

Ex.No:1(a)	BASIC LINUX COMMANDS
Date:	

Aim:

To implement basic Linux Commands.

Commands:

1. pwd: It displays the current working directory.

```
[student@fedora ~]$ pwd
/home/student
```

2. ls: It lists all the files and sub-directories present in the current directory.

```
[exam@fedora ~]$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
[exam@fedora ~]$
```

3. mkdir: It is used to create a new directory.

```
[student@fedora ~]$ mkdir dir
[student@fedora ~]$
```

4. cd: It is used to change the path to the specified directory.

```
[exam@fedora ~]$ mkdir dir1
[exam@fedora ~]$ cd dir1
[exam@fedora dir1]$
```

5. cd ..: It is used to change path to the root directory.

```
[exam@fedora dir1]$ cd ..
[exam@fedora ~]$
```

6. touch: It is used to create an empty file.

```
[student@fedora ~]$ touch file1
[student@fedora ~]$
```

7. echo: It is used to append to text to a file.

```
[student@fedora ~]$ echo "hello world" >>file1
[student@fedora ~]$ cat file1
hello world
```

8. echo: (>>) It appends the given text to the file and overwrites the existing text in the file.

```
[student@fedora ~]$ echo "hello world" >>file1
[student@fedora ~]$ cat file1
hello world
```

9. echo -e(>) It appends the given text to the file and does not overwrite the existing text in the file.

```
[student@fedora ~]$ echo -e "text1\ntext2" >>file1
[student@fedora ~]$ cat file1
commands
hello world
basic commands
computer
networks
echo
text1
text2
[student@fedora ~]$
```

10. head: It is used to view the top lines of a file.

```
[student@fedora ~]$ head -3 file1
commands
hello world
basic commands
[student@fedora ~]$
```

11. tail: It is used to view the bottom lines of the file.

```
[student@fedora ~]$ tail -2 file1
computer
networks
[student@fedora ~]$
```

12. stat: It is used to display the statistics of the file.

```
[student@fedora ~]$ stat file1
  File: file1
  Size: 54          Blocks: 8          IO Block: 4096   regular file
Device: 0,42    Inode: 4517       Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/  student)   Gid: ( 1000/  student)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2022-09-20 14:27:15.820165566 +0530
Modify: 2022-09-20 14:27:03.883113878 +0530
Change: 2022-09-20 14:27:03.883113878 +0530
 Birth: 2022-09-20 14:16:50.680564926 +0530
[student@fedora ~]$
```

13. grep: It is used to search for a pattern in the file.

```
[student@fedora ~]$ grep "hello" file1
hello world
[student@fedora ~]$
```

14. grep-i: It is used to ignore the case and search for a pattern in the file.

```
[student@fedora ~]$ grep -i "HELLO" file2
hello world
[student@fedora ~]$
```

15. cp: It is used to copy the text from a source file to a destination file.

```
[student@fedora ~]$ cp file1 file2
[student@fedora ~]$ cat file2
commands
hello world
basic commands
computer
networks
echo
text1
text2
[student@fedora ~]$
```

16. mv: It is used to move a text from a source file to a destination file.

```
[student@fedora ~]$ mv file1 file3
[student@fedora ~]$ cat file3
commands
hello world
basic commands
computer
networks
echo
text1
text2
```

17. wc: It counts the number of words, characters and lines for a given file.

```
[student@fedora ~]$ wc file2
 8 10 71 file2
[student@fedora ~]$
```

18. wc-l: It displays the number of lines for a given file.

```
[student@fedora ~]$ wc -l file2
8 file2
[student@fedora ~]$
```

19. wc-L: It displays the length of the longest line in a file.

```
[student@fedora ~]$ wc -L file2
14 file2
[student@fedora ~]$
```

Result:

Thus, the Linux Commands were executed successfully.

Ex.No:1(b)	BASIC NETWORKING COMMANDS
Date:	

Aim:

To study basic networking commands and perform the configuration of a network interface in Linux Operating System.

IFCONFIG COMMAND:

ifconfig (interface configurator) command is used to,

1. View IP Address and Hardware / MAC address assigned to interface, gateway, DNS Server

,MTU (Maximum transmission unit) size and many other network configuration parameters.

```
[root@localhost ~]# ifconfig
```

```
eth0 Link encap:Ethernet HWaddr 00:0C:29:28:FD:4C
```

```
inet addr:192.168.50.2 Bcast:192.168.50.255 Mask:255.255.255.0
```

```
inet6 addr: fe80::20c:29ff:fe28:fd4c/64 Scope:Link
```

```
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

```
RX packets:6093 errors:0 dropped:0 overruns:0 frame:0
```

```
TX packets:4824 errors:0 dropped:0 overruns:0 carrier:0
```

```
collisions:0 txqueuelen:1000
```

```
RX bytes:6125302 (5.8 MiB) TX bytes:536966 (524.3 KiB)
```

```
Interrupt:18 Base address:0x2000
```

```
lo Link encap:Local Loopback
```

```
inet addr:127.0.0.1 Mask:255.0.0.0
```

```
inet6 addr: ::1/128 Scope:Host
```

```
UP LOOPBACK RUNNING MTU:16436 Metric:1
```

```
RX packets:8 errors:0 dropped:0 overruns:0 frame:0
```

```
TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
```

```
collisions:0 txqueuelen:0
```

```
RX bytes:480 (480.0 b) TX bytes:480 (480.0 b)
```

2. Assign IP Address to interface

```
[root@localhost ~]#ifconfig eth0 192.168.50.5 netmask 255.255.255.0
```

```
[root@localhost ~]#ifconfig eth0
```

```
eth0 Link encap:Ethernet HWaddr 00:0C:29:28:FD:4C
```

```
inet addr:192.168.50.5 Bcast:192.168.50.255 Mask:255.255.255.0
```

```
inet6 addr: fe80::20c:29ff:fe28:fd4c/64 Scope:Link
```

```
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

```
RX packets:6119 errors:0 dropped:0 overruns:0 frame:0
```

```
TX packets:4841 errors:0 dropped:0 overruns:0 carrier:0
```

```
collisions:0 txqueuelen:1000
```

```
RX bytes:6127464 (5.8 MiB) TX bytes:539648 (527.0 KiB)
```

```
Interrupt:18 Base address:0x2000
```

3. Enable or disable interface on demand.

```
[root@localhost ~]#ifdown eth0 – disables an interface named eth0
```

```
[root@localhost ~]#ifup eth0 – Enables an interface named eth0
```

PING COMMAND:

1. PING (Packet InterNet Groper) command is the best way to test connectivity between two nodes.

2. Both in Local Area Network (LAN) or Wide Area Network (WAN).

3. Ping use ICMP (Internet Control Message Protocol) to communicate to other devices.

4. #ping hostname(ping localhost)

5. #ping ip address (ping 4.2.2.2)

6. #ping fully qualified domain name(ping www.facebook.com)

```
[root@localhost ~]#ping 4.2.2.2
```

```
PING 4.2.2.2 (4.2.2.2) 56(84) bytes of data.
```

```
64 bytes from 4.2.2.2: icmp_seq=1 ttl=44 time=203 ms
```

```
64 bytes from 4.2.2.2: icmp_seq=2 ttl=44 time=201 ms
```

```
64 bytes from 4.2.2.2: icmp_seq=3 ttl=44 time=201 ms
```

HOSTNAME COMMAND:

This command displays the network name of the host machine.

```
[root@localhost network-scripts]# hostname
```

```
localhost.localdomain
```

```
[root@localhost network-scripts]#
```

