EX: 1 **Configuration of Network in Linux Environment**

DATE: 3/8/22

AIM:

To use the network configuration basic commands in linux environment.

COMMANDS:

1. **ifconfig command:**

   Description: Interface Configuration(ifconfig) command is used to initialize an interface, assign IP address to interface and enable or disable interface on demand.

   Syntax:

   ifconfig

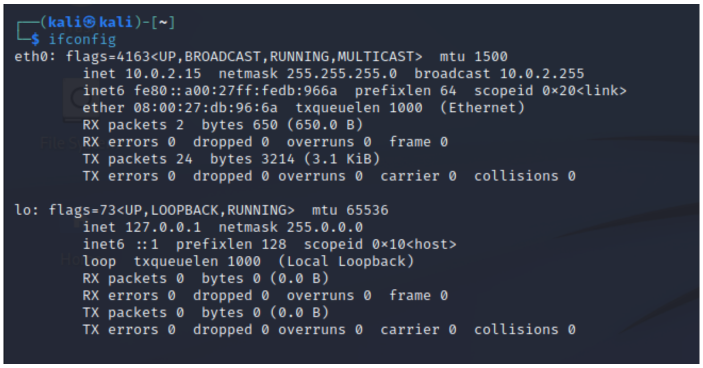
   Command:

   ifconfig eth0

   ifconfig l0

   ifconfig Wlan0

   Output: It shows the IP address of 3 networks, Ethernet, Local network.



```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fedb:966a  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:db:96:6a  txqueuelen 1000  (Ethernet)
        RX packets 2  bytes 650 (650.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 24  bytes 3214 (3.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2. **ip:**

   Description: This is the latest and updated version of ifconfig command. This command gives the details of all networks like ifconfig.

   Syntax:

   ip a

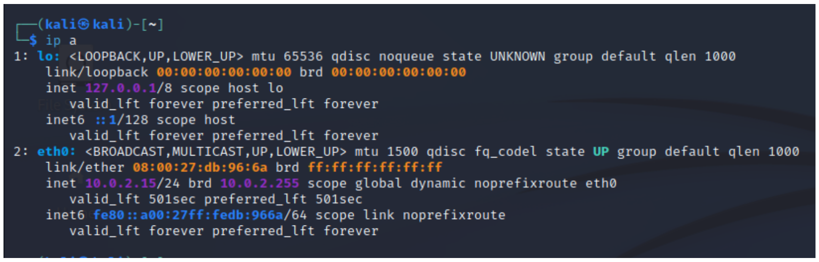
   ip addr

   ip a show eth0

   ip a show l0

   ip a show wlan0

Output:



```
┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:db:96:6a brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
       valid_lft 501sec preferred_lft 501sec
    inet6 fe80::a00:27ff:fedb:966a/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

3. **traceroute**

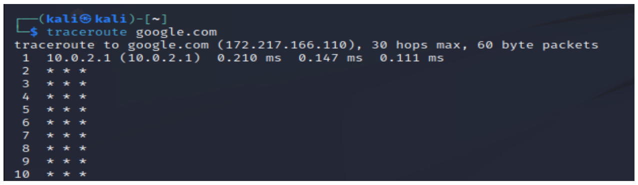
Description: It is used to troubleshoot the network. It provides the name and identifies every device on the path. It follows the route to the destination.

Syntax:

traceroute <destination>

$traceroute google.com

Output: It displays the specified hostname, IP address, size of the packets.



```
┌──(kali㉿kali)-[~]
└─$ traceroute google.com
traceroute to google.com (172.217.166.110), 30 hops max, 60 byte packets
 1  10.0.2.1 (10.0.2.1)  0.210 ms  0.147 ms  0.111 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
```

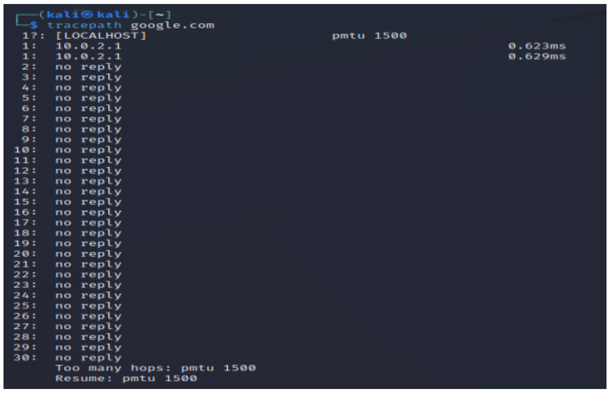
4. **tracepath**

Description: It is used to detect network delays.

Syntax:

tracepath <destination>

Output:



```
┌──(kali㉿kali)-[~]
└─$ tracepath google.com
 1?: [LOCALHOST]                      pmtu 1500
 1:  10.0.2.1                                             0.623ms
 1:  10.0.2.1                                             0.629ms
 2:  no reply
 3:  no reply
 4:  no reply
 5:  no reply
 6:  no reply
 7:  no reply
 8:  no reply
 9:  no reply
10:  no reply
11:  no reply
12:  no reply
13:  no reply
14:  no reply
15:  no reply
16:  no reply
17:  no reply
18:  no reply
19:  no reply
20:  no reply
21:  no reply
22:  no reply
23:  no reply
24:  no reply
25:  no reply
26:  no reply
27:  no reply
28:  no reply
29:  no reply
30:  no reply
     Too many hops: pmtu 1500
     Resume: pmtu 1500
```

5. **ping**

Description: It checks for the network connectivity between two nodes.

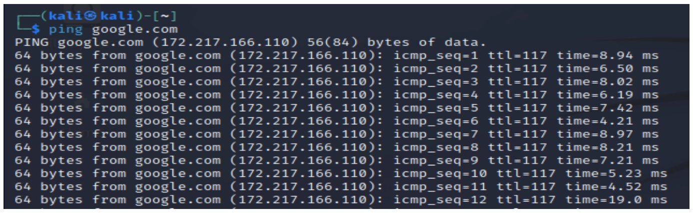
Syntax:

ping <destination>

Example:

$ping google.com

Output: It shows the successful connection to google.com and you can use the IP address to ping directly.



```
┌──(kali㉿kali)-[~]
└─$ ping google.com
PING google.com (172.217.166.110) 56(84) bytes of data.
64 bytes from google.com (172.217.166.110): icmp_seq=1 ttl=117 time=8.94 ms
64 bytes from google.com (172.217.166.110): icmp_seq=2 ttl=117 time=6.50 ms
64 bytes from google.com (172.217.166.110): icmp_seq=3 ttl=117 time=8.02 ms
64 bytes from google.com (172.217.166.110): icmp_seq=4 ttl=117 time=6.19 ms
64 bytes from google.com (172.217.166.110): icmp_seq=5 ttl=117 time=7.42 ms
64 bytes from google.com (172.217.166.110): icmp_seq=6 ttl=117 time=4.21 ms
64 bytes from google.com (172.217.166.110): icmp_seq=7 ttl=117 time=8.97 ms
64 bytes from google.com (172.217.166.110): icmp_seq=8 ttl=117 time=8.21 ms
64 bytes from google.com (172.217.166.110): icmp_seq=9 ttl=117 time=7.21 ms
64 bytes from google.com (172.217.166.110): icmp_seq=10 ttl=117 time=5.23 ms
64 bytes from google.com (172.217.166.110): icmp_seq=11 ttl=117 time=4.52 ms
64 bytes from google.com (172.217.166.110): icmp_seq=12 ttl=117 time=19.0 ms
```

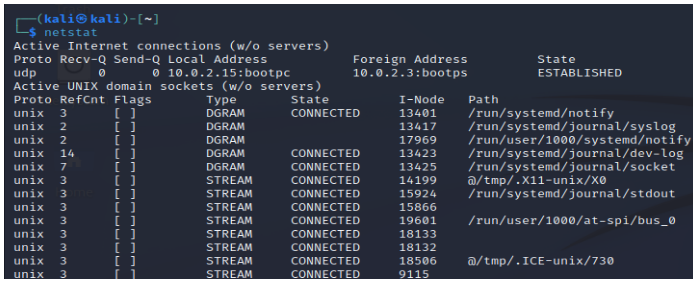
6. **netstat:**

Description: It provides statistical figures about different interfaces which include open sockets, routing tables, and connection information.

Syntax:

netstat

Output: It observes the output displaying all the open sockets.



```
┌──(kali㉿kali)-[~]
└─$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 10.0.2.15:bootpc        10.0.2.3:bootps         ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  3      [ ]         DGRAM      CONNECTED     13401    /run/systemd/notify
unix  2      [ ]         DGRAM                    13417    /run/systemd/journal/syslog
unix  2      [ ]         DGRAM                    17969    /run/user/1000/systemd/notify
unix  14     [ ]         DGRAM      CONNECTED     13423    /run/systemd/journal/dev-log
unix  7      [ ]         DGRAM      CONNECTED     13425    /run/systemd/journal/socket
unix  3      [ ]         STREAM     CONNECTED     14199    @/tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTED     15924    /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     15866
unix  3      [ ]         STREAM     CONNECTED     19601    /run/user/1000/at-spi/bus_0
unix  3      [ ]         STREAM     CONNECTED     18133
unix  3      [ ]         STREAM     CONNECTED     18132
unix  3      [ ]         STREAM     CONNECTED     18506    @/tmp/.ICE-unix/730
unix  3      [ ]         STREAM     CONNECTED     9115
```

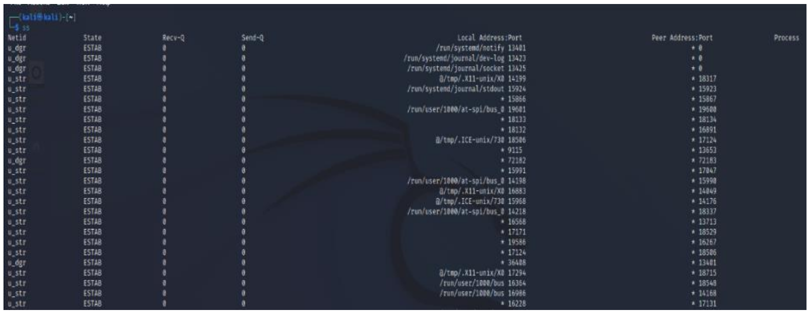
7. **ss:**

Description: Linux ss command is the replacement for netstat command. It fetches all the information from within the kernel userspace.

Syntax:

ss

Output: This command gives information about all TCP, UDP, and UNIX socket connections.



```
┌──(kali㉿kali)-[~]
└─$ ss
Netid   State   Recv-Q   Send-Q                Local Address:Port          Peer Address:Port   Process
u_dgr   ESTAB   0        0                 /run/systemd/notify 13401                  * 0
u_dgr   ESTAB   0        0        /run/systemd/journal/dev-log 13423                  * 0
u_dgr   ESTAB   0        0         /run/systemd/journal/socket 13425                  * 0
u_str   ESTAB   0        0                 @/tmp/.X11-unix/X0 14199                   * 18317
u_str   ESTAB   0        0         /run/systemd/journal/stdout 15924                  * 15923
u_str   ESTAB   0        0                                   * 15866                  * 15867
u_str   ESTAB   0        0         /run/user/1000/at-spi/bus_0 19601                  * 19600
u_str   ESTAB   0        0                                   * 18133                  * 18134
u_str   ESTAB   0        0                                   * 18132                  * 16891
u_str   ESTAB   0        0                 @/tmp/.ICE-unix/730 18506                  * 17124
u_str   ESTAB   0        0                                   * 9115                   * 13653
u_dgr   ESTAB   0        0                                   * 72182                  * 72183
u_str   ESTAB   0        0                                   * 15991                  * 17047
u_str   ESTAB   0        0         /run/user/1000/at-spi/bus_0 14198                  * 15990
u_str   ESTAB   0        0                 @/tmp/.X11-unix/X0 16883                   * 14049
u_str   ESTAB   0        0                 @/tmp/.ICE-unix/730 15968                  * 14176
u_str   ESTAB   0        0         /run/user/1000/at-spi/bus_0 14218                  * 18337
u_str   ESTAB   0        0                                   * 16568                  * 13713
u_str   ESTAB   0        0                                   * 17171                  * 18529
u_str   ESTAB   0        0                                   * 19586                  * 16267
u_str   ESTAB   0        0                                   * 17124                  * 18506
u_dgr   ESTAB   0        0                                   * 36488                  * 13401
u_str   ESTAB   0        0                 @/tmp/.X11-unix/X0 17294                   * 18715
u_str   ESTAB   0        0                 /run/user/1000/bus 16364                   * 18548
u_str   ESTAB   0        0                 /run/user/1000/bus 16986                   * 14168
u_str   ESTAB   0        0                                   * 16228                  * 17131
```

8. **dig:**

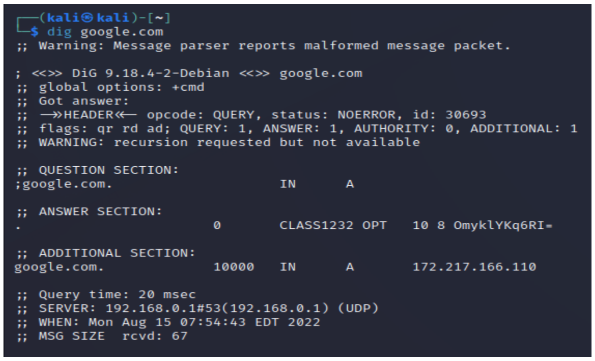
Description: It is mainly used to verify DNS mappings, MX Records, host addresses. This command is used in DNS lookup to query the DNS name server.

Syntax:

dig <domainname>

$dig google.com

Output:



```
┌──(kali㉿kali)-[~]
└─$ dig google.com
;; Warning: Message parser reports malformed message packet.

; <<>> DiG 9.18.4-2-Debian <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30693
;; flags: qr rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
.                       0       CLASS1232 OPT   10 8 OmyklYKq6RI=

;; ADDITIONAL SECTION:
google.com.             10000   IN      A       172.217.166.110

;; Query time: 20 msec
;; SERVER: 192.168.0.1#53(192.168.0.1) (UDP)
;; WHEN: Mon Aug 15 07:54:43 EDT 2022
;; MSG SIZE  rcvd: 67
```

9. **nslookup:**

Description: *Linux* nslookup is also a command used for DNS related queries.

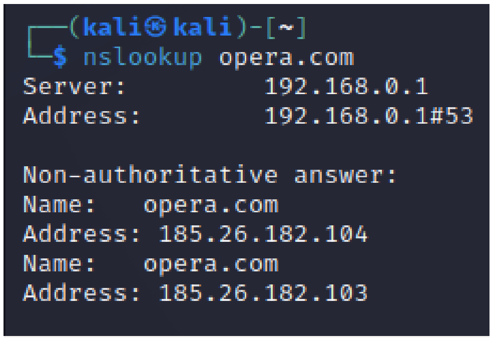
Syntax:

nslookup <domain name>

Example:

nslookup mindmajix.com

Output: It displays the record information relating to mindmajix.com



```
┌──(kali㉿kali)-[~]
└─$ nslookup opera.com
Server:         192.168.0.1
Address:        192.168.0.1#53

Non-authoritative answer:
Name:   opera.com
Address: 185.26.182.104
Name:   opera.com
Address: 185.26.182.103
```

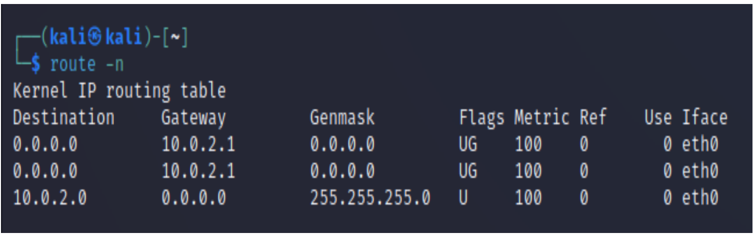
10. route:

Description: It displays and manipulates the routing table existing for your system.

Syntax:

route-n

Output: To add a gateway.

+

```
┌──(kali㉿kali)-[~]
└─$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.1        0.0.0.0         UG    100    0        0 eth0
0.0.0.0         10.0.2.1        0.0.0.0         UG    100    0        0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 eth0
```

RESULT:

Thus, the basic Linux commands are executed successfully.

Ex No:2a **STUDY OF SOCKET PROGRAMMING IN PYTHON**

DATE: 9/8/22

AIM:

To study the concepts of socket programming using python.

PROGRAM:

IMPLEMENTATION OF ECHO CLIENT/SERVER APPLICATION USING TCP

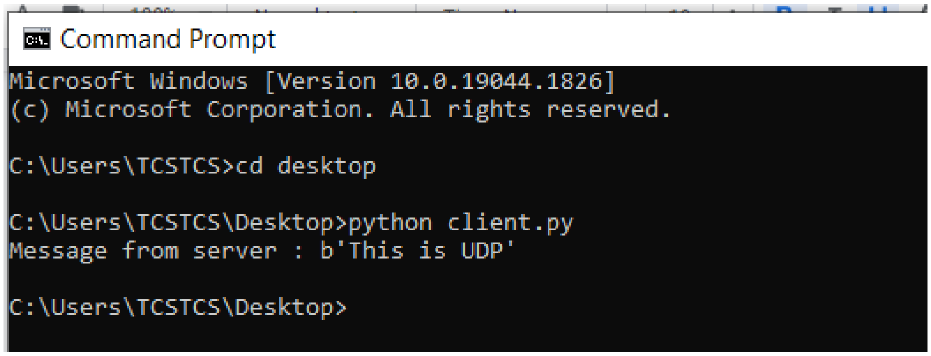
CLIENT CODE:

```
from socket import *
s = socket(AF_INET, SOCK_STREAM)
s.connect(("127.0.0.1",8000)) # Connect
op='hai'
s.send(op.encode('utf-8')) # Send request
data = s.recv(100).decode()# Get response print(data)
s.close()
```

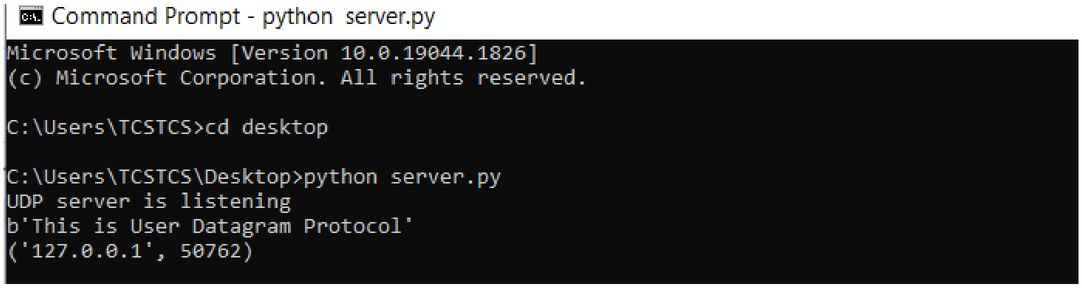
SERVER CODE:

```
from socket import *
s = socket(AF_INET,SOCK_STREAM)
s.bind(("",8000))
s.listen(5)
while True:
        c,a = s.accept()
        print("Received connection from", a)
        data=c.recv(100).decode()
        print(data)
        c.send(data.encode('utf-8'))
c.close()
```

OUTPUT:



```
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\TCSTCS>cd desktop

C:\Users\TCSTCS\Desktop>python client.py
Message from server : b'This is UDP'

C:\Users\TCSTCS\Desktop>
```



```
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\TCSTCS>cd desktop

C:\Users\TCSTCS\Desktop>python server.py
UDP server is listening
b'This is User Datagram Protocol'
('127.0.0.1', 50762)
```

IMPLEMENTATION OF ECHO CLIENT/SERVER APPLICATION USING UDP

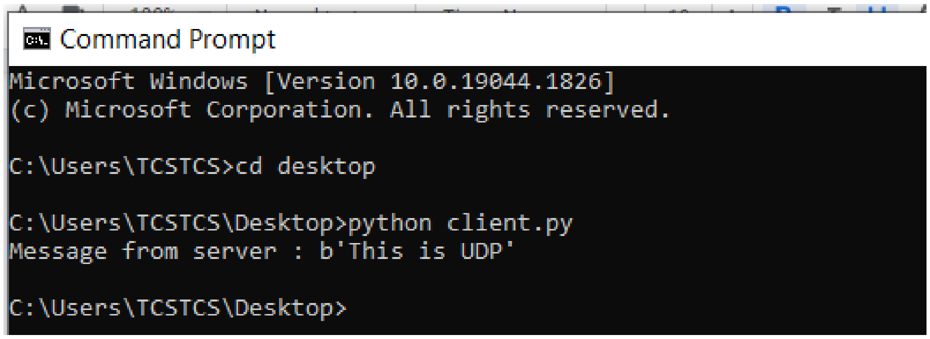
CLIENT CODE:

```
from socket import *
client_socket = socket(AF_INET, SOCK_DGRAM)
message = "This is User Datagram Protocol"
client_socket.sendto(message.encode('utf-8'), ("127.0.0.1", 12000)) data, address =
client_socket.recvfrom(1024)
print("Message from server : {}".format(data))
```

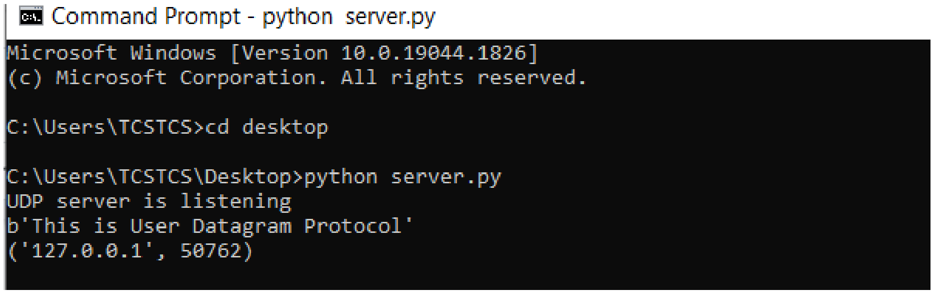
SERVER CODE:

```
from socket import *
server_socket = socket(AF_INET, SOCK_DGRAM)
server_socket.bind(('127.0.0.1', 12000))
print("UDP server is listening")
while True:
        message, address = server_socket.recvfrom(1024)
        print(message)
        print(address)

server_socket.sendto("This is UDP".encode('utf-8'), address)
```

OUTPUT:



```
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\TCSTCS>cd desktop

C:\Users\TCSTCS\Desktop>python client.py
Message from server : b'This is UDP'

C:\Users\TCSTCS\Desktop>
```



```
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\TCSTCS>cd desktop

C:\Users\TCSTCS\Desktop>python server.py
UDP server is listening
b'This is User Datagram Protocol'
('127.0.0.1', 50762)
```

RESULT:

Thus, the implementation of an echo using sockets in TCP and UDP has been written, executed and the output was verified successfully.

Ex No:2b **CHAT PROGRAM USING SOCKETS**

DATE: 10/8/22

AIM:

To implement chat program using sockets

ALGORITHM:

a. **Server Side:**

   i. Import Socket and Time module in Python.

   ii. Use gethostname() and gethostbyname() to retrieve the hostname of the machine and translate host name to IPV4 format address.

   iii. Use bind() function to bind the socket to a given address.

   iv. Use listen() function to accept connection requests from clients.

   v. Use accept() function to wait and accept connection from clients.

   vi. Use the recv() function to retrieve messages through TCP.

   vii. Then decoding is done by decode() function.

   viii. Open a while loop. Get the message as input.

   ix. If input is '[e]' then exit the chat room.

   x. Else, send the message to the client and display it.

   xi. Display the received and sent messages.

b. **Client Side:**

   i. Import Socket and Time module in Python.

   ii. Get the hostname and IP address of the host.

   iii. Enter the address of the server to chat with.

   iv. Connect to that server socket using socket() function.

   v. Send and receive messages using send() and recv() functions.

   vi. Open a while loop. Print the messages received from the server.

   vii. If the message is '[e]', then leave the server and exit the connection.

   viii. Else, display the received and sent messages.

