191AIC404J – COMPUTER NETWORKS LAB MANUAL

SYLLABUS

1. Study of Network Components, Basic Network Commands and Network Configuration Commands.
2. Python Program for Chat Program using TCP Socket.
3. Python program for Sliding Window Protocol using TCP Sockets.
4. Python Program for DNS using UDP Sockets.
5. Implement Routing Protocol using Cisco Packet tracer simulator tool.
6. Performance comparison of TCP and UDP protocols using Cisco Packet Tracer Simulator tool.

1. Study of Network Components, Basic Network Commands and Network Configuration Commands.

Aim:

To learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute. Capture ping and traceroute PDUs using a network protocol analyzer and examine.

TCPDUMP:

Tcpdump command is a famous network packet analysing tool that is used to display TCP\IP & other network packets being transmitted over the network attached to the system on which tcpdump have been installed. Tcpdump uses libpcap library to capture the network packets & is available on almost all Linux/Unix flavors.

Tcpdump command can read the contents from a network interface or from a previously created packet file or we can also write the packets to a file to be used for later. One must use the tcpdump command as root or as a user with sudo privileges.

INSTALLATION:

a. Fedora

On Fedora, install tcpdump using the following command,**$ dnf install tcpdump**

b. Ubuntu/Debian/Linux Mint

On Ubuntu or Debian or Linux Mint, install tcp dumo using the following command,

```
$ apt-get install tcpdump
```

EXAMPLES:

1. **Get packets from all interfaces**

   To get the network packets from all network interfaces, run the following command,

   ```
   $ tcpdump -i any
   ```

2. **Get packets from a single interfaces**

   To get the network packets from a single interface, use

```
$ tcpdump -i eth0
```

3. **Writing captured packets to file**

   To write all the captured packets to a file, use the ‘-w’ option,

```
$ tcpdump -i eth1 -w packets_file
```

4. **Reading an old tcpdump file**

   To read an already created, old tcpdump file, use the following command,

```
$ tcpdump -r packets_file
```

5. **Getting more packets information with readable timestamps**

To get more information regarding the packets along with readable timestamp, use **$ tcpdump -ttttnnvvS**

6. **Check packets of whole network**

   To get the packets for whole network, execute the following command from terminal

```
$ tcpdump net 192.168.1.0/24
```

7. **Check packets based on IP address**

   Get all the packets based on the IP address, whether source or destination or both, using the following command,

```
$ tcpdump host 192.168.1.100
```

   To get packets based on source or destination of an IP address, use

```
$ tcpdump src 192.168.1.100
```

```
$ tcpdump dst 192.168.1.100
```

8. **Check packets for a protocol or port number**

   To check all the packets used based on the protocol, run the following command

```
$ tcpdump ssh
```

   To get packets for a single port ot for a range of ports, use

```
$ tcpdump port 22
```

```
$ tcpdump portrange 22-125
```

We can also use **‘src’** & **‘dst’** options to get packets for ports based on source & destination.We can also combine two conditions with AND (and , &&), OR (or. ||) & EXCEPT (not , !). This helps when we have analyze network packets based on the some conditions.

9. **Using AND:**

   We can use ‘and’ or symbol ‘&&’ to combine two conditions or mote with tcpdump. An example would be,

```
$ tcpdump src 192.168.1.100 && port 22 -w ssh_packets
```

10. **Using OR**

    OR will check the command agtcpdump -i eth0 src port not 22ainst one the mentioned conditions in the command, like

```
$ tcpdump src 192.168.1.100 or dst 192.168.1.50 && port 22 -w ssh_packets
```

```
$ tcpdump port 443 or 80 -w http_packets
```

11. **Using EXCEPT**

    EXCEPT will be used when we want not fulfill a condition, like

```
$ tcpdump -i eth0 src port not 22
```

This will monitor all the traffic on eth0 but will not capture port 22.

NETSTAT COMMAND:

Netstat is a command line utility that tells us about all the tcp/udp/unix socket connections on our system. It provides list of all connections that are currently

established or are in waiting state. This tool is extremely useful in identifying the port numbers on which an application is working and we can also make sure if an application is working or not on the port it is supposed to work.Netstat command also displays various other network related information such as routing tables, interface statistics, masquerade connections, multicast memberships etc.,

1- Checking all connections

To list out all the connections on a system, we can use ‘a’ option with netstat command,

$ netstat -a
This will produce all tcp, udp & unix connections from the system.

2- Checking all tcp or udp or unix socket connections

To list only the tcp connections our system, use ‘t’ options with netstat,

$ netstat -at

Similarly to list out only the udp connections on our system, we can use ‘u’ option with netstat,

$ netstat -au
To only list out Unix socket connections, we can use ‘x’ options,

$ netstat -ax

3- List process id/Process Name with

To get list of all connections along with PID or process name, we can use ‘p’ option & it can be used in combination with any other netstat option,

$ netstat -ap
4- List only port number & not the name

To speed up our output, we can use ‘n’ option as it will perform any reverse lookup & produce output with only numbers. Since no lookup is performed, our output will much faster.

$ netstat -an
5- Print only listening ports

To print only the listening ports , we will use ‘l’ option with netstat. It will not be used with ‘a’ as it prints all ports,

$ netstat -l

6- Print network stats

To print network statistics of each protocol like packet received or transmitted, we can use ‘s’ options with netstat,

$ netstat -s

7- Print interfaces stats

To display only the statistics on network interfaces, use ‘I’ option,

$ netstat -i

8-Display multicast group information

With option ‘g’ , we can print the multicast group information for IPV4 & IPV6,

$ netstat -g

9- Display the network routing information

To print the network routing information, use ‘r’ option,

$ netstat -r

10- Continuous output

To get continuous output of netstat, use ‘c’ option

$ netstat -c

11- Filtering a single port

To filter a single port connections, we can combine ‘grep’ command with netstat,

$ netstat -anp | grep 3306

12- Count number of connections

To count the number of connections from port, we can further add ‘wc’ command with netstat & grep command,

$ netstat -anp | grep 3306 | wc -l

This will print the number of connections for the port mysql port i.e. 3306.

ifconfig command : Learn with some examples

BY SHUSAIN · PUBLISHED SEPTEMBER 8, 2017 · UPDATED MARCH 12, 2018

‘Ifconfig’ , short for ‘Interface Configuration, is a very important utility of Linux systems. It is used to check & configure network interfaces using the terminal or CLI of Linux machines. With ifconfig command we can check information regarding the network interfaces, we can configure them or we can also enable and disable them.Any person using a Linux distro should know how to use ifconfig, especially if the distro you are using is CLI based.

Check information regarding Network Interfaces

To check complete information regarding the network interfaces, open the terminal & execute

$ ifconfig

OR

$ ifconfig -a

Only difference between the two mentioned command is that , while ‘ifconfig’ shows information regarding all the active interfaces ; ‘ifconfig -a’ shows information of all the interfaces including interfaces that are currently not active also.

Check information regarding a particular Network Interface

We might have a number of network interfaces with names like ‘eth0’ , ‘eth1’, ‘wlan0’ etc. So if we want to see information regarding a particular network interface, then run

$ ifconfig eth0

Enabling or disabling an interface

With ifconfig, we can enable or disable a particular interface & the way we do it is

$ ifconfig eth0 up

This is to bring UP an interface, for disabling an interface the command is

$ ifconfig eth0 down

Remember, you will need sudo privileges to perform this acction.

(Recommended Read: Granting SUDO access to a local user account)

Configuring a network interfaces

We can also configure the network interfaces using the ifconfig command i.e. we can assign an IP address, netmask & broadcast address using ifconfig.