TRACEROUTE COMMAND:

Traceroute is a command which can show the path a packet takes from the computer to one specified as destination. It will list all the routers it passes through until it reaches its destination, or

fails to and is discarded. In addition to this, it will tell how long each 'hop' from router to router

takes.

```
[root@localhost network-scripts]# traceroute www.google.com
```

```
traceroute to www.google.com (172.217.163.132), 30 hops max, 60 byte packets
```

```
1 gateway (172.16.8.1) 0.165 ms 0.131 ms 0.119 ms
```

```
2 220.225.219.38 (220.225.219.38) 7.131 ms 7.133 ms 7.472 ms
```

```
3 * * *
```

```
4 220.227.42.241 (220.227.42.241) 6.951 ms 7.100 ms 7.323 ms
```

```
5 80.81.65.93 (80.81.65.93) 6.869 ms 6.826 ms 6.962 ms
```

```
6 ae10.0.cjr01.sin001.flagtel.com (62.216.128.233) 39.441 ms 221.191 ms 40.260 ms
```

```
7 80.77.1.254 (80.77.1.254) 40.248 ms 38.916 ms 38.304 ms
```

```
8 108.170.240.164 (108.170.240.164) 39.610 ms 108.170.240.242
```

```
(108.170.240.242) 38.963 ms 108.170.240.172 (108.170.240.172) 38.860 ms
```

```
9 72.14.235.152 (72.14.235.152) 39.912 ms * 72.14.234.96 (72.14.234.96) 38.783 ms
```

```
10 216.239.48.226 (216.239.48.226) 122.220 ms 72.14.233.122
```

```
(72.14.233.122) 121.132 ms 216.239.48.226 (216.239.48.226) 124.187 ms
```

```
(216.239.43.77) 125.901 ms *
```

```
11 maa05s04-in-f4.1e100.net (172.217.163.132) 27.956 ms * *
```

```
[root@localhost network-scripts]#
```

ROUTE COMMAND:

Route command is used to show/manipulate the IP routing table. It is primarily used to setup static

routes to specific host or networks via an interface.

```
[root@localhost network-scripts]# route
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
-------------	---------	---------	-------	--------	-----	-----	-------

default gateway	0.0.0.0	UG	100	0	0		enp3s0
-----------------	---------	----	-----	---	---	--	--------

172.16.8.0	0.0.0.0	255.255.252.0	U	100	0	0	enp3s0
------------	---------	---------------	---	-----	---	---	--------

```
[root@localhost network-scripts]#
```

NETSTAT COMMAND:

Netstat (network statistics) is a command line tool for monitoring network connections both incoming and outgoing as well as viewing routing tables, interface statistics etc.

```
[root@localhost network-scripts]# netstat -r
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
-------------	---------	---------	-------	-----	--------	------	-------

default gateway	0.0.0.0	UG	0	0	0		enp3s0
-----------------	---------	----	---	---	---	--	--------

172.16.8.0	0.0.0.0	255.255.252.0	U	0	0	0	enp3s0
------------	---------	---------------	---	---	---	---	--------

```
[root@localhost network-scripts]#
```

NSLOOKUP COMMAND:

nslookup command. nslookup (name server lookup) is a tool used to perform DNS lookups inLinux.

It is used to display DNS details, such as the IP address of a particular computer, the MX records

for a domain or the NS servers of a domain. nslookup can operate in two modes: interactive and

non-interactive.

```
[root@localhost network-scripts]# nslookup www.google.com
```

Server: 172.16.8.1

Address: 172.16.8.1

Non-authoritative answer:

Name: www.google.com

Address: 172.217.163.132

Name: www.google.com

Address: 2404:6800:4007:80e::2004

[root@localhost network-scripts]#

ARP COMMAND:

Arp command is used to display the arp cache table

[root@localhost network-scripts]# arp

Address HWtype HWaddress Flags Mask Iface

172.16.9.82 ether ec:a8:6b:69:d3:e1 C enp3s0

172.16.10.118 ether 00:11:5b:ff:cc:a5 C enp3s0

gateway ether 08:35:71:f2:b4:a1 C enp3s0

172.16.10.91 ether 00:e0:4c:b2:5b:06 C enp3s0

172.16.10.124 ether 00:0f:ea:93:f4:55 C enp3s0

172.16.8.51 ether 00:1a:4d:a6:98:30 C enp3s0

[root@localhost network-scripts]# ^C

WHOIS COMMAND:

It's queries an official Internet database to determine the current owner of a network domain name

or host name

[root@localhost network-scripts]# whois google.com

[Querying whois.verisign-grs.com]

[Redirected to whois.markmonitor.com]

[Querying whois.markmonitor.com]

[whois.markmonitor.com]

Domain Name: google.com

Registry Domain ID: 2138514_DOMAIN_COM-VRSN

Registrar WHOIS Server: whois.markmonitor.com

Registrar URL: <http://www.markmonitor.com>

Updated Date: 2018-02-21T10:45:07-0800

Creation Date: 1997-09-15T00:00:00-0700

NETWORK CONFIGURATION IN LINUX

1. Open the network interface configuration file of your computer and write the values of the following configuration parameters

Output:

- a) IPADDR : 172.16.8.138
- b) HWADDR : 00:27:0e:13:ea:6a
- c) PREFIX/SUBNET MASK : 23
- d) GATEWAY : 172.16.8.1
- e) DNS : 172.16.8.1
- f) NAME : New 302-3-ethernet connection
- g) TYPE : Ethernet

2. Display the network configuration details of your network interface.

Output:

TYPE = Ethernet

PROXY_METHOD=none

BROWSER_ONLY=no

BOOTPROTO= "static"

IPADDR=172.16.8.13

PREFIX=23

GATEWAY=172.16.8.1

DNS=172.16.8.1

DEFROUTE=yes

IPV4_FAILURE_FATAL=yes

IPV6INIT=yes

IPV6_AUTOCONF=yes

IPV6_DEFROUTE=yes

IPV6_FAILURE_FATAL=no

IPV6_ADDR_GEN_MODE=Stable_privacy

NAME= New 302-3-ethernet connection

UUID=C7628055-dbb6-470f-a68-c71dafdefa

ONBOOT=yes

3. Assign the following IP address to your device 172.16.8.127 and display the change in IP address.

Output:

To change the IP address:

```
[root@localhost network-scripts]#ifconfig eth0 172.16.8.127 netmask 255.255.255.0
```

To verify the Change:

```
[root@localhost network-scripts]# ifconfig
```

```
eth0 Link encap:Ethernet HWaddr 00:0C:29:28:FD:4C
```

```
inet addr:192.168.50.5 Bcast:172.16.8.127 Mask:255.255.255.0
```

```
inet6 addr: fe80::20c:29ff:fe28:fd4c/64 Scope:Link
```

```
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

```
RX packets:6119 errors:0 dropped:0 overruns:0 frame:0
```

```
TX packets:4841 errors:0 dropped:0 overruns:0 carrier:0
```

```
collisions:0 txqueuelen:1000
```

```
RX bytes:6127464 (5.8 MiB) TX bytes:539648 (527.0 KiB)
```

```
Interrupt:18 Base address:0x2000
```

4. Disconnect the network interface and check the dis-connectivity through a command
and display the results. Finally activate the interface.

Output:

To Disconnect the Interface:

```
[root@localhost network-scripts]#ifdown enp3so
```

Device 'enp3so' successfully disconnected.

(Check for a internet connectivity. It will not happen)

To connect to the Interface:

```
[root@localhost network-scripts]#ifup enp3so
```

Connection successfully activated.

5. Check whether the following hosts are reachable from your device and explain the response.

Output:

a) 172.16.8.2

```
[root@localhost network-scripts]#ping 172.16.8.2
```

64 bytes from 172.16.8.2:icmp_seq=1 ttl=64 time=3.14ms

64 bytes from 172.16.8.2:icmp_seq=1 ttl=64 time=2.12ms

64 bytes from 172.16.8.2:icmp_seq=1 ttl=64 time=3.44ms

64 bytes from 172.16.8.2:icmp_seq=1 ttl=64 time=1.14ms

172.16.8.2 is reachable.

b)DNS Server

```
[root@localhost network-scripts]#ping 172.16.8.1
```

64 bytes from 172.16.8.2:icmp_seq=1 ttl=64 time=3.14ms

64 bytes from 172.16.8.2:icmp_seq=1 ttl=64 time=2.12ms

64 bytes from 172.16.8.2:icmp_seq=1 ttl=64 time=3.44ms

64 bytes from 172.16.8.2:icmp_seq=1 ttl=64 time=1.14ms

DNS is reachable.

c) GATEWAY

```
[root@localhost network-scripts]#ping 172.16.8.1
```

64 bytes from 172.16.8.2:icmp_seq=1 ttl=64 time=3.14ms

64 bytes from 172.16.8.2:icmp_seq=1 ttl=64 time=2.12ms

64 bytes from 172.16.8.2:icmp_seq=1 ttl=64 time=3.44ms

64 bytes from 172.16.8.2:icmp_seq=1 ttl=64 time=1.14ms

GATEWAY is reachable.

6. Disconnect the network interface and Check whether the following hosts are reachable from your device and explain the response.

Output:

```
[root@localhost network-scripts]#ifdown enp3so
```

Device 'enp3so' successfully disconnected.

a)127.0.0.1

```
[root@localhost network-scripts]#ping 127.0.0.1
```

64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.034ms

64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.021ms

64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.034ms

64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.014ms

127.0.0.1 is reachable because it is loopback address.

b)GATEWAY

```
[root@localhost network-scripts]#ping 172.16.8.1
```

GATEWAY is unreachable.

7. Find the IPv4 and IPv6 addresses of the following URLs.

Output:

a)www.google.com

```
nslookup www.google.com
```

IPV4 172.217.163.68

IPV6 2404:6800:4007:80c::2004

b)www.facebook.com

```
nslookup www.facebook.com
```

IPV4 157.240.24.35

IPV6 2a03:2880:f139:83:face:booc:0:25de

c)www.rajalakshmi.org

```
nslookup www.rajalakshmi.org
```

IPV4 220.227.30.5

8. Display the ARP table of your machine.

Output:

```
[root@localhost network-scripts]#arp
```

Address	HWType	HWaddress	Flags	Mark	Iface
---------	--------	-----------	-------	------	-------

172.16.8.1	ether	00:27:0e:13:ea:6a	c		enp3s0
------------	-------	-------------------	---	--	--------

172.16.8.2	ether	00:27:0e:13:ea:6a	c		enp3s0
------------	-------	-------------------	---	--	--------

9. Display the kernel routing table of your machine.

Output:

```
[root@localhost network-scripts]#route
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	metric	Ref	Iface
-------------	---------	---------	-------	--------	-----	-------

Default gateway	0.0.0.0	ua	100 0	U		enp3s0
-----------------	---------	----	-------	---	--	--------

172.16.0.0	0.0.0.0	255.255.0.0	U	100 0	O	enp3s0
------------	---------	-------------	---	-------	---	--------

10. List out the hops(IP addresses of the intermediate nodes) taken by a packet to reach its destination.

Output:

```
[root@localhost network-scripts]#traceroute www.google.com
```

Traceroute to www.google.com (172.217.163.68) 30 hops.60 byte packets.

Result:

Thus, the basic networking commands were executed on Linux operating system.

Ex.No:2	STUDY OF SOCKET PROGRAMMING IN PYTHON
Date:	

AIM:

To study the concepts of socket programming in python.

INTRODUCTION:

In networks, the services provided to the user follow the traditional client/server model. One computer acts as a server to provide a certain service and another computer represents the client side which makes use of this service. In order to communicate over the network a network socket comes into play, mostly only referred to as a socket.

SOCKET DEFINITION:

A network socket is an endpoint of a two-way communication link between two programs or processes - client and server in our case - which are running on the network. This can be on the same computer as well as on different systems which are connected via the network. Both client/server communicate with each other by writing to or reading from the network socket. The technical equivalent in reality is a telephone communication between two participants. The network socket represents the corresponding number of the telephone line, or a contract in case of cell phones.

SOCKET PROGRAMMING IN PYTHON:

- Python's core networking library is Socket Module.
- Python's socket module has both class-based and instances-based methods.
- Class-based method is an intuitive approach which doesn't need an instance of a socket object.
- For example, in order to print a machine's IP address, you don't need a socket object. Instead, just call the socket's class-based methods.
- In instance-based method, if some data needed to be sent to a server application, it is more intuitive to create a socket object to perform that explicit operation.
- This module has everything you need to build socket servers and clients.

SAMPLE CLASS METHODS:

1. gethostname method: Used to get the name of the machine

```
>>> import socket

>>> socket.gethostname()

'DESKTOP-K146K1I'
```

2. gethostbyname method: used to get the IP address of the machine

```
>>> host=socket.gethostname()

>>> socket.gethostbyname(host)

'172.16.11.202'

remote='www.gmail.com' (remote host IP address can also be got)

>>> socket.gethostbyname(remote)

'172.217.163.37'
```

3. inet_aton() and inet_ntoa() methods: Converting IP address in decimal notation to

binary format

```
import socket
```

```
for ip_addr in ['127.0.0.1', '192.168.0.1']:
```

```
    packed_ip_addr = socket.inet_aton(ip_addr) [decimal notation to binary
    format]
```

```
    unpacked_ip_addr = socket.inet_ntoa(packed_ip_addr) [binary format to
    decimal notation]
```

```
    print(packed_ip_addr)
```

```
    print(unpacked_ip_addr)
```

OUTPUT:

```
b'\x7f\x00\x00\x01'
```

```
127.0.0.1
```

```
b'\xc0\xa8\x00\x01'
```

```
192.168.0.1
```

4. getserverport() method: used to get the service(protocol) name by giving its port number

and transport layer protocol name.

```
import socket

protocolname = 'tcp'

for port in [80, 25]:

    serp=socket.getservbyport(port,protocolname)

    print(serp)
```

OUTPUT:

http

smtp

5. htonl() and ntohl() methods: used to convert data from host to network byte order and

vice versa

```
import socket

data=1234

data1=socket.htonl(data)

print(data1)

print(socket.ntohl(data1))
```

OUTPUT:

3523477504

1234

SAMPLE INSTANCE METHODS:

Instance method Description

sock.bind((adrs, port)) Bind the socket to the address and port

sock.accept() Return a client socket (with peer address information)

sock.listen(backlog) Place the socket into the listening state, able to pend backlog outstanding connection requests