PROGRAM:

Chatserver.py

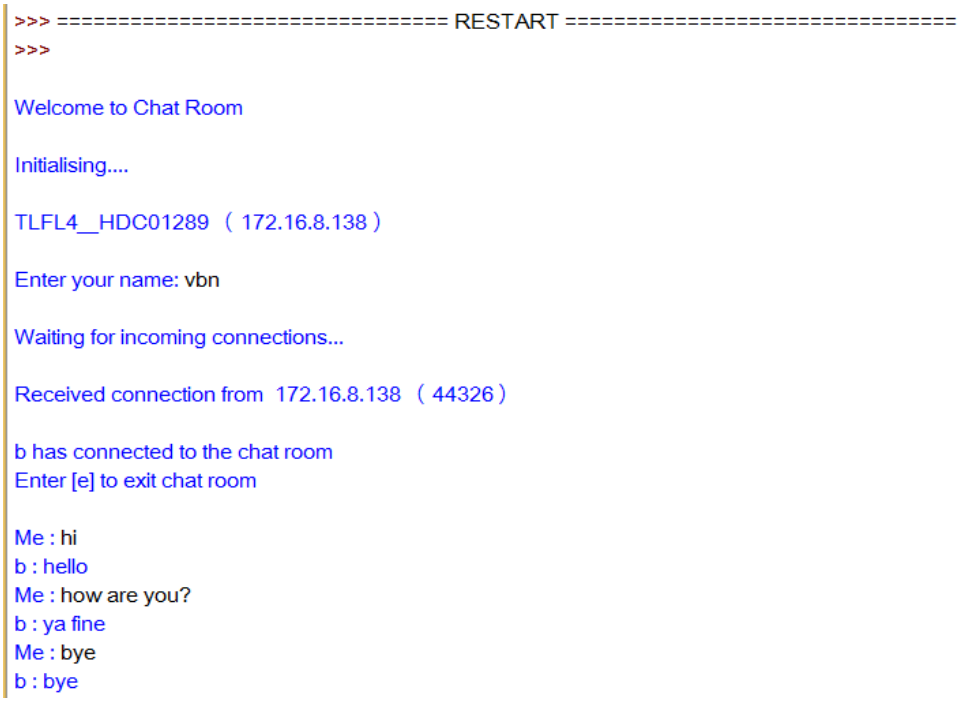
```
import time, socket, sys

print("\nWelcome to Chat Room\n")

print("Initialising....\n")

time.sleep(1)

s = socket.socket()

host = socket.gethostname()

ip = socket.gethostbyname(host)

port = 8000
```

```
s.bind((host, port))
print(host, "(", ip, ")\n")
name = input(str("Enter your name: "))
s.listen(1)
while True:
    message = input(str("Me : "))
    if message == "[e]":
        message = "Left chat room!"
        conn.send(message.encode())
        print("\n")
        break
    conn.send(message.encode())
    message = conn.recv(8000)
    message = message.decode()
    print(s_name, ":", message)
```

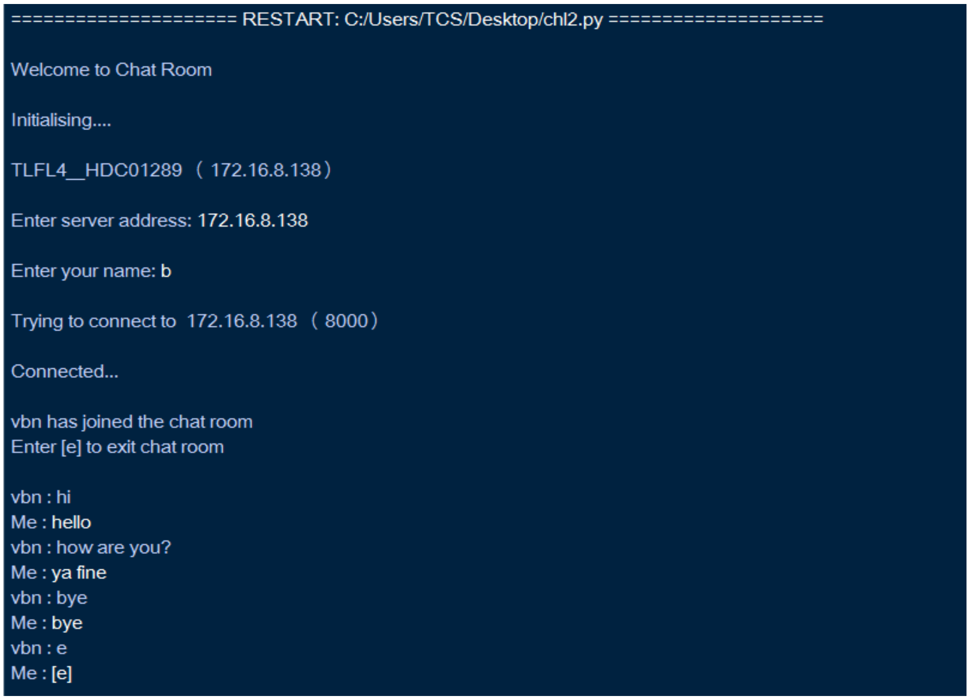
Chatclient.py

```
import time, socket, sys
 to ", host, "(", port, ")\n")
time.sleep(1)
s.connect((host, port))
s_name = s_name.decode()
print(s_name, "has joined the chat room\nEnter [e] to exit chat room\n")
while True:
    message = s.recv(8000)
    message = message.decode()
        message = "Left chat room!"
        s.send(message.encode())
        print("\n")
        break
```

Output:



```
>>> ================================ RESTART ================================
>>>

Welcome to Chat Room

Initialising....

TLFL4__HDC01289  ( 172.16.8.138 )

Enter your name: vbn

Waiting for incoming connections...

Received connection from  172.16.8.138  ( 44326 )

b has connected to the chat room
Enter [e] to exit chat room

Me : hi
b : hello
Me : how are you?
b : ya fine
Me : bye
b : bye
```



```
==================== RESTART: C:/Users/TCS/Desktop/chl2.py ====================

Welcome to Chat Room

Initialising....

TLFL4__HDC01289  ( 172.16.8.138)

Enter server address: 172.16.8.138

Enter your name: b

Trying to connect to  172.16.8.138  ( 8000)

Connected...

vbn has joined the chat room
Enter [e] to exit chat room

vbn : hi
Me : hello
vbn : how are you?
Me : ya fine
vbn : bye
Me : bye
vbn : e
Me : [e]
```

RESULT:

Thus, the implementation of a chat application using sockets has been written, executed and the output was verified successfully.

Ex No: 2c **UPPER CASE CONVERTOR USING UDP SOCKET**
DATE: 16/8/22

AIM:

To convert uppercase to lowercase using UDP socket.

ALGORITHM:

UDP Server:

1. Create a UDP socket.
2. Bind the socket to the server address.
3. Wait until the datagram packet arrives from the client.
4. Process the datagram packet and send a reply to the client.
5. Go back to Step 3.

UDP Client:

1. Create a UDP socket.
2. Send a message to the server.
3. Wait until response from the server is received.
4. Process reply and go back to step 2, if necessary.
5. Close socket descriptor and exit.

PROGRAM:

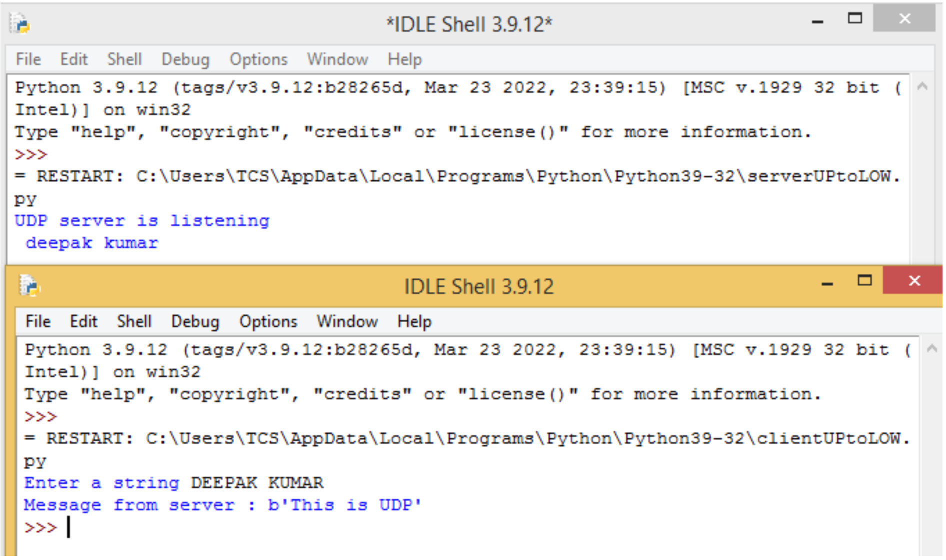
Server.py

```
from socket import *
server_socket = socket(AF_INET, SOCK_DGRAM)
server_socket.bind(('127.0.0.1', 12000))
print("UDP server is listening")
while True:
    message, address = server_socket.recvfrom(1024)
    print(message.lower().decode())
    server_socket.sendto("This is UDP".encode('utf-8'), address)
```

Client.py

```
from socket import *
client_socket = socket(AF_INET, SOCK_DGRAM)
message =input("Enter a string")
client_socket.sendto(message.encode('utf-8'), ("127.0.0.1", 12000))
data, address = client_socket.recvfrom(1024)
print("Message from server : {}".format(data))
```

OUTPUT:


IDLE Shell 3.9.12
File Edit Shell Debug Options Window Help
IDLE Shell 3.9.12
File Edit Shell Debug Options Window Help

```
Python 3.9.12 (tags/v3.9.12:b28265d, Mar 23 2022, 23:39:15) [MSC v.1929 32 bit (
Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\TCS\AppData\Local\Programs\Python\Python39-32\serverUPtoLOW.
py
UDP server is listening
 deepak kumar
```

```
Python 3.9.12 (tags/v3.9.12:b28265d, Mar 23 2022, 23:39:15) [MSC v.1929 32 bit (
Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\TCS\AppData\Local\Programs\Python\Python39-32\clientUPtoLOW.
py
Enter a string DEEPAK KUMAR
Message from server : b'This is UDP'
>>>
```

RESULT:

Thus, uppercase alphabets were converted to lowercase using socket programming in python and the output has been verified.

Ex No:2d **CALCULATOR APPLICATION USING TCP SOCKET**
DATE: 17/8/22

AIM:
To implement a calculator application using TCP socket.

ALGORITHM:

Server Side:

- Create a socket using socket( ) system call.
- Bind server’s address and port using bind( ) system call.
- Convert the socket into a listening socket using listen( ) sytem call.
- Wait for client connection to complete using accept( ) system call.
- Receive the Client request using recv() system call which consists of the input and operation to be performed.
- The calculation is performed and the result is passed back to the client by the server.

Client Side:

- Create a socket.
- Fill in the internet socket address structure (with server information).
- Connect to the server using the connect system call.
- The client passes the operator and input numbers to the server.
- Read the result sent by the server, write it to standard output.
- Close the socket connection.

PROGRAM:

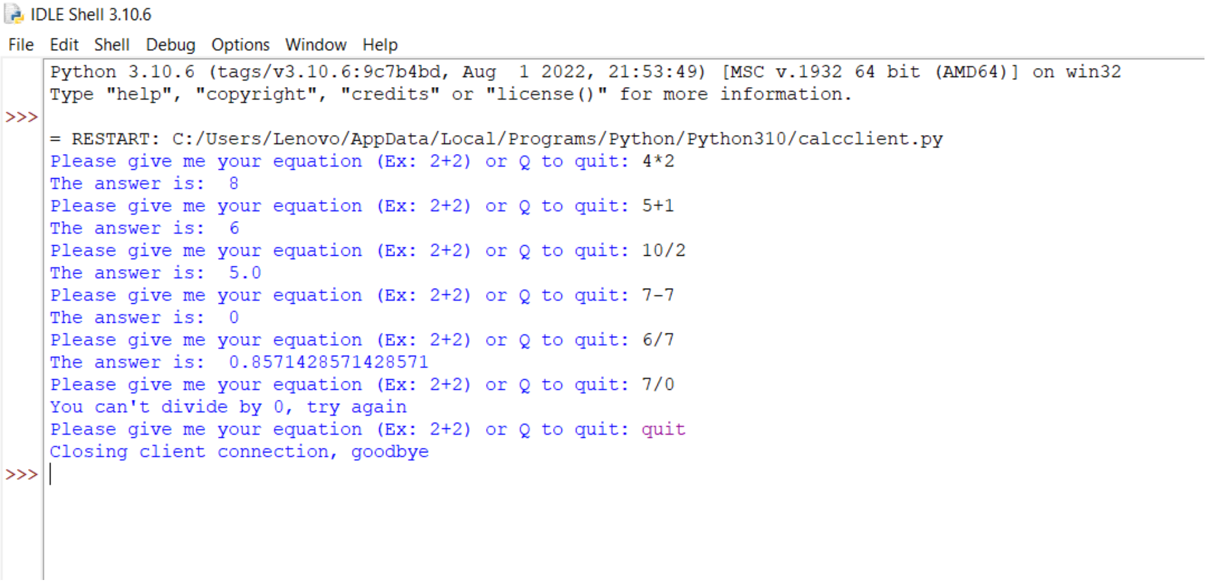
Server.py

```
from socket import *
from sys import *

s =socket(AF_INET,SOCK_STREAM)
s.bind(("127.0.0.1", 8000))
s.listen(5)
print("Server is up and running")
while True:
    c, addr = s.accept()
    print('Got connection from', addr)
    while True:
        try:
            equation=c.recv(1024).decode()
            if equation == "Q" or equation == "q" or equation == "Quit" or equation == "quit" or
equation == "quit()":
                c.send("Quit".encode())
                break
            else:
                print("You gave me the equation:", equation)
                result = eval(equation)
                c.send(str(result).encode())
        except (ZeroDivisionError):
            c.send("ZeroDiv".encode())
        except (ArithmeticError):
            c.send("MathError".encode())
        except (SyntaxError):
            c.send("SyntaxError".encode())
        except (NameError):
            c.send("NameError".encode())

    c.close()
```

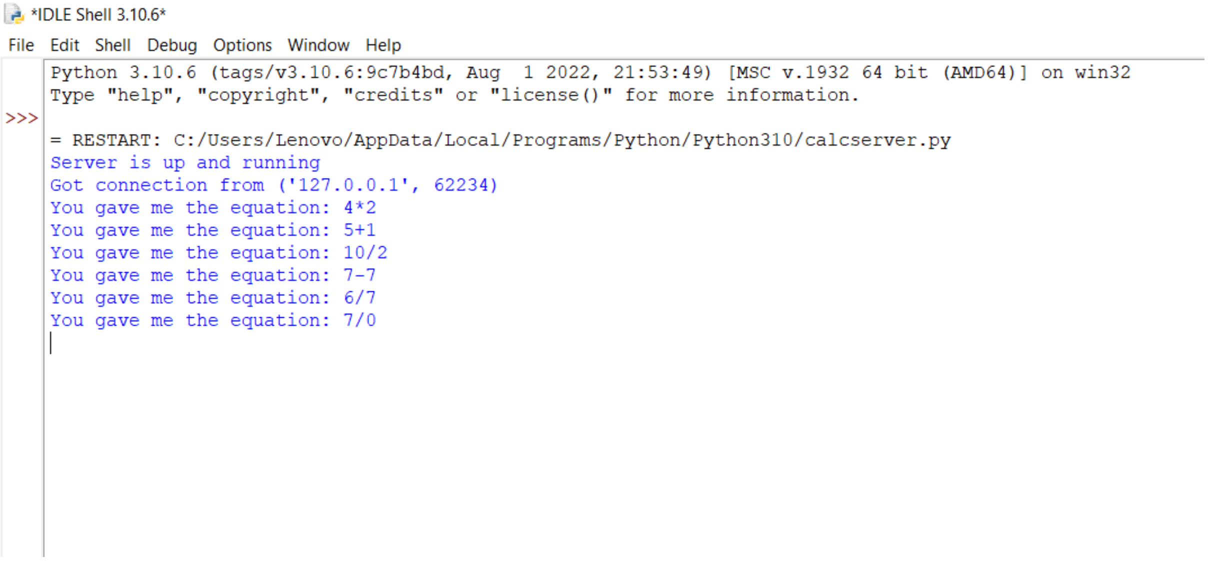
Client.py

```
from socket import *
from sys import *
from ipaddress import *

s = socket(AF_INET,SOCK_STREAM)
s.connect(("127.0.0.1", 8000))
while(True):
    equ=input("Please give me your equation (Ex: 2+2) or Q to quit: ")
    s.send(equ.encode())
    result = s.recv(1024).decode()

    if result == "Quit":
        print("Closing client connection, goodbye")
        break
    elif result == "ZeroDiv":
        print("You can't divide by 0, try again")
    elif result == "MathError":
        print("There is an error with your math, try again")
    elif result == "SyntaxError":
        print("There is a syntax error, please try again")
    elif result == "NameError":
        print("You did not enter an equation, try again")
    else:
        print("The answer is: ", result)
s.close()
```

OUTPUT:



```
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug  1 2022, 21:53:49) [MSC v.1932 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Lenovo/AppData/Local/Programs/Python/Python310/calcclient.py
Please give me your equation (Ex: 2+2) or Q to quit: 4*2
The answer is:  8
Please give me your equation (Ex: 2+2) or Q to quit: 5+1
The answer is:  6
Please give me your equation (Ex: 2+2) or Q to quit: 10/2
The answer is:  5.0
Please give me your equation (Ex: 2+2) or Q to quit: 7-7
The answer is:  0
Please give me your equation (Ex: 2+2) or Q to quit: 6/7
The answer is:  0.8571428571428571
Please give me your equation (Ex: 2+2) or Q to quit: 7/0
You can't divide by 0, try again
Please give me your equation (Ex: 2+2) or Q to quit: quit
Closing client connection, goodbye
>>>
```



```
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug  1 2022, 21:53:49) [MSC v.1932 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Lenovo/AppData/Local/Programs/Python/Python310/calcserver.py
Server is up and running
Got connection from ('127.0.0.1', 62234)
You gave me the equation: 4*2
You gave me the equation: 5+1
You gave me the equation: 10/2
You gave me the equation: 7-7
You gave me the equation: 6/7
You gave me the equation: 7/0
```

RESULT:

Thus a calculator application between client and server using socket programming in python is executed and the output has been verified.

Ex No:2e **FILE TRANSFER USING SOCKET PROGRAMMING**
DATE: 19/8/22

AIM:
To create file transfer applications using socket programming in python.

PROCEDURE:
Server Side:

- Import socket package
- Set the port number
- Set host variable to get hostname by gethostname() method.
- s.bind method binds host and port address.
- Set listen value to 5
- accept method is used to accept connection from server.
- Get the data using the recv method.
- Send the connection using the send method.
- Close the connection.

Client Side:

- Import socket package
- Call the socket function and store it in a variable.
- Get host variable to get hostname by gethostname() method.
- Set the port address.
- Connect to the server and send the message.
- Receive the data using the recv method.
- Close the connection.

PROGRAM:

CLIENT

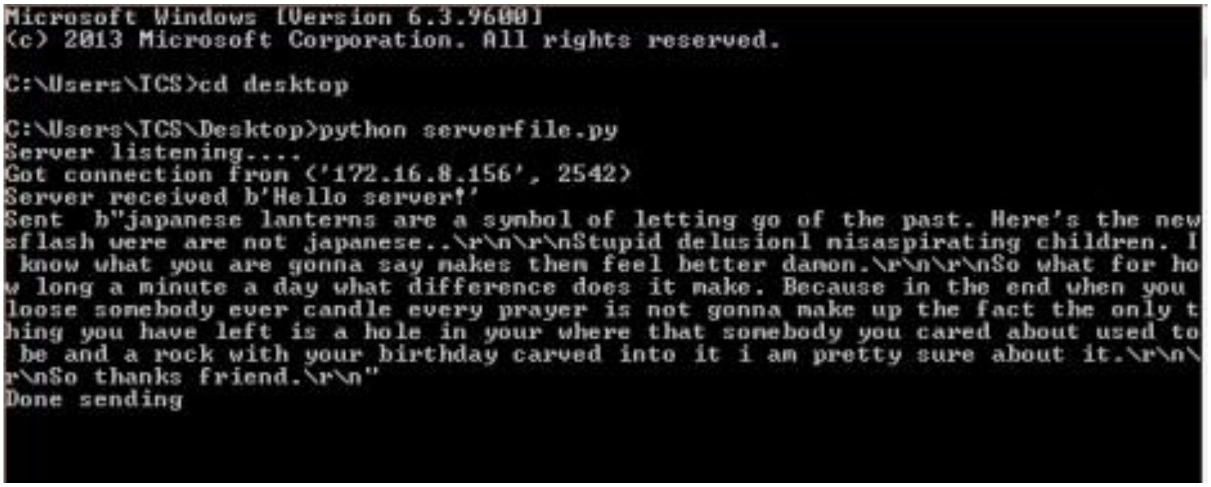
```
import socket  # Import socket module
s = socket.socket()                # Create a socket object
host = socket.gethostname()   # Get local machine name
port = 60000   # Reserve a port for your service.
s.connect((host, port))
s.send("Hello server!".encode('utf-8'))
with open('received_file.txt', 'wb') as f:
        print('file opened')
        while True:
                print('receiving data...')
                data = s.recv(1024)
                print('data :', (data))
                if not data:
                        break
        f.write(data)
f.close()
print('Successfully get the file')
s.close()
print('connection closed')
```

SERVER

```
import socket  # Import socket module
port = 60000   # Reserve a port for your service.
s = socket.socket()             # Create a socket object
host = socket.gethostname()   # Get local machine name
s.bind((host, port))      # Bind to the port
s.listen(5)                 # Now wait for client connection.
print('Server listening..')
while True:
    conn, addr = s.accept()       # Establish connection with client.
    print('Got connection from', addr)
    data = conn.recv(1024)
    print('Server received', repr(data))
    filename='file.txt'
    f = open(filename,'rb')
    l = f.read(1024)
    while (l):
        conn.send(l)
        print('Sent ',repr(l))
        l = f.read(1024)
    f.close()
    print('Done sending')
    # conn.send('Thank you for connecting'.encode('utf-8'))
    conn.close()
```

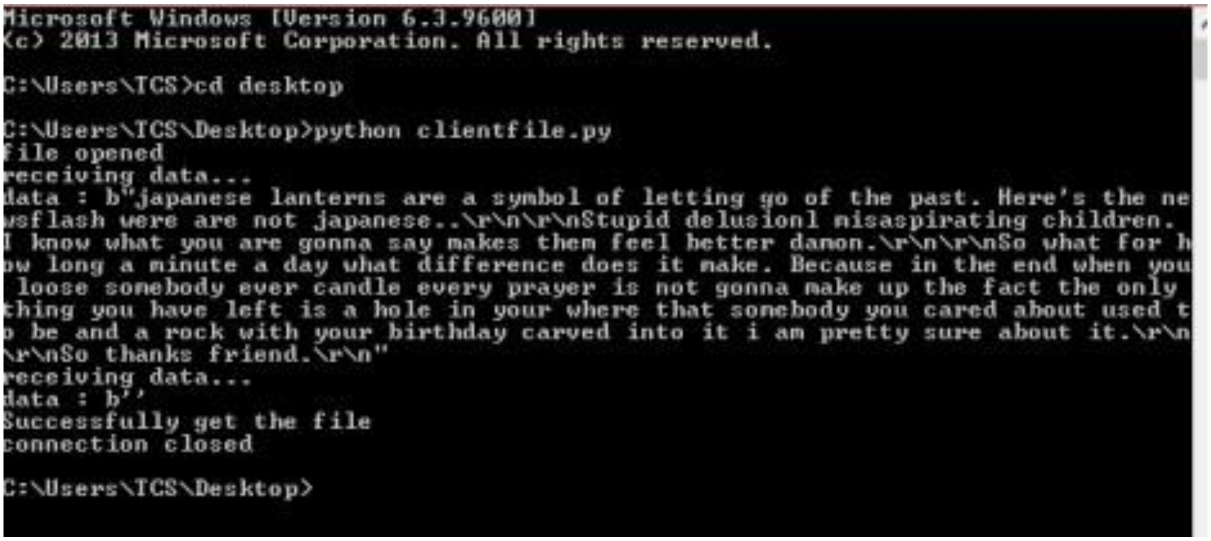
OUTPUT:

SERVER



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\TCS>cd desktop

C:\Users\TCS\Desktop>python serverfile.py
Server listening....
Got connection from ('172.16.8.156', 2542)
Server received b'Hello server!'
Sent  b"japanese lanterns are a symbol of letting go of the past. Here's the new
sflash were are not japanese..\r\n\r\nStupid delusionl misaspirating children. I
 know what you are gonna say makes them feel better damon.\r\n\r\nSo what for ho
w long a minute a day what difference does it make. Because in the end when you
loose somebody ever candle every prayer is not gonna make up the fact the only t
hing you have left is a hole in your where that somebody you cared about used to
 be and a rock with your birthday carved into it i am pretty sure about it.\r\n\
r\nSo thanks friend.\r\n"
Done sending
```

CLIENT



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\TCS>cd desktop

C:\Users\TCS\Desktop>python clientfile.py
file opened
receiving data...
data : b"japanese lanterns are a symbol of letting go of the past. Here's the ne
wsflash were are not japanese..\r\n\r\nStupid delusionl misaspirating children.
I know what you are gonna say makes them feel better damon.\r\n\r\nSo what for h
ow long a minute a day what difference does it make. Because in the end when you
 loose somebody ever candle every prayer is not gonna make up the fact the only
thing you have left is a hole in your where that somebody you cared about used t
o be and a rock with your birthday carved into it i am pretty sure about it.\r\n
\r\nSo thanks friend.\r\n"
receiving data...
data : b''
Successfully get the file
connection closed

C:\Users\TCS\Desktop>
```

RESULT:

Thus, transfer of files between client and server using socket programming in python is executed and the output has been verified.

Ex No:2f **CONNECTING AND PING A SERVER USING SOCKETS**

DATE: 24/8/22

AIM:

To connect and ping a Server using Sockets

ALGORITHM:

- Import Socket module in Python.
- Create a socket object.
- Set the default port number of the socket to 80.
- Use gethostname() to retrieve the hostname of the server and translate host name to IPV4 format address.
- Use connect() function to connect to the server by specifying host IP and port number.
- Print the host Ip and Port number and also print “Connection is Successful”.

