats the Step 2 and 3 until all the nodes are added in the tree.

Source Code:

```
inf=float("inf")
def neighbour(x):
#finding the neighbours of a given node
neighindex=[]
for j in range(n):
 if(j!=x):
      if(LSDB[x][j]<inf):
 #eg: neighbour of 0th node is [1,3]
      neighindex=neighindex+[j]
 return neighindex
def createTemp():
 temp={}
 # initially a dictionary corresponding to the root node where key as the
 # index of node and values are distance to every other node
 for i in range(len(D)):
 if(i not in tree):
 # temp consists of all the index and distances of those nodes from root that is not present
 # in the tree.
```

```
 temp[i]=D[i]
 return temp
print('Enter the number of nodes: ')
n=int(input()) #total number of nodes
LSDB=[[inf for i in range(n)]for j in range(n)]
for i in range(n):
 print('Enter the distances for',chr(i+65))
 for j in range(n):
 a=input()
 if(a!='inf'):
 LSDB[i][j]=int(a)
print('Enter the node') b=ord(input())-65
tree=[b] #consider the root node as 0->A node
print('Tree => ',tree)
D=[inf for i in range(n)] #distance matrix #List of distances from a given
#node to every other node after Link state routing is applied
parent=[tree[0] for i in range(n)] #parent matrix
#now going to fill the D matrix with available info from LSDB
for j in range(n):
 if(j==tree[0]): #if it is root
 D[j]=0
 elif(LSDB[tree[0]][j]<inf): #if it is neighbour
 D[j]=LSDB[tree[0]][j]
while(True):
 temp=createTemp()
 print("temp=",temp)
 print("D=",D)
mini=min(temp.values())
ind=list(temp.values()).index(mini)
w=list(temp.keys())[ind]
tree=tree+[w]
 print(" w => tree:- ",w,tree)
 for x in neighbour(w): # Updating the distances for all the neighbours of w node
 if(x not in tree):
 if((D[w]+LSDB[w][x])<D[x]):
 D[x]=D[w]+LSDB[w][x]
 parent[x]=w # through the node w the least distance is generated
 if(len(tree)==n):
 break
route=[chr(i+65) for i in range(n)] #path followed from root node to every other node
print('Initially the path matrix: ',route)
for i in range(n):
 ind=i
 while(True):
```

```
 route[i]=route[i]+"-->"+chr(parent[ind]+65) # for G, i==6 #G->F->E->B->A
 ind=parent[ind]
 if(ind==tree[0]): #on reaching the index of root node the loop ends meaning the travel
 #is complete.
 break
print('\nThe Distance of every node from the root node and its path is given by\n')
print("Node\tDistance\tPath")
for i in range(n):
 print(chr(i+65),'\t',D[i],sep="",end="")
 print("\t\t"+route[i])
```

Output:

```
Enter the number of nodes:
4

Enter the distances for A
0
3
5
inf

Enter the distances for B
3
0
1
inf

Enter the distances for C
5
1
0
5
```

```
Enter the distances for D
inf
inf
5
0

Enter the node
A
Tree => [0]
temp= {1: 3, 2: 5, 3: inf}
D= [0, 3, 5, inf]
 w => tree:- 1 [0, 1]
temp= {2: 4, 3: inf}
D= [0, 3, 4, inf]
 w => tree:- 2 [0, 1, 2]
 temp= {3: 9}
 D= [0, 3, 4, 9]
 w => tree:- 3 [0, 1, 2, 3]
 Initially the path matrix: ['A', 'B', 'C', 'D']
 The Distance of every node from the root node and its path is given by Node Distance Path
 A 0 A-->A
 B 3 B-->A
 C 4 C-->B-->A
 D 9 D-->C-->B-->A
```

Result:

Thus, the Link State Routing algorithm was successfully implemented for the network graph provided by the user

Expt No : 9 a	PERFORMANCE EVALUATION OF ROUTING PROTOCOLS USING SIMULATION TOOL
Date :	

Aim:

To evaluate the performance of Routing Performance using Simulation Tool

Procedure:

- In the TCL script, when the user configures AODV as a routing protocol by using the command,$ns node- config -adhocRouting AODV the pointer moves to the “start” and this “start” moves the pointer to the Command function of AODVprotocol.
- In the Command function, the user can find two timers in the “start”
  * btimer.handle((Event*) 0);
  * htimer.handle((Event*) 0);
- Let’s consider the case of htimer, the flow points toHelloTimer::handle(Event*) function and the user can see the following lines:
  agent -> sendHello();
  double interval = MinHelloInterval + ((MaxHelloInterval - Min-HelloInterval) *Random::uniform());
  assert(interval -> = 0);
  Scheduler::instance().schedule(this, &intr, interval);
- These lines are calling the sendHello() function by setting the appropriate interval of Hello Packets
- Now, the pointer is in AODV::sendHello() function and the user can seeScheduler::instance().schedule(target , p, 0.0) which will schedule the packets.
- In the destination node AODV::recv(Packet*p, Handler*) is called, but actually this is done after the node is receiving a packet.
- AODV::recv(Packet*p, Handler*) function then calls therecvAODV(p) function.
- Hence, the flow goes to the AODV::recvAODV(Packet *p) function, which will check different packets types and call the respective function.
- In this example, flow can go to case
  AODVTYPE HELLO:
  recvHello(p);
  break;
- Finally, in the recvHello() function, the packet is received.

Here you can download the following tcl file of aodv protocolsimple-wireless.tcl

Pre-Lab Learning Material:

The reactive on demand routing protocols establish the route to a particular destination only if it is needed. Adhoc on-demand Distance Vector (AODV) is one of the commonly used reactive on demand routing protocols in mobile ad hoc network (MANET). AODV is a reactive enhancement of the DSDV protocol. The route discovery process involves ROUTE REQUEST (RREQ) and ROUTE REPLY (RREP) packets. The source node initiates the route requested through the route discovery process using RREQ packets. The generated route request is forwarded to the neighbors of the source node and this process is repeated till it reaches the destination. On receiving a RREQ packet, an intermediate node with route to destination, it generates a RREP containing the number of hops required to reach the destination.