`sock.connect((adrs, port))` Connect the socket to the defined host and port

`sock.recv(buflen[, flags])` Receive data from the socket, up to buflen bytes

`sock.recvfrom(buflen[, flags])`

Receive data from the socket, up to buflen bytes,

returning also the remote host and port from which

the data came

`sock.send(data[, flags])` Send the data through the socket

`sock.sendto(data[, flags], addr)` Send the data through the socket

`sock.close()` Close the socket

`sock.getsockopt(lvl, optname)` Get the value for the specified socket option

`sock.setsockopt(lvl, optname, val)` Set the value for the specified socket option

HOW TO WORK WITH TCP SOCKETS IN PYTHON:

Socket programming with TCP

Client must contact server:

- ❖ Server must be first running
- ❖ Server must have created socket that welcomes client's contact

client connects to server by:

- ❖ creating TCP socket, specifying IP address, port number of server process
- ❖ client socket is now bound to that specific server

server accepts connect by:

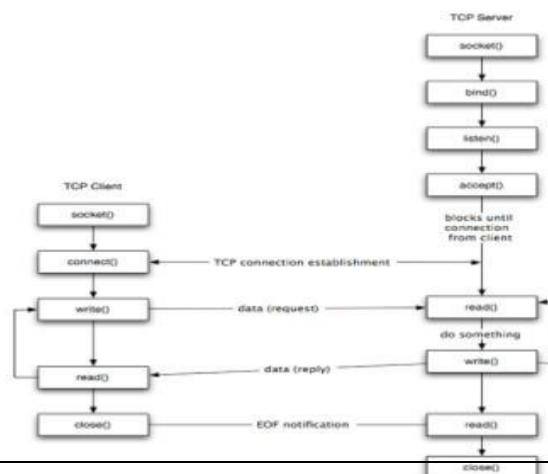
- ❖ creating new connection-specific socket
- ❖ allows server to talk with multiple clients

application

TCP provides
stream transfer
server.

viewpoint:

reliable, in-order byte-
("pipe") between client and



HOW TO WORK WITH UDP SOCKETS IN PYTHON:

UDP: no “connection” between client & server

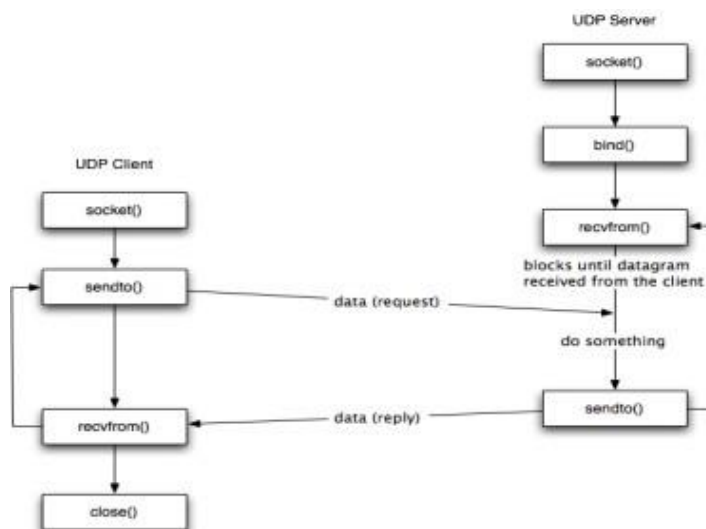
- ❖ no handshaking before sending data
- ❖ sender explicitly attaches IP destination address and port # to each

packet ❖ rcvr extracts sender IP address and port# from received

packet UDP: transmitted data may be lost or received out-of-order

Application viewpoint:

- ❖ UDP provides *unreliable* transfer of groups of bytes (“datagrams”) between client and server



Result:

Thus, the Socket programming is studied.

Ex.No:3	ECHO SERVER AND CLIENT USING TCP SOCKETS
Date:	

Aim:

To develop Echo Server and Client program using TCP socket.

Algorithm:**TCP_Server.py**

1. Import python socket package
2. Obtain host address by using gethost_name
3. Use port number 5000 to make socket function call.
4. Bind the newly created socket connection
5. Send the data to the client by encoding it.
6. Accept client request
7. Close the connection after sending data.

TCP_Client.py

1. Import python socket package
2. Initialize port as 5000
3. Make socket function call to the server IP and port as 5000
4. After connection established, send encoded data to the server
5. Decode the received data from the server.

Code:

```
//tcpServer.py
```

```

import socket

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

print('Socket Created')

sockfd.bind(('localhost',55555))

sockfd.listen(3)

print('Waiting for connections')

while True:

    clientfd,addr=sockfd.accept()

    receivedMsg=clientfd.recv(1024).decode()

    print("Connected with ",addr)

    print("Message Received from Client: ",receivedMsg)

    clientfd.send(bytes(receivedMsg,'utf-8'))

    print("Message reply sent to Client!")

    print("Do you want to continue(type y or n):")

    choice=input()

    if choice=='n':

        break

//tcpclient.py

import socket

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

clientfd.connect(('localhost',55555))

name=input("Enter your message:")

clientfd.send(bytes(name,'utf-8'))

print("Message Received from Server: ",clientfd.recv(1024).decode())

```

Output:

```
Command Prompt - python tcpser.py
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\TCS>python
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
^C
C:\Users\TCS>cd Desktop

C:\Users\TCS\Desktop>python tcpser.py
Socket Created
Waiting for connections
Connected with ('127.0.0.1', 43424)
Message Received from Client:  hello
Message reply sent to Client!
Do you want to continue(type y or n):
```

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\TCS>cd Desktop

C:\Users\TCS\Desktop>python tcpcli.py
Enter your message:hello
Message Received from Server:  hello

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

Result:

Thus, the echo server and client using TCP sockets are executed successfully.

Ex.No:4	ECHO SERVER AND CLIENT USING UDP SOCKETS
Date:	

Aim:

To develop Echo server and client program using UDP sockets.

Algorithm:

UDP_Server.py

1. Import the socket package
2. Initialize local IP address and local port.
3. Create a socket using socket() function
4. Bind the IP address and port number.
5. Accept client request for connection.
6. Read the contents sent by client and display it.
7. Close the connection

UDP_Client.py

1. Import the socket package
2. Initialize server IP address and local port.
3. Create a socket using socket() function.
4. Send the encoded client message using sendto() function.
5. Receive server reply message and print.
6. Close the connection.

Code:

udpEchoServer.py

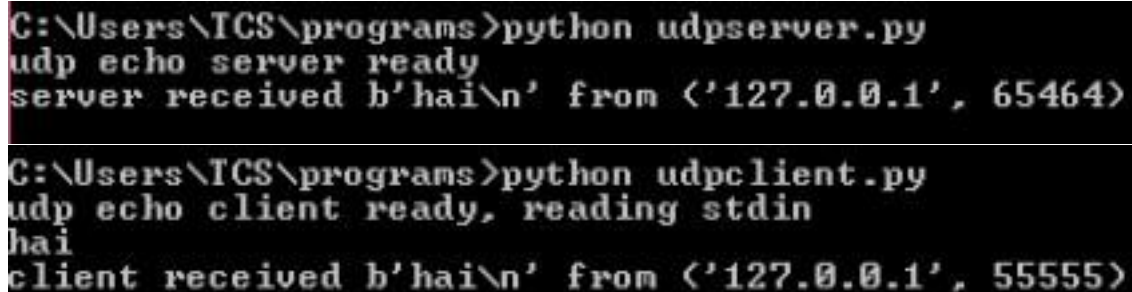
```
import sys
from socket import *
ECHO_PORT = 55555
BUFSIZE = 1024

s = socket(AF_INET, SOCK_DGRAM)
s.bind(('', ECHO_PORT))
print('udp echo server ready')
while 1:
    data, addr = s.recvfrom(BUFSIZE)
    print('server received %r from %r' %
          (data, addr))
    s.sendto(data, addr)
```

udpEchoClient.py

```
import sys
from socket import *
ECHO_PORT = 55555
BUFSIZE = 1024
host = "127.0.0.1"
addr = host, ECHO_PORT
s = socket(AF_INET, SOCK_DGRAM)
s.bind(("", 0))
print ('udp echo client ready,
reading stdin') while 1:
    line = sys.stdin.readline()
    if not line:
        break
    s.sendto(line.encode(), addr)
    data, fromaddr = s.recvfrom(BUFSIZE)
    print ('client received %r from %r' % (data, fromaddr))
```