PROGRAM:

CLIENT CODE:

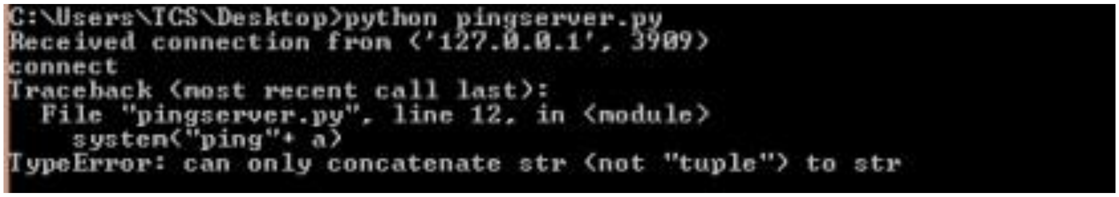
```
from socket import *

from os import system

s = socket(AF_INET, SOCK_STREAM)

s.connect(("127.0.0.1",8000)) # Connect

op='connect'

s.send(op.encode('utf-8')) # Send request

data = s.recv(100).decode()# Get response print(data)

system("ping "+ gethostname())

s.close()
```

SERVER CODE:

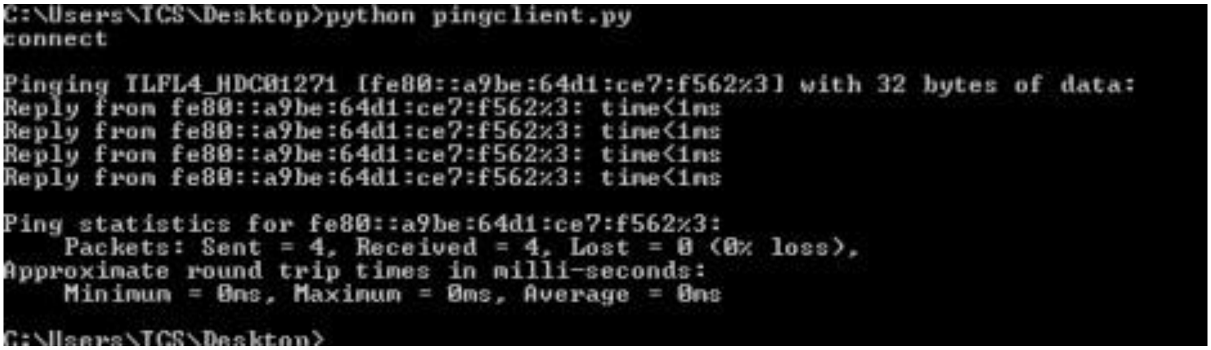
```
from socket import *

from os import system

s = socket(AF_INET,SOCK_STREAM)

s.bind(("",8000))

s.listen(5)
```

```
while True:
        c,a = s.accept()
        print("Received connection from", a)
        data=c.recv(100).decode()
        print(data)
        c.send(data.encode('utf-8'))
        result = "ping" + str(a)
        system(result)
c.close()
```

OUTPUT:



```
C:\Users\TCS\Desktop>python pingserver.py
Received connection from ('127.0.0.1', 3909)
connect
Traceback (most recent call last):
  File "pingserver.py", line 12, in <module>
    system("ping"+ a)
TypeError: can only concatenate str (not "tuple") to str
```



```
C:\Users\TCS\Desktop>python pingclient.py
connect

Pinging ILFL4_HDC01271 [fe80::a9be:64d1:ce7:f562%3] with 32 bytes of data:
Reply from fe80::a9be:64d1:ce7:f562%3: time<1ms
Reply from fe80::a9be:64d1:ce7:f562%3: time<1ms
Reply from fe80::a9be:64d1:ce7:f562%3: time<1ms
Reply from fe80::a9be:64d1:ce7:f562%3: time<1ms

Ping statistics for fe80::a9be:64d1:ce7:f562%3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\TCS\Desktop>
```

RESULT:

Thus, a program to connect and ping a Server using Sockets has been written, executed and the output was verified successfully.

Ex No: 3 IMPLEMENTATION OF PACKET SNIFFING USING RAW SOCKET

DATE: 26/8/22

AIM:

To study packet sniffing concept and implement it using raw sockets.

How To Parse/Extract Captured Packets?

```
import socket
import struct
import binascii
s= socket.socket(socket.AF_INET,socket.SOCK_RAW,socket.IPPROTO_IP)
s.bind(("127.0.0.1",0))
packet=s.recvfrom(65565)
print(packet)
ethernet_header = packet[0][0:14]
eth_header = struct.unpack("!6s6s2s", ethernet_header)
print("ETHERNET HEADER")
print("****************")
print("Destination Address")
print( binascii.hexlify(eth_header[0]))
print("Source Address")
print( binascii.hexlify(eth_header[1]))
print("Type")
print( binascii.hexlify(eth_header[2]))
print("IP HEADER")
print("*********")
ipheader = packet[0][14:34]
ip_header = struct.unpack("!12s4s4s", ipheader)
print("Destination Address")
print (socket.inet_ntoa(ip_header[1]))
print("Source Address")
print(socket.inet_ntoa(ip_header[2]))
```

OUTPUT:

```
(b'E\x00\x004\x00\x87@\x00\x80\x06\x00\x00\x7f\x00\x00\x01\x7f\x00\x00\x01\xd8_\x17a\x9
5\x08fs\x00\x00\x00\x00\x80\x02\xff\xff\x8b\xad\x00\x00\x02\x04\xff\xd7\x01\x03\x03\x08\x0
1\x01\x04\x02', ('127.0.0.1', 0))
ETHERNET HEADER
```

```
****************
Destination Address
b'450000340087'
Source Address
b'400080060000'
Type
b'7f00'
IP HEADER
*********
Destination Address
102.115.0.0
Source Address
0.0.128.2
```

RESULT:

Thus, a study on packet sniffing concept and implement it using raw sockets was done

Ex No: 4a **STUDY OF REMOTE PROCEDURE CALL- XMLRPC**
DATE: 2/9/22

AIM:

To study the concepts of Remote Procedure Call-XML RPC.

Sample Code for Arithmetic operations using RPC

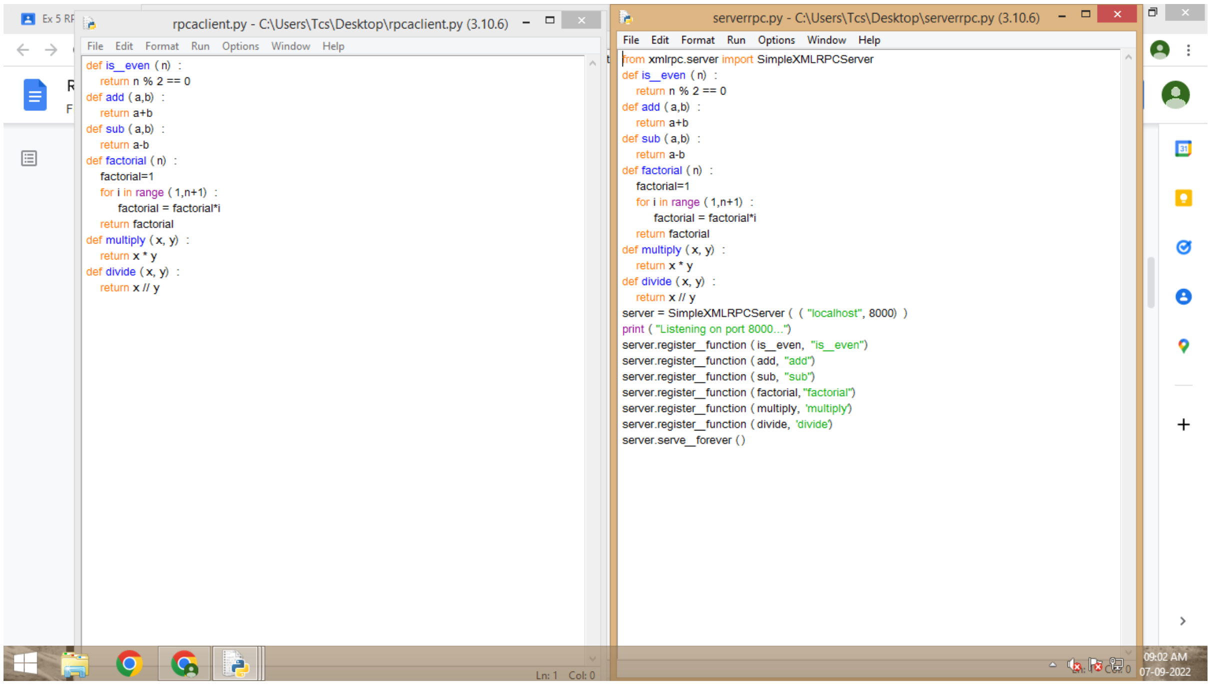
XML RPC PROGRAM- SERVER SIDE:

```
from xmlrpc.server import SimpleXMLRPCServer
# Define a function
def is_even(n):
   return n % 2 == 0
def add(a,b):
   return a+b
def sub(a,b):
   return a-b
def factorial(n):
   factorial=1
   for i in range(1,n+1):
       factorial = factorial*i
   return factorial
def multiply(x, y):
   return x * y
def divide(x, y):
   return x // y
# Create server
server = SimpleXMLRPCServer(("localhost", 8000))
print("Listening on port 8000...")
# Register a function under a different name
server.register_function(is_even, "is_even")
server.register_function(add, "add")
server.register_function(sub, "sub")
server.register_function(factorial,"factorial")
#server.register_function(factorial,"factorial")
server.register_function(multiply, 'multiply')
server.register_function(divide, 'divide')
# Run the server's main loop
server.serve_forever()
```

XML RPC PROGRAM- CLIENT SIDE:

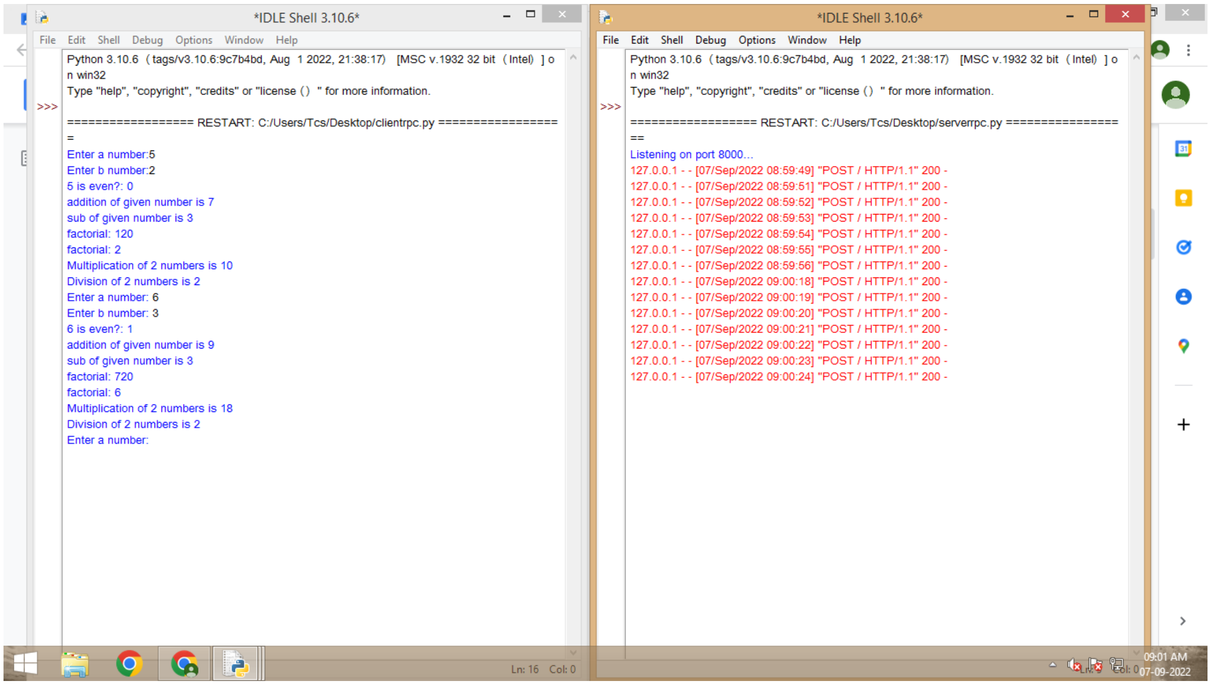
```
import xmlrpc.client
proxy= xmlrpc.client.ServerProxy('http://localhost:8000/') # local server
for i in range(5):
    a=int(input("Enter a number:"))
    b=int(input("Enter b number:"))
    print("%d is even?: %d" % (a, (proxy.is_even(a)))) #access XML-RPC server through proxy
    print("addition of given number is %d "%((proxy.add(a,b))))
    print("sub of given number is %d "%((proxy.sub(a,b))))
    print("factorial: %d" %((proxy.factorial(a))))
    print("factorial: %d" %((proxy.factorial(b))))
            print("Multiplication of 2 numbers is %d" %(proxy.multiply(a,b))
    print("Division of 2 numbers is %d" %(proxy.divide(a,b))
```

OUTPUT:


Ex 5 RP
rpcaclient.py - C:\Users\Tcs\Desktop\rpcaclient.py (3.10.6)
File Edit Format Run Options Window Help
serverrpc.py - C:\Users\Tcs\Desktop\serverrpc.py (3.10.6)
File Edit Format Run Options Window Help
Ln: 1 Col: 0
09:02 AM
07-09-2022

```
def is__even ( n)  :
   return n % 2 == 0
def add ( a,b)  :
   return a+b
def sub ( a,b)  :
   return a-b
def factorial ( n)  :
   factorial=1
   for i in range ( 1,n+1)  :
       factorial = factorial*i
   return factorial
def multiply ( x, y)  :
   return x * y
def divide ( x, y)  :
   return x // y
```

```
from xmlrpc.server import SimpleXMLRPCServer
def is__even ( n)  :
   return n % 2 == 0
def add ( a,b)  :
   return a+b
def sub ( a,b)  :
   return a-b
def factorial ( n)  :
   factorial=1
   for i in range ( 1,n+1)  :
       factorial = factorial*i
   return factorial
def multiply ( x, y)  :
   return x * y
def divide ( x, y)  :
   return x // y
server = SimpleXMLRPCServer (  ( "localhost", 8000)  )
print ( "Listening on port 8000...")
server.register__function ( is__even,  "is__even")
server.register__function ( add,  "add")
server.register__function ( sub,  "sub")
server.register__function ( factorial, "factorial")
server.register__function ( multiply,  'multiply')
server.register__function ( divide,  'divide')
server.serve__forever ()
```


IDLE Shell 3.10.6
File Edit Shell Debug Options Window Help
Ln: 16 Col: 0
IDLE Shell 3.10.6
File Edit Shell Debug Options Window Help
09:01 AM
07-09-2022

```
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug  1 2022, 21:38:17) [MSC v.1932 32 bit (Intel)] o
n win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: C:/Users/Tcs/Desktop/clientrpc.py =================
=
Enter a number:5
Enter b number:2
5 is even?: 0
addition of given number is 7
sub of given number is 3
factorial: 120
factorial: 2
Multiplication of 2 numbers is 10
Division of 2 numbers is 2
Enter a number: 6
Enter b number: 3
6 is even?: 1
addition of given number is 9
sub of given number is 3
factorial: 720
factorial: 6
Multiplication of 2 numbers is 18
Division of 2 numbers is 2
Enter a number:
```

```
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug  1 2022, 21:38:17) [MSC v.1932 32 bit (Intel)] o
n win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: C:/Users/Tcs/Desktop/serverrpc.py ================
==
Listening on port 8000...
127.0.0.1 - - [07/Sep/2022 08:59:49] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 08:59:51] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 08:59:52] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 08:59:53] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 08:59:54] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 08:59:55] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 08:59:56] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 09:00:18] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 09:00:19] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 09:00:20] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 09:00:21] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 09:00:22] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 09:00:23] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 09:00:24] "POST / HTTP/1.1" 200 -
```

RESULT:

Thus, Arithmetic operations using Remote Procedure Call has been successfully executed

EX NO: 4b **REMOTE PROCEDURE CALL FOR LIST OPERATIONS XMLRPC**
DATE: 7/9/22

AIM:

To Implement an XML RPC code for the following functions,

a. No of items in a list
b. Smallest element in a list
c. Largest element in the list
d. Converting a list to a set.

PROGRAM:

SERVER SIDE:

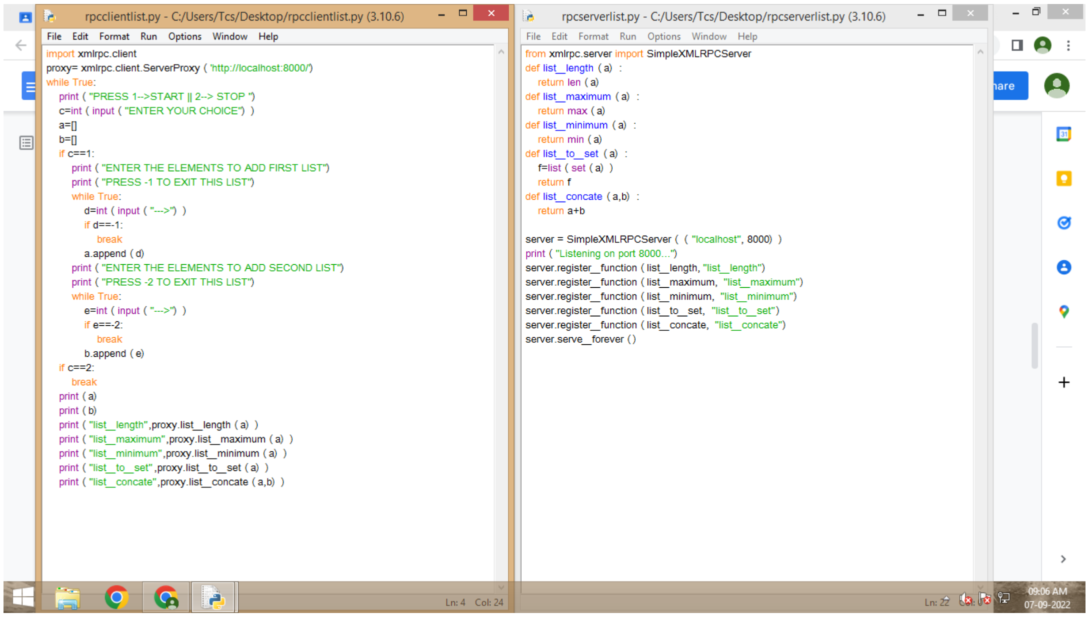
```
from xmlrpc.server import SimpleXMLRPCServer
def list_length(a):
   return len(a)
def list_maximum(a):
   return max(a)
def list_minimum(a):
   return min(a)
def list_to_set(a):
   f=list(set(a))
   return f
def list_concate(a,b):
   return a+b
server = SimpleXMLRPCServer(("localhost", 8000))
print("Listening on port 8000...")
server.register_function(list_length,"list_length")
server.register_function(list_maximum, "list_maximum")
server.register_function(list_minimum, "list_minimum")
server.register_function(list_to_set, "list_to_set")
server.register_function(list_concate, "list_concate")
server.serve_forever()
```

CLIENT SIDE:

```
import xmlrpc.client
proxy= xmlrpc.client.ServerProxy('http://localhost:8000/')
while True:
   print("PRESS 1-->STRAT || 2--> STOP ")
```

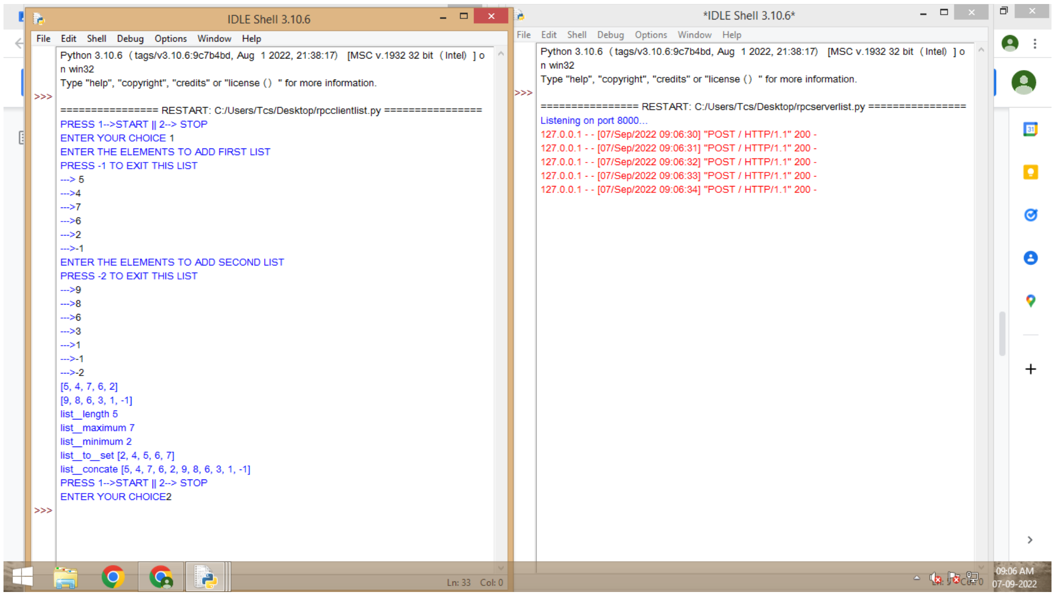
```
c=int(input("ENTER YOUR CHOICE"))
a=[]
b=[]
if c==1:
    print("ENTER THE ELEMENTS TO ADD FIRST LIST")
    print("PRESS -1 TO EXIT THIS LIST")
    while True:
        d=int(input("--->"))
        if d==-1:
            break
        a.append(d)
    print("ENTER THE ELEMENTS TO ADD SECOND LIST")
    print("PRESS -2 TO EXIT THIS LIST")
    while True:
        e=int(input("--->"))
        if e==-2:
            break
        b.append(e)
if c==2:
    break
print(a)
print(b)
print("list_length",proxy.list_length(a))
print("list_maximum",proxy.list_maximum(a))
print("list_minimum",proxy.list_minimum(a))
print("list_to_set",proxy.list_to_set(a))
print("list_concate",proxy.list_concate(a,b))
```

OUTPUT:


rpcclientlist.py - C:/Users/Tcs/Desktop/rpcclientlist.py (3.10.6)
File Edit Format Run Options Window Help
rpcserverlist.py - C:/Users/Tcs/Desktop/rpcserverlist.py (3.10.6)
File Edit Format Run Options Window Help
Ln: 4 Col: 24
Ln: 22
09:06 AM
07-09-2022

```
import xmlrpc.client
proxy= xmlrpc.client.ServerProxy ( 'http://localhost:8000/')
while True:
    print ( "PRESS 1-->START || 2--> STOP ")
    c=int ( input ( "ENTER YOUR CHOICE") )
    a=[]
    b=[]
    if c==1:
        print ( "ENTER THE ELEMENTS TO ADD FIRST LIST")
        print ( "PRESS -1 TO EXIT THIS LIST")
        while True:
            d=int ( input ( "--->") )
            if d==-1:
                break
            a.append ( d)
        print ( "ENTER THE ELEMENTS TO ADD SECOND LIST")
        print ( "PRESS -2 TO EXIT THIS LIST")
        while True:
            e=int ( input ( "--->") )
            if e==-2:
                break
            b.append ( e)
    if c==2:
        break
    print ( a)
    print ( b)
    print ( "list__length",proxy.list__length ( a) )
    print ( "list__maximum",proxy.list__maximum ( a) )
    print ( "list__minimum",proxy.list__minimum ( a) )
    print ( "list__to__set",proxy.list__to__set ( a) )
    print ( "list__concate",proxy.list__concate ( a,b) )
```

```
from xmlrpc.server import SimpleXMLRPCServer
def list__length ( a) :
   return len ( a)
def list__maximum ( a) :
   return max ( a)
def list__minimum ( a) :
   return min ( a)
def list__to__set ( a) :
   f=list ( set ( a) )
   return f
def list__concate ( a,b) :
   return a+b

server = SimpleXMLRPCServer ( ( "localhost", 8000) )
print ( "Listening on port 8000...")
server.register__function ( list__length, "list__length")
server.register__function ( list__maximum, "list__maximum")
server.register__function ( list__minimum, "list__minimum")
server.register__function ( list__to__set, "list__to__set")
server.register__function ( list__concate, "list__concate")
server.serve__forever ()
```


IDLE Shell 3.10.6
File Edit Shell Debug Options Window Help
IDLE Shell 3.10.6
File Edit Shell Debug Options Window Help
Ln: 33 Col: 0
09:06 AM
07-09-2022

```
Python 3.10.6 ( tags/v3.10.6:9c7b4bd, Aug  1 2022, 21:38:17)  [MSC v.1932 32 bit ( Intel) ] o
n win32
Type "help", "copyright", "credits" or "license ()  " for more information.
>>>
================ RESTART: C:/Users/Tcs/Desktop/rpcclientlist.py ================
PRESS 1-->START || 2--> STOP
ENTER YOUR CHOICE 1
ENTER THE ELEMENTS TO ADD FIRST LIST
PRESS -1 TO EXIT THIS LIST
---> 5
--->4
--->7
--->6
--->2
--->-1
ENTER THE ELEMENTS TO ADD SECOND LIST
PRESS -2 TO EXIT THIS LIST
--->9
--->8
--->6
--->3
--->1
--->-1
--->-2
[5, 4, 7, 6, 2]
[9, 8, 6, 3, 1, -1]
list__length 5
list__maximum 7
list__minimum 2
list__to__set [2, 4, 5, 6, 7]
list__concate [5, 4, 7, 6, 2, 9, 8, 6, 3, 1, -1]
PRESS 1-->START || 2--> STOP
ENTER YOUR CHOICE2
>>>
```

```
Python 3.10.6 ( tags/v3.10.6:9c7b4bd, Aug  1 2022, 21:38:17)  [MSC v.1932 32 bit ( Intel) ] o
n win32
Type "help", "copyright", "credits" or "license ()  " for more information.
>>>
================ RESTART: C:/Users/Tcs/Desktop/rpcserverlist.py ================
Listening on port 8000...
127.0.0.1 - - [07/Sep/2022 09:06:30] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 09:06:31] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 09:06:32] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 09:06:33] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2022 09:06:34] "POST / HTTP/1.1" 200 -
```

RESULT:

Thus, the program for list operations using Remote Procedure Call has been executed successfully.