All intermediate nodes that participates in relaying this reply to the source node creates a forward route to destination. AODV minimizes the number of packets involved in route discovery by establishing routes on-demand.

Sample Exercise:

Evaluate the performance of AODV routing protocol using simulation tool.

Source Code:

```
### Setting The Simulator Objects

      set ns_ [new Simulator]
#create the nam and trace file:
      set tracefd [open aodv.tr w]
      $ns_ trace-all $tracefd

     set namtrace [open aodv.nam w]
      $ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

     set topo [new Topography]
      $topo load_flatgrid $val(x) $val(y)
      create-god $val(nn)
      set chan_1_ [new $val(chan)]

#### Setting The Distance Variables


     # For model 'TwoRayGround'
     set dist(5m) 7.69113e-06 set
     dist(9m) 2.37381e-06 set
     dist(10m) 1.92278e-06 set
     dist(11m) 1.58908e-06 set
```

```
        dist(12m) 1.33527e-06 set
        dist(13m) 1.13774e-06 set
        dist(14m) 9.81011e-07 set
        dist(15m) 8.54570e-07 set
        dist(16m) 7.51087e-07 set
        dist(20m) 4.80696e-07 set
        dist(25m) 3.07645e-07 set
        dist(30m) 2.13643e-07 set
        dist(35m) 1.56962e-07 set
        dist(40m) 1.56962e-10 set
        dist(45m) 1.56962e-11 set
        dist(50m) 1.20174e-13
        Phy/WirelessPhy set CSThresh_ $dist(50m)
        Phy/WirelessPhy set RXThresh_ $dist(50m)

    # Defining Node Configuration
                $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) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON \
                 -movementTrace ON \
                 -channel $chan_1_
### Creating The WIRELESS NODES

        set Server1 [$ns_ node]
        set Server2 [$ns_ node]
        set n2 [$ns_ node]
        set n3 [$ns_ node]
        set n4 [$ns_ node]
        set n5 [$ns_ node]
        set n6 [$ns_ node]
        set n7 [$ns_ node]
        set n8 [$ns_ node]
        set n9 [$ns_ node]
        set n10 [$ns_ node]
        set n11 [$ns_ node]
        set n12 [$ns_ node]
        set n13 [$ns_ node]
        set n14 [$ns_ node]
        set n15 [$ns_ node]
        set n16 [$ns_ node]
        set n17 [$ns_ node]
```

```
set n18 [$ns_ node]
set n19 [$ns_ node]
set n20 [$ns_ node]
set n21 [$ns_ node]
set n22 [$ns_ node]

set opt(seed) 0.1
set a [ns-random $opt(seed)]
set i 0
while {$i < 5} {
incr i
}


### Setting The Initial Positions of Nodes

$Server1 set X_ 513.0
$Server1 set Y_ 517.0
$Server1 set Z_ 0.0

$Server2 set X_ 1445.0
$Server2 set Y_ 474.0
$Server2 set Z_ 0.0

$n2 set X_ 36.0
$n2 set Y_ 529.0
$n2 set Z_ 0.0

$n3 set X_ 143.0
$n3 set Y_ 666.0
$n3 set Z_ 0.0

$n4 set X_ 201.0
$n4 set Y_ 552.0
$n4 set Z_ 0.0

$n5 set X_ 147.0
$n5 set Y_ 403.0
$n5 set Z_ 0.0

$n6 set X_ 230.0
$n6 set Y_ 291.0
$n6 set Z_ 0.0

$n7 set X_ 295.0
$n7 set Y_ 419.0
$n7 set Z_ 0.0
```

```
$n8 set X_ 363.0
 $n8 set Y_ 335.0
 $n8 set Z_ 0.0

$n9 set X_ 334.0
 $n9 set Y_ 647.0
 $n9 set Z_ 0.0

$n10 set X_ 304.0
 $n10 set Y_ 777.0
 $n10 set Z_ 0.0


 $n11 set X_ 412.0
 $n11 set Y_ 194.0
 $n11 set Z_ 0.0


 $n12 set X_ 519.0
 $n12 set Y_ 361.0
 $n12 set Z_ 0.0

$n13 set X_ 569.0
 $n13 set Y_ 167.0
 $n13 set Z_ 0.0

$n14 set X_ 349.0
 $n14 set Y_ 546.0
 $n14 set Z_ 0.0

$n15 set X_ 466.0
 $n15 set Y_ 668.0
 $n15 set Z_ 0.0

$n16 set X_ 489.0
 $n16 set Y_ 794.0
 $n16 set Z_ 0.0

$n17 set X_ 606.0
 $n17 set Y_ 711.0
 $n17 set Z_ 0.0

$n18 set X_ 630.0
 $n18 set Y_ 626.0
 $n18 set Z_ 0.0

$n19 set X_ 666.0
 $n19 set Y_ 347.0
 $n19 set Z_ 0.0

$n20 set X_ 741.0
 $n20 set Y_ 152.0
 $n20 set Z_ 0.0
```

```
$n21 set X_ 882.0
 $n21 set Y_ 264.0
 $n21 set Z_ 0.0

 $n22 set X_ 761.0
 $n22 set Y_ 441.0
 $n22 set Z_ 0.0

 ## Giving Mobility to Nodes

 $ns_ at 0.75 "$n2 setdest 379.0 349.0 20.0"
 $ns_ at 0.75 "$n3 setdest 556.0 302.0 20.0"
 $ns_ at 0.20 "$n4 setdest 309.0 211.0 20.0"
 $ns_ at 1.25 "$n5 setdest 179.0 333.0 20.0"
 $ns_ at 0.75 "$n6 setdest 139.0 63.0 20.0"
 $ns_ at 0.75 "$n7 setdest 320.0 27.0 20.0"
 $ns_ at 1.50 "$n8 setdest 505.0 124.0 20.0"
 $ns_ at 1.25 "$n9 setdest 274.0 487.0 20.0"
 $ns_ at 1.25 "$n10 setdest 494.0 475.0 20.0"
 $ns_ at 1.25 "$n11 setdest 899.0 757.0 25.0"
 $ns_ at 0.50 "$n12 setdest 598.0 728.0 25.0"
 $ns_ at 0.25 "$n13 setdest 551.0 624.0 25.0"
 $ns_ at 1.25 "$n14 setdest 397.0 647.0 25.0"
 $ns_ at 1.25 "$n15 setdest 748.0 688.0 25.0"
 $ns_ at 1.25 "$n16 setdest 842.0 623.0 25.0"
 $ns_ at 1.25 "$n17 setdest 678.0 548.0 25.0"
 $ns_ at 0.75 "$n18 setdest 741.0 809.0 20.0"
 $ns_ at 0.75 "$n19 setdest 437.0 799.0 20.0"
 $ns_ at 0.20 "$n20 setdest 159.0 722.0 20.0"
 $ns_ at 1.25 "$n21 setdest 700.0 350.0 20.0"
 $ns_ at 0.75 "$n22 setdest 839.0 444.0 20.0"

 ## Setting The Node Size

 $ns_ initial_node_pos $Server1 75
 $ns_ initial_node_pos $Server2 75
 $ns_ initial_node_pos $n2 40