To assign an IP address to interface using ifconfig, the command is

$ ifconfig eth0 192.168.5.45
Next we will assign the netmask for assigned IP address & command to do it is

$ ifconfig netmask 255.255.240.0
astly to assign a broadcast address, command is

$ ifconfig broadcast 192.168.5.255
We can also combine all these command into a single command to configure all the network information in one go. We can however can't configure gateway & DNS using ifconfig command.

To assign a gateway, run the following command,

$ route add default gw 192.168.5.10
& to add DNS, execute

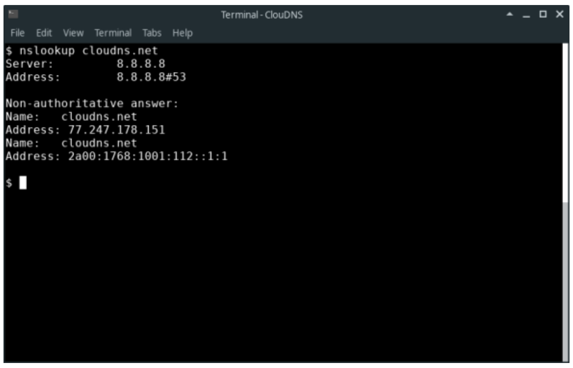
$ echo "nameserver 8.8.8.8" > /etc/resolv.conf
Creating an alias or assigning multiple IP to an interface

We can also assign multiple IP addresses to a single interface using ifconfig, this process is also known as IP aliasing. To accomplish this, we need to execute the following command,
$ ifconfig eth0:1 192.168.5.50 netmask 255.255.240.0 broadcast 192.168.5.255& if we want to remove the IP address to an alias, we can simply bring down the interface using
$ ifconfig eth0:1 down

We have also discussed IP aliasing in a more detail in our tutorial **ASSIGNING MULTIPLE IP TO A SINGLE NIC.**
Nslookup commands

Nslookup – it is a powerful network administration command-line tool, available for many of the popular computer operating systems for querying Domain Name System (DNS) to obtain domain names or IP addresses, mapping or for any other specific DNS Records.

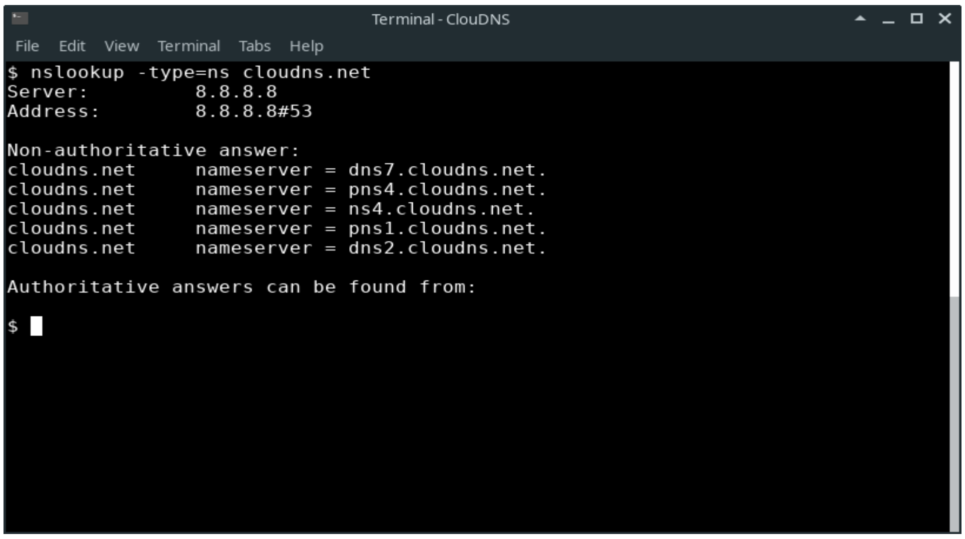
1. How to find the A record of a domain.
Command line:
$ nslookup example.com



```
$ nslookup cloudns.net
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   cloudns.net
Address: 77.247.178.151
Name:   cloudns.net
Address: 2a00:1768:1001:112::1:1

$ 
```

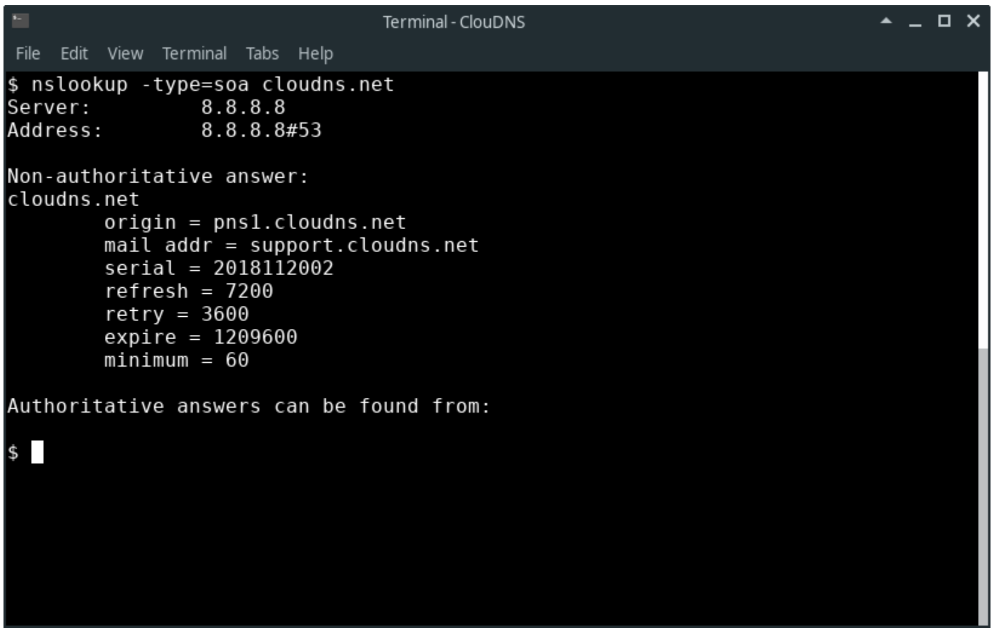
2. How to check the NS records of a domain.