EX NO: 4C **REMOTE PROCEDURE CALL FOR SORTING AN ARRAY - XMLRPC**

DATE: 9/9/22

AIM:

To Implement an XML RPC code for sorting array.

PROGRAM:

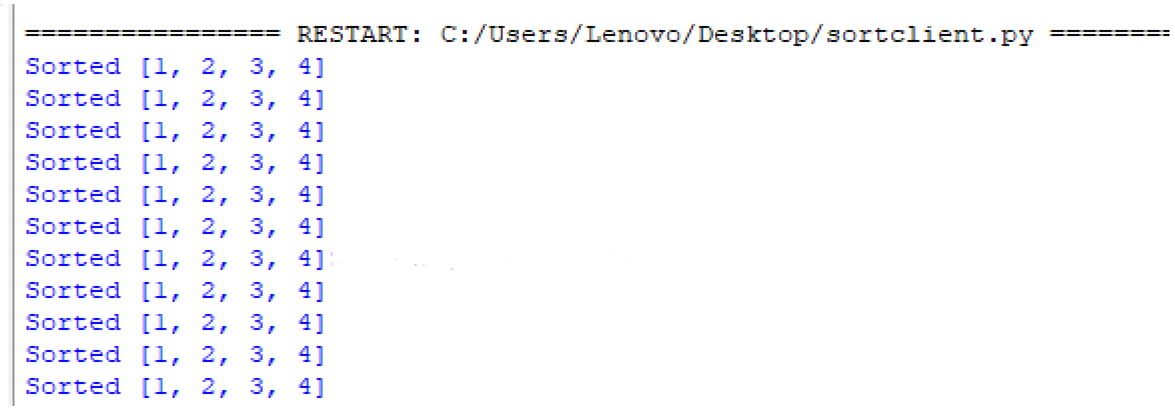
SERVER SIDE:

```
from xmlrpc.server import SimpleXMLRPCServer
def selection_sort(x):
        x.sort()
return x
server = SimpleXMLRPCServer(("localhost", 8000))
print("Listening on port 8000...")
server.register_function(selection_sort,"selection_sort")
server.serve_forever()
```

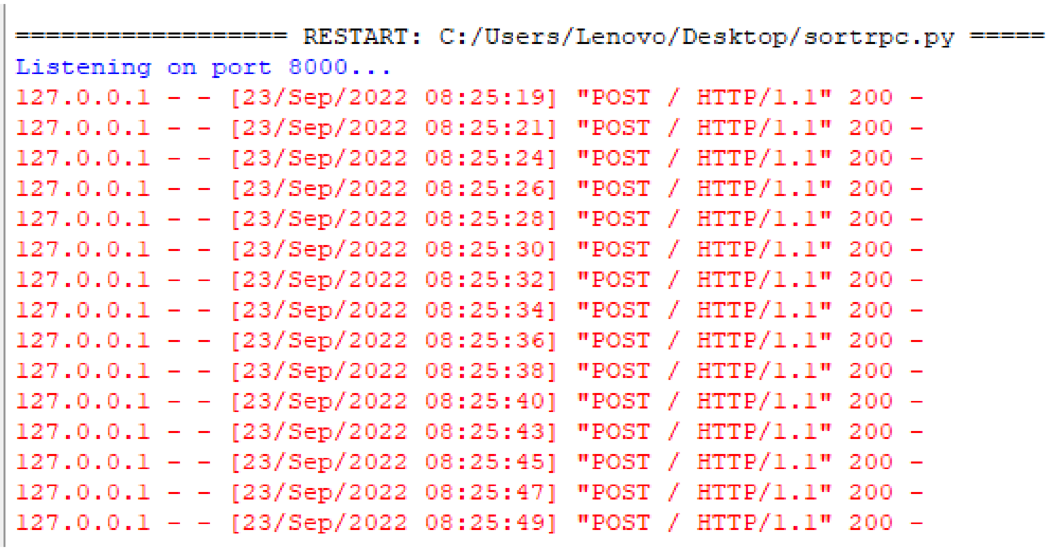
CLIENT SIDE:

```
import xmlrpc.client
proxy= xmlrpc.client.ServerProxy('http://localhost:8000/')
while True:
        a = [1, 3, 4, 2]
        print("Sorted",proxy.selection_sort(a))
```

OUTPUT:



```
================ RESTART: C:/Users/Lenovo/Desktop/sortclient.py ========
Sorted [1, 2, 3, 4]
Sorted [1, 2, 3, 4]
Sorted [1, 2, 3, 4]
Sorted [1, 2, 3, 4]
Sorted [1, 2, 3, 4]
Sorted [1, 2, 3, 4]
Sorted [1, 2, 3, 4]
Sorted [1, 2, 3, 4]
Sorted [1, 2, 3, 4]
Sorted [1, 2, 3, 4]
Sorted [1, 2, 3, 4]
```



```
================== RESTART: C:/Users/Lenovo/Desktop/sortrpc.py =====
Listening on port 8000...
127.0.0.1 - - [23/Sep/2022 08:25:19] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/Sep/2022 08:25:21] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/Sep/2022 08:25:24] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/Sep/2022 08:25:26] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/Sep/2022 08:25:28] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/Sep/2022 08:25:30] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/Sep/2022 08:25:32] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/Sep/2022 08:25:34] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/Sep/2022 08:25:36] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/Sep/2022 08:25:38] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/Sep/2022 08:25:40] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/Sep/2022 08:25:43] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/Sep/2022 08:25:45] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/Sep/2022 08:25:47] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/Sep/2022 08:25:49] "POST / HTTP/1.1" 200 -
```

RESULT:

Thus, Remote Procedure Call for sorting an array has been successfully executed.

Ex No: 5 **STUDY OF PACKET TRACER-Installation and User Interface Overview**

DATE: 22/9/22

AIM:

To study the Packet tracer tool Installation and User Interface Overview

ALGORITHM:

INSTALLING PACKET TRACER:

To download Packet Tracer, go to https://www.netacad.com and log in with your Cisco Networking Academy credentials; then, click on the Packet Tracer graphic and download the package appropriate for your operating system. (Can be used to download in your laptop).

Windows

Installation in Windows is pretty simple and straightforward; the setup comes in a single file named Packettracer_Setup6.0.1.exe. Open this file to begin the setup wizard, accept the license agreement, choose a location, and start the installation.

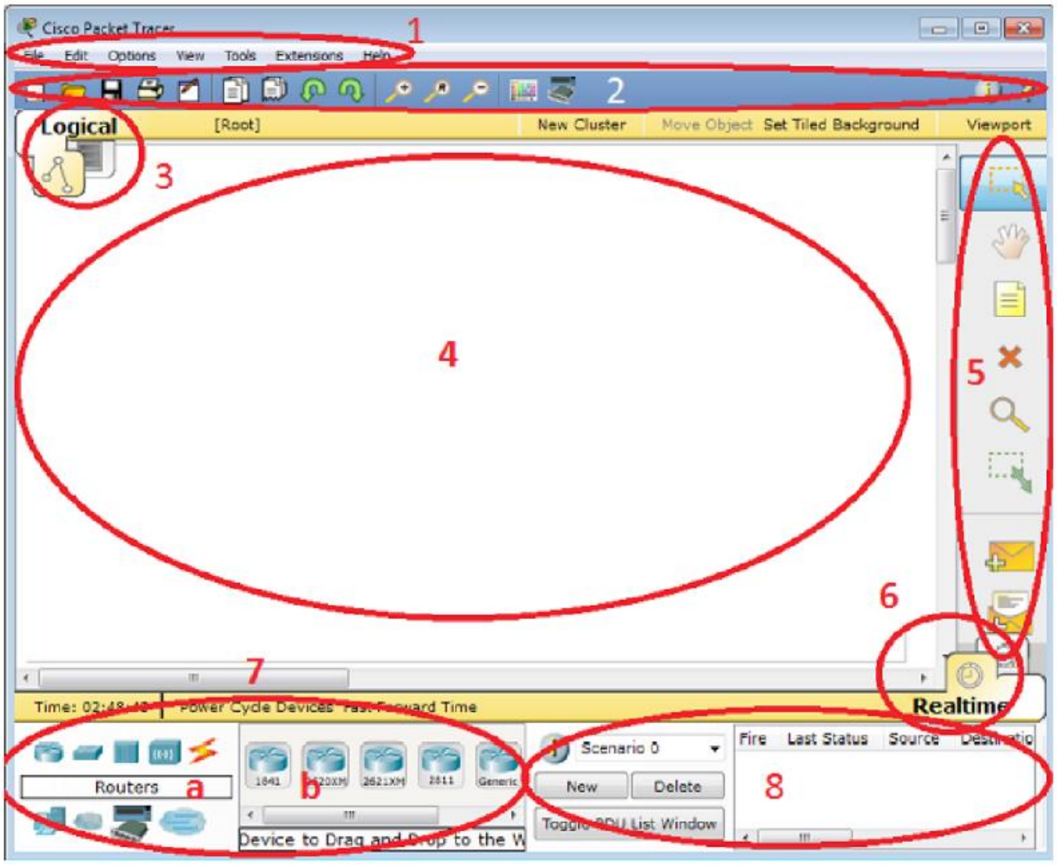
Linux

Linux users with an Ubuntu/Debian distribution should download the file for Ubuntu, and those using Fedora/Redhat/CentOS must download the file for Fedora. Grant executable permission to this file by using chmod, and execute it to begin the installation.

```
chmod +x PacketTracer601_i386_installer-rpm.bin
./PacketTracer601_i386_installer-rpm.bin
```

USER INTERFACE OVERVIEW:

The layout of Packet Tracer is divided into several components. The components of the Packet Tracer interface are as follows: match the numbering with explanations



1. Menu bar – This is a common menu found in all software applications; it is used to open, save, print, change preferences, and so on.

2. Main toolbar – This bar provides shortcut icons to menu options that are commonly accessed, such as open, save, zoom, undo, and redo, and on the right-hand side is an icon for entering network information for the current network.

3. Logical/Physical workspace tabs – These tabs allow you to toggle between the Logical and Physical work areas.

4. Workspace – This is the area where topologies are created and simulations are displayed.

5. Common tools bar – This toolbar provides controls for manipulating topologies, such as select, move layout, place note, delete, inspect, resize shape, and add simple/complex PDU.

6. Real-time/Simulation tabs – These tabs are used to toggle between the real and simulation modes. Buttons are also provided to control the time, and to capture the

packets.

7. Network component box – This component contains all of the network and end devices available with Packet Tracer, and is further divided into two areas: Area 7a: Device-type selection box – This area contains device categories Area 7b: Device-specific selection box – When a device category is selected, this selection box displays the different device models within that category

8. User-created packet box – Users can create highly-customized packets to test their topology from this area, and the results are displayed as a list.

RESULT:

Thus, STUDY OF PACKET TRACER has been successfully executed

Ex No:6a **REATING A SIMPLE NETWORK TOPOLOGY**

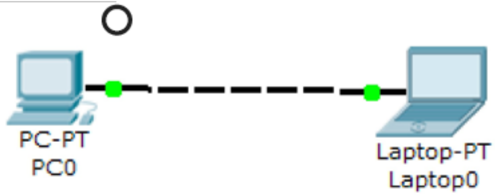
DATE: 27/9/22

AIM:

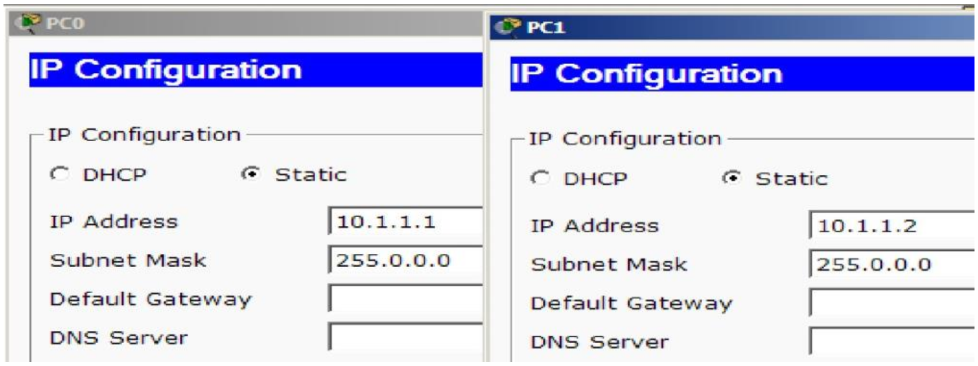
To create a simple network topology using packet tracer-

SAMPLE 1: A SIMPLE TOPOLOGY WITH TWO END DEVICES

1. From the network component box, click on End Devices and drag-and-drop a Generic PC icon and a Generic laptop icon into the Workspace.
2. Click on Connections, then click on Copper Cross-Over, then on PC0, and select Fast Ethernet. After this, click on Laptop0 and select Fast Ethernet. The link status LED should show up in green, indicating that the link is up.



3. Click on the PC, go to the Desktop tab, click on IP Configuration, and enter an IP address and subnet mask. In this topology, the default gateway and DNS server information is not needed as there are only two end devices in the network.
4. Close the window, open the laptop, and assign an IP address to it in the same way. Make sure that both IP addresses are in the same subnet.


PC0

IP Configuration

IP Configuration

○ DHCP ● Static

IP Address 10.1.1.1

Subnet Mask 255.0.0.0

Default Gateway

DNS Server

PC1

IP Configuration

IP Configuration

○ DHCP ● Static

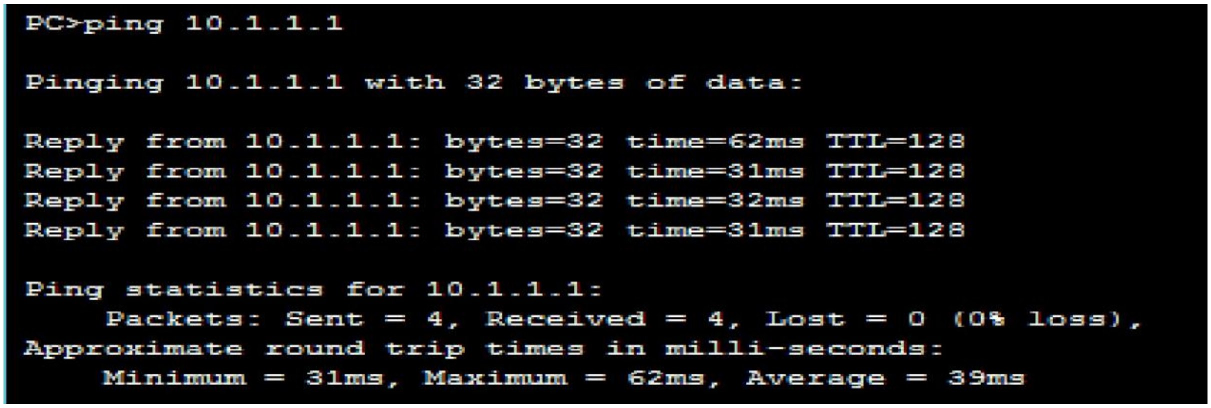
IP Address 10.1.1.2

Subnet Mask 255.0.0.0

Default Gateway

DNS Server

5. Close the IP Configuration box, open the command prompt, and ping the IP address of the device at the end to check connectivity. (in desktop tab, command prompt is also there)

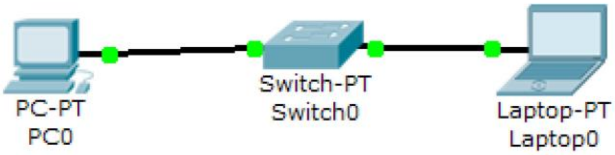


```
PC>ping 10.1.1.1

Pinging 10.1.1.1 with 32 bytes of data:

Reply from 10.1.1.1: bytes=32 time=62ms TTL=128
Reply from 10.1.1.1: bytes=32 time=31ms TTL=128
Reply from 10.1.1.1: bytes=32 time=32ms TTL=128
Reply from 10.1.1.1: bytes=32 time=31ms TTL=128

Ping statistics for 10.1.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 31ms, Maximum = 62ms, Average = 39ms
```

SAMPLE 2: TOPOLOGY WITH NETWORK DEVICE AND END DEVICES

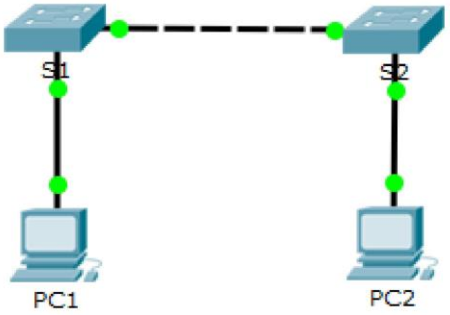
This topology uses a network device (Ethernet switch) so that more than two end devices can be connected, by performing the following steps:

1. Click on Switches from the device-type selection box and insert any switch (except Switch-PT-Empty) into the workspace.
2. From the network component box, click on End Devices and drag-and-drop a Generic PC icon and a Generic laptop icon into the Workspace.
3. Choose the Copper Straight-Through cable and connect the PC and laptop with the switch. At this point, the link indicators on the switch are orange in color because the switch ports are undergoing the listening and learning states of the Spanning Tree Protocol (STP).

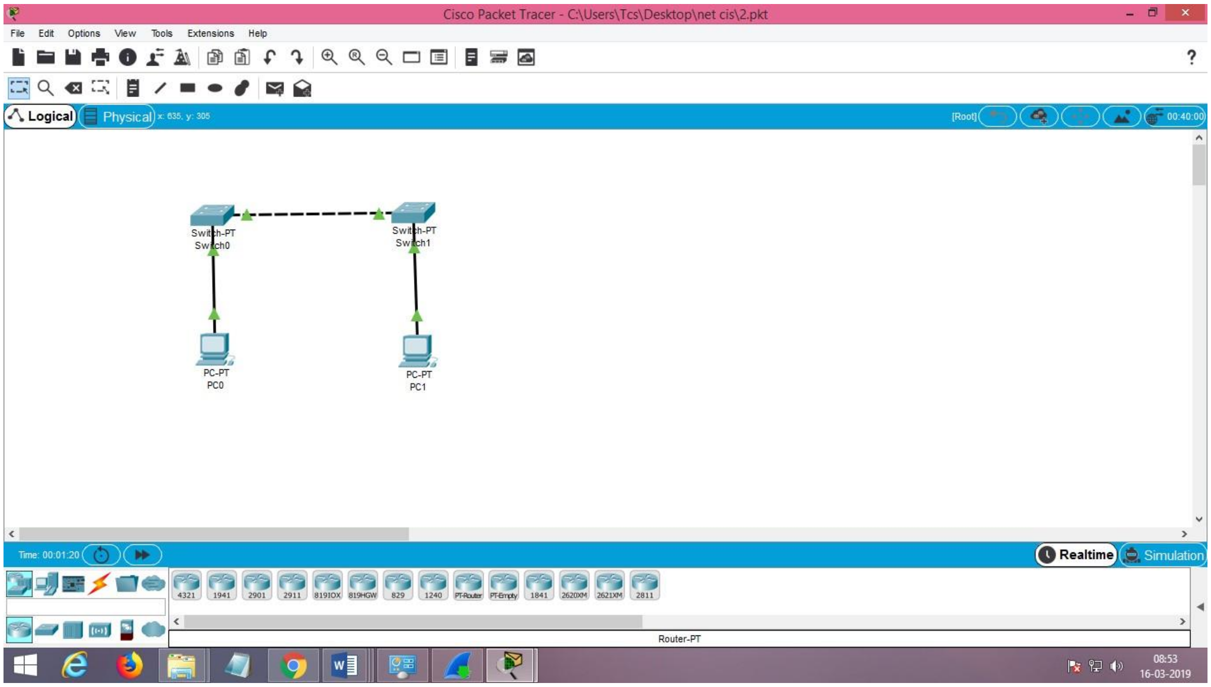


4. Once the link turns green, as shown in the previous screenshot, ping again to check the connectivity.
5. To save this topology, navigate to File | Save As and choose a location. The topology will be saved with a.pkt extension, with the devices in the same state.

Exercise:
Create a Simple topology as shown below and configure the PCs and switch s1 and s2.



OUTPUT:



1. Click on Switches from the device-type selection box and insert 2 switches (except Switch-PT-Empty) into the workspace.
2. From the network component box, click on End Devices and drag-and-drop 2 Generic PC icon.
3. Choose the Copper Straight-Through cable and connect the 2 PCs with the switches. At this point, the link indicators on the switch are orange in color because the switch ports are undergoing the listening and learning states of the Spanning Tree Protocol (STP).

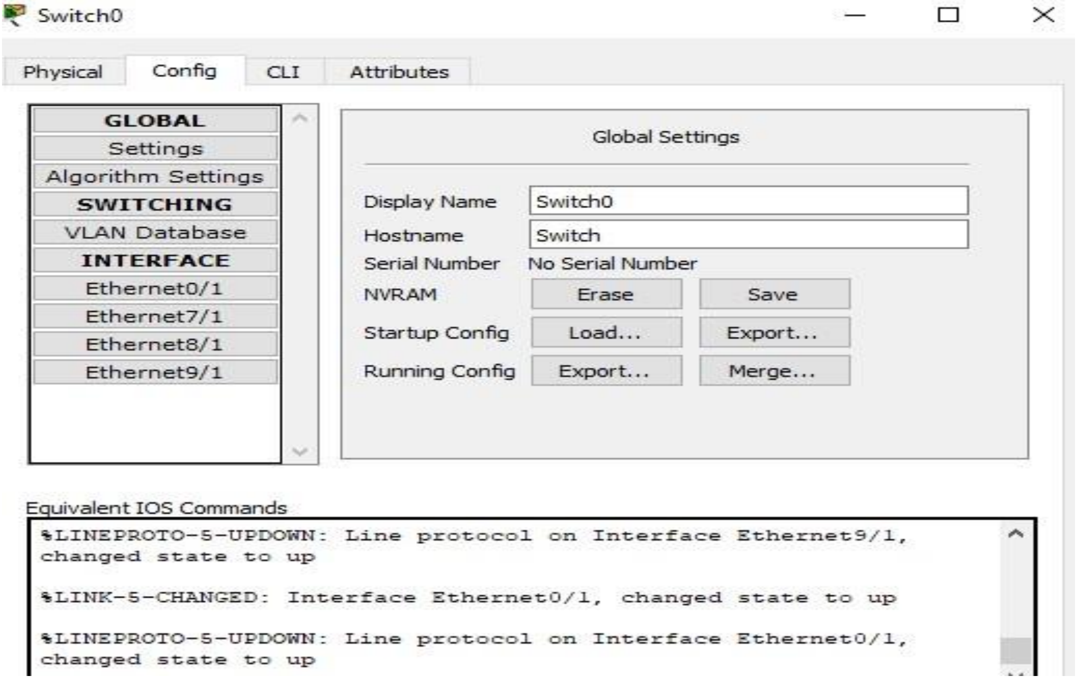
RESULT:

Thus, REATING A SIMPLE NETWORK TOPOLOGY has been successfully executed

Ex NO:6b **CONFIGURING A NETWORK DEVICE (ROUTERS AND SWITCHES)**
DATE: 28/9/22

AIM:
To configure network devices (switches and routers)

DESCRIPTION:
To configure Cisco routers and switches, Packet Tracer provides a Config tab that contains GUI options for the most common configurations. Config Tab details of a switch is shown in the screen shot.