 $ns_ initial_node_pos $n3 40
 $ns_ initial_node_pos $n4 40
 $ns_ initial_node_pos $n5 40
 $ns_ initial_node_pos $n6 40
 $ns_ initial_node_pos $n7 40
 $ns_ initial_node_pos $n8 40
 $ns_ initial_node_pos $n9 40
 $ns_ initial_node_pos $n10 40
 $ns_ initial_node_pos $n11 40
 $ns_ initial_node_pos $n12 40
```

```
$ns_ initial_node_pos $n13 40
$ns_ initial_node_pos $n14 40
$ns_ initial_node_pos $n15 40
$ns_ initial_node_pos $n16 40
$ns_ initial_node_pos $n17 40
$ns_ initial_node_pos $n18 40
$ns_ initial_node_pos $n19 40
$ns_ initial_node_pos $n20 40
$ns_ initial_node_pos $n21 40
$ns_ initial_node_pos $n22 40


#### Setting The Labels For Nodes

$ns_ at 0.0 "$Server1 label Server1"
$ns_ at 0.0 "$Server2 label Server2"

#Setting Color For Server

$Server1 color maroon
$ns_ at 0.0 "$Server1 color maroon"

$Server2 color maroon
$ns_ at 0.0 "$Server2 color maroon"

## SETTING ANIMATION RATE


$ns_ at 0.0 "$ns_ set-animation-rate 15.0ms"

  # COLORING THE NODES
$n9 color blue
$ns_ at 4.71 "$n9 color blue"
$n5 color blue
$ns_ at 7.0 "$n5 color blue"
$n2 color blue
$ns_ at 7.29 "$n2 color blue"

$n16 color blue
$ns_ at 7.59 "$n16 color blue"

$n9 color maroon
$ns_ at 7.44 "$n9 color maroon"

$ns_ at 7.43 "$n9 label TTLover"
$ns_ at 7.55 "$n9 label \"\""

$n12 color blue
$ns_ at 7.85 "$n12 color blue"

#### Establishing Communication
```

```
set udp0 [$ns_ create-connection UDP $Server1 LossMonitor $n18 0]
 $udp0 set fid_ 1
 set cbr0 [$udp0 attach-app Traffic/CBR]
 $cbr0 set packetSize_ 1000
 $cbr0 set interval_ .07
 $ns_ at 0.0 "$cbr0 start"
 $ns_ at 4.0 "$cbr0 stop"

 set udp1 [$ns_ create-connection UDP $Server1 LossMonitor $n22 0]
 $udp1 set fid_ 1
 set cbr1 [$udp1 attach-app Traffic/CBR]
 $cbr1 set packetSize_ 1000
 $cbr1 set interval_ .07
 $ns_ at 0.1 "$cbr1 start"
 $ns_ at 4.1 "$cbr1 stop"


 set udp2 [$ns_ create-connection UDP $n21 LossMonitor $n20 0]
 $udp2 set fid_ 1
 set cbr2 [$udp2 attach-app Traffic/CBR]
 $cbr2 set packetSize_ 1000
 $cbr2 set interval_ .07
 $ns_ at 2.4 "$cbr2 start"
 $ns_ at 4.1 "$cbr2 stop"

 set udp3 [$ns_ create-connection UDP $Server1 LossMonitor $n15 0]
 $udp3 set fid_ 1
 set cbr3 [$udp3 attach-app Traffic/CBR]
 $cbr3 set packetSize_ 1000
 $cbr3 set interval_ 5
 $ns_ at 4.0 "$cbr3 start"
 $ns_ at 4.1 "$cbr3 stop"

 set udp4 [$ns_ create-connection UDP $Server1 LossMonitor $n14 0]
 $udp4 set fid_ 1
 set cbr4 [$udp4 attach-app Traffic/CBR]
 $cbr4 set packetSize_ 1000
 $cbr4 set interval_ 5
 $ns_ at 4.0 "$cbr4 start"
 $ns_ at 4.1 "$cbr4 stop"

 set udp5 [$ns_ create-connection UDP $n15 LossMonitor $n16 0]
 $udp5 set fid_ 1
 set cbr5 [$udp5 attach-app Traffic/CBR]
 $cbr5 set packetSize_ 1000
 $cbr5 set interval_ 5
 $ns_ at 4.0 "$cbr5 start"
```

```
$ns_ at 4.1 "$cbr5 stop"

set udp6 [$ns_ create-connection UDP $n15 LossMonitor $n17 0]
$udp6 set fid_ 1
set cbr6 [$udp6 attach-app Traffic/CBR]
$cbr6 set packetSize_ 1000
$cbr6 set interval_ 5
$ns_ at 4.0 "$cbr6 start"
$ns_ at 4.1 "$cbr6 stop"

set udp7 [$ns_ create-connection UDP $n14 LossMonitor $n4 0]
$udp7 set fid_ 1
set cbr7 [$udp7 attach-app Traffic/CBR]
$cbr7 set packetSize_ 1000
$cbr7 set interval_ 5
$ns_ at 4.0 "$cbr7 start"
$ns_ at 4.1 "$cbr7 stop"

set udp8 [$ns_ create-connection UDP $n14 LossMonitor $n9 0]
$udp8 set fid_ 1
set cbr8 [$udp8 attach-app Traffic/CBR]
$cbr8 set packetSize_ 1000
$cbr8 set interval_ 5
$ns_ at 4.0 "$cbr8 start"
$ns_ at 4.1 "$cbr8 stop"

set udp9 [$ns_ create-connection UDP $n4 LossMonitor $n3 0]
$udp9 set fid_ 1
set cbr9 [$udp9 attach-app Traffic/CBR]
$cbr9 set packetSize_ 1000
$cbr9 set interval_ 5
$ns_ at 4.0 "$cbr9 start"
$ns_ at 4.1 "$cbr9 stop"

set udp10 [$ns_ create-connection UDP $n4 LossMonitor $n2 0]
$udp10 set fid_ 1
set cbr10 [$udp10 attach-app Traffic/CBR]
$cbr10 set packetSize_ 1000
$cbr10 set interval_ 5
$ns_ at 4.0 "$cbr10 start"
$ns_ at 4.1 "$cbr10 stop"

set udp11 [$ns_ create-connection UDP $n9 LossMonitor $n16 0]
$udp11 set fid_ 1
set cbr11 [$udp11 attach-app Traffic/CBR]
$cbr11 set packetSize_ 1000
$cbr11 set interval_ 5
```

```
$ns_ at 4.0 "$cbr11 start"
$ns_ at 4.1 "$cbr11 stop"

set udp12 [$ns_ create-connection UDP $n9 LossMonitor $n10 0]
$udp12 set fid_ 1
set cbr12 [$udp12 attach-app Traffic/CBR]
$cbr12 set packetSize_ 1000
$cbr12 set interval_ 5
$ns_ at 4.0 "$cbr12 start"
$ns_ at 4.1 "$cbr12 stop"

#ANNOTATIONS DETAILS

$ns_ at 0.0 "$ns_ trace-annotate \"MOBILE NODE MOVEMENTS\""
$ns_ at 4.1 "$ns_ trace-annotate \"NODE27 CACHE THE DATA FRO SERVER\""
#$ns_ at 4.59 "$ns_ trace-annotate \"PACKET LOSS AT NODE27\""
$ns_ at 4.71 "$ns_ trace-annotate \"NODE10 CACHE THE DATA\""

### PROCEDURE TO STOP

proc stop {} {

        global ns_ tracefd
        $ns_ flush-trace
        close $tracefd
        exec nam datacache.nam &
        exit 0

    }

puts "Starting Simulation....... "
$ns_ at 25.0 "stop"
$ns_ run
```