Output:



```
C:\Users\TCS\programs>python udpserver.py
udp echo server ready
server received b'hai\n' from ('127.0.0.1', 65464)

C:\Users\TCS\programs>python udpclient.py
udp echo client ready, reading stdin
hai
client received b'hai\n' from ('127.0.0.1', 55555)
```

Result:

Thus, the echo server and client using UDP sockets are executed successfully.

Ex.No:5	PING TO TEST SERVER CONNECTIVITY USING SOCKETS
Date:	

Aim:

To develop ping program to test server connectivity using sockets.

Algorithm:

Server.py

1. Import the socket package
2. Initialize local IP address and local port.
3. Create a socket using socket() function
4. Bind the IP address and port number.
5. Accept client request for connection.
6. Print the received connection details
7. Send reply message to the client.
8. Close the connection.

Client.py

1. Import the socket package
2. Initialize server IP address and local port.
3. Create a socket using socket() function.
4. Start the timer.
5. Send message to the server.
6. The reply message of the server is received.
7. The timer is stopped.
8. Print the round trip time statistics.

Code:

//udpserver.py

```
import sys
from socket import *

ECHO_PORT = 55555
BUFSIZE = 1024
s = socket(AF_INET, SOCK_DGRAM)
s.bind(('', ECHO_PORT))
print ('udp echo server ready')
while 1:
    data, addr = s.recvfrom(BUFSIZE)
    print( 'server received %r from %r' %
    (data, addr)) s.sendto(data, addr)
```



```
//rtt.py import sys
from socket import *
import time
start=time.time()
ECHO_PORT = 55555
BUFSIZE = 1024
host = "172.16.8.101"
addr = host, ECHO_PORT
s = socket(AF_INET, SOCK_DGRAM)
s.bind(("", 0))
print ('udp echo client ready,
reading stdin') while 1:
    line = sys.stdin.readline()
    if not line:
        break
    s.sendto(line.encode(), addr)
    data, fromaddr = s.recvfrom(BUFSIZE)
    print ('client received %r from %r' %
    (data, fromaddr)) end=time.time()
    rtt=abs(start-end)
    print(f"RTT={rtt}")
```

Output:



```
Command Prompt - python rtt.py
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\TCS>cd desktop

C:\Users\TCS\Desktop>python rtt.py
udp echo client ready, reading stdin
hello
client received b'hello\n' from ('172.16.8.101', 55555)
RTT=4.841574430465698
```

Result:

Thus, the ping to test server connectivity using sockets is executed successfully.

Ex.No:6	CHAT PROGRAM USING SOCKETS
Date:	

Aim:

To develop Chat server and client program using sockets in python.

Algorithm:

Server.py

1. Import the socket package
2. Initialize local IP address and local port.
3. Initialize flag to true.
4. Create a socket using socket() function
5. Bind the IP address and port number.
6. Loop till the flag is true
 - a. Accept client request for connection.
 - b. Print the received message.
 - c. Send the message to the client.
 - d. Stop the loop by setting flag to false
7. Close the connection.

Client.py

1. Import the socket package
2. Initialize server IP address and local port.
3. Initialize flag to true.
4. Create a socket using socket() function.
5. Loop till the flag is true.
 - a. Client sends the message to server
 - b. Client receives the reply message from server
 - c. Stop the loop by setting flag to false
7. Close the connection.

Code:

Server Side

```

from socket import*
s=socket(AF_INET,SOCK_STREAM)
s.bind(("",8000))
s.listen(5)
print("=====ChatApp=====")
c,a=s.accept()
while True:
    data=c.recv(100).decode()
    print("<--",data)
    if (data=="bye" or ""):
        p="bye"
        c.send(p.encode('utf-8'))
        print("Chat End")
        break
    msg=input("-->"),
    c.send(msg.encode('utf-8'))

```

Client Side

```

from socket import*
s=socket(AF_INET,SOCK_STREAM)
s.connect(("127.0.0.1",8000))
print("=====ChatApp=====")
while True:
    msg=input("-->"),
    s.send(msg.encode('utf-8'))
    data=s.recv(100).decode()
    print("<--",data)

```

Output:

Client side

```

=====ChatApp=====
===== -->Hi
<-- Hello
-->How are You?
<-- Fine
-->bye
<-- bye
-->

```

Server Side:

```

<-- Hi
-->Hello
<-- How are You?
-->Fine
<-- bye
Chat End

```

Result:

Thus, the chat program using sockets is executed successfully.

Ex.No:7	FILE TRANSFER USING SOCKETS
Date:	

Aim:

To develop program to do file transfer using sockets.

Algorithm:**Server.py**

1. Import the socket package
2. Create a socket using socket() function
3. Bind the IP address and port number.
4. Open the target file in write mode
5. Initialize flag to true
6. Loop till flag is true
 - a. Receive the bytes from the client in chunks of 1024 bytes
 - b. Write the data received into the target file
 - c. If no data received then set flag to false
7. Close the target file
8. Send the message to client that transfer is complete
9. Close the connection

Client.py

1. Import the socket package
2. Create a socket using socket() function
3. Bind the IP address and port number.
4. Open the source file to be copied in read mode
5. Set flag to true
6. Loop

- a. Read the file in chunks of 1024 bytes
- b. Send the read data to the server
- c. If no more data to send then set flag to false
7. Send completion message to the server
8. Close the source file
9. Close the connection.

Code:

Server Side:

```
import socket
s = socket.socket()
s.bind(("localhost", 5000))
s.listen(1)
c,a = s.accept()
filetodown =
open("C:/Users/TCS/Desktop/79.doc",
"wb") while True:
print("Receiving. ...")
data = c.recv(1024)
if data == b"DONE":
print("Done Receiving.")
break
filetodown.write(data)
filetodown.close()
c.send("Thank you for connecting.")
c.shutdown(2)
c.close()
s.close()
```

Client Side:

```
import socket
s = socket.socket()
s.connect(("localhost", 5000))
filetosend =
open("C:/Users/TCS/Desktop/79.doc",
"rb") data = filetosend.read(1024)
while data:
print("Sending. . .")
s.send(data)
data = filetosend.read(1024)
filetosend.close()
s.send(b"DONE")
print("Done Sending.")
print(s.recv(1024))
s.shutdown(2)
s.close()
```

Output:

```
Python 2.7.10 Shell
File Edit Shell Debug Options Window Help
Python 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Receiving...
Done Receiving.
>>> |

Python 2.7.10 Shell
File Edit Shell Debug Options Window Help
Python 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Done Sending.
Thank you for connecting.
>>> |
```

Result:

Thus, the File Transfer using sockets is executed successfully.

Ex.No:8	PACKET SNIFFING USING RAW SOCKETS
Date:	

Aim:

To develop program to do packet sniffing using raw sockets.