Display Name: Switch0
Hostname: Switch
Serial Number: No Serial Number
NVRAM
Startup Config
Running Config

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet9/1,
changed state to up

%LINK-5-CHANGED: Interface Ethernet0/1, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/1,
changed state to up
```

Using the Config tab, the following can be configured:

1. Global settings
2. Routing (on a router and a layer 3 switch)
3. VLAN database (on a switch)
4. Interface settings Global settings

This section slightly differs from the switch and the router. Switches have options for modifying the speed and duplex setting and for assigning a port to VLAN. On routers, the VLAN section is replaced by the IP address configuration.

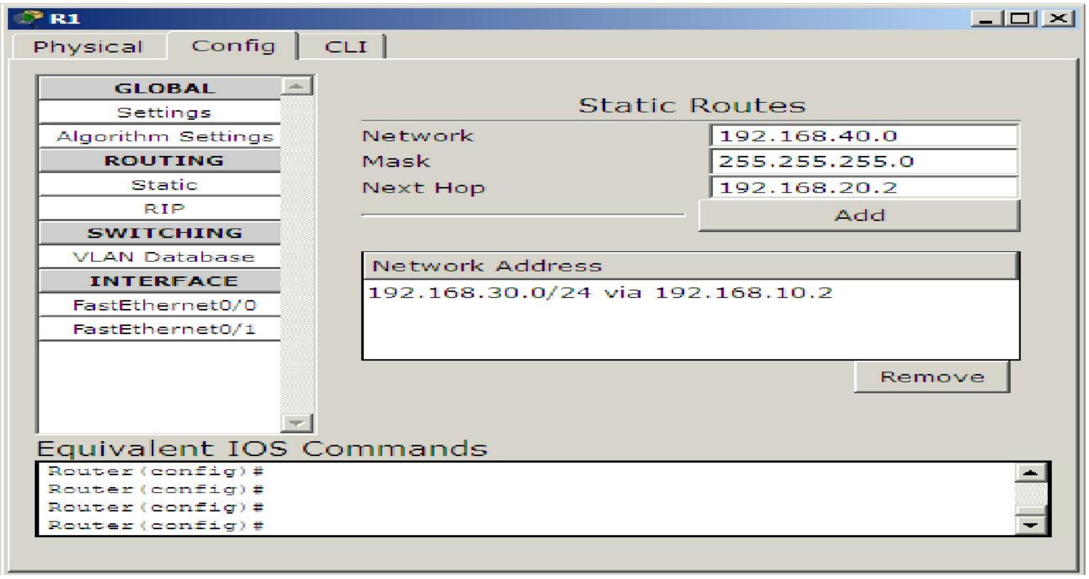
Configuring a Router is done by performing the following steps:

1. Click on a router icon, go to the Config tab, select an interface, and configure the IP address. Make sure that you select the On checkbox in this section to bring the port state up. For example, if there are four router connected to each other then, the following IP addresses can be assigned to the Routers.

Router	Interface	IP Address
R1	**FastEthernet0/0**	192.168.10.1
	FastEthernet0/1	192.168.20.1
R2	**FastEthernet0/0**	192.168.10.2
	FastEthernet0/1	192.168.30.1
R3	**FastEthernet0/0**	192.168.20.2
	FastEthernet0/1	192.168.40.1
R4	**FastEthernet0/0**	192.168.30.2
	FastEthernet0/1	192.168.40.2

Routing

This section has options for configuring Static and dynamic routing (RIP). To configure static routing, enter the network address, netmask, and its next hop address, and then click on Add.



Static Routes

Network 192.168.40.0

Mask 255.255.255.0

Next Hop 192.168.20.2

Network Address

192.168.30.0/24 via 192.168.10.2

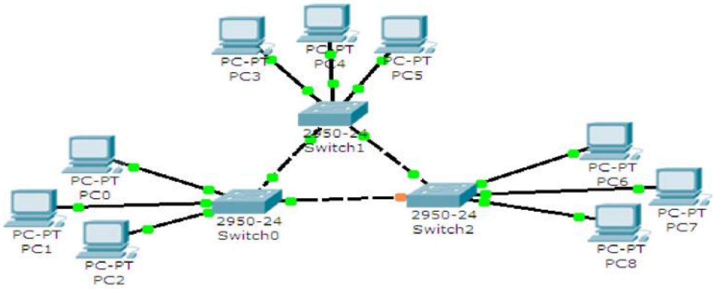
Equivalent IOS Commands

```
Router(config)#
Router(config)#
Router(config)#
Router(config)#
```

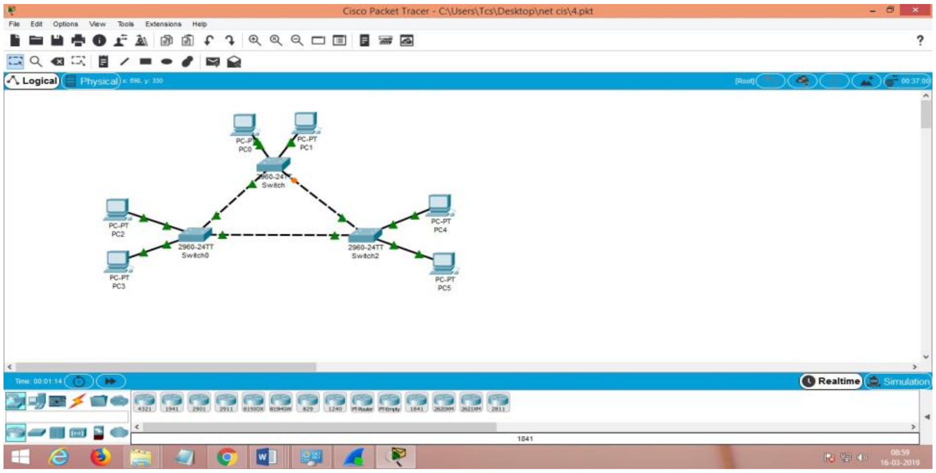
To configure Routing Information Protocol (RIP), it is enough to add only network IP.

Exercises:

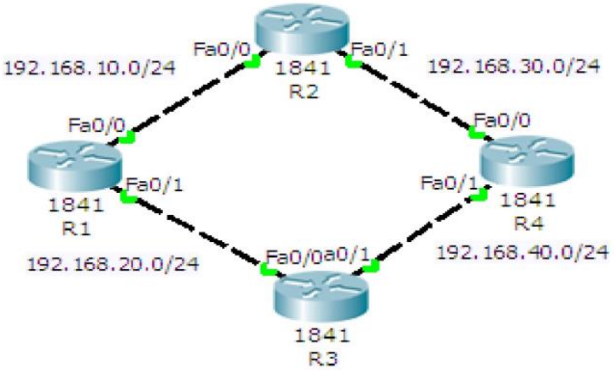
1. Create a topology shown below and configure the Initial Setting. (PC and Switch Configuration)



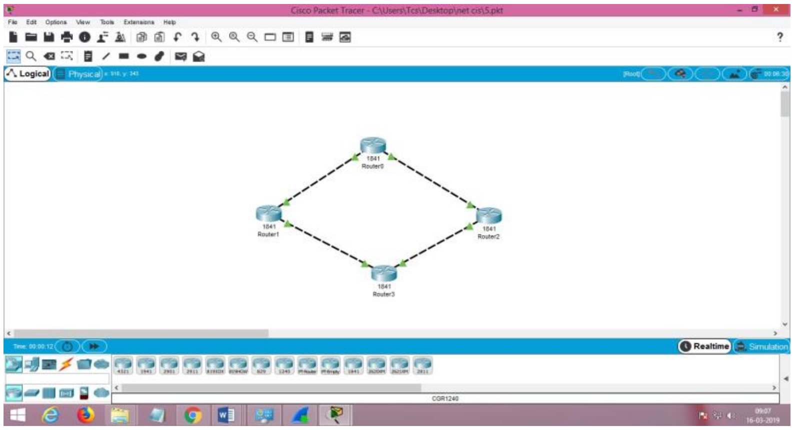
OUTPUT:



2.Peform Static Routing Configuration of Routers connected as shown below



OUTPUT:



Configuring a Router is done by performing the following steps:

1. Click on a router icon, go to the Config tab, select an interface, and configure the IP address. Make sure that you select the On checkbox in this section to bring the port state up. For example, if there are four router connected to each other then, the following IP addresses can be assigned to the Routers.

Router	**Interface**	**IP Address**
R1	**FastEthernet0/0**	192.168.10.1
	FastEthernet0/1	192.168.20.1
R2	**FastEthernet0/0**	192.168.10.2
	FastEthernet0/1	192.168.30.1
R3	**FastEthernet0/0**	192.168.20.2
	FastEthernet0/1	192.168.40.1
R4	**FastEthernet0/0**	192.168.30.2
	FastEthernet0/1	192.168.40.2

RESULT:

Thus, CONFIGURING A NETWORK DEVICE has been successfully executed

Ex NO:6c **TESTING CONNECTIVITY USING SIMPLE AND COMPLEX PDUs PTIONS**
DATE: 30/9/22

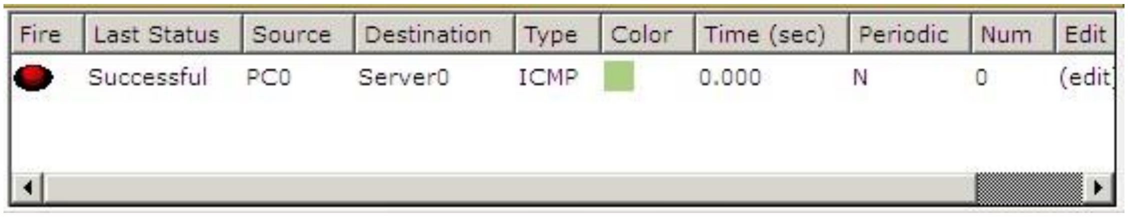
AIM:
To test the connectivity of a network using simple and complex PDUs.

REAL-TIME MODE

Simple PDU
The Add Simple PDU option uses only ICMP (Internet Control Message Protocol). Create a topology with a PC and a server.

1. Add a PC and a server to the workspace and connect them using a copper crossover cable.
2. Assign IP addresses to both of them in the same subnet. Example, PC1:

   192.168.0.1/255.255.255.0 and PC2: 192.168.0.2/255.255.255.0.
3. From the common tools bar, click on the closed envelope icon or use the shortcut key P.
4. The pointer will change to an envelope symbol. Click on the PC first and then on the server. Now look at the User Created Packet box. Status Successful, the source, the destination, and the type of packet that was sent will be shown.

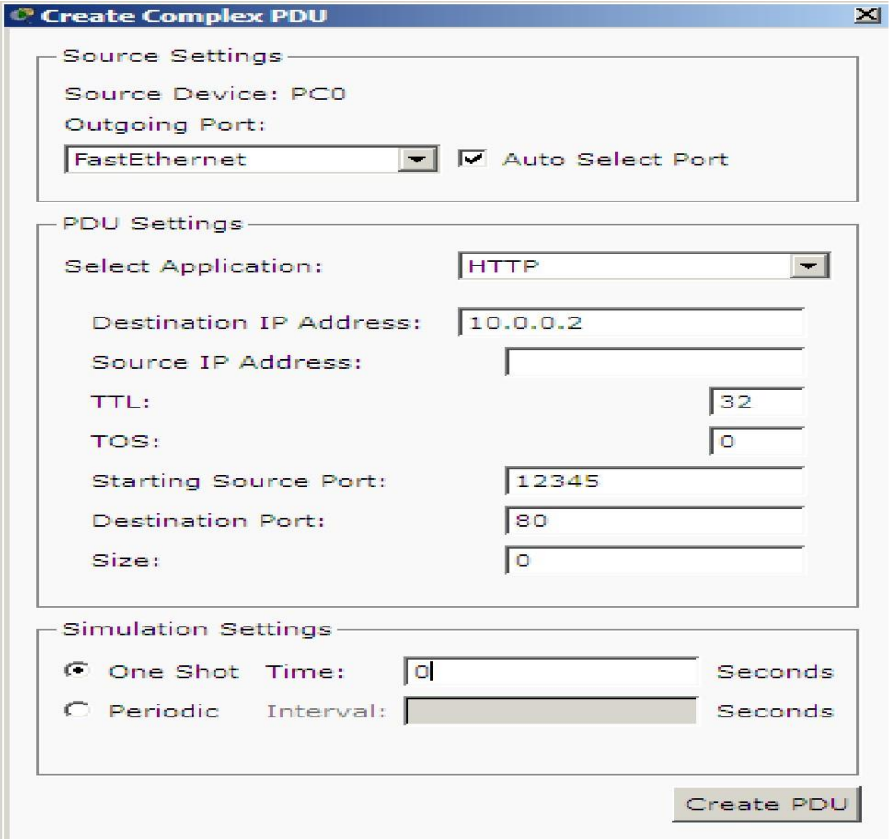


Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit
●	Successful	PC0	Server0	ICMP	🟩	0.000	N	0	(edit)

Complex PDU

Complex PDUs is also shown with the same PC-Server topology:

1. Click on the open envelope icon or press C; this is the Add Complex PDU option.
2. Click on the PC and the Create Complex PDU dialog box opens. Select the application and fill the Destination IP address (IP of the server), Starting Source Port, and Time fields, and then click on the Create PDU button.



Create Complex PDU

Source Settings

Source Device: PC0

Outgoing Port:

FastEthernet ☑ Auto Select Port

PDU Settings

Select Application: HTTP

Destination IP Address: 10.0.0.2

Source IP Address:

TTL: 32

TOS: 0

Starting Source Port: 12345

Destination Port: 80

Size: 0

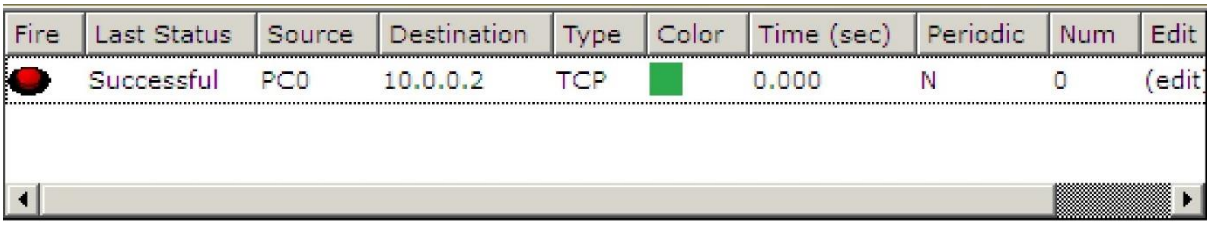
Simulation Settings

● One Shot Time: 0 Seconds

○ Periodic Interval: Seconds

Create PDU

Now click on the server and then look at the user-created packet box. An entry indicates a successful TCP three-way handshake as shown in the following screenshot:

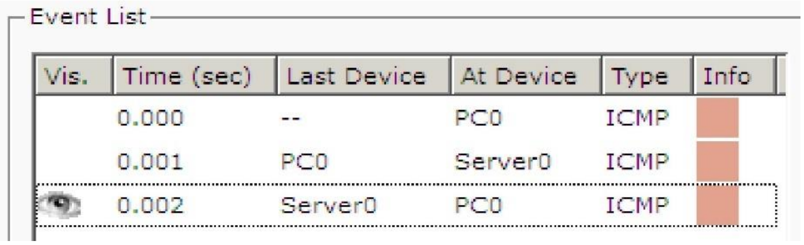


Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit
●	Successful	PC0	10.0.0.2	TCP	🟩	0.000	N	0	(edit)

SIMULATION MODE:

Use the real time/simulation tab to switch to the simulation mode. Using simulation mode, packets flowing from one node to can be seen.

Click on the Auto Capture / Play button to begin packet capture. Try a Simple PDU, as described in the previous section, and the event list will be populated with three entries, indicating the creation of an ICMP packet, ICMP echo sent, and ICMP reply received



Event List

Vis.	Time (sec)	Last Device	At Device	Type	Info
	0.000	--	PC0	ICMP	■
	0.001	PC0	Server0	ICMP	■
👁	0.002	Server0	PC0	ICMP	■

If you click on a packet (the envelope icon), you'll be presented with the packet information categorized according to OSI layers. The Outbound PDU Details tab lists each layer's information in a packet format:

The simulation mode has a Play Controls section that works similar to the controls of a media player and is as follows:

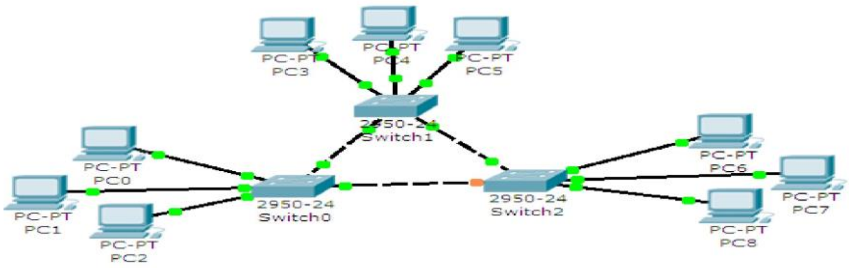
Back: This button moves the process one step back each time it is clicked on.

Auto Capture / Play: Pressing this button results in all of the network traffic (chosen under event filters) being continuously captured until this button is pressed again.

Capture/Forward: This is the manual mode of the previous button. This has to be pressed each time to move the packet from one place to another.

Exercise:

Test the connectivity by sending simple PDU between PC1 and PC7.



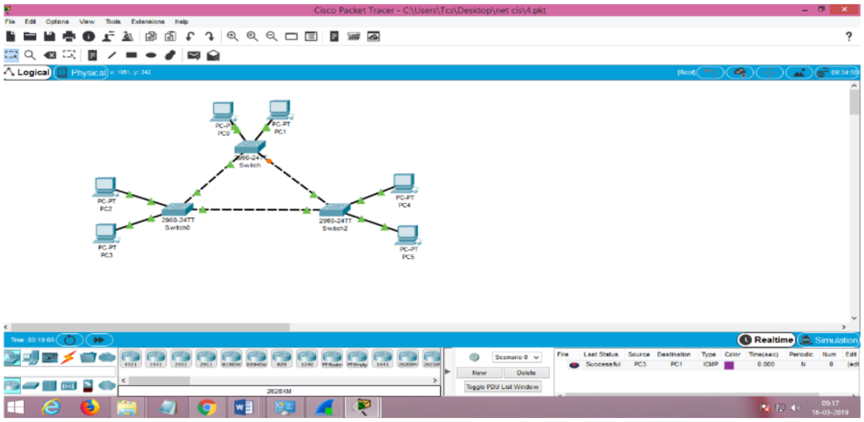
OUTPUT:

Simple PDU

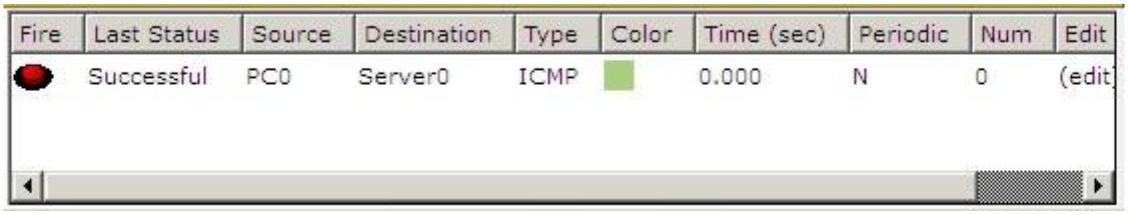
The Add Simple PDU option uses only ICMP (Internet Control Message Protocol). Create a topology with a PC and a server.

5. Add a PC and a server to the workspace and connect them using a copper crossover cable.
6. Assign IP addresses to both of them in the same subnet. Example, PC1:

   192.168.0.1/255.255.255.0 and PC2: 192.168.0.2/255.255.255.0.
7. From the common tools bar, click on the closed envelope icon or use the shortcut key P. The pointer will change to an envelope symbol. Click on the PC first and then on the server.

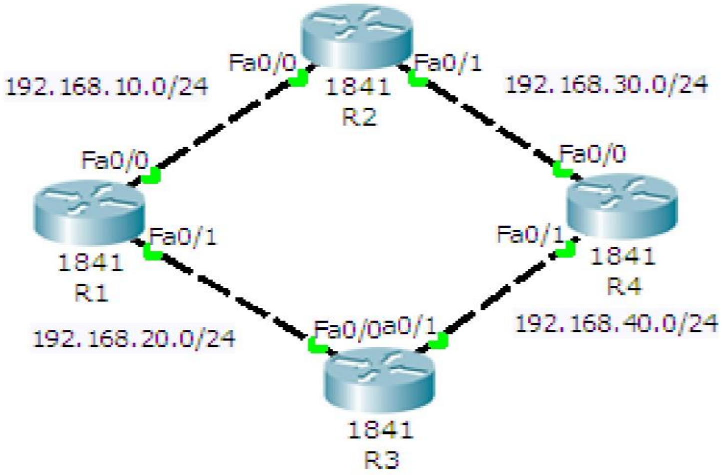


Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
●	Successful	PC3	PC1	ICMP	🟪	0.000	N	0	(edi

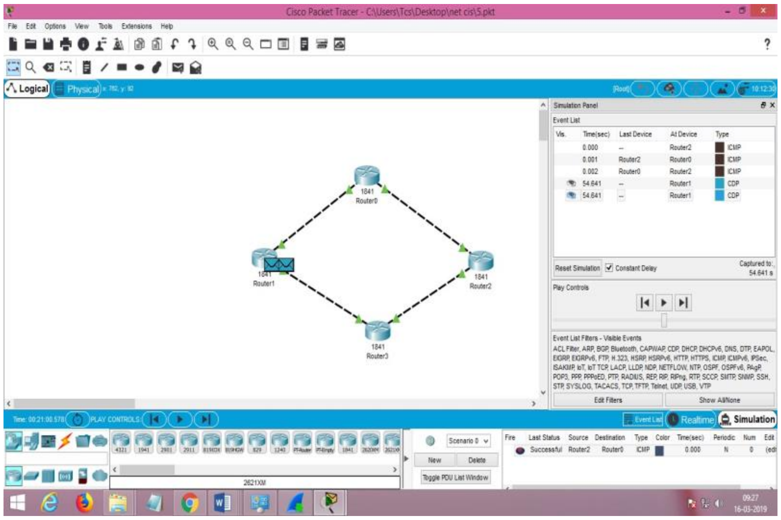


Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit
●	Successful	PC0	Server0	ICMP	🟩	0.000	N	0	(edit)

2.Test the connectivity by sending simple PDU between R2 and R3.



OUTPUT:



Vis.	Time(sec)	Last Device	At Device	Type
	0.000	--	Router2	ICMP
	0.001	Router2	Router0	ICMP
	0.002	Router0	Router2	ICMP
	54.641	--	Router1	CDP
	54.641	--	Router1	CDP

Reset Simulation ☑ Constant Delay Captured to: 54.641 s

Play Controls

Event List Filters - Visible Events

ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Edit Filters Show All/None

Scenario 0 New Delete Toggle PDU List Window

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
●	Successful	Router2	Router0	ICMP	■	0.000	N	0	(edi

09:27 16-03-2019

RESULT:

Thus, CONFIGURING A NETWORK DEVICE has been successfully executed

Ex NO: 6d **EXAMINE THE ARP TABLE**

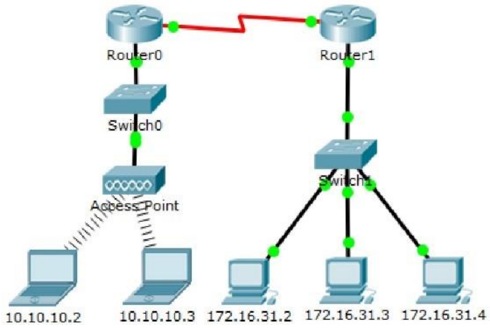
Date: 11/10/22

AIM:

To test the connectivity of a network using simple and complex PDUs.

DESCRIPTION:

TOPOLOGY: Create a topology as shown below.



Address the devices as given in the table

Device	**Interface**	**MAC Address**	**Switch Interface**
Router0	Gg0/0	0001.6458.2501	G0/1
S0/0/0	N/A	N/A	
Router1	G0/0	00E0.F7B1.8901	G0/1
S0/0/0	N/A	N/A	
10.10.10.2	Wireless	0060.2F84.4AB6	F0/2
10.10.10.3	Wireless	0060.4706.572B	F0/2
172.16.31.2	F0	000C.85CC.1DA7	F0/1
172.16.31.3	F0	0060.7036.2849	F0/2
172.16.31.4	G0	0002.1640.8D75	F0/3

Objectives

1. Examine an ARP Request
2. Examine a Switch MAC Address Table

Examine an ARP Request

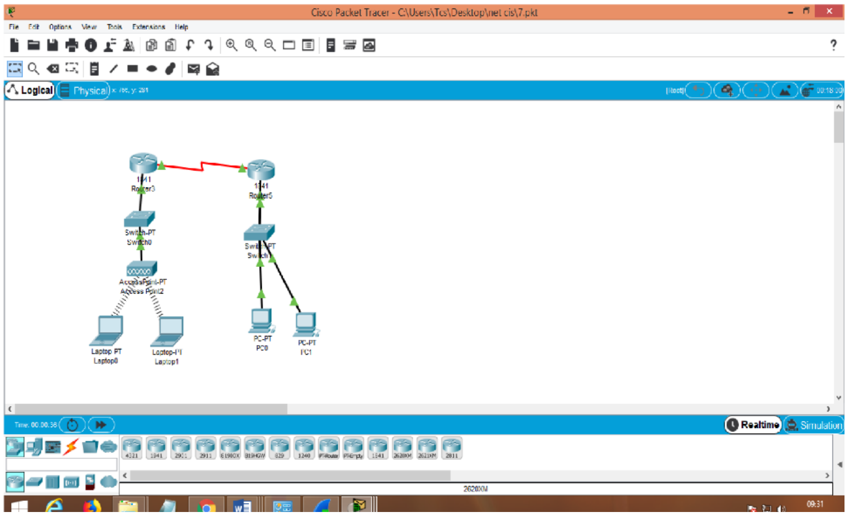
a. Enter Simulation mode.
b. Click 172.16.31.2 and open the Command Prompt.
c. Enter the arp -d command to clear the ARP table.
d. Enter the command ping 172.16.31.3.

e. Two PDUs will be generated.
f. The ping command cannot complete the ICMP packet without knowing the MAC address of the destination.
g. So the computer sends an ARP broadcast frame to find the MAC address of the destination.
h. Click Capture/Forward once. The ARP PDU moves Switch1 while the ICMP PDU disappears, waitingfor the ARP reply.
i. Click Capture/Forward to move the PDU to the next device.
j. Click Capture/Forward until the PDU returns to 172.16.31.2.
k. The ICMP packet reappears.
l. Switch back to Realtime and the ping completes.
m. Click 172.16.31.2 and enter the arp –a command. The Mac address of 172.16.31.3 will be added to the ARP table.

Examine a Switch MAC Address Table

a. From 172.16.31.2, enter the ping 172.16.31.4 command.
b. Click 10.10.10.2 and open the Command Prompt.
c. Enter the ping 10.10.10.3 command.
d. Click Switch1and then the CLI tab. Enter the show mac-address-table command.
e. Click Switch0, then the CLI tab. Enter the show mac-address-table command.

OUTPUT:



RESULT:

Thus, EXAMINE THE ARP TABLE has been successfully executed

EX NO: 7 SCAN PORTS USING NMAP TOOL

DATE: 12/10/22

Aim:

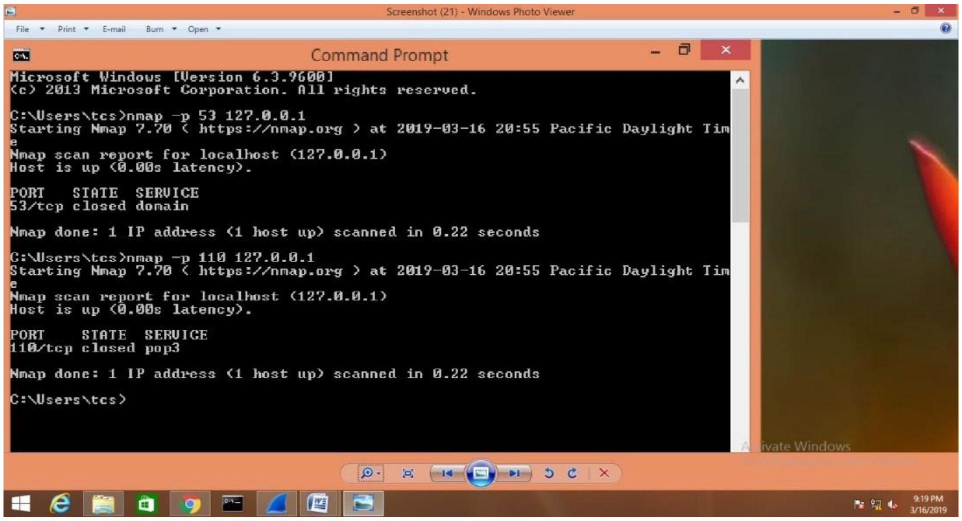
To scan various ports using the NMAP tool.

1) Perform a port scan over DNS and POP3 ports command:

DNS: nmap –p 53 127.0.0.1

POP3: nmap –p 110 127.0.0.1

OUTPUT


Screenshot (21) - Windows Photo Viewer
File Print E-mail Burn Open
Command Prompt

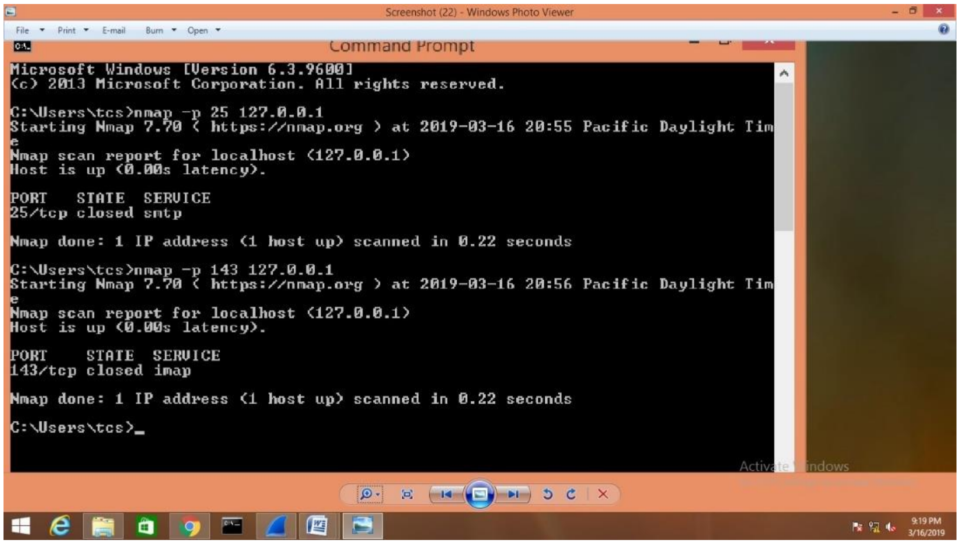
```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\tcs>nmap -p 53 127.0.0.1
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-16 20:55 Pacific Daylight Tim
e
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00s latency).