Command line:
$nslookup -type=ns example.com



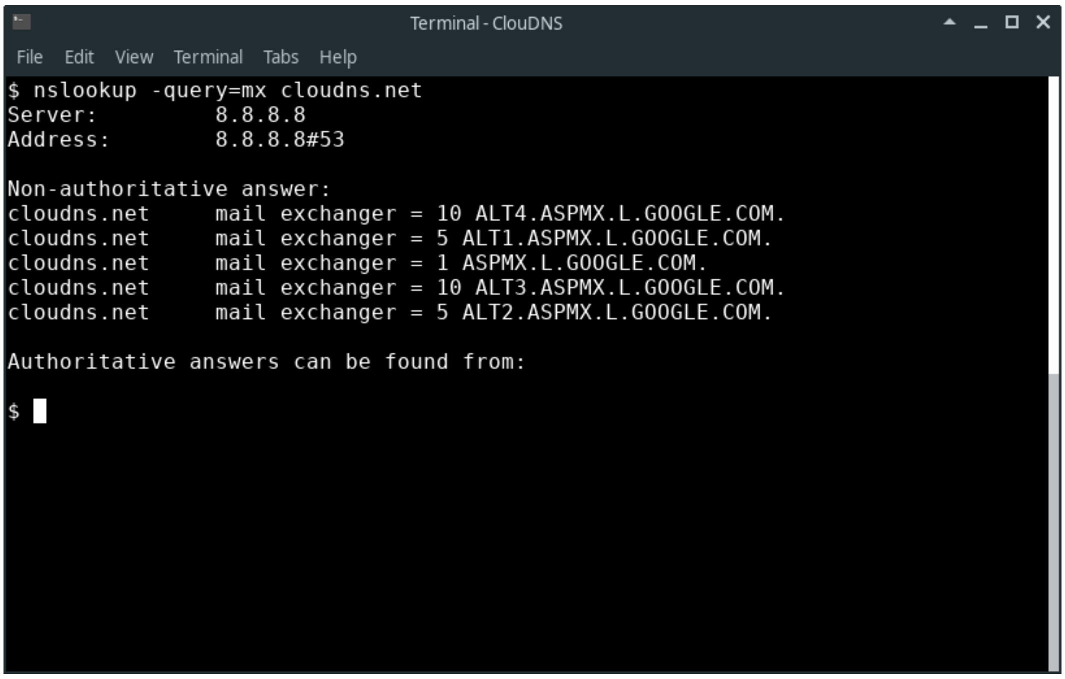
```
$ nslookup -type=ns cloudns.net
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
cloudns.net     nameserver = dns7.cloudns.net.
cloudns.net     nameserver = pns4.cloudns.net.
cloudns.net     nameserver = ns4.cloudns.net.
cloudns.net     nameserver = pns1.cloudns.net.
cloudns.net     nameserver = dns2.cloudns.net.

Authoritative answers can be found from:

$
```

3. How to query the SOA record of a domain.
Command line:
$nslookup -type=soa example.com



```
$ nslookup -type=soa cloudns.net
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
cloudns.net
        origin = pns1.cloudns.net
        mail addr = support.cloudns.net
        serial = 2018112002
        refresh = 7200
        retry = 3600
        expire = 1209600
        minimum = 60

Authoritative answers can be found from:

$
```

4. How to find the MX records responsible for the email exchange.
Command line:
$ nslookup -query=mx example.com

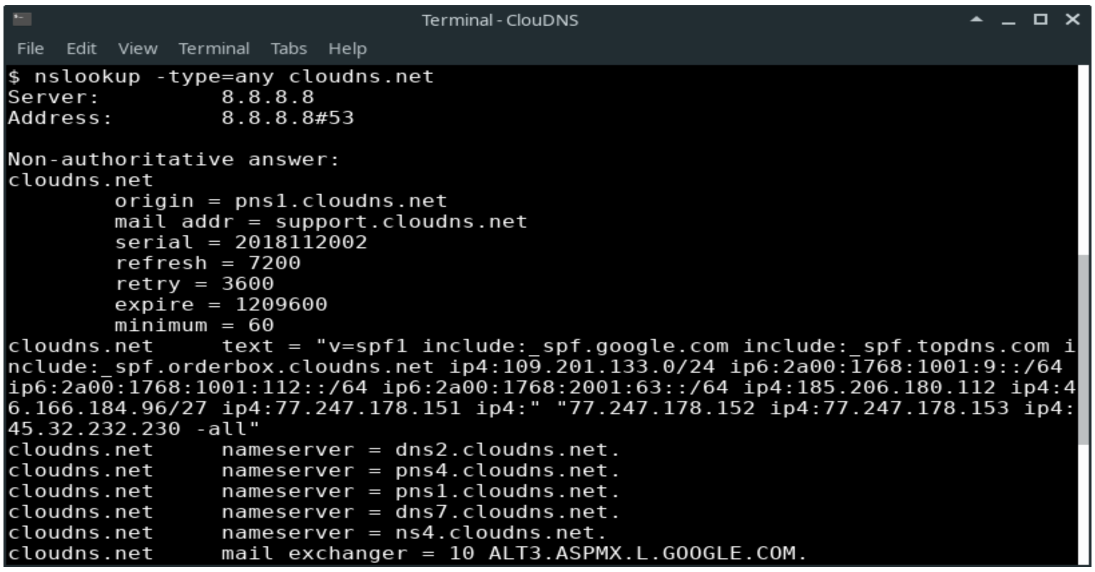


```
$ nslookup -query=mx cloudns.net
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
cloudns.net     mail exchanger = 10 ALT4.ASPMX.L.GOOGLE.COM.
cloudns.net     mail exchanger = 5 ALT1.ASPMX.L.GOOGLE.COM.
cloudns.net     mail exchanger = 1 ASPMX.L.GOOGLE.COM.
cloudns.net     mail exchanger = 10 ALT3.ASPMX.L.GOOGLE.COM.
cloudns.net     mail exchanger = 5 ALT2.ASPMX.L.GOOGLE.COM.

Authoritative answers can be found from:

$ 
```

5. How to find all of the available DNS records of a domain.

Command line:

$ nslookup -type=any example.com

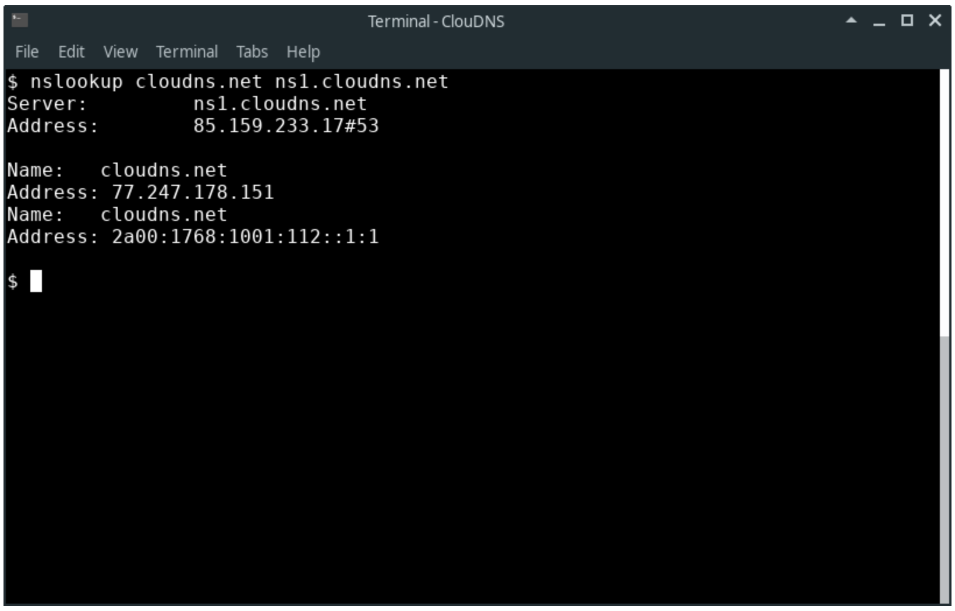


```
$ nslookup -type=any cloudns.net
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
cloudns.net
        origin = pns1.cloudns.net
        mail addr = support.cloudns.net
        serial = 2018112002
        refresh = 7200
        retry = 3600
        expire = 1209600
        minimum = 60
cloudns.net     text = "v=spf1 include:_spf.google.com include:_spf.topdns.com i
nclude:_spf.orderbox.cloudns.net ip4:109.201.133.0/24 ip6:2a00:1768:1001:9::/64
ip6:2a00:1768:1001:112::/64 ip6:2a00:1768:2001:63::/64 ip4:185.206.180.112 ip4:4
6.166.184.96/27 ip4:77.247.178.151 ip4:" "77.247.178.152 ip4:77.247.178.153 ip4:
45.32.232.230 -all"
cloudns.net     nameserver = dns2.cloudns.net.
cloudns.net     nameserver = pns4.cloudns.net.
cloudns.net     nameserver = pns1.cloudns.net.
cloudns.net     nameserver = dns7.cloudns.net.
cloudns.net     nameserver = ns4.cloudns.net.
cloudns.net     mail exchanger = 10 ALT3.ASPMX.L.GOOGLE.COM.
```

6. How to check the using of a specific DNS Server.

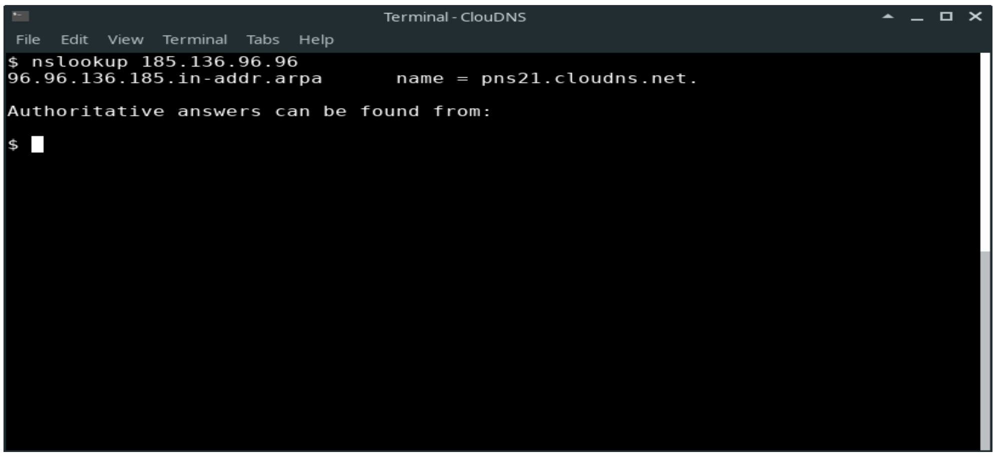
Command line:

$ nslookup example.com ns1.nsexample.com



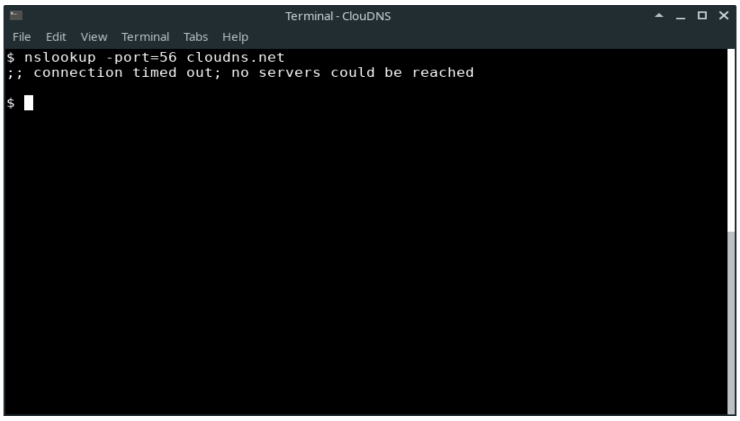
```
$ nslookup cloudns.net ns1.cloudns.net
Server:         ns1.cloudns.net
Address:        85.159.233.17#53

Name:   cloudns.net
Address: 77.247.178.151
Name:   cloudns.net
Address: 2a00:1768:1001:112::1:1

$ 
```

7. How to check the Reverse DNS Lookup.
Command line:
$ nslookup 10.20.30.40



```
$ nslookup 185.136.96.96
96.96.136.185.in-addr.arpa      name = pns21.cloudns.net.

Authoritative answers can be found from:

$ 
```

8. How to change the port number for the connection.
Command line:
$ nslookup -port=56 example.com

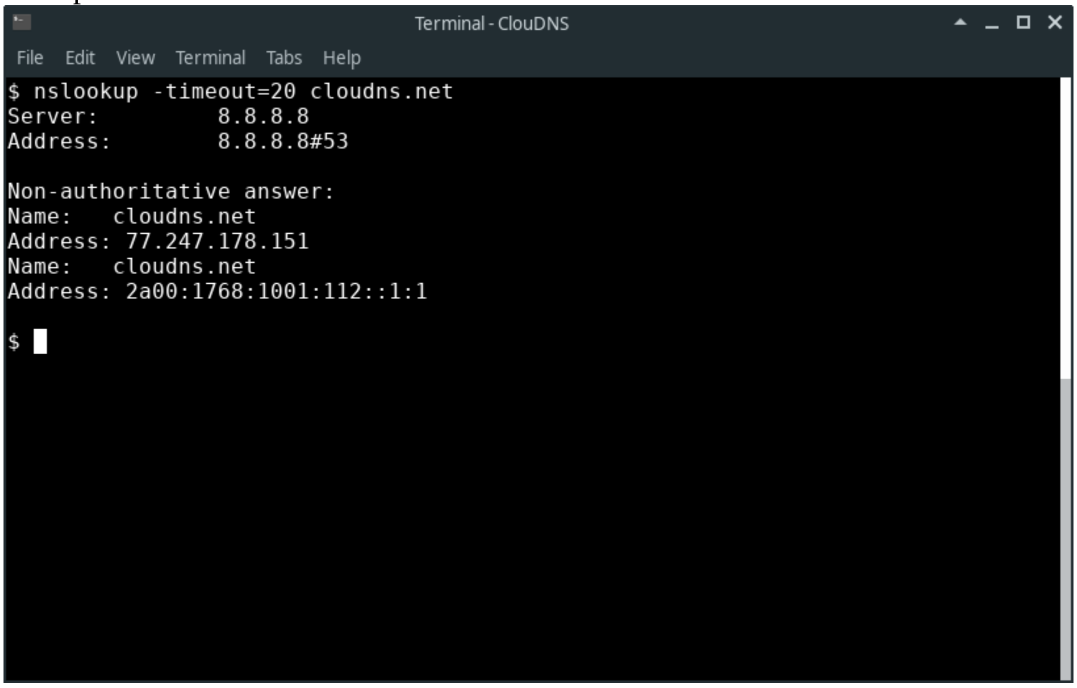


```
$ nslookup -port=56 cloudns.net
;; connection timed out; no servers could be reached

$ 
```

9.How to change the timeout interval for a reply.

Command line:

$ nslookup -timeout=20
example.com



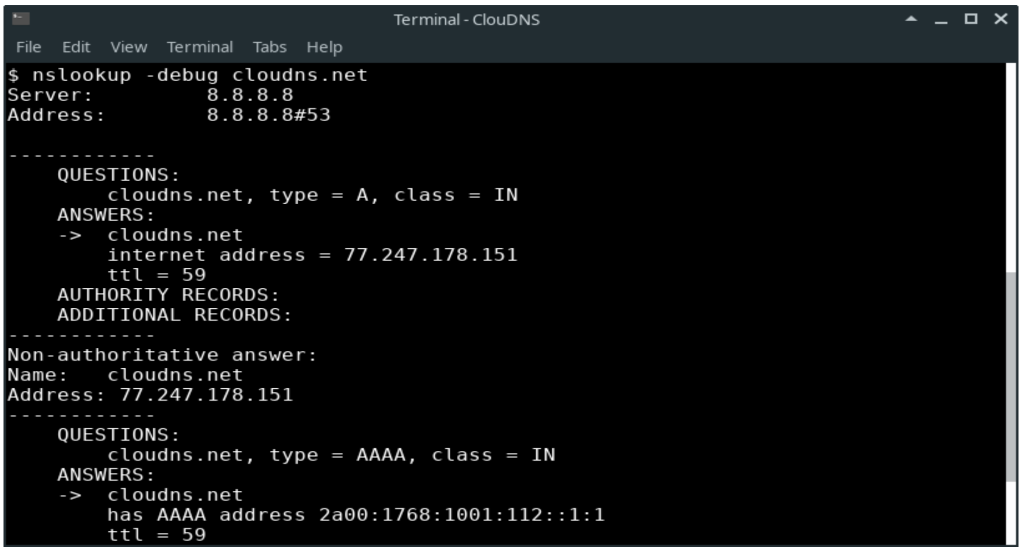
```
$ nslookup -timeout=20 cloudns.net
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   cloudns.net
Address: 77.247.178.151
Name:   cloudns.net
Address: 2a00:1768:1001:112::1:1

$
```

10. How to enable debug mode.

Debug mode provides important and detailed information both for the question and for the received answer.

Command line:

$ nslookup -debug example.com



```
$ nslookup -debug cloudns.net
Server:         8.8.8.8
Address:        8.8.8.8#53

------------
    QUESTIONS:
        cloudns.net, type = A, class = IN
    ANSWERS:
    ->  cloudns.net
        internet address = 77.247.178.151
        ttl = 59
    AUTHORITY RECORDS:
    ADDITIONAL RECORDS:
------------
Non-authoritative answer:
Name:   cloudns.net
Address: 77.247.178.151
------------
    QUESTIONS:
        cloudns.net, type = AAAA, class = IN
    ANSWERS:
    ->  cloudns.net
        has AAAA address 2a00:1768:1001:112::1:1
        ttl = 59
```

Result:

Thus the learning of commands like tcpdump, netstat, ifconfig, nslookup and traceroute has done. Capture ping and traceroute PDUs using a network protocol analyzer and examined.

2. Python Program for Chat Program using TCP Socket.

Aim:

To write Applications using TCP sockets like chat server.

SERVER PROGRAM:

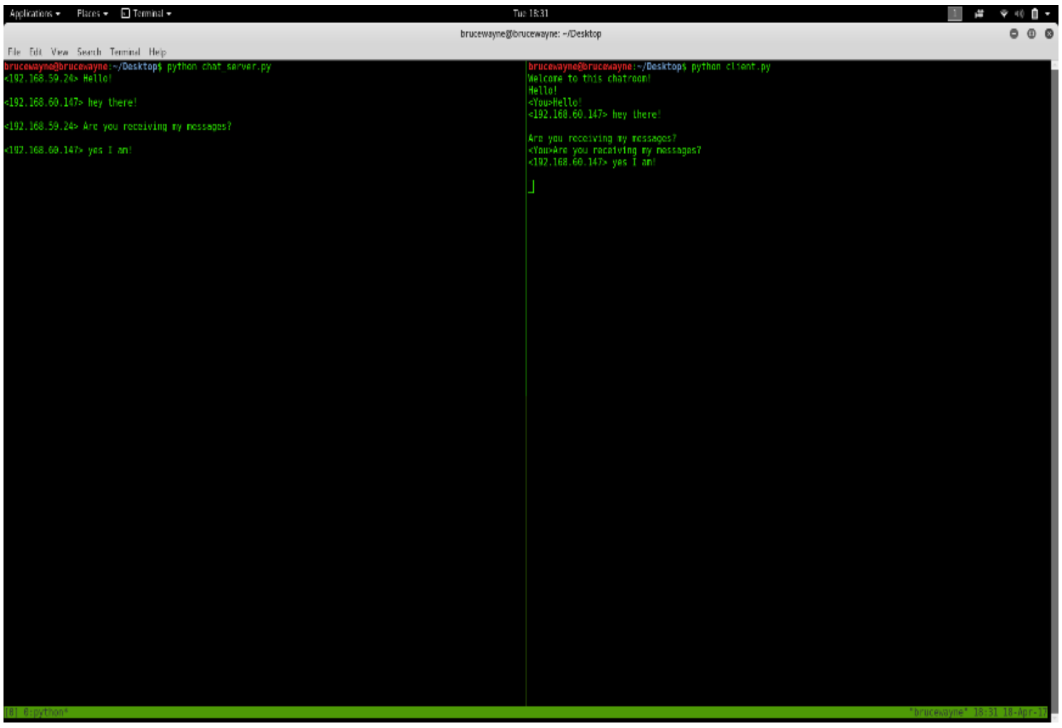
```
import socket
import select
import sys
'''Replace "thread" with "_thread" for python 3'''
from thread import *

"""The first argument AF_INET is the address domain of the
socket. This is used when we have an Internet Domain with
any two hosts The second argument is the type of socket.
SOCK_STREAM means that data or characters are read in
a continuous flow."""
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

# checks whether sufficient arguments have been provided
if len(sys.argv) != 3:
    print ("Correct usage: script, IP address, port number")
    exit()

# takes the first argument from command prompt as IP address
IP_address = str(sys.argv[1])

# takes second argument from command prompt as port number
Port = int(sys.argv[2])

"""
binds the server to an entered IP address and at the
specified port number.
The client must be aware of these parameters
"""
server.bind((IP_address, Port))

"""
listens for 100 active connections. This number can be
increased as per convenience.
"""
server.listen(100)

list_of_clients = []
```

```
def clientthread(conn, addr):

    # sends a message to the client whose user object is conn
    conn.send("Welcome to this chatroom!")

    while True:
            try:
                message = conn.recv(2048)
                if message:

                    """prints the message and address of the
                    user who just sent the message on the server
                    terminal"""
                    print ("<" + addr[0] + "> " + message)

                    # Calls broadcast function to send message to all
                    message_to_send = "<" + addr[0] + "> " + message
                    broadcast(message_to_send, conn)

                else:
                    """message may have no content if the connection
                    is broken, in this case we remove the connection"""
                    remove(conn)

            except:
                continue

"""Using the below function, we broadcast the message to all
clients who's object is not the same as the one sending
the message """
def broadcast(message, connection):
    for clients in list_of_clients:
        if clients!=connection:
            try:
                clients.send(message)
            except:
                clients.close()

                # if the link is broken, we remove the client
                remove(clients)

"""The following function simply removes the object
from the list that was created at the beginning of
the program"""
def remove(connection):
    if connection in list_of_clients:
        list_of_clients.remove(connection)

while True:
    """Accepts a connection request and stores two parameters,
    conn which is a socket object for that user, and addr
    which contains the IP address of the client that just
    connected"""
    conn, addr = server.accept()

    """Maintains a list of clients for ease of broadcasting
    a message to all available people in the chatroom"""
    list_of_clients.append(conn)

    # prints the address of the user that just connected
    print (addr[0] + " connected")

    # creates and individual thread for every user
    # that connects
    start_new_thread(clientthread,(conn,addr))

conn.close()
server.close()
```

CLIENT PROGRAM:

```
# Python program to implement client side of chat room.

import socket

import select

import sys

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

if len(sys.argv) != 3:

        print ("Correct usage: script, IP address, port number")

        exit()

IP_address = str(sys.argv[1])

Port = int(sys.argv[2])

server.connect((IP_address, Port))

while True:

        # maintains a list of possible input streams

        sockets_list = [sys.stdin, server]
```

```
read_sockets,write_socket, error_socket = select.select(sockets_list,[],[])
for socks in read_sockets:
        if socks == server:
                message = socks.recv(2048)
                print (message)
        else:
                message = sys.stdin.readline()
                server.send(message)
                sys.stdout.write("<You>")
                sys.stdout.write(message)
                sys.stdout.flush()

server.close()
```


Applications
Places
Terminal
Tue 18:31
brucewayne@brucewayne: ~/Desktop
File Edit View Search Terminal Help

```
brucewayne@brucewayne:~/Desktop$ python chat_server.py
<192.168.59.24> Hello!

<192.168.60.147> hey there!

<192.168.59.24> Are you receiving my messages?

<192.168.60.147> yes I am!
```

```
brucewayne@brucewayne:~/Desktop$ python client.py
Welcome to this chatroom!
Hello!
<You>Hello!
<192.168.60.147> hey there!

Are you receiving my messages?
<You>Are you receiving my messages?
<192.168.60.147> yes I am!
```

3. Program for Sliding Window Protocol using TCP Sockets

SERVER PROGRAM

```
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>

int main()
{
        int sock,size,connect;
        char senddata[50],data[50];
        int val,count,i,port;
        struct sockaddr_in ser,cli;
        printf("\n\n Server Running ...... ");
        if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1)
        {
           perror("\n Socket Creation Error");
            exit(-1);
        }
        printf("\nEnter the port number  : ");
        scanf("%d",&port);
        ser.sin_family = AF_INET;
        ser.sin_port = htons(port);
        ser.sin_addr.s_addr=INADDR_ANY;
        bzero(&(ser.sin_zero),8);
if(bind(sock,(struct sockaddr *)&ser,sizeof(struct sockaddr)) == -1)
        {
            perror("\n\t Error in Bind");
            exit(-1);
        }
if (listen(sock,2)==-1)
        {
             perror("\n\t Error in Listen");
             exit(-1);
        }
printf("\n\t Waiting for connection ");
   size=sizeof(struct sockaddr);
   connect=accept(sock,(struct sockaddr *)&cli,&size);
if(connect==-1)
    {
      perror("\n\t Connection Failed :");
      exit(-1);
    }
    printf("\n\t Connected Successfully");
    printf("\n");
```

```
        // get the pocket number from client
    recv(connect,&val,sizeof(val),0);
    count=val;
      while(1)
      {
            i=recv(connect,&data,sizeof(data),0);
            data[i]='\0';
            if (strcmp(data,"end")==0)
           {
               printf("\n\t Finished");
               break;
           }
           if(count!=val)
           {
            strcpy(senddata,"packet missing");
                send(connect,&count,sizeof(count),0);
                send(connect,senddata,strlen(senddata),0);
            }
          else
          {
             printf("\n The packet Number is : %d",val);
             printf("\n The data is :%s",data);
             count++;
             strcpy(senddata,"send nextdata");
             send(connect,&count,sizeof(count),0);
             send(connect,senddata,strlen(senddata),0);
          }

        printf("\n The Expected Packet now is: %d \n",count);
        recv(connect,&val,sizeof(val),0);
     }
close(connect);
close(sock);
return 0;
}
```

CLIENT PROGRAM

```
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>

int main()
{
        int sock,val,i,count,port;
        char recvdata[50],sentdata[50];
        struct sockaddr_in server_addr;
        printf("\n\n Client Running ...... ");
        if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1)
        {
            perror("Socket");
            exit(1);
        }
        printf("\nEnter the port number");
        scanf("%d",&port);
        server_addr.sin_family = AF_INET;
        server_addr.sin_port = htons(port);
        server_addr.sin_addr.s_addr= htonl(INADDR_ANY);
        bzero(&(server_addr.sin_zero),8);
        if (connect(sock, (struct sockaddr *)&server_addr, sizeof(struct sockadd
r)) == -1)
        {
            perror("Connect");
            exit(1);
        }
        while(1)
        {
                //get the pack number from client
                printf("\n Enter packet number ");
                scanf("%d",&val);
                // sent the value to server
                send(sock,&val,sizeof(val),0);
                // get the data from the user
                printf("\n\n Enter data ");
                scanf("%s",sentdata);
                // sent the to server
                send(sock,sentdata,strlen(sentdata),0);
                   if(strcmp(sentdata,"end")==0)
                   break;
                // recev the result from server
                recv(sock,&count,sizeof(count),0);
                i=recv(sock,recvdata,50,0);
                recvdata[i]='\0';
                printf("\n %s %d",recvdata,count);
        }
        close(sock);
        return 0;
}
```

OUTPUT
SERVER

```
[11ca013@mcalinux network]$ cc 17slideser.c
[11ca013@mcalinux network]$  ./a.out


 Server Running ......
```

```
Enter the port number  : 3326

         Waiting for connection
         Connected Successfully

 The packet Number is : 18
 The data is :google
 The Expected Packet now is: 19

 The packet Number is : 19
 The data is :yahoo
 The Expected Packet now is: 20

 The Expected Packet now is: 20

 The packet Number is : 20
 The data is :ok
 The Expected Packet now is: 21
```

CLIENT

```
[11ca013@mcalinux network]$ cc 17slidecli.c
[11ca013@mcalinux network]$ ./a.out

 Client Running ......
Enter the port number3326
 Enter packet number 18


 Enter data google

 send nextdata 19
 Enter packet number 19


 Enter data yahoo

 send nextdata 20
 Enter packet number 26

 Enter data hi

 packet missing 20
 Enter packet number 20


 Enter data ok
```

4. Python Program for DNS using UDP Sockets.

A Record: It is fundamental type of DNS record, here A stands for address. It shows the IP address of the domain.

```
# Import libraries
import dns.resolver

# Finding A record
result = dns.resolver.query('srmeaswari.edu.in’, 'A')

# Printing record
for val in result:
    print('A Record : ', val.to_text())
```

Output:

```
A Record :  34.218.62.116
```

AAAA Record: This is an IP address record, used to find the IP of the computer connected to the domain. It is conceptually similar to A record but specifies only the IPv6 address of the server rather than IPv4.

```
# Import libraries
import dns.resolver

# Finding AAAA record
result = dns.resolver.query(' srmeaswari.edu.in’, 'AAAA')

# Printing record
for val in result:
    print('AAAA Record : ', ipval.to_text())
```

Output:

```
NoAnswer: The DNS response does not contain an answer to the question: srmeaswari.edu.in. IN AAAA
```

PTR Record: PTR stands for pointer record, used to translate IP addresses to the domain name or hostname. It is used to reverse the DNS lookup.

```
# Import libraries
import dns.resolver

# Finding PTR record
result = dns.resolver.query('116.62.218.34.in-addr.arpa', 'PTR')

# Printing record
for val in result:
    print('PTR Record : ', val.to_text())
```

Output:

PTR Record : ec2-34-218-62-116.us-west-2.compute.amazonaws.com.

- **NS Record:** Nameserver(NS) record gives information that which server is authoritative for the given domain i.e. which server has the actual DNS records. Multiple NS records are possible for a domain including the primary and the backup name servers.

```
# Import libraries
import dns.resolver

# Finding NS record
result = dns.resolver.query('geeksforgeeks.org', 'NS')

# Printing record
for val in result:
        print('NS Record : ', val.to_text())
```

Output:

```
NS Record :  ns-1520.awsdns-62.org.

NS Record :  ns-1569.awsdns-04.co.uk.

NS Record :  ns-245.awsdns-30.com.

NS Record :  ns-869.awsdns-44.net.
```

5. Implement Routing Protocol using Cisco Packet tracer simulator tool.

AIM:

To stimulate and save the distance vector routing algorithm using simulation

PROCEDURE:

STEP1:insert PC-PT(PC0)and IP configuration Class A and subnet mask 255.0.0.0

STEP2:insert PC-PT(pc1)and IP configuration

STEP3:connect PC0 to switch 0 using straight wire

STEP4:connect PC1 to switch1 and PC2 to swith 2 using straight wire

STEP5:connect switch 0 to router-1,switch-1 to router2

STEP6:assign IP address Class A to serial i/o and router1

Step7:assign IP address Class A to serial 2/0 and serial 3/0 and router 2 and subnetmask 255.0.0.0

STEP8:Assign IP address Class A to serial 2/0 of router 3 and subnetmask

STEP9:connect router-1 to router-2 using cable connect router-2 to router-3 using cable

PROGRAM:

Adding IP Address to Router 1

Router>en **(Enabling privilege mode)**

Router#conf t **(Enabling global configuration mode)**

Router(config)#int s0/0 **(Selecting serial port s0/0)**

Router(config-if)#ip add 10.0.0.1 255.0.0.0 **(Assigning IP Address)**

Router(config-if)#clock rate 64000 **(Adding clock rate to the DCE side of the cable)**

Router(config-if)#no shut **(Enabling the port)**

Router(config-if)#exit **(Exit from global configuration mode)**

Router(config)#int s0/1

Router(config-if)#ip add 40.0.0.1 255.0.0.0

```
Router(config-if)#clock rate 64000
Router(config-if)#no shut
Router(config-if)#exit

Router(config)#int s0/2
Router(config-if)#ip add 70.0.0.1 255.0.0.0
Router(config-if)#clock rate 64000
Router(config-if)#no shut
Router(config-if)#exit

Router(config)#int f0/0
Router(config-if)#ip add 90.0.0.1 255.0.0.0
Router(config-if)#no shut
```

Adding IP Address to Router 2

```
Router>en
Router#conf t
Router(config)#int s0/0
Router(config-if)#ip add 10.0.0.2 255.0.0.0
Router(config-if)#no shut

Router(config-if)#exit
Router(config)#int s0/1
Router(config-if)#ip add 20.0.0.1 255.0.0.0
Router(config-if)#clock rate 64000
Router(config-if)#no shut
```

Adding IP Address to Router 3

```
Router>
Router>en
Router#conf t
Router(config)#int s0/0
Router(config-if)#ip add 20.0.0.2 255.0.0.0
Router(config-if)#no shut
Router(config-if)#Exit
```

```
 Router(config)#int s0/1
Router(config-if)#ip add 30.0.0.1 255.0.0.0
Router(config-if)#clock rate 64000
Router(config-if)#no shut
Router(config-if)#exit
Router(config)#exit
```

Adding IP Address to Router 4

```
Router>en
Router#conf t
Router(config)#int s0/0
Router(config-if)#ip add 30.0.0.2 255.0.0.0
Router(config-if)#no shut
 Router(config-if)#int s0/1
Router(config-if)#ip add 60.0.0.2 255.0.0.0
Router(config-if)#no shut
 Router(config-if)#int s0/2
Router(config-if)#ip add 80.0.0.2 255.0.0.0
Router(config-if)#no shut
 Router(config-if)#int f0/0
Router(config-if)#ip add 100.0.0.1 255.0.0.0
Router(config-if)#no shut
```

Adding IP Address to Router 5

```
Router>en
Router#conf t
Router(config)#int s0/0
Router(config-if)#ip add 40.0.0.2 255.0.0.0
Router(config-if)#no shut

Router(config-if)#int s0/1
Router(config-if)#ip add 50.0.0.1 255.0.0.0
Router(config-if)#clock rate 64000
Router(config-if)#no shut
Router(config-if)#exit
```

Adding IP Address to Router 6

```
Router>en
Router#conf t
Router(config)#int s0/0
Router(config-if)#ip add 50.0.0.2 255.0.0.0
Router(config-if)#no shut

Router(config-if)#int s0/1
Router(config-if)#ip add 60.0.0.1 255.0.0.0
Router(config-if)#clock rate 64000
Router(config-if)#no shut
Router(config-if)#exit
```

Adding IP Address to Router 7

```
Router>en
Router#conf t
Router(config)#int s0/0
Router(config-if)#ip add 70.0.0.2 255.0.0.0
Router(config-if)#no shut

Router(config-if)#int s0/1
Router(config-if)#ip add 80.0.0.1 255.0.0.0
Router(config-if)#clock rate 64000
Router(config-if)#no shut
Router(config-if)#exit
```

Add IP Address to both the PC

PC 1 IP Address: 90.0.0.2

Gateway: 90.0.0.1

PC 2 IP Address: 100.0.0.2

Gateway: 100.0.0.1

RIP configuration commands for Router 1

Router#conf t **(Enabling global configuration mode)**

```
Router(config)#router rip (Enabling RIP Protocol)
Router(config-router)#network 10.0.0.0 (Adding network ID 10.0.0.0)
Router(config-router)#network 40.0.0.0 (Adding network ID 40.0.0.0)
Router(config-router)#network 90.0.0.0 (Adding network ID 90.0.0.0)
Router(config-router)#exit
```

RIP configuration commands for Router 2

```
Router>en
Router#conf t
Router(config)#router rip
Router(config-router)#network 10.0.0.0
Router(config-router)#network 20.0.0.0
Router(config-router)#exit
```

RIP configuration commands for Router 3

```
Router#conf t
Router(config)#router rip
Router(config-router)#network 20.0.0.0
Router(config-router)#network 30.0.0.0
Router(config-router)#exit
Router(config)#exit
```

RIP configuration commands for Router 4

```
Router#conf t
Router(config)#router rip
Router(config-router)#network 30.0.0.0
Router(config-router)#network 80.0.0.0
Router(config-router)#network 60.0.0.0
Router(config-router)#network 100.0.0.0
Router(config-router)#exit
Router(config)#exit
```

RIP configuration commands for Router 5

```
Router#conf t
Router(config)#router rip
Router(config-router)#network 40.0.0.0
Router(config-router)#network 50.0.0.0
Router(config-router)#exit
```

Router(config)#exit

RIP configuration commands for Router 6

Router#conf t

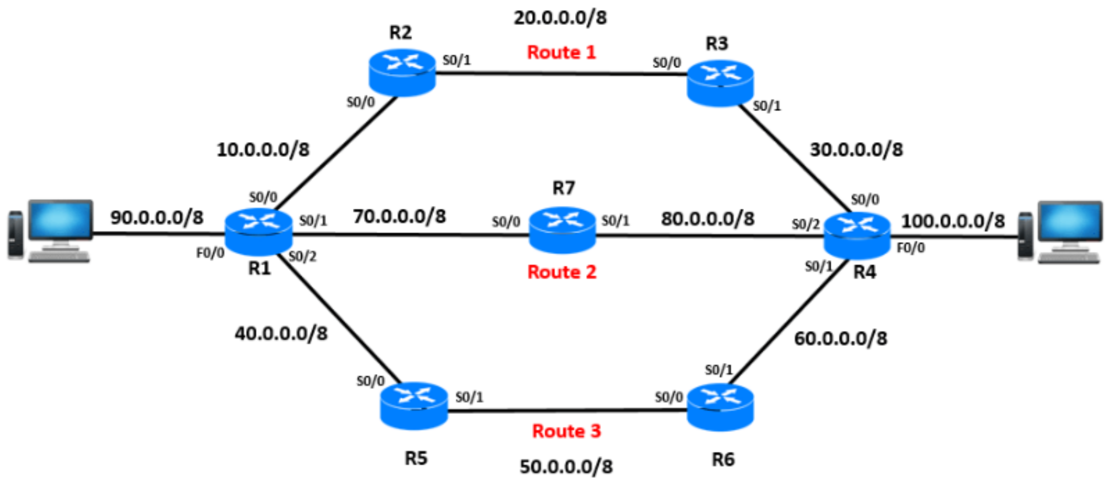
Router(config)#router rip

Router(config-router)#network 50.0.0.0

Router(config-router)#network 60.0.0.0

Router(config-router)#exit

Router(config)#exit



RESULT:

Thus to stimulate and save distance routing algorithm was successfully executed.