Algorithm;

1. Import socket package
2. Create socket connection
3. Bind the IP address
4. Set socket options with setsockopt()
5. Set the flag to true
6. Loop till flag is true
 - a. Receive destination MAC and source MAC address
 - b. Print protocol details
 - c. Set flag to false
7. Close the connection

Coding:

```
import socket

import struct

import binascii

import textwrap

def main():

    host = socket.gethostbyname(socket.gethostname())

    print('IP: {}'.format(host))

    conn = socket.socket(socket.AF_INET, socket.SOCK_RAW,
socket.IPPROTO_IP) conn.bind((host, 0))

    conn.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1) # Enable promiscuous
mode

    conn.ioctl(socket.SIO_RCVALL, socket.RCVALL_ON)

    while True:

        raw_data, addr = conn.recvfrom(65536)

        dest_mac, src_mac, eth_proto, data = ethernet_frame(raw_data)

        print("\nEthernet Frame:")

        print("Destination MAC: {}".format(dest_mac))

        print("Source MAC: {}".format(src_mac))

        print("Protocol: {}".format(eth_proto))

    def ethernet_frame(data):

        dest_mac, src_mac, proto = struct.unpack('!6s6s2s', data[:14])

        return get_mac_addr(dest_mac), get_mac_addr(src_mac), get_protocol(proto), data[14:]

    def get_mac_addr(bytes_addr):

        bytes_str = map('{:02x}'.format, bytes_addr)

        mac_address = ':'.join(bytes_str).upper()

        return mac_address

    def get_protocol(bytes_proto):

        bytes_str = map('{:02x}'.format, bytes_proto)
```

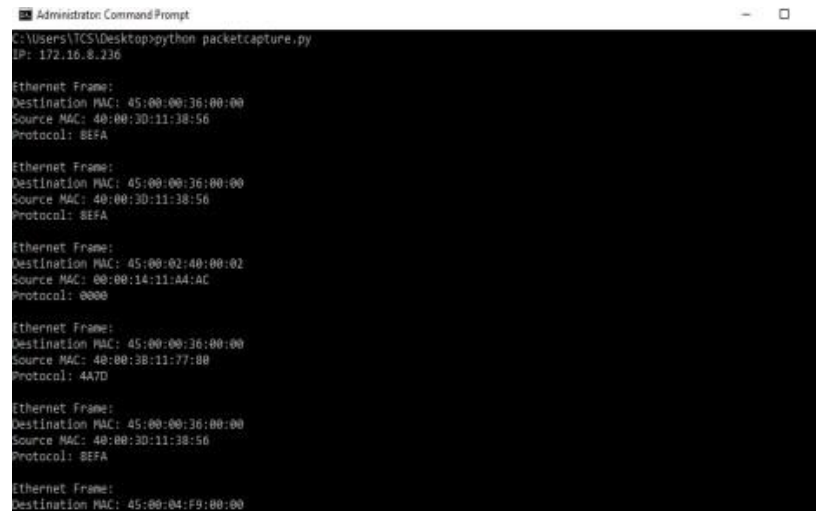


```
protocol = ".join(bytes_str).upper()
```

```
return protocol
```

```
main()
```

Output:



```
Administrator: Command Prompt
C:\Users\TCS\Desktop>python packetcapture.py
IP: 172.16.8.236

Ethernet Frame:
Destination MAC: 45:00:00:36:00:00
Source MAC: 48:00:30:11:38:56
Protocol: 8EFA

Ethernet Frame:
Destination MAC: 45:00:00:36:00:00
Source MAC: 48:00:30:11:38:56
Protocol: 8EFA

Ethernet Frame:
Destination MAC: 45:00:02:40:00:02
Source MAC: 00:00:14:11:A4:AC
Protocol: 0000

Ethernet Frame:
Destination MAC: 45:00:00:36:00:00
Source MAC: 48:00:38:11:77:00
Protocol: 4A7D

Ethernet Frame:
Destination MAC: 45:00:00:36:00:00
Source MAC: 48:00:30:11:38:56
Protocol: 8EFA

Ethernet Frame:
Destination MAC: 45:00:04:F9:00:00
```

Result:

Thus, the Packet Sniffing using raw sockets is executed successfully.

Ex.No:9	ARITHMETIC OPERATIONS USING RPC
Date:	

Aim:

To perform arithmetic operations using XML RPC.

Algorithm:

Server.py

1. Import XMLRPCServer package
2. Define functions for addition, subtraction, multiplication, division and modulus
3. Initialize simple XMLRPCServer with IP address (or localhost) and port number
4. Register the functions add, sub, mul, div and mod with the server
5. Handle the request
6. Close the connection

Client.py

1. Import XMLRPC Client package
2. Define functions for addition, subtraction, multiplication, division and modulus
3. Initialize simple XMLRPC Client with Server IP address (or localhost) and port number
4. Get two numbers a and b for arithmetic operations
5. Call add() function and print the result
6. Call sub() function and print the result
7. Call mul() function and print the result
8. Call div() function and print the result
9. Call mod() function and print the result
10. Close the connection

Coding:

Server.py

```

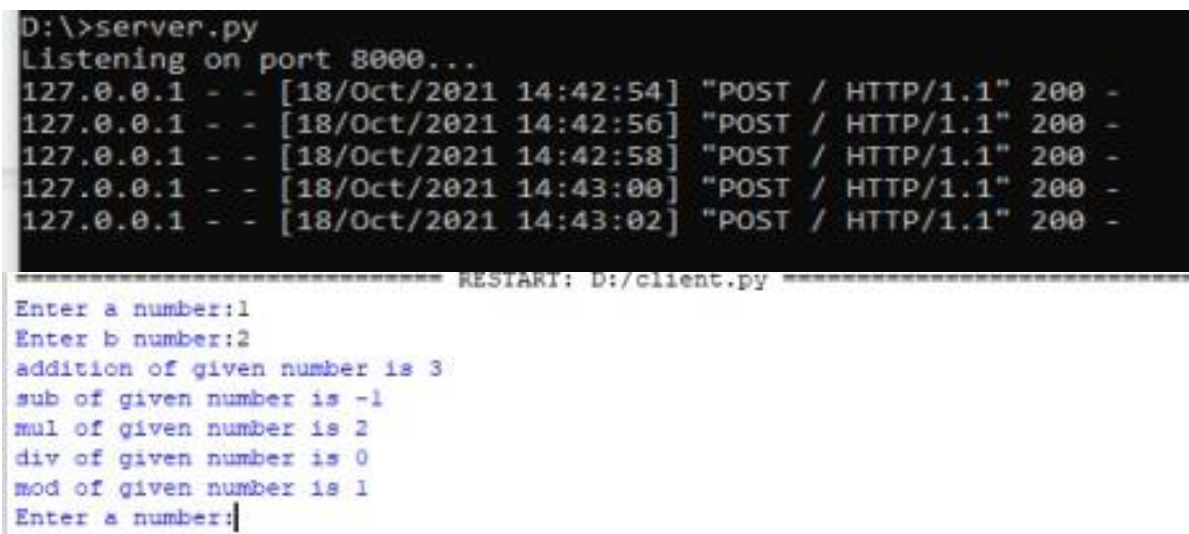
from xmlrpc.server import SimpleXMLRPCServer
def add(a,b):
    return a+b
def sub(a,b):
    return a-b
def mul(a,b):
    return a*b
def div(a,b):
    return a/b
def mod(a,b):
    return a%b
server = SimpleXMLRPCServer(("localhost", 8000))
print("Listening on port 8000...")
server.register_function(add, "add")
server.register_function(sub, "sub")
server.register_function(mul, "mul")
server.register_function(div, "div")
server.register_function(mod, "mod")
server.serve_forever()