PORT   STATE  SERVICE
53/tcp closed domain

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds

C:\Users\tcs>nmap -p 110 127.0.0.1
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-16 20:55 Pacific Daylight Tim
e
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00s latency).

PORT    STATE  SERVICE
110/tcp closed pop3

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds

C:\Users\tcs>
```

2.Perform a port scan over SMTP and IMAP ports

Command :

SMTP: nmap –p 25 127.0.0.1
IMAP: : nmap –p 143 127.0.0.1

OUTPUT:


Screenshot (22) - Windows Photo Viewer
File Print E-mail Burn Open
Command Prompt
Activate Windows
9:19 PM
3/16/2019

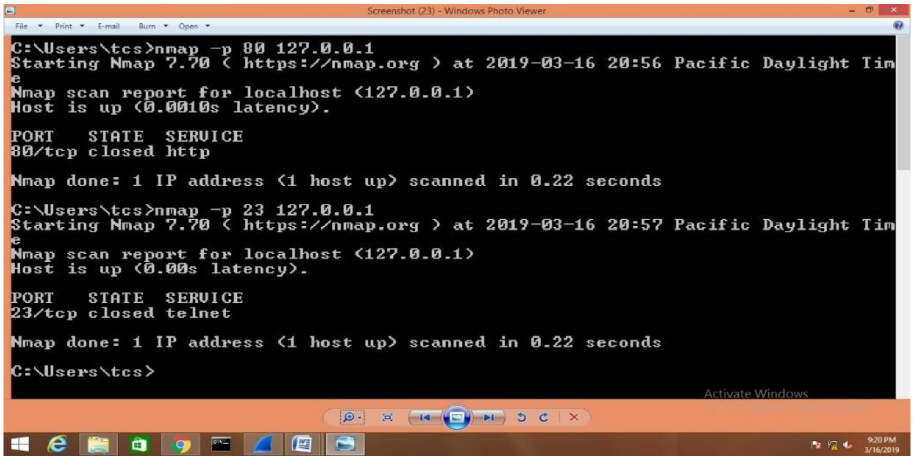
```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\tcs>nmap -p 25 127.0.0.1
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-16 20:55 Pacific Daylight Tim
e
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00s latency).

PORT   STATE  SERVICE
25/tcp closed smtp

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds

C:\Users\tcs>nmap -p 143 127.0.0.1
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-16 20:56 Pacific Daylight Tim
e
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00s latency).

PORT    STATE  SERVICE
143/tcp closed imap

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds

C:\Users\tcs>_
```

3.Perform a port scan over HTTP and Telnet port

Command:

HTTP : nmap –p 80 127.0.0.1
Telnet port: : nmap –p 23 127.0.0.1

OUTPUT


Screenshot (23) - Windows Photo Viewer
File Print E-mail Burn Open
Activate Windows
9:20 PM
3/16/2019

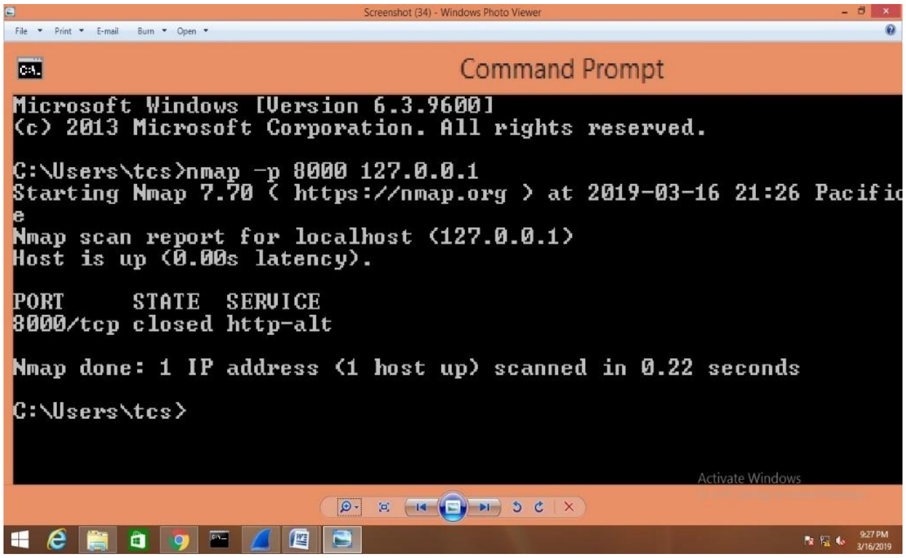
```
C:\Users\tcs>nmap -p 80 127.0.0.1
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-16 20:56 Pacific Daylight Tim
e
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0010s latency).

PORT   STATE  SERVICE
80/tcp closed http

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds

C:\Users\tcs>nmap -p 23 127.0.0.1
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-16 20:57 Pacific Daylight Tim
e
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00s latency).

PORT   STATE  SERVICE
23/tcp closed telnet

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds

C:\Users\tcs>
```

4. Perform a port scan over any network application created by you

Command:

nmap –p 8000 127.0.0.1

OUTPUT


Screenshot (34) - Windows Photo Viewer
File Print E-mail Burn Open
Command Prompt
Activate Windows
9:27 PM
3/16/2019

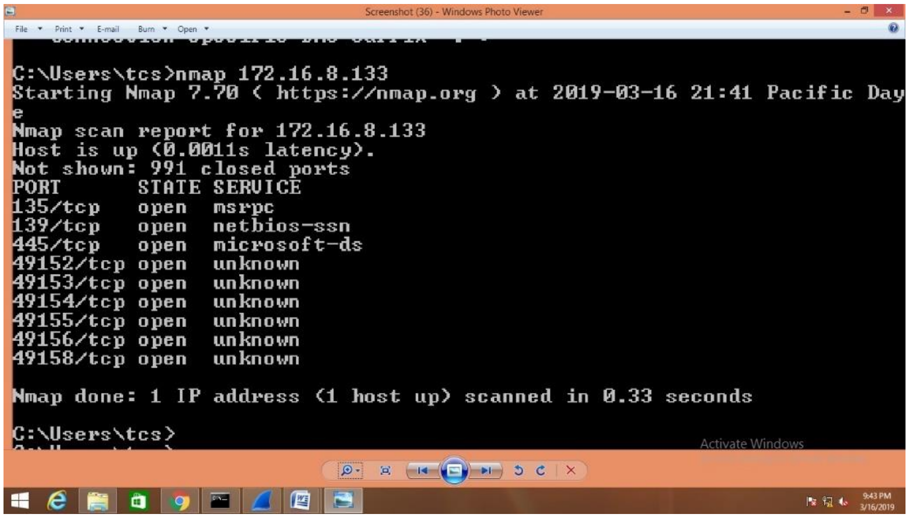
```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\tcs>nmap -p 8000 127.0.0.1
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-16 21:26 Pacific
e
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00s latency).

PORT     STATE  SERVICE
8000/tcp closed http-alt

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds

C:\Users\tcs>
```

5. Perform a scan over the target host 172.16.9.83.

Command:

nmap –p 172.16.8.133

OUTPUT


Screenshot (36) - Windows Photo Viewer
File Print E-mail Burn Open
Activate Windows
9:43 PM
3/16/2019

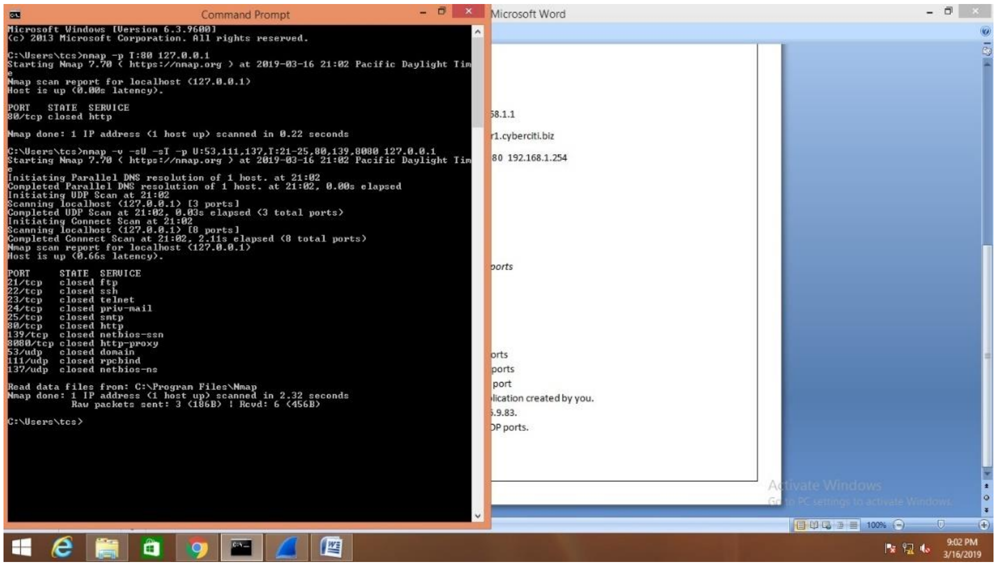
```
C:\Users\tcs>nmap 172.16.8.133
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-16 21:41 Pacific Day
e
Nmap scan report for 172.16.8.133
Host is up (0.0011s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49156/tcp open  unknown
49158/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 0.33 seconds

C:\Users\tcs>
```

6. Perform a port scan over TCP ports and UDP ports.

Command:

nmap –p T:80 127.0.0.1
nmap –p U:53 127.0.0.1

OUTPUT



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\tcs>nmap -p T:80 127.0.0.1
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-16 21:02 Pacific Daylight Tim
e
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00s latency).

PORT   STATE  SERVICE
80/tcp closed http

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds

C:\Users\tcs>nmap -v -sU -sT -p U:53,111,137,T:21-25,80,139,8080 127.0.0.1
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-16 21:02 Pacific Daylight Tim
e
Initiating Parallel DNS resolution of 1 host. at 21:02
Completed Parallel DNS resolution of 1 host. at 21:02, 0.00s elapsed
Initiating UDP Scan at 21:02
Scanning localhost (127.0.0.1) [3 ports]
Completed UDP Scan at 21:02, 0.03s elapsed (3 total ports)
Initiating Connect Scan at 21:02
Scanning localhost (127.0.0.1) [8 ports]
Completed Connect Scan at 21:02, 2.11s elapsed (8 total ports)
Nmap scan report for localhost (127.0.0.1)
Host is up (0.66s latency).

PORT     STATE  SERVICE
21/tcp   closed ftp
22/tcp   closed ssh
23/tcp   closed telnet
24/tcp   closed priv-mail
25/tcp   closed smtp
80/tcp   closed http
139/tcp  closed netbios-ssn
8080/tcp closed http-proxy
53/udp   closed domain
111/udp  closed rpcbind
137/udp  closed netbios-ns

Read data files from: C:\Program Files\Nmap
Nmap done: 1 IP address (1 host up) scanned in 2.32 seconds
           Raw packets sent: 3 (186B) | Rcvd: 6 (456B)

C:\Users\tcs>
```

RESULT:

Thus, SCAN PORTS USING NMAP TOOL has been successfully executed

EX NO: 8 CAPTURING AND ANALYSING PACKETS USING WIRESHARK TOOL

DATE: 21/10/22

Aim

To filter, capture, view, packets in Wireshark Tool.

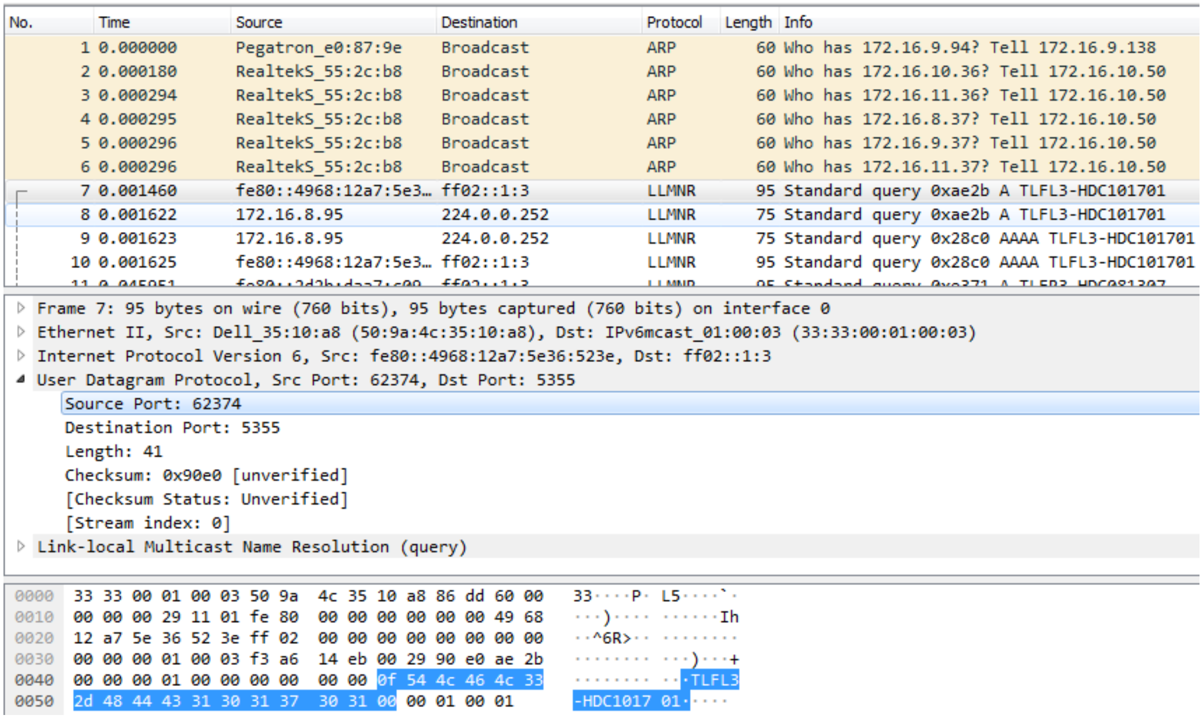
Exercises

1. Capture 100 packets from the Ethernet: IEEE 802.3 LAN Interface and save it.

Procedure

- Select Local Area Connection in Wireshark.
- Go to capture option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Save the packets.

Output



```
No.   Time       Source               Destination   Protocol  Length Info
    1 0.000000   Pegatron_e0:87:9e    Broadcast     ARP       60 Who has 172.16.9.94? Tell 172.16.9.138
    2 0.000180   RealtekS_55:2c:b8    Broadcast     ARP       60 Who has 172.16.10.36? Tell 172.16.10.50
    3 0.000294   RealtekS_55:2c:b8    Broadcast     ARP       60 Who has 172.16.11.36? Tell 172.16.10.50
    4 0.000295   RealtekS_55:2c:b8    Broadcast     ARP       60 Who has 172.16.8.37? Tell 172.16.10.50
    5 0.000296   RealtekS_55:2c:b8    Broadcast     ARP       60 Who has 172.16.9.37? Tell 172.16.10.50
    6 0.000296   RealtekS_55:2c:b8    Broadcast     ARP       60 Who has 172.16.11.37? Tell 172.16.10.50
    7 0.001460   fe80::4968:12a7:5e3… ff02::1:3     LLMNR     95 Standard query 0xae2b A TLFL3-HDC101701
    8 0.001622   172.16.8.95          224.0.0.252   LLMNR     75 Standard query 0xae2b A TLFL3-HDC101701
    9 0.001623   172.16.8.95          224.0.0.252   LLMNR     75 Standard query 0x28c0 AAAA TLFL3-HDC101701
   10 0.001625   fe80::4968:12a7:5e3… ff02::1:3     LLMNR     95 Standard query 0x28c0 AAAA TLFL3-HDC101701
```

```
▷ Frame 7: 95 bytes on wire (760 bits), 95 bytes captured (760 bits) on interface 0
▷ Ethernet II, Src: Dell_35:10:a8 (50:9a:4c:35:10:a8), Dst: IPv6mcast_01:00:03 (33:33:00:01:00:03)
▷ Internet Protocol Version 6, Src: fe80::4968:12a7:5e36:523e, Dst: ff02::1:3
◢ User Datagram Protocol, Src Port: 62374, Dst Port: 5355
    Source Port: 62374
    Destination Port: 5355
    Length: 41
    Checksum: 0x90e0 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
▷ Link-local Multicast Name Resolution (query)
```

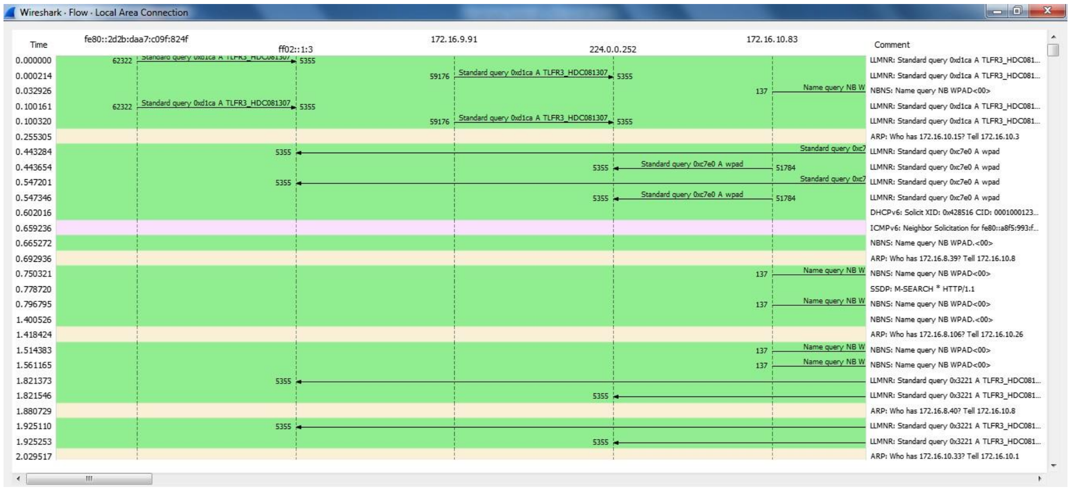
```
0000  33 33 00 01 00 03 50 9a  4c 35 10 a8 86 dd 60 00   33····P· L5····`·
0010  00 00 00 29 11 01 fe 80  00 00 00 00 00 00 49 68   ···)···· ······Ih
0020  12 a7 5e 36 52 3e ff 02  00 00 00 00 00 00 00 00   ··^6R>·· ········
0030  00 00 00 01 00 03 f3 a6  14 eb 00 29 90 e0 ae 2b   ········ ···)···+
0040  00 00 00 01 00 00 00 00  00 00 0f 54 4c 46 4c 33   ········ ··TLFL3
0050  2d 48 44 43 31 30 31 37  30 31 00 00 01 00 01      -HDC1017 01·····
```

2) Create a Filter to display only TCP/UDP packets, inspect the packets and provide the flowgraph.

Procedure

- Select Local Area Connection in Wireshark.
- Go to capture option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search TCP packets in the search bar.
- To see flow graph click Statistics Flow graph.
- Save the packets.