RESULT:

The program to evaluate the performance of Routing Protocols using Simulation tool .

Expt No : 9 b	RIP CONFIGURATION
Date :	

Aim:

To configure RIP for the topology mentioned using packet tracer tool.

Algorithm:

Step 1: Design the network topology.

Step 2: Enable the serial port in both the Routers

Step 3: In the Router R1, do the following Steps:

a. From global configuration mode enter in (Gigabit) interface mode, assign IP

b. address with subnet mask

c. Invoke the interface using no shutdown

d. From global configuration mode enter in (serial) interface mode, assign IP address

e. with subnet mask

f. d. Invoke the interface using no shutdown

Step 4: Repeat the Step 3 for Router R2 with corresponding IP addresses.

Step 5: Try pinging hosts between the networks separated by the routers

Step 6: To create Forwarding table for each router, follow the Steps:

a. In R1, traverse to global configuration.

b. Enter the commands, R1(config)#router rip

c. R1(config)#version 2

d. R1(config)#network 10.1.1.0

e. R1(config)#network 192.168.10.0

f. R1(config)#show ip route

Step 7: Repeat the Step 6 for Router R2 with corresponding network ids

Step 8: Now try to ping the nodes between the networks

Step 9: To revise / update the forwarding table in Router R1

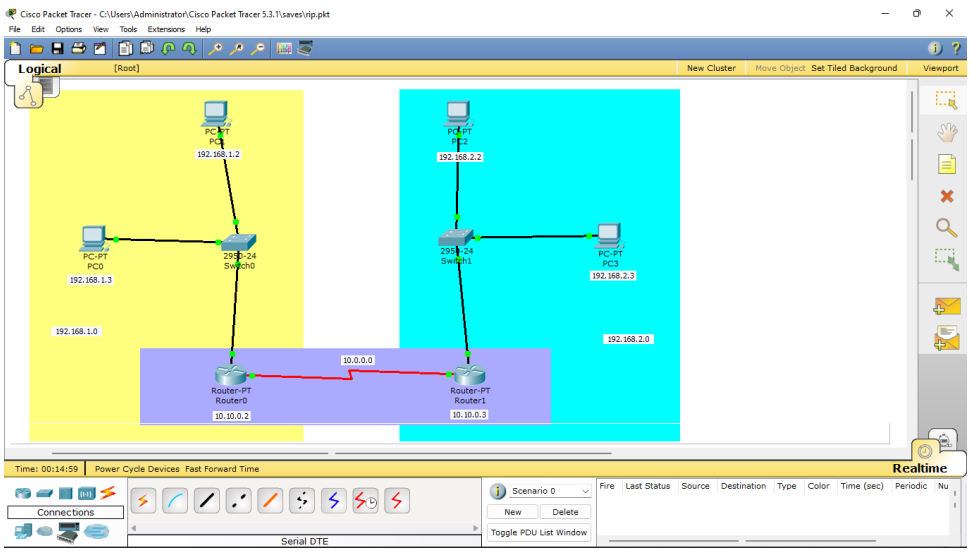
a. R1# debug ip rip

b. R1# no debug all

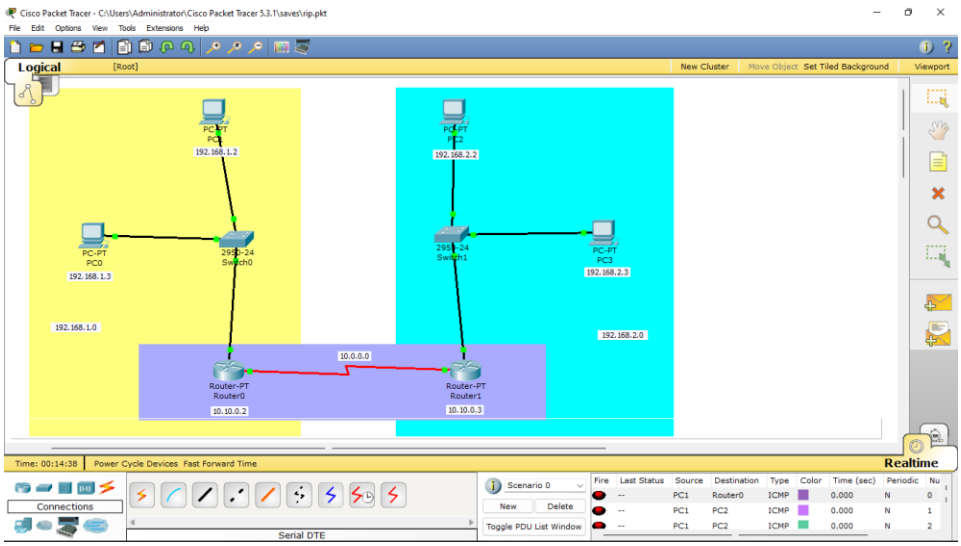
STEP 10: Now RIP has been configured for the networks.

Program & Output:

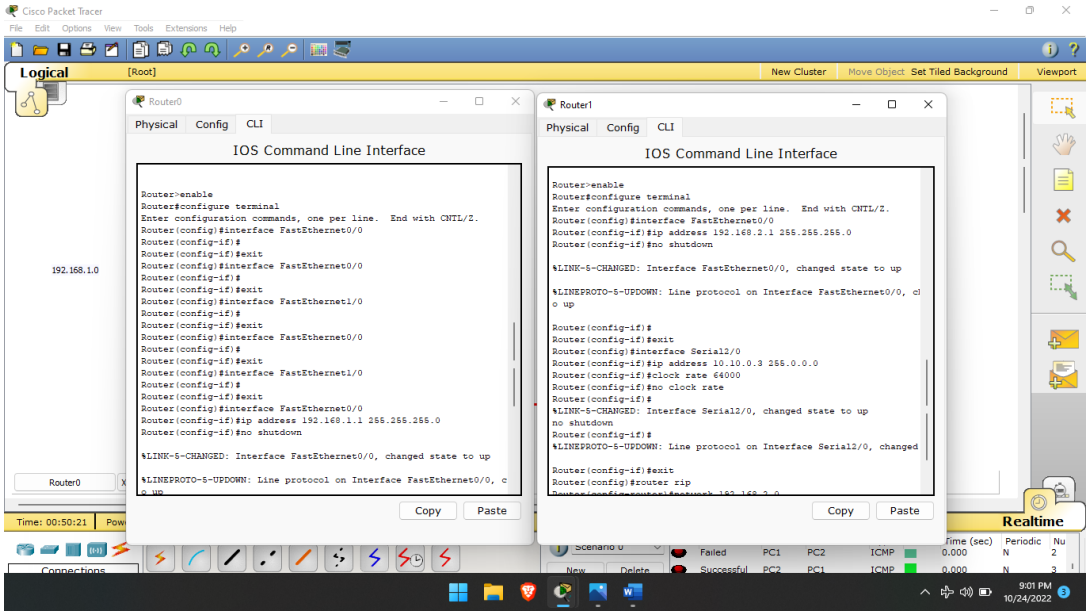
Initial Configuration

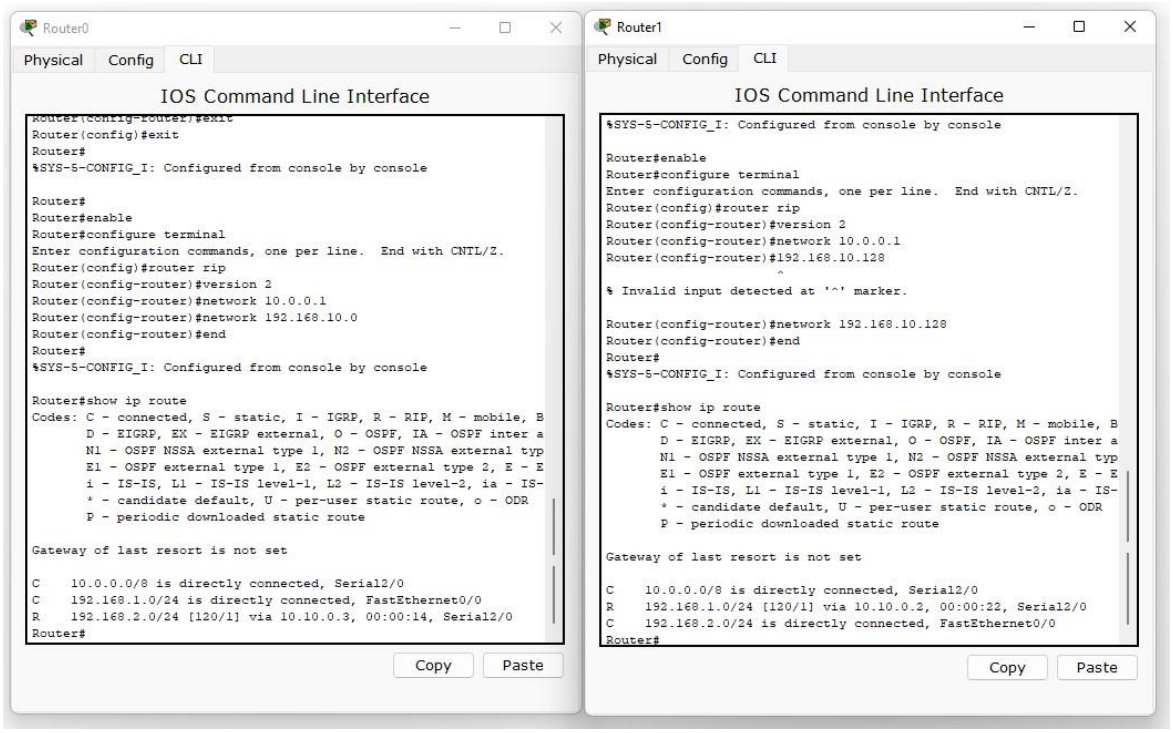


Router 0 and Router 1 interface configuration



Router 0 and Router 1 RIP configuration





Result:

Thus, RIP for the topology mentioned is configured successfully using the packet tracer tool

Expt No : 9 c	OSPF CONFIGURATION
Date :	

Aim:

To configure OSPF for the topology mentioned using the packet tracer tool.

Algorithm:

Step 1: Design the topology as shown above.

Step 2: Enable the serial port in both the Routers

Step 3: In the Router R1, do the following Steps:

a. From global configuration mode enter in (Gigabit) interface mode, assign IP

b. address with subnet mask.

c. Invoke the interface using no shutdown.

d. From global configuration mode enter in (serial) interface mode, assign IP address.

e. with subnet mask.

f. Invoke the interface using no shutdown.

Step 4: Repeat the Step 3 for Router R2 with corresponding ip addresses.

Step 5: Try pinging hosts between the networks separated by the routers

Step 6: To create Forwarding table for each router, follow the Steps:

a. In R1, traverse to global configuration.

b. Enter the commands, R1(config)#router ospf 1.

c. R1(config)#network 10.1.1.0 0.0.0.3 area 0.

d. R1(config)#network 192.168.10.1 0.0.0.127 area 0.

e. R1(config)#show ip route.

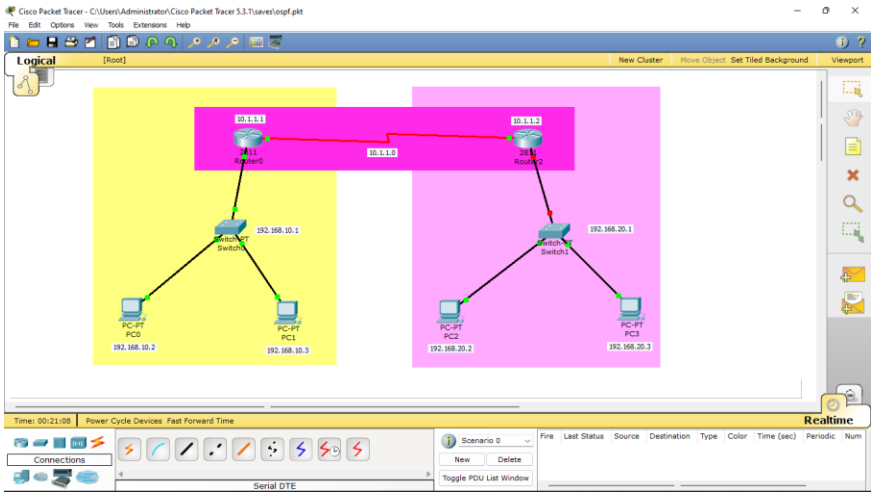
Step 7: Repeat the Step 6 for Router R2 with corresponding network ids

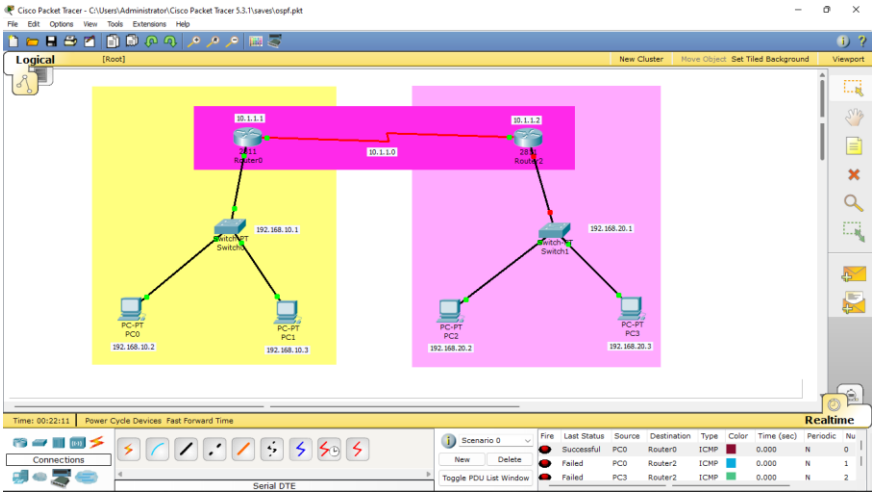
Step 8: Now try to ping the nodes between the networks

Step 9: Now OSPF has been configured for the networks.

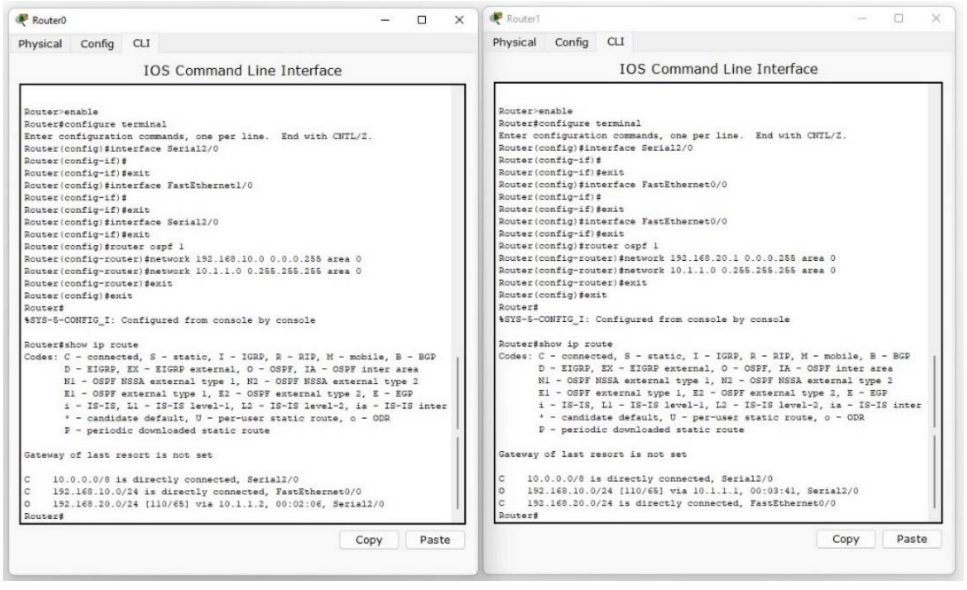
Program & Output:

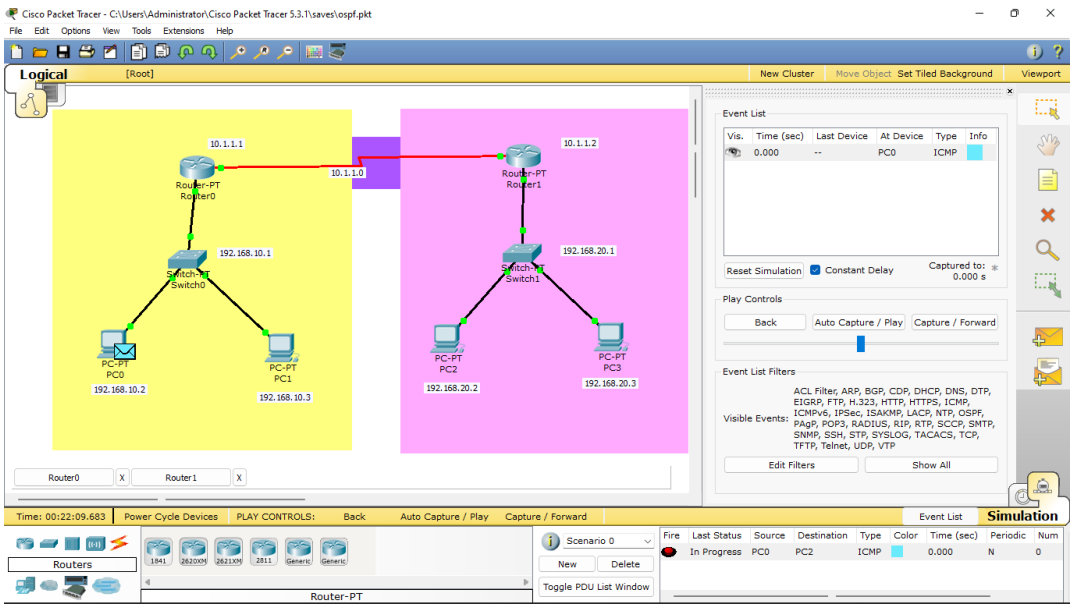
Initial Configuration

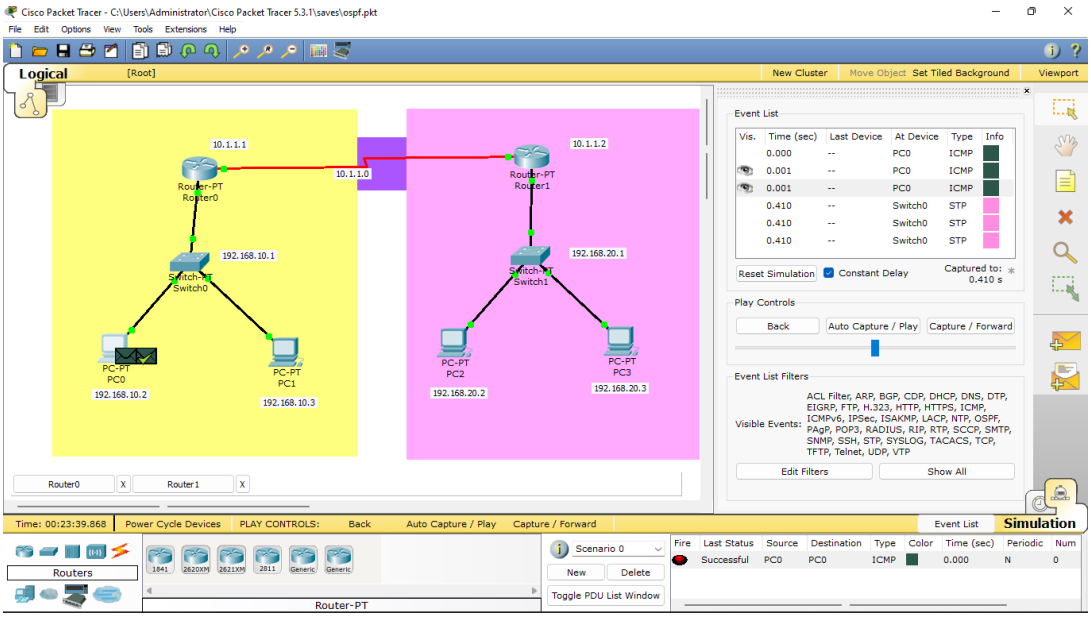




Router 0 and Router 1 interface configuration







RESULT:

Thus, OSPF for the topology mentioned is configured successfully using the packet tracer tool.

Expt No : 10	CYCLIC REDUNDANCY CODE
Date :	

AIM:

To implement the Cyclic Redundancy Check algorithm using java.

Algorithm:

ENCODER:

Step 1: The communicating parties agrees upon the size of message, M(x) and the generator polynomial, G(x).

Step 2: If r is the order of G(x), r bits are appended to the low order end of M(x). This makes the block size bits, the value of which is xrM(x).

Step 3: The block xrM(x) is divided by G(x) using modulo 2 division.

Step 4: The remainder after division is added to xrM(x) using modulo 2 addition. The result is the frame to be transmitted, T(x). The encoding procedure makes exactly divisible by G(x).

DECODER:

Step 1: The receiver divides the incoming data frame T(x) unit by G(x) using modulo 2 division. Mathematically, if E(x) is the error, then modulo 2 division of [M(x)+E(x)] by G(x) is done.

Step 2: If there is no remainder, then it implies that E(x). The data frame is accepted.

Step 3: A remainder indicates a non-zero value of E(x), or in other words presence of an error. So, the data frame is rejected. The receiver may then send an erroneous acknowledgment back to the sender for retransmission.

Source Code:

```
import java.util.*;
class CRC {
	public static void main(String args[]) {
		Scanner scan = new Scanner(System.in);
		int n;
		System.out.println("Enter the size of the data:");
		n = scan.nextInt();
		int data[] = new int[n];
		System.out.println("Enter the data, bit by bit:");
```

```
        for(int i=0 ; i < n ; i++) {
                System.out.println("Enter bit number " + (n-i) + ":");
                data[i] = scan.nextInt();
        }
        System.out.println("Enter the size of the divisor:");
        n = scan.nextInt();
        int divisor[] = new int[n];
        System.out.println("Enter the divisor, bit by bit:");
        for(int i=0 ; i < n ; i++) {
                System.out.println("Enter bit number " + (n-i) + ":");
                divisor[i] = scan.nextInt();
        }
        int remainder[] = divide(data, divisor);
        for(int i=0 ; i < remainder.length-1 ; i++) {
                System.out.print(remainder[i]);
        }
        System.out.println("\nThe CRC code generated is:");
        for(int i=0 ; i < data.length ; i++) {
                System.out.print(data[i]);

        }

        for(int i=0 ; i < remainder.length-1 ; i++) {
                System.out.print(remainder[i]);
        }

        System.out.println();
        int sent_data[] = new int[data.length + remainder.length - 1];
        System.out.println("Enter the data to be sent:");
        for(int i=0 ; i < sent_data.length ; i++) {
                System.out.println("Enter bit number " + (sent_data.length-i)+ ":");
                sent_data[i] = scan.nextInt();
        }
        receive(sent_data, divisor);
}
static int[] divide(int old_data[], int divisor[]) {
        int remainder[] , i;
        int data[] = new int[old_data.length + divisor.length];
```

```
                System.arraycopy(old_data, 0, data, 0, old_data.length);
                remainder = new int[divisor.length];
                System.arraycopy(data, 0, remainder, 0, divisor.length);
                for(i=0 ; i < old_data.length ; i++) {
                        System.out.println((i+1) + ".) First data bit is : "+ remainder[0]);
                        System.out.print("Remainder : ");
                        if(remainder[0] == 1) {
                                // We have to exor the remainder bits with divisor bits
                                for(int j=1 ; j < divisor.length ; j++) {
                                        remainder[j-1] = exor(remainder[j], divisor[j]);
                                        System.out.print(remainder[j-1]);
                                }
                        }
                        else {
                                        for(int j=1 ; j < divisor.length ; j++) {
                                        remainder[j-1] = exor(remainder[j], 0);
                                        System.out.print(remainder[j-1]);
                                } }
                        remainder[divisor.length-1] = data[i+divisor.length];
                        System.out.println(remainder[divisor.length-1]);
                }
                return remainder;
        }
        static int exor(int a, int b) {
                if(a == b) {
                        return 0;
                }
                return 1;
        }
        static void receive(int data[], int divisor[]) {
                int remainder[] = divide(data, divisor);
for(int i=0 ; i < remainder.length ; i++) {
                if(remainder[i] != 0) {
                                System.out.println("There is an error in received data...");
                                return;
                        }
                }
                System.out.println("Data was received without any error.");
        }
```

Output:

```
Enter the size of the data:
7
Enter the data, bit by bit:
Enter bit number 7:  1
Enter bit number 6:  0
Enter bit number 5:  0
Enter bit number 4:  1
Enter bit number 3:  1
Enter bit number 2:  0
Enter bit number 1:  1
Enter the size of the divisor:
Enter the divisor, bit by bit:

Enter bit number 4:  1
Enter bit number 3:  0
Enter bit number 2:  1
Enter bit number 1:  1

1.) First data bit is :  1          Remainder : 0101
2.) First data bit is :  0          Remainder : 1010
3.) First data bit is :  1          Remainder : 0011
4.) First data bit is :  0          Remainder : 0110
5.) First data bit is :  0          Remainder : 1100
6.) First data bit is :  1          Remainder : 1110
7.) First data bit is :  1          Remainder : 1010

101
The CRC code generated is: 1001101101
Enter the data to be sent: Enter bit number 10: 1
Enter bit number 9: 0
Enter bit number 8: 0
Enter bit number 7: 1
Enter bit number 6: 1
Enter bit number 5: 0
Enter bit number 4: 1
Enter bit number 3: 1
```

Enter bit number 2: 0
Enter bit number 1: 1

1.) First data bit is : 1 Remainder : 0101
2.) First data bit is : 0 Remainder : 1010
3.) First data bit is : 1 Remainder : 0011
4.) First data bit is : 0 Remainder : 0111
5.) First data bit is : 0 Remainder : 1110
6.) First data bit is : 1 Remainder : 1011
7.) First data bit is : 1 Remainder : 0000
8.) First data bit is : 0 Remainder : 0000
9.) First data bit is : 0 Remainder : 0000
10.) First data bit is : 0 Remainder : 0000

Data was received without any error.

Result:

Thus, the Cyclic Redundancy Check algorithm is implemented successfully using python sockets