```

Client.py

```
import xmlrpc.client
proxy= xmlrpc.client.ServerProxy('http://localhost:8000/')
for i in range(5):
    a=int(input("Enter a number:"))
    b=int(input("Enter b number:"))
    print("addition of given number is %d "%((proxy.add(a,b))))
    print("sub of given number is %d "%((proxy.sub(a,b))))
    print("mul of given number is %d "%((proxy.mul(a,b))))
    print("div of given number is %d "%((proxy.div(a,b))))
    print("mod of given number is %d "%((proxy.mod(a,b))))
```

Output:



The screenshot shows a terminal window with two parts. The top part shows the server's log output for 'server.py', indicating it is listening on port 8000 and receiving five POST requests from 127.0.0.1. The bottom part, separated by a dashed line, shows the client's execution of 'client.py'. It prompts for two numbers, 1 and 2, and then displays the results of five arithmetic operations: addition (3), subtraction (-1), multiplication (2), division (0), and modulo (1). The prompt 'Enter a number:' is shown at the end.

```
D:\>server.py
Listening on port 8000...
127.0.0.1 - - [18/Oct/2021 14:42:54] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [18/Oct/2021 14:42:56] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [18/Oct/2021 14:42:58] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [18/Oct/2021 14:43:00] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [18/Oct/2021 14:43:02] "POST / HTTP/1.1" 200 -
===== RESTART: D:/client.py =====
Enter a number:1
Enter b number:2
addition of given number is 3
sub of given number is -1
mul of given number is 2
div of given number is 0
mod of given number is 1
Enter a number:|
```

Result:

Thus, the Arithmetic Operations using RPC is executes successfully.

Ex.No:10a	STUDY OF WIRESHARK TOOL FOR PACKET SNIFFING
Date:	

AIM:

To study packet sniffing concept using Wireshark Tool.

DESCRIPTION:

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and display them in human-readable format. Wireshark includes filters, color coding, and other

features that let you dig deep into network traffic and inspect individual packets. You can use Wireshark to inspect a suspicious program's network traffic, analyze the traffic flow on your network, or troubleshoot network problems.

What we can do with Wireshark:

- Capture network traffic
- Decode packet protocols using dissectors
- Define filters – capture and display
- Watch smart statistics
- Analyze problems
- Interactively browse that traffic

Wireshark used for:

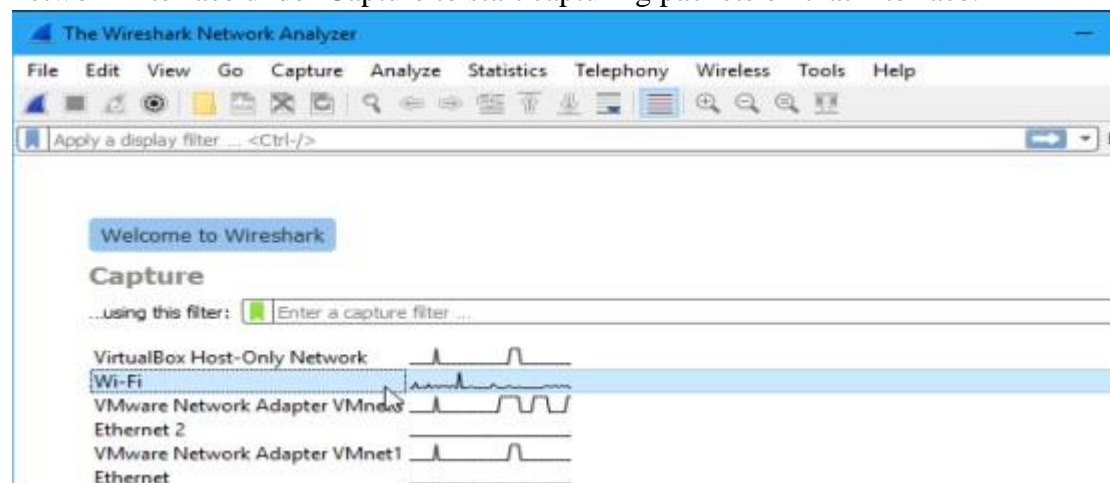
- Network administrators: troubleshoot network problems
- Network security engineers: examine security problems
- Developers: debug protocol implementations
- People: learn network protocol internals

Getting Wireshark

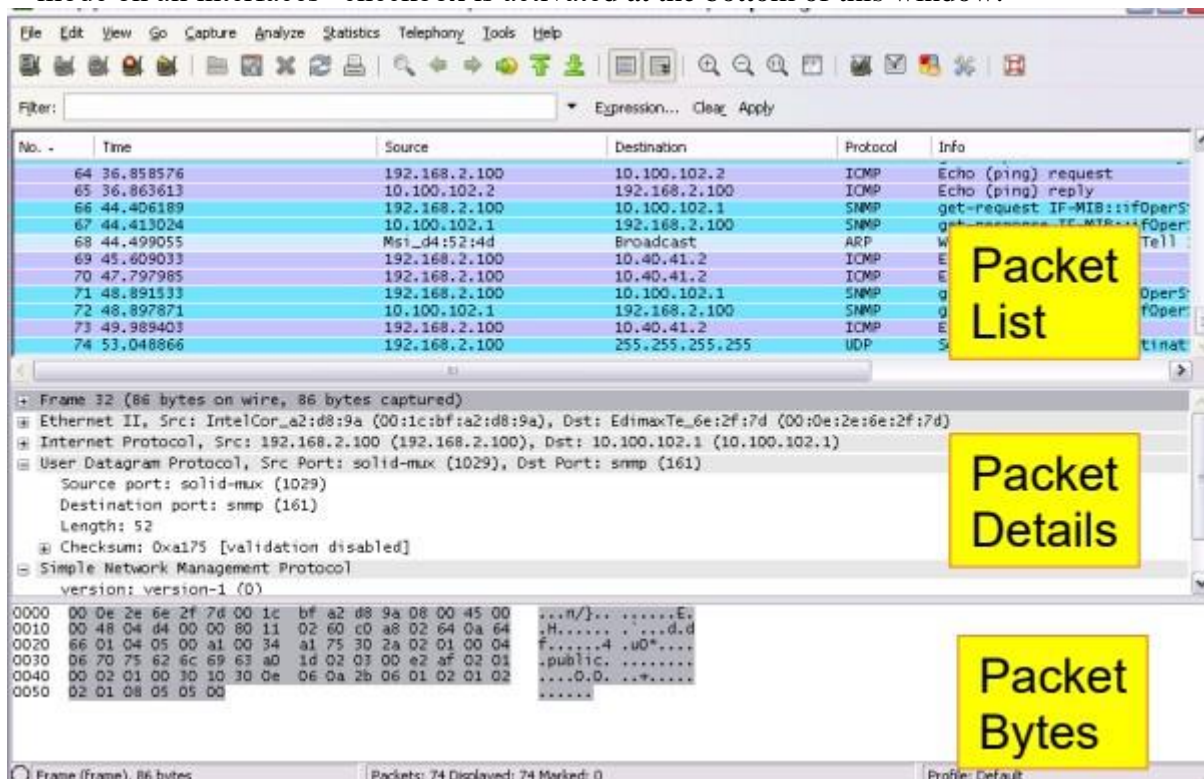
Wireshark can be downloaded for Windows or macOS from its official website. For Linux or another UNIX-like system, Wireshark will be found in its package repositories. For Ubuntu, Wireshark will be found in the Ubuntu Software Center.