Flow Graph


Wireshark · Flow · Local Area Connection
fe80::2d2b:daa7:c09f:824f
ff02::1:3
172.16.9.91
224.0.0.252
172.16.10.83
62322
Standard query 0xd1ca A TLFR3_HDC081307
5355
59176
137
Name query NB W
Standard query 0xc7
Standard query 0xc7e0 A wpad
51784

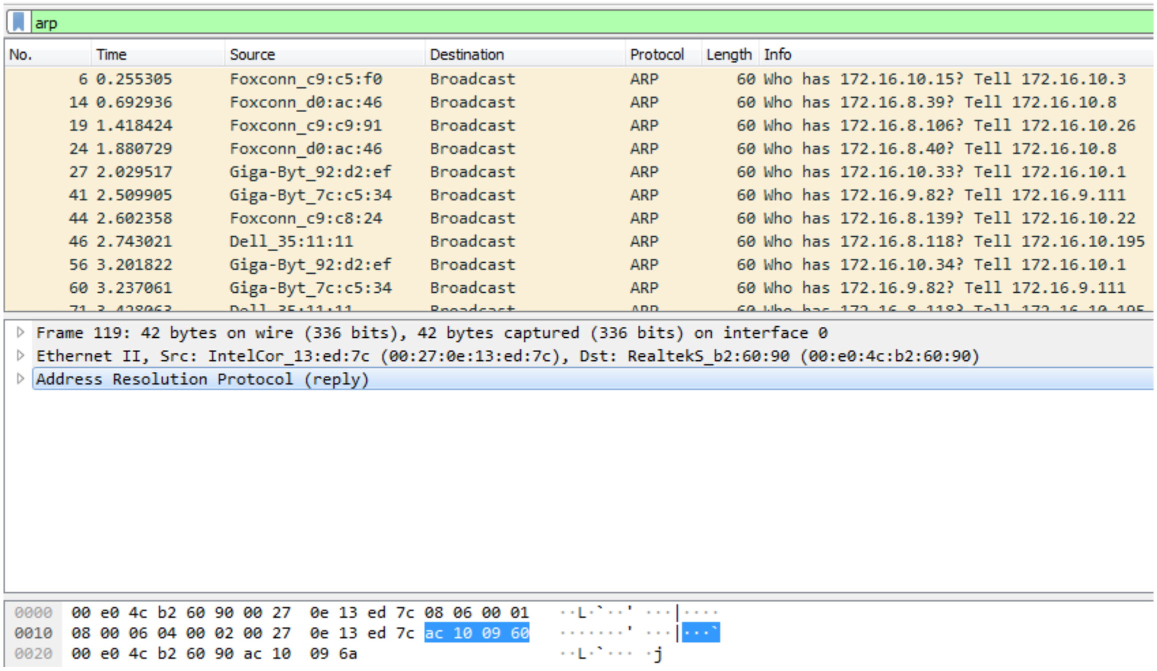
```
Time        Comment
0.000000    LLMNR: Standard query 0xd1ca A TLFR3_HDC081...
0.000214    LLMNR: Standard query 0xd1ca A TLFR3_HDC081...
0.032926    NBNS: Name query NB WPAD<00>
0.100161    LLMNR: Standard query 0xd1ca A TLFR3_HDC081...
0.100320    LLMNR: Standard query 0xd1ca A TLFR3_HDC081...
0.255305    ARP: Who has 172.16.10.15? Tell 172.16.10.3
0.443284    LLMNR: Standard query 0xc7e0 A wpad
0.443654    LLMNR: Standard query 0xc7e0 A wpad
0.547201    LLMNR: Standard query 0xc7e0 A wpad
0.547346    LLMNR: Standard query 0xc7e0 A wpad
0.602016    DHCPv6: Solicit XID: 0x428516 CID: 0001000123...
0.659236    ICMPv6: Neighbor Solicitation for fe80::a8f5:993:f...
0.665272    NBNS: Name query NB WPAD.<00>
0.692936    ARP: Who has 172.16.8.39? Tell 172.16.10.8
0.750321    NBNS: Name query NB WPAD<00>
0.778720    SSDP: M-SEARCH * HTTP/1.1
0.796795    NBNS: Name query NB WPAD<00>
1.400526    NBNS: Name query NB WPAD.<00>
1.418424    ARP: Who has 172.16.8.106? Tell 172.16.10.26
1.514383    NBNS: Name query NB WPAD<00>
1.561165    NBNS: Name query NB WPAD<00>
1.821373    LLMNR: Standard query 0x3221 A TLFR3_HDC081...
1.821546    LLMNR: Standard query 0x3221 A TLFR3_HDC081...
1.880729    ARP: Who has 172.16.8.40? Tell 172.16.10.8
1.925110    LLMNR: Standard query 0x3221 A TLFR3_HDC081...
1.925253    LLMNR: Standard query 0x3221 A TLFR3_HDC081...
2.029517    ARP: Who has 172.16.10.33? Tell 172.16.10.1
```

2.Create a Filter to display only ARP packets and inspect the packets.

Procedure

- Select Local Area Connection in Wireshark.
- Go to capture option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search ARP packets in the search bar.
- Save the packets.

Output



arp

```
No.   Time       Source              Destination  Protocol  Length  Info
    6 0.255305   Foxconn_c9:c5:f0    Broadcast    ARP       60 Who has 172.16.10.15? Tell 172.16.10.3
   14 0.692936   Foxconn_d0:ac:46    Broadcast    ARP       60 Who has 172.16.8.39? Tell 172.16.10.8
   19 1.418424   Foxconn_c9:c9:91    Broadcast    ARP       60 Who has 172.16.8.106? Tell 172.16.10.26
   24 1.880729   Foxconn_d0:ac:46    Broadcast    ARP       60 Who has 172.16.8.40? Tell 172.16.10.8
   27 2.029517   Giga-Byt_92:d2:ef   Broadcast    ARP       60 Who has 172.16.10.33? Tell 172.16.10.1
   41 2.509905   Giga-Byt_7c:c5:34   Broadcast    ARP       60 Who has 172.16.9.82? Tell 172.16.9.111
   44 2.602358   Foxconn_c9:c8:24    Broadcast    ARP       60 Who has 172.16.8.139? Tell 172.16.10.22
   46 2.743021   Dell_35:11:11       Broadcast    ARP       60 Who has 172.16.8.118? Tell 172.16.10.195
   56 3.201822   Giga-Byt_92:d2:ef   Broadcast    ARP       60 Who has 172.16.10.34? Tell 172.16.10.1
   60 3.237061   Giga-Byt_7c:c5:34   Broadcast    ARP       60 Who has 172.16.9.82? Tell 172.16.9.111
```

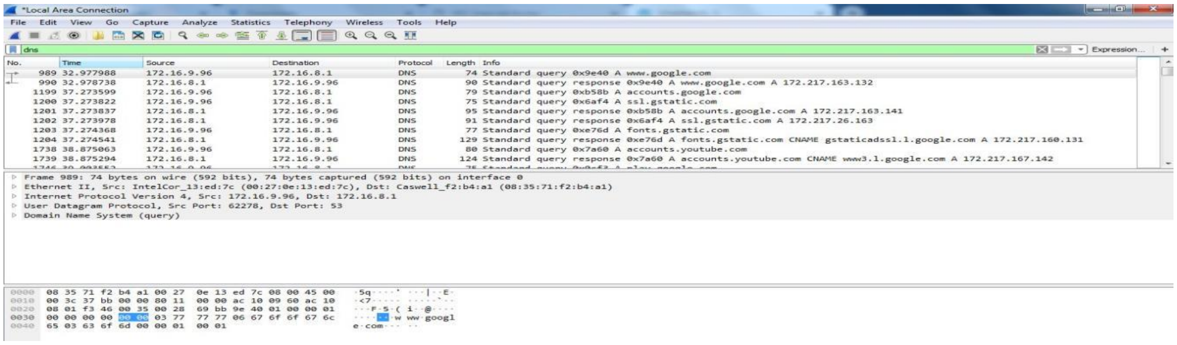
```
▷ Frame 119: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▷ Ethernet II, Src: IntelCor_13:ed:7c (00:27:0e:13:ed:7c), Dst: RealtekS_b2:60:90 (00:e0:4c:b2:60:90)
▷ Address Resolution Protocol (reply)
```

```
0000  00 e0 4c b2 60 90 00 27  0e 13 ed 7c 08 06 00 01   ··L·`··' ···|····
0010  08 00 06 04 00 02 00 27  0e 13 ed 7c ac 10 09 60   ·······' ···|···`
0020  00 e0 4c b2 60 90 ac 10  09 6a                     ··L·`··· ·j
```

3.Create a Filter to display only DNS packets and provide the flow graph.

Procedure

- Select Local Area Connection in Wireshark.
- Go to capture   option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search DNS packets in the search bar.
- To see flow graph click Statistics Flow graph.
- Save the packets.



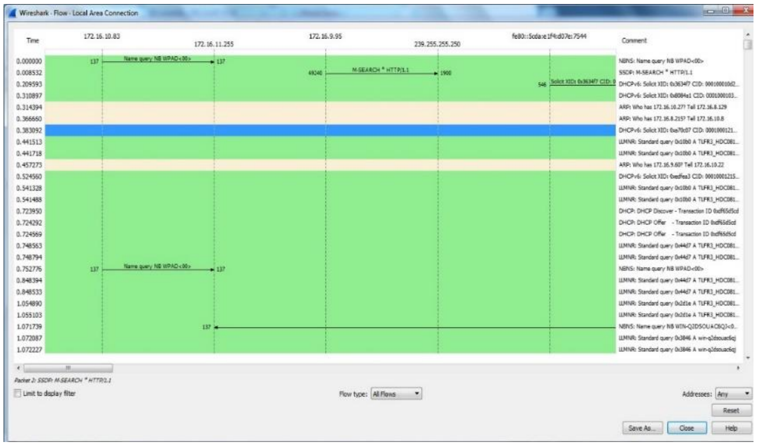
*Local Area Connection
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression…
+

dns

```
No.   Time        Source        Destination   Protocol  Length  Info
  989 32.977988   172.16.9.96   172.16.8.1    DNS        74 Standard query 0x9e40 A www.google.com
  990 32.978738   172.16.8.1    172.16.9.96   DNS        90 Standard query response 0x9e40 A www.google.com A 172.217.163.132
 1199 37.273599   172.16.9.96   172.16.8.1    DNS        79 Standard query 0xb58b A accounts.google.com
 1200 37.273822   172.16.9.96   172.16.8.1    DNS        75 Standard query 0x6af4 A ssl.gstatic.com
 1201 37.273837   172.16.8.1    172.16.9.96   DNS        95 Standard query response 0xb58b A accounts.google.com A 172.217.163.141
 1202 37.273978   172.16.8.1    172.16.9.96   DNS        91 Standard query response 0x6af4 A ssl.gstatic.com A 172.217.26.163
 1203 37.274368   172.16.9.96   172.16.8.1    DNS        77 Standard query 0xe76d A fonts.gstatic.com
 1204 37.274541   172.16.8.1    172.16.9.96   DNS       129 Standard query response 0xe76d A fonts.gstatic.com CNAME gstaticadssl.l.google.com A 172.217.160.131
 1738 38.875063   172.16.9.96   172.16.8.1    DNS        80 Standard query 0x7a60 A accounts.youtube.com
 1739 38.875294   172.16.8.1    172.16.9.96   DNS       124 Standard query response 0x7a60 A accounts.youtube.com CNAME www3.l.google.com A 172.217.167.142
```

```
▷ Frame 989: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▷ Ethernet II, Src: IntelCor_13:ed:7c (00:27:0e:13:ed:7c), Dst: Caswell_f2:b4:a1 (08:35:71:f2:b4:a1)
▷ Internet Protocol Version 4, Src: 172.16.9.96, Dst: 172.16.8.1
▷ User Datagram Protocol, Src Port: 62278, Dst Port: 53
▷ Domain Name System (query)
```

```
0000  08 35 71 f2 b4 a1 00 27  0e 13 ed 7c 08 00 45 00   ·5q····' ···|··E·
0010  00 3c 37 bb 00 00 80 11  00 00 ac 10 09 60 ac 10   ·<7····· ·····`··
0020  08 01 f3 46 00 35 00 28  69 bb 9e 40 01 00 00 01   ···F·5·( i··@····
0030  00 00 00 00 00 00 03 77  77 77 06 67 6f 6f 67 6c   ·······w ww·googl
0040  65 03 63 6f 6d 00 00 01  00 01                     e·com··· ··
```

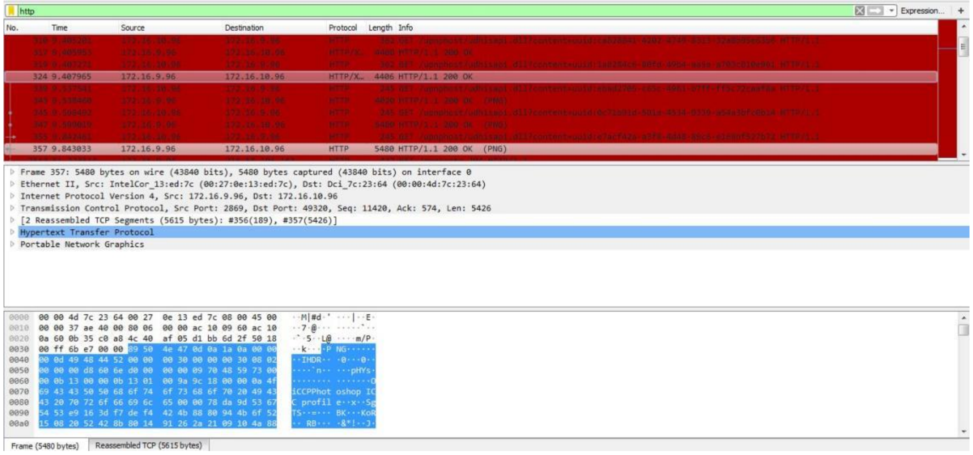


```
0.000000  NBNS: Name query NB WPAD<00>
0.008532  SSDP: M-SEARCH * HTTP/1.1
0.209593  DHCPv6: Solicit XID: 0x3634f7 CID: 000100010d2...
0.310897  DHCPv6: Solicit XID: 0x8084a1 CID: 0001000103...
0.314394  ARP: Who has 172.16.10.27? Tell 172.16.8.129
0.366660  ARP: Who has 172.16.8.215? Tell 172.16.10.8
0.383092  DHCPv6: Solicit XID: 0xa70c07 CID: 0001000121...
0.441513  LLMNR: Standard query 0x10b0 A TLFR3_HDC081...
0.441718  LLMNR: Standard query 0x10b0 A TLFR3_HDC081...
0.457273  ARP: Who has 172.16.9.60? Tell 172.16.10.22
0.524560  DHCPv6: Solicit XID: 0xedfea3 CID: 00010001215...
0.541328  LLMNR: Standard query 0x10b0 A TLFR3_HDC081...
0.541488  LLMNR: Standard query 0x10b0 A TLFR3_HDC081...
0.723950  DHCP: DHCP Discover - Transaction ID 0xdf65d5cd
0.724292  DHCP: DHCP Offer    - Transaction ID 0xdf65d5cd
0.724569  DHCP: DHCP Offer    - Transaction ID 0xdf65d5cd
0.748563  LLMNR: Standard query 0x44d7 A TLFR3_HDC081...
0.748794  LLMNR: Standard query 0x44d7 A TLFR3_HDC081...
0.752776  NBNS: Name query NB WPAD<00>
0.848394  LLMNR: Standard query 0x44d7 A TLFR3_HDC081...
0.848533  LLMNR: Standard query 0x44d7 A TLFR3_HDC081...
1.054890  LLMNR: Standard query 0x2d1e A TLFR3_HDC081...
1.055103  LLMNR: Standard query 0x2d1e A TLFR3_HDC081...
1.071739  NBNS: Name query NB WIN-Q2DSOUAC6QJ<0...
1.072087  LLMNR: Standard query 0x3846 A win-q2dsouac6qj
1.072227  LLMNR: Standard query 0x3846 A win-q2dsouac6qj

Packet 2: SSDP: M-SEARCH * HTTP/1.1
```

4.Create a Filter to display only HTTP packets and inspect the packets

Procedure

- Select Local Area Connection in Wireshark.
- Go to capture option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search HTTP packets in the search bar.
- Save the packets.



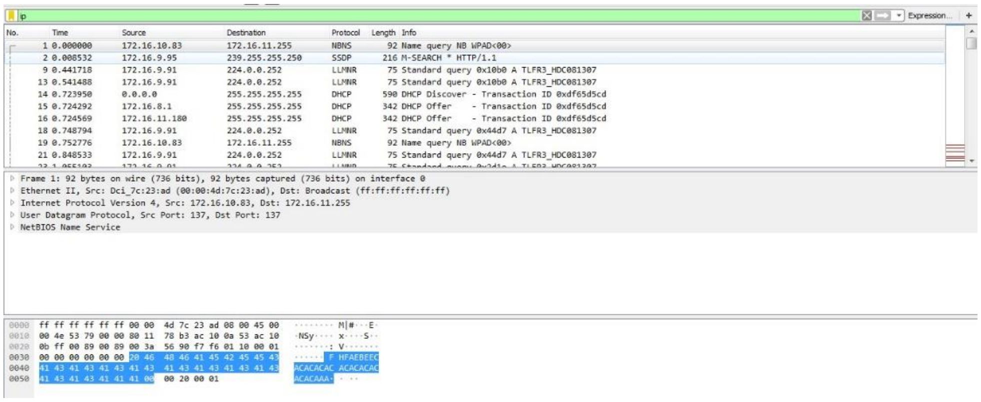
```
http

No.  Time      Source        Destination   Protocol Length Info
 316 9.405201  172.16.10.96  172.16.9.96   HTTP     302 GET /upnphost/udhisapi.dll?content=uuid:ca628841-4202-4749-8313-32a8595e6358 HTTP/1.1
 317 9.405953  172.16.9.96   172.16.10.96  HTTP/X…  4480 HTTP/1.1 200 OK
 319 9.407274  172.16.10.96  172.16.9.96   HTTP     302 GET /upnphost/udhisapi.dll?content=uuid:1a8284c6-00fd-4964-aa5a-a703c010e901 HTTP/1.1
 324 9.407965  172.16.9.96   172.16.10.96  HTTP/X…  4406 HTTP/1.1 200 OK
 339 9.537541  172.16.10.96  172.16.9.96   HTTP     245 GET /upnphost/udhisapi.dll?content=uuid:ebbd2706-c65c-4961-87ff-ff5c72caafaa HTTP/1.1
 343 9.538460  172.16.9.96   172.16.10.96  HTTP     4020 HTTP/1.1 200 OK  (PNG)
 345 9.588492  172.16.10.96  172.16.9.96   HTTP     245 GET /upnphost/udhisapi.dll?content=uuid:0c71a01d-501a-4534-0339-a54a3bfc0b14 HTTP/1.1
 347 9.590019  172.16.9.96   172.16.10.96  HTTP     5480 HTTP/1.1 200 OK  (PNG)
 355 9.842461  172.16.10.96  172.16.9.96   HTTP     245 GET /upnphost/udhisapi.dll?content=uuid:e7acf42a-a3f8-4d48-89c6-e1880f527b72 HTTP/1.1
 357 9.843033  172.16.9.96   172.16.10.96  HTTP     5480 HTTP/1.1 200 OK  (PNG)

▷ Frame 357: 5480 bytes on wire (43840 bits), 5480 bytes captured (43840 bits) on interface 0
▷ Ethernet II, Src: IntelCor_13:ed:7c (00:27:0e:13:ed:7c), Dst: Dci_7c:23:64 (00:00:4d:7c:23:64)
▷ Internet Protocol Version 4, Src: 172.16.9.96, Dst: 172.16.10.96
▷ Transmission Control Protocol, Src Port: 2869, Dst Port: 49320, Seq: 11420, Ack: 574, Len: 5426
▷ [2 Reassembled TCP Segments (5615 bytes): #356(189), #357(5426)]
▷ Hypertext Transfer Protocol
▷ Portable Network Graphics

0000  00 00 4d 7c 23 64 00 27  0e 13 ed 7c 08 00 45 00   ··M|#d·' ···|··E·
0010  00 00 37 ae 40 00 80 06  00 00 ac 10 09 60 ac 10   ··7·@··· ·····`··
0020  0a 60 0b 35 c0 a8 4c 40  af 05 d1 bb 6d 2f 50 18   ·`·5··L@ ····m/P·
0030  00 ff 6b e7 00 00 89 50  4e 47 0d 0a 1a 0a 00 00   ··k····P NG······
0040  00 0d 49 48 44 52 00 00  00 30 00 00 00 30 08 02   ··IHDR·· ·0···0··
0050  00 00 00 d8 60 6e d0 00  00 00 09 70 48 59 73 00   ····`n·· ···pHYs·
0060  00 0b 13 00 00 0b 13 01  00 9a 9c 18 00 00 0a 4f   ········ ·······O
0070  69 43 43 50 50 68 6f 74  6f 73 68 6f 70 20 49 43   iCCPPhot oshop IC
0080  43 20 70 72 6f 66 69 6c  65 00 00 78 da 9d 53 67   C profil e··x··Sg
0090  54 53 e9 16 3d f7 de f4  42 4b 88 80 94 4b 6f 52   TS··=··· BK···KoR
00a0  15 08 20 52 42 8b 80 14  91 26 2a 21 09 10 4a 88   ·· RB··· ·&*!··J·

Frame (5480 bytes)   Reassembled TCP (5615 bytes)
```

5. Create a Filter to display only IP/ICMP packets and inspect the packets.

Procedure

- Select Local Area Connection in Wireshark.
- Go to capture option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search ICMP/IP packets in the search bar.

- Save the packets



```
No.   Time       Source          Destination      Protocol Length Info
    1 0.000000   172.16.10.83    172.16.11.255    NBNS        92 Name query NB WPAD<00>
    2 0.008532   172.16.9.95     239.255.255.250  SSDP       216 M-SEARCH * HTTP/1.1
    9 0.441718   172.16.9.91     224.0.0.252      LLMNR       75 Standard query 0x10b0 A TLFR3_HDC081307
   13 0.541488   172.16.9.91     224.0.0.252      LLMNR       75 Standard query 0x10b0 A TLFR3_HDC081307
   14 0.723950   0.0.0.0         255.255.255.255  DHCP       590 DHCP Discover - Transaction ID 0xdf65d5cd
   15 0.724292   172.16.8.1      255.255.255.255  DHCP       342 DHCP Offer    - Transaction ID 0xdf65d5cd
   16 0.724569   172.16.11.180   255.255.255.255  DHCP       342 DHCP Offer    - Transaction ID 0xdf65d5cd
   18 0.748794   172.16.9.91     224.0.0.252      LLMNR       75 Standard query 0x44d7 A TLFR3_HDC081307
   19 0.752776   172.16.10.83    172.16.11.255    NBNS        92 Name query NB WPAD<00>
   21 0.848533   172.16.9.91     224.0.0.252      LLMNR       75 Standard query 0x44d7 A TLFR3_HDC081307
```

```
▷ Frame 1: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
▷ Ethernet II, Src: Dci_7c:23:ad (00:00:4d:7c:23:ad), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▷ Internet Protocol Version 4, Src: 172.16.10.83, Dst: 172.16.11.255
▷ User Datagram Protocol, Src Port: 137, Dst Port: 137
▷ NetBIOS Name Service
```

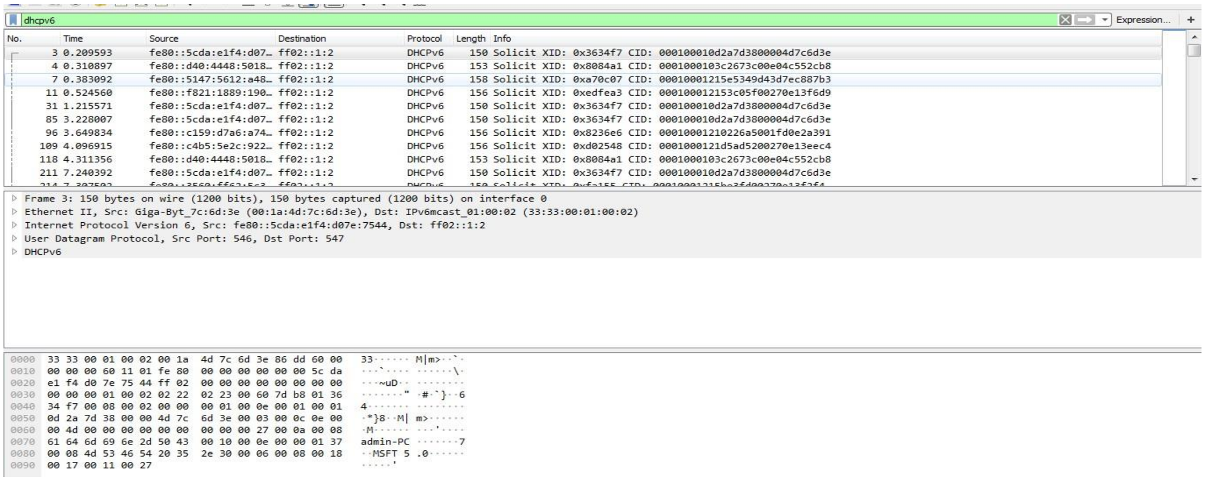
```
0000  ff ff ff ff ff ff 00 00  4d 7c 23 ad 08 00 45 00   ········ M|#···E·
0010  00 4e 53 79 00 00 80 11  78 b3 ac 10 0a 53 ac 10   ·NSy···· x····S··
0020  0b ff 00 89 00 89 00 3a  56 90 f7 f6 01 10 00 01   ·······: V·······
0030  00 00 00 00 00 00 20 46  48 46 41 45 42 45 45 43   ······ F HFAEBEEC
0040  41 43 41 43 41 43 41 43  41 43 41 43 41 43 41 43   ACACACAC ACACACAC
0050  41 43 41 43 41 41 41 00  00 20 00 01               ACACAAA· ····
```

6.Create a Filter to display only DHCP packets and inspect the packets.