EX:7 Comparison of TCP/UDP performance using Simulation tool

AIM:

To Comparison of performance TCP/UDP using simulation tool.

PROCEDURE:

STEP1: Insert PC0 and IP Configuration 192.15.17 and Subnet masking(PC-PT).

STEP2: Insert PC and IP Configuration 192.15.13 and Subnet masking(PC-PT).

STEP3: Connect the above two PC's to a switch 0(2950-24).

STEP4: Insert PC2 and IP Configuration 172.15.8.7 and Subnet masking(PC-PT)(255.255.255.0).

STEP5: Insert PC3(PC-PT) and IP Configuration 172.15.9.6 and Subnet masking(225.225.0.0).

STEP6: Connect PC2 and PC3 to switch 1(2950-24) using straight wire.

STEP7: Connect switch 0 and switch 1 to a router(1841) using straight wire.

STEP8: Assign the IP address to router 172.15.6.7 for fast Ethernet 0/1 and Subnet masking.Specify the gateway in PC0 and PC1 as the IP router 0/0(172.15.1.2).

STEP9: Specify gateway to PC2 and PC3 as IP of router 0/1(172.15.6.7).

Program:

UDP

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

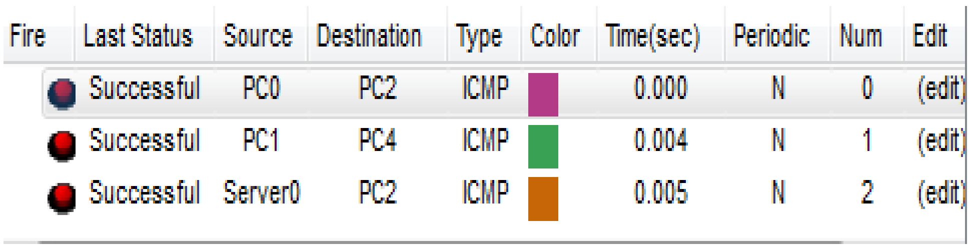
```
    set n2 [$ns node]
    set n3 [$ns node]
$n3 label “destination”
$ns duplex-link $n0 $n2 10Mb 1ms DropTail
$ns duplex-link $n1 $n2 10Mb 1ms DropTail
$ns duplex-link $n2 $n3 10Mb 1ms DropTail
$ns queue-limit $n0 $n2 10
$ns queue-limit $n1 $n2 10
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
set udp2 [new Agent/UDP]
$ns attach-agent $n2 $udp2
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
$ns connect $udp0 $null0
$ns connect $udp1 $null0
$ns connect $udp2 $null0
$ns at 0.1 "$cbr1 start"
$ns at 0.2 "$cbr0 start"

$ns at 5 "finish"
$ns run
```

TCP

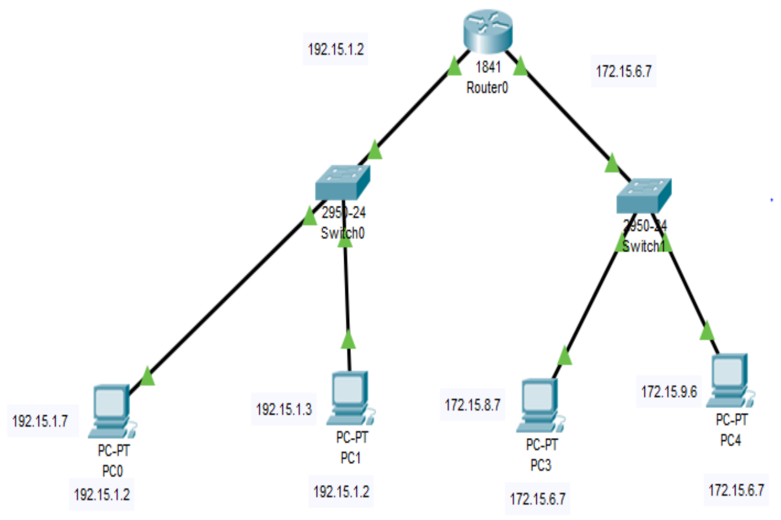
```
set ns [new Simulator]
set nf [open mi.nam w]
$ns namtrace-all $nf
set tf [open tcp.tr w]
$ns trace-all $tf
proc finish { } {
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam tcp.nam &
exit 0    }
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$n3 label “destination”
$ns duplex-link $n0 $n2 10Mb 1ms DropTail
$ns duplex-link $n1 $n2 10Mb 1ms DropTail
$ns duplex-link $n2 $n3 10Mb 1ms DropTail
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 set packet_Size_ 500
$ftp0 set interval_ 0.005
$ftp0 attach-agent $tcp0
set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
set ftp1 [new Application/FTP]
$ftp1 set packet_Size_ 500
$ftp1 set interval_ 0.005
$ftp1 attach-agent $tcp1

set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
$ns connect $tcp0 $sink0
$ns connect $tcp1 $sink1
$ns at 0.1 "$ftp1 start"
$ns at 0.1 "$ftp0 start"
$ns at 5 "finish"
$ns run
```

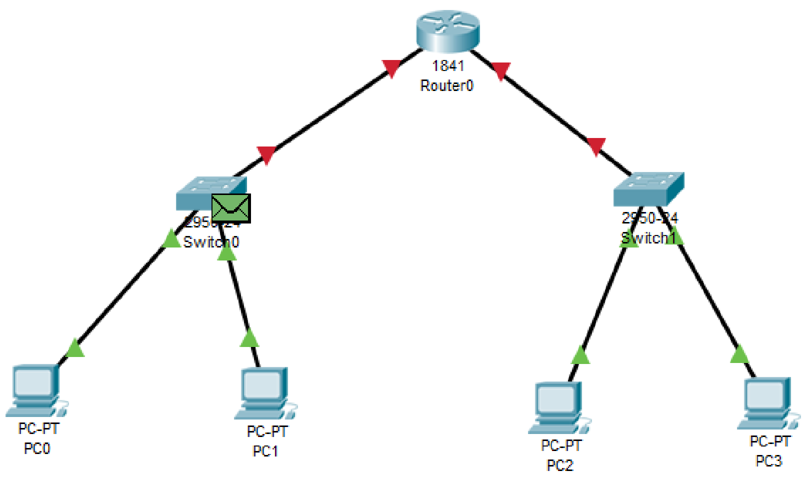
OUTPUT:

The PDU has been generated from PC0 to PC3



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
●	Successful	PC0	PC2	ICMP	🟪	0.000	N	0	(edit)
●	Successful	PC1	PC4	ICMP	🟩	0.004	N	1	(edit)
●	Successful	Server0	PC2	ICMP	🟧	0.005	N	2	(edit)





RESULT:

Thus the comparison of TCP/UDP using Simulation Components were executed successfully.

THE END