Capturing Packets

After downloading and installing Wireshark, launch it and double-click the name of a network interface under Capture to start capturing packets on that interface.



As soon as you click the interface's name, you'll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system. If you have promiscuous mode enabled—it's enabled by default—you'll also see all the other packets on the network instead of only packets addressed to your network adapter. To check if promiscuous mode is enabled, click Capture > Options and verify the "Enable promiscuous mode on all interfaces" checkbox is activated at the bottom of this window.



The “Packet List” Pane

The packet list pane displays all the packets in the current capture file. The “Packet List” pane Each line in the packet list corresponds to one packet in the capture file. If you select a line in this pane, more details will be displayed in the “Packet Details” and “Packet Bytes” panes.

The “Packet Details” Pane

The packet details pane shows the current packet (selected in the “Packet List” pane) in a more detailed form. This pane shows the protocols and protocol fields of the packet selected in the “Packet List” pane. The protocols and fields of the packet shown in a tree which can be expanded and collapsed.

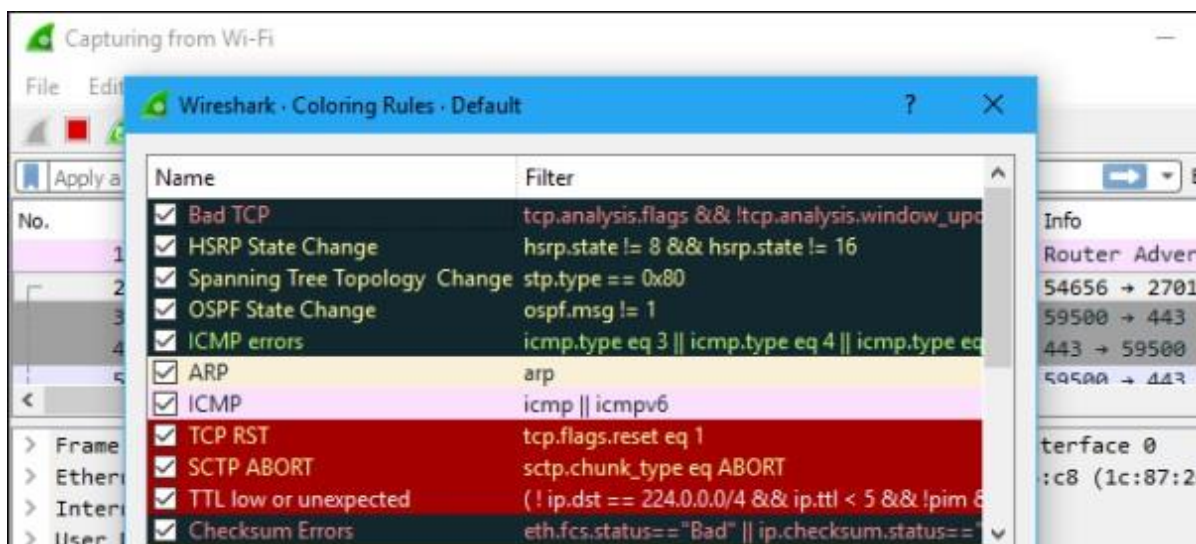
The “Packet Bytes” Pane

The packet bytes pane shows the data of the current packet (selected in the “Packet List” pane) in a hexdump style.

Color Coding

You'll probably see packets highlighted in a variety of different colors. Wireshark uses colors to help you identify the types of traffic at a glance. By default, light purple is TCP traffic, light blue is UDP traffic, and black identifies packets with errors—for example, they could have been delivered out of order.

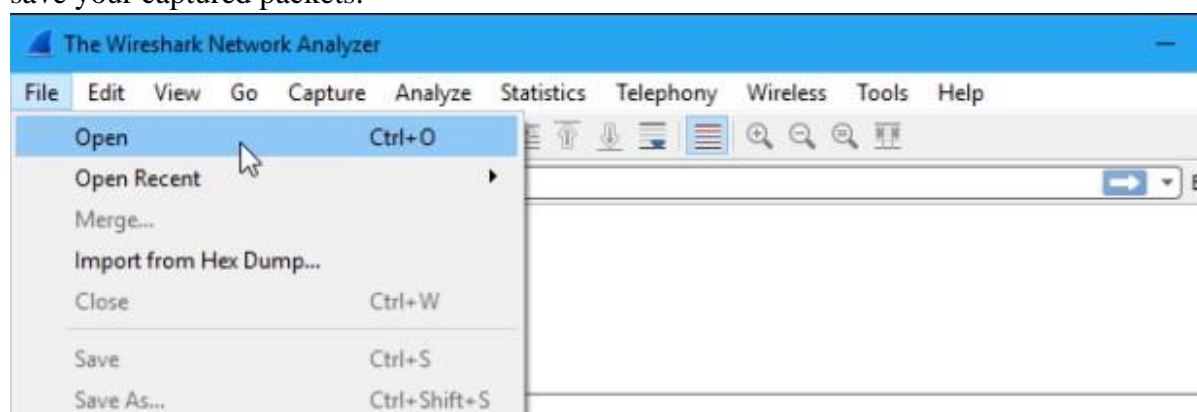
To view exactly what the color codes mean, click View > Coloring Rules. You can also customize and modify the coloring rules from here, if you like.



Sample Captures

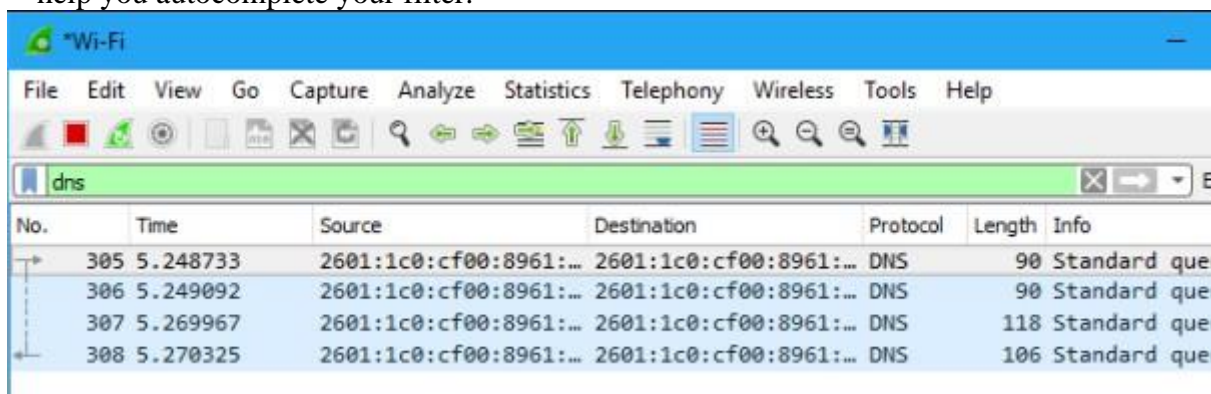
If there's nothing interesting on your own network to inspect, Wireshark's wiki has you covered. The wiki contains a page of sample capture files that you can load and inspect. Click File > Open in Wireshark and browse for your downloaded file to open one.

You can also save your own captures in Wireshark and open them later. Click File > Save to save your captured packets.



Filtering Packets