Procedure

- Select Local Area Connection in Wireshark.
- Go to capture   option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search DHCP packets in the search bar.
- Save the packets

Output



```
No.   Time       Source                   Destination   Protocol Length Info
    3 0.209593   fe80::5cda:e1f4:d07…     ff02::1:2     DHCPv6      150 Solicit XID: 0x3634f7 CID: 000100010d2a7d3800004d7c6d3e
    4 0.310897   fe80::d40:4448:5018…     ff02::1:2     DHCPv6      153 Solicit XID: 0x8084a1 CID: 0001000103c2673c00e04c552cb8
    7 0.383092   fe80::5147:5612:a48…     ff02::1:2     DHCPv6      158 Solicit XID: 0xa70c07 CID: 00010001215e5349d43d7ec887b3
   11 0.524560   fe80::f821:1889:190…     ff02::1:2     DHCPv6      156 Solicit XID: 0xedfea3 CID: 000100012153c05f00270e13f6d9
   31 1.215571   fe80::5cda:e1f4:d07…     ff02::1:2     DHCPv6      150 Solicit XID: 0x3634f7 CID: 000100010d2a7d3800004d7c6d3e
   85 3.228007   fe80::5cda:e1f4:d07…     ff02::1:2     DHCPv6      150 Solicit XID: 0x3634f7 CID: 000100010d2a7d3800004d7c6d3e
   96 3.649834   fe80::c159:d7a6:a74…     ff02::1:2     DHCPv6      156 Solicit XID: 0x8236e6 CID: 00010001210226a5001fd0e2a391
  109 4.096915   fe80::c4b5:5e2c:922…     ff02::1:2     DHCPv6      156 Solicit XID: 0xd02548 CID: 0001000121d5ad5200270e13eec4
  118 4.311356   fe80::d40:4448:5018…     ff02::1:2     DHCPv6      153 Solicit XID: 0x8084a1 CID: 0001000103c2673c00e04c552cb8
  211 7.240392   fe80::5cda:e1f4:d07…     ff02::1:2     DHCPv6      150 Solicit XID: 0x3634f7 CID: 000100010d2a7d3800004d7c6d3e
```

```
▷ Frame 3: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits) on interface 0
▷ Ethernet II, Src: Giga-Byt_7c:6d:3e (00:1a:4d:7c:6d:3e), Dst: IPv6mcast_01:00:02 (33:33:00:01:00:02)
▷ Internet Protocol Version 6, Src: fe80::5cda:e1f4:d07e:7544, Dst: ff02::1:2
▷ User Datagram Protocol, Src Port: 546, Dst Port: 547
▷ DHCPv6
```

```
0000  33 33 00 01 00 02 00 1a  4d 7c 6d 3e 86 dd 60 00   33······ M|m>··`·
0010  00 00 00 60 11 01 fe 80  00 00 00 00 00 00 5c da   ···`···· ······\·
0020  e1 f4 d0 7e 75 44 ff 02  00 00 00 00 00 00 00 00   ···~uD·· ········
0030  00 00 00 01 00 02 02 22  02 23 00 60 7d b8 01 36   ·······" ·#·`}··6
0040  34 f7 00 08 00 02 00 00  00 01 00 0e 00 01 00 01   4······· ········
0050  0d 2a 7d 38 00 00 4d 7c  6d 3e 00 03 00 0c 0e 00   ·*}8··M| m>······
0060  00 4d 00 00 00 00 00 00  00 00 00 27 00 0a 00 08   ·M······ ···'····
0070  61 64 6d 69 6e 2d 50 43  00 10 00 0e 00 00 01 37   admin-PC ·······7
0080  00 08 4d 53 46 54 20 35  2e 30 00 06 00 08 00 18   ··MSFT 5 .0······
0090  00 17 00 11 00 27                                  ·····'
```

RESULT:

Thus, SCAN PORTS USING NMAP TOOL has been successfully executed

EX NO :9 ANALYZE WEB LOGS USING WEBALIZER

DATE: 26/10/22

Aim

To analyze the different types of web logs using Webalizer tool.

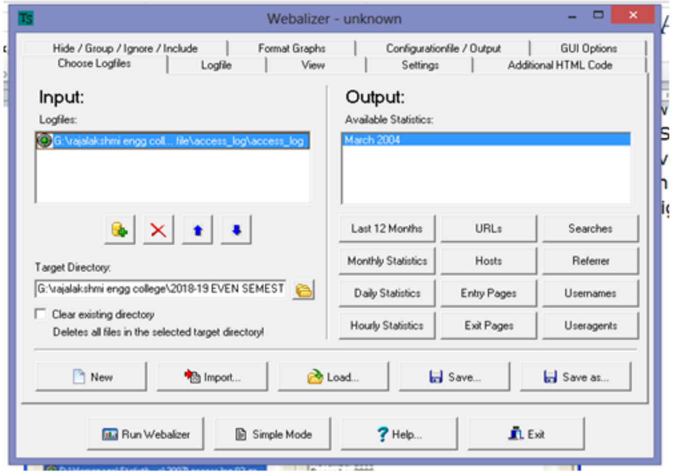
Description

Running steps

Step1: Run webalizer windows version

Step2. Input web log file (download from web)

Step3: Press Run webalizer



Webalizer - unknown

Input:

Logfiles:

G:\rajalakshmi engg coll... file\access_log\access_log

Target Directory:

G:\rajalakshmi engg college\2018-19 EVEN SEMEST

☐ Clear existing directory

Deletes all files in the selected target directory!

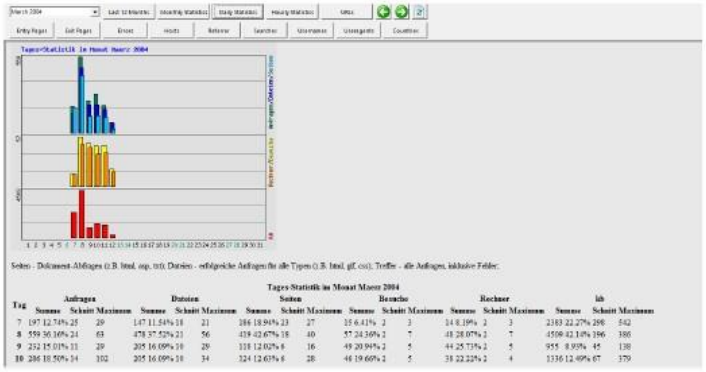
Output:

Available Statistics:

March 2004

Output:

Monthly statistics

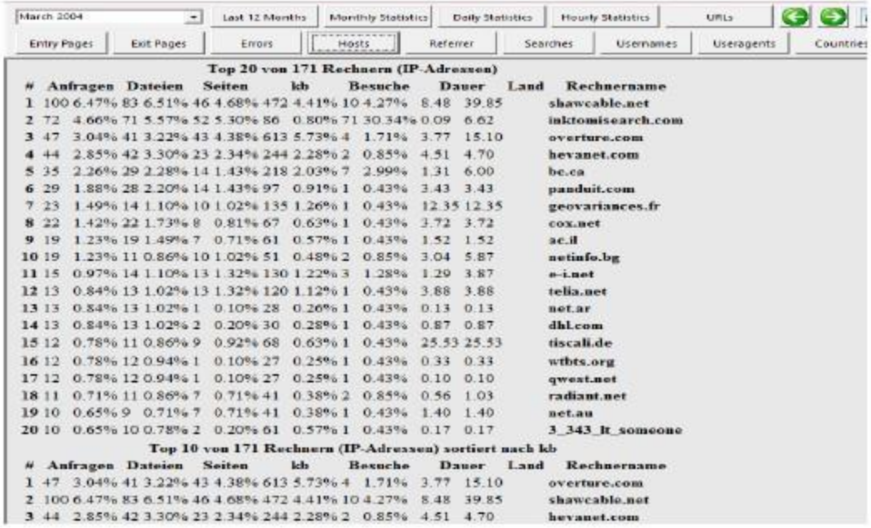


Seiten - Dokument-Abfragen (z.B. html, asp, txt); Dateien - erfolgreiche Anfragen für alle Typen (z.B. html, gif, css); Treffer - alle Anfragen, inklusive Fehler.

Tages-Statistik im Monat Maerz 2004

Tag	Anfragen			Dateien			Seiten			Besuche			Rechner			kB		
	Summe	Schnitt	Maximum	Summe	Schnitt	Maximum	Summe	Schnitt	Maximum	Summe	Schnitt	Maximum	Summe	Schnitt	Maximum	Summe	Schnitt	Maximum
7	197 12.74%	25	29	147 11.54%	18	21	186 18.94%	23	27	15 6.41%	2	3	14 8.19%	2	3	2383 22.27%	298	542
8	559 36.16%	24	63	478 37.52%	21	56	419 42.67%	18	40	57 24.36%	2	7	48 28.07%	2	7	4509 42.14%	396	386
9	232 15.01%	11	29	205 16.09%	10	29	118 12.02%	6	16	49 20.94%	2	5	44 25.73%	2	5	955 8.93%	45	138
10	286 18.50%	14	102	205 16.09%	10	34	124 12.63%	6	28	46 19.66%	2	5	38 22.22%	2	4	1336 12.49%	67	379

Hosts



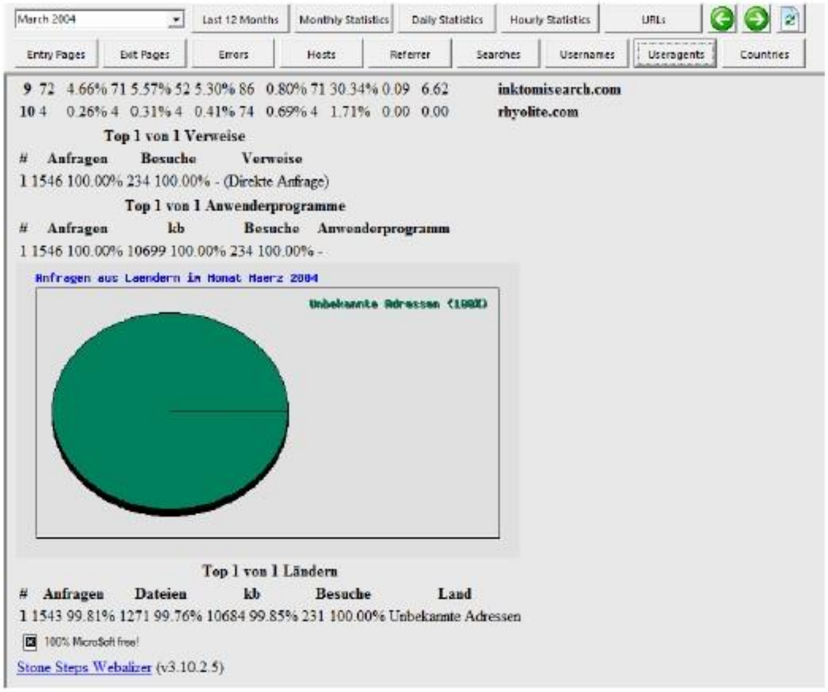
Top 20 von 171 Rechnern (IP-Adressen)

#	Anfragen		Dateien		Seiten		kb		Besuche		Dauer		Land	Rechnername
1	100	6.47%	83	6.51%	46	4.68%	472	4.41%	10	4.27%	8.48	39.85		shawcable.net
2	72	4.66%	71	5.57%	52	5.30%	86	0.80%	71	30.34%	0.09	6.62		inktomisearch.com
3	47	3.04%	41	3.22%	43	4.38%	613	5.73%	4	1.71%	3.77	15.10		overture.com
4	44	2.85%	42	3.30%	23	2.34%	244	2.28%	2	0.85%	4.51	4.70		hevanet.com
5	35	2.26%	29	2.28%	14	1.43%	218	2.03%	7	2.99%	1.31	6.00		bc.ca
6	29	1.88%	28	2.20%	14	1.43%	97	0.91%	1	0.43%	3.43	3.43		panduit.com
7	23	1.49%	14	1.10%	10	1.02%	135	1.26%	1	0.43%	12.35	12.35		geovariances.fr
8	22	1.42%	22	1.73%	8	0.81%	67	0.63%	1	0.43%	3.72	3.72		cox.net
9	19	1.23%	19	1.49%	7	0.71%	61	0.57%	1	0.43%	1.52	1.52		ac.il
10	19	1.23%	11	0.86%	10	1.02%	51	0.48%	2	0.85%	3.04	5.87		netinfo.bg
11	15	0.97%	14	1.10%	13	1.32%	130	1.22%	3	1.28%	1.29	3.87		e-i.net
12	13	0.84%	13	1.02%	13	1.32%	120	1.12%	1	0.43%	3.88	3.88		telia.net
13	13	0.84%	13	1.02%	1	0.10%	28	0.26%	1	0.43%	0.13	0.13		net.ar
14	13	0.84%	13	1.02%	2	0.20%	30	0.28%	1	0.43%	0.87	0.87		dhl.com
15	12	0.78%	11	0.86%	9	0.92%	68	0.63%	1	0.43%	25.53	25.53		tiscali.de
16	12	0.78%	12	0.94%	1	0.10%	27	0.25%	1	0.43%	0.33	0.33		wtbts.org
17	12	0.78%	12	0.94%	1	0.10%	27	0.25%	1	0.43%	0.10	0.10		qwest.net
18	11	0.71%	11	0.86%	7	0.71%	41	0.38%	2	0.85%	0.56	1.03		radiant.net
19	10	0.65%	9	0.71%	7	0.71%	41	0.38%	1	0.43%	1.40	1.40		net.au
20	10	0.65%	10	0.78%	2	0.20%	61	0.57%	1	0.43%	0.17	0.17		3_343_lt_someone

Top 10 von 171 Rechnern (IP-Adressen) sortiert nach kb

#	Anfragen		Dateien		Seiten		kb		Besuche		Dauer		Land	Rechnername
1	47	3.04%	41	3.22%	43	4.38%	613	5.73%	4	1.71%	3.77	15.10		overture.com
2	100	6.47%	83	6.51%	46	4.68%	472	4.41%	10	4.27%	8.48	39.85		shawcable.net
3	44	2.85%	42	3.30%	23	2.34%	244	2.28%	2	0.85%	4.51	4.70		hevanet.com

User-agents



9	72	4.66%	71	5.57%	52	5.30%	86	0.80%	71	30.34%	0.09	6.62	inktomisearch.com
10	4	0.26%	4	0.31%	4	0.41%	74	0.69%	4	1.71%	0.00	0.00	rhyolite.com

Top 1 von 1 Verweise

#	Anfragen		Besuche		Verweise
1	1546	100.00%	234	100.00%	- (Direkte Anfrage)

Top 1 von 1 Anwenderprogramme

#	Anfragen		kb		Besuche		Anwenderprogramm
1	1546	100.00%	10699	100.00%	234	100.00%	-

Top 1 von 1 Ländern

#	Anfragen		Dateien		kb		Besuche		Land
1	1543	99.81%	1271	99.76%	10684	99.85%	231	100.00%	Unbekannte Adressen

☒ 100% MicroSoft free!

Stone Steps Webalizer (v3.10.2.5)

RESULT:

Thus, SCAN PORTS USING NMAP TOOL has been successfully executed

Ex No: 10 **STUDY THE DIFFERENT KINDS OF CABLES**

DATE 28/10/22

AIM:

To study the different kinds of cables.

EXERCISE: Select the best answer

1. When selecting a network card, you should consider all of the following except:

a. the speed and type of network to which you are attaching

b. the MAC address of the card

c. the type of cable or wireless connection used

d. the type of expansion slot in which to install the card

2. A network using a__________ topology has a central hub to which all other computers and devices are connected.

3. A ______ is a communications device that directs data to the correct network by determining the most efficient available route from the sending computer to the receiving computer.

4. Which of the following cables can be used to connect a PC and a Hub?

a) Straight cable

b) Cross cable

c) Rollover cable

d) Any of the above

5. Which type of connector is mostly used these days for connectivity with the network card?

a) BNC

b) RJ-45

c) RJ-33

d) RJ-11

6. Which cable uses RJ-45 connectors?

a) Shielded Twisted Pair Cable

b) Unshielded Twisted Pair Cable

c) Thick Coaxial Cable

d) Thin Coaxial Cable

7. How is a unique MAC address assigned to a network card?

a) A unique address is automatically assigned whenever you boot up the computer

b) A unique address is automatically assigned when the card drivers are installed

c) It is built into the card when the card is manufactured

d) Network administrator must assign the address

8. What do you call a fiber optic cable in which the light signals follow multiple paths?

a) Broadband

b) Baseband

c) Multimode

d) Single mode

9. Which of the following is used as a high-speed network backbone media?

a) Fiber optic cable

b) Thin Coaxial cable

c) Unshielded Twisted Pair Cable

d) Thick Coaxial Cable

10. Which cabling media does not suffer from Electro-Magnetic Interference (EMI)?

a) Fiber optic cable

b) Thin Coaxial cable

c) Unshielded Twisted Pair Cable

d) Thick Coaxial Cable

Ex No: 11 SETING UP A LOCAL AREA NETWORK USING A SWITCH

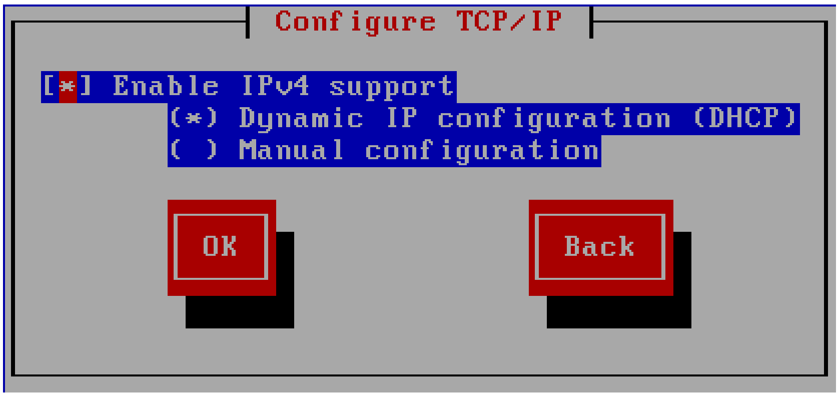
DATE: 2/11/22

AIM:

To study and perform the setting up of a LAN using SWITCH.

Configuring TCP/IP for your LAN(during Installation of OS):

During OS Installation, in the part of network installation, the **Configure TCP/IP** dialog appears.



```
[*] Enable IPv4 support
    (*) Dynamic IP configuration (DHCP)
    ( ) Manual configuration
```

This dialog asks for your IP and other network addresses.

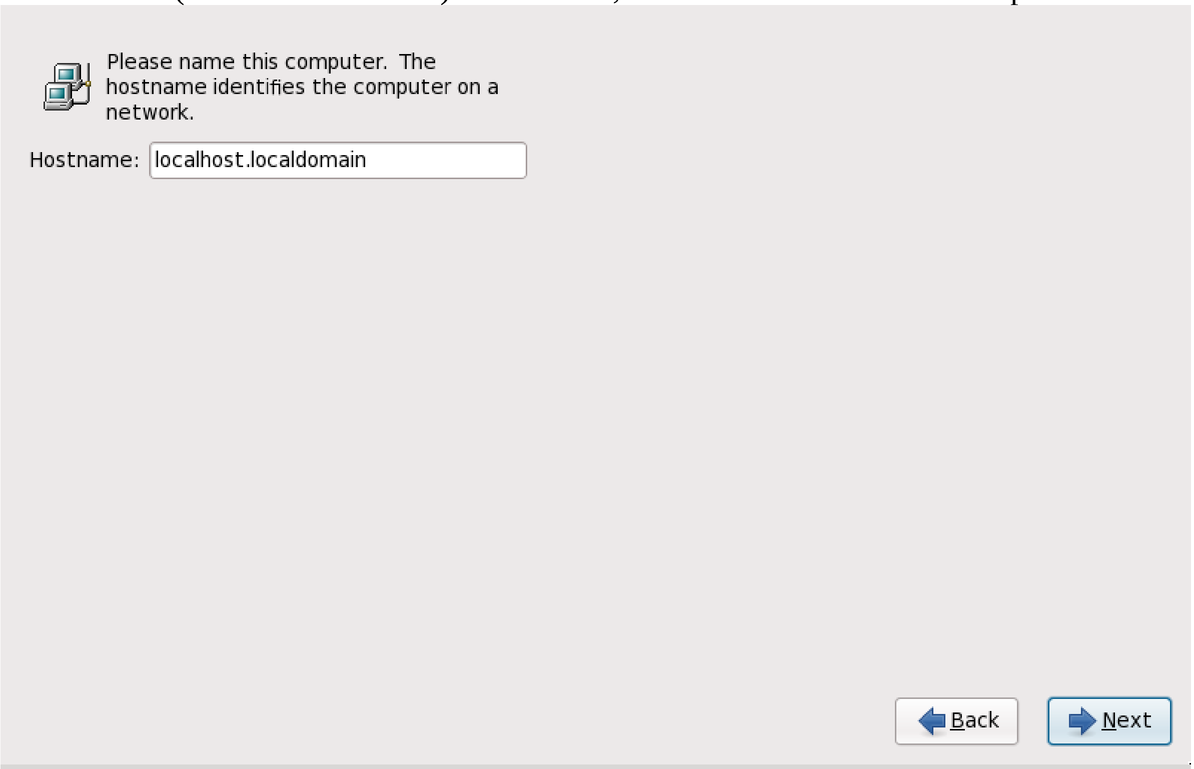
1. You can choose to configure the IP address and Netmask of the device via DHCP or manually.
2. By default, the installation program uses DHCP to automatically provide network settings.
3. If you use a cable or DSL modem, router, firewall, or other network hardware to communicate with the Internet, DHCP is a suitable option.
4. If your network has no DHCP server, clear the check box labelled **Use dynamic IP configuration (DHCP)**. Enter the IP address you are using during installation.

Network Configuration

Fedora, the Linux Operating System, contains support for both IPv4 and IPv6. However, by default, the installation program configures network interfaces on your computer for IPv4.

1. During Network Configuration, a Setup prompts you to supply a host name and domain name for the computer, in the format **hostname. domainname**.
2. Many networks have a DHCP (Dynamic Host Configuration Protocol) service that automatically supplies connected systems with a domain name, leaving the user to enter a hostname.

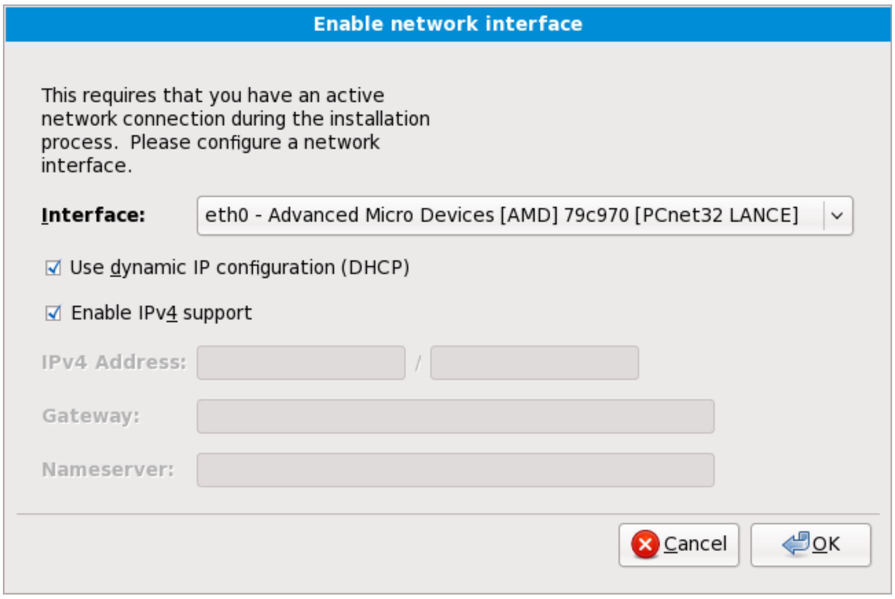
3. Unless you have a specific need to customize the host name and domain name, the default setting **localhost. localdomain** is a good choice for most users.
4. To set up a network that is behind an Internet firewall or router, you may want to use hostname. localdomain for your Fedora system.
5. If you have more than one computer on this network, you should give each one a separate host name in this domain.
6. In some networks, the DHCP provider also provides the name of the computer, or hostname. The complete hostname includes both the name of the machine and the name of the domain of which it is a member, such as machine1.example.com. The machine name (or "short hostname") is machine1, and the domain name is example.com.



Please name this computer. The hostname identifies the computer on a network.

Hostname: localhost.localdomain

Configuring the network interface:



Enable network interface

This requires that you have an active network connection during the installation process. Please configure a network interface.

Interface: eth0 - Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]

☑ Use dynamic IP configuration (DHCP)

☑ Enable IPv4 support

IPv4 Address: /

Gateway:

Nameserver:

If your network does not have DHCP enabled, or if you need to override the DHCP settings, select the network interface that you plan to use from the **Interfaces** menu. Clear the checkbox for **Use dynamic IP configuration (DHCP)**. You can now enter an IPv4 address and netmask for this system in the form ***address*** / ***netmask***, along with the gateway address and nameserver address for your network.

LAN setup:

With an Ethernet NIC, appropriate cables, and a switch, you are ready to set up your wired Ethernet LAN.

Steps for setting up an Ethernet LAN are:

- Power down each computer and physically install the NIC card (following the manufacturer's instructions).
- Using cables appropriate for your NIC cards and switch, connect each NIC to the switch.
- Power up each computer.
- If Linux is not installed yet, install the software and reboot (as instructed).
- If Linux is already installed, when the system comes up, your Ethernet card and interface (eth0) should be ready to use.

RESULT:

Thus, SETING UP A LOCAL AREA NETWORK USING A SWITCH has been successfully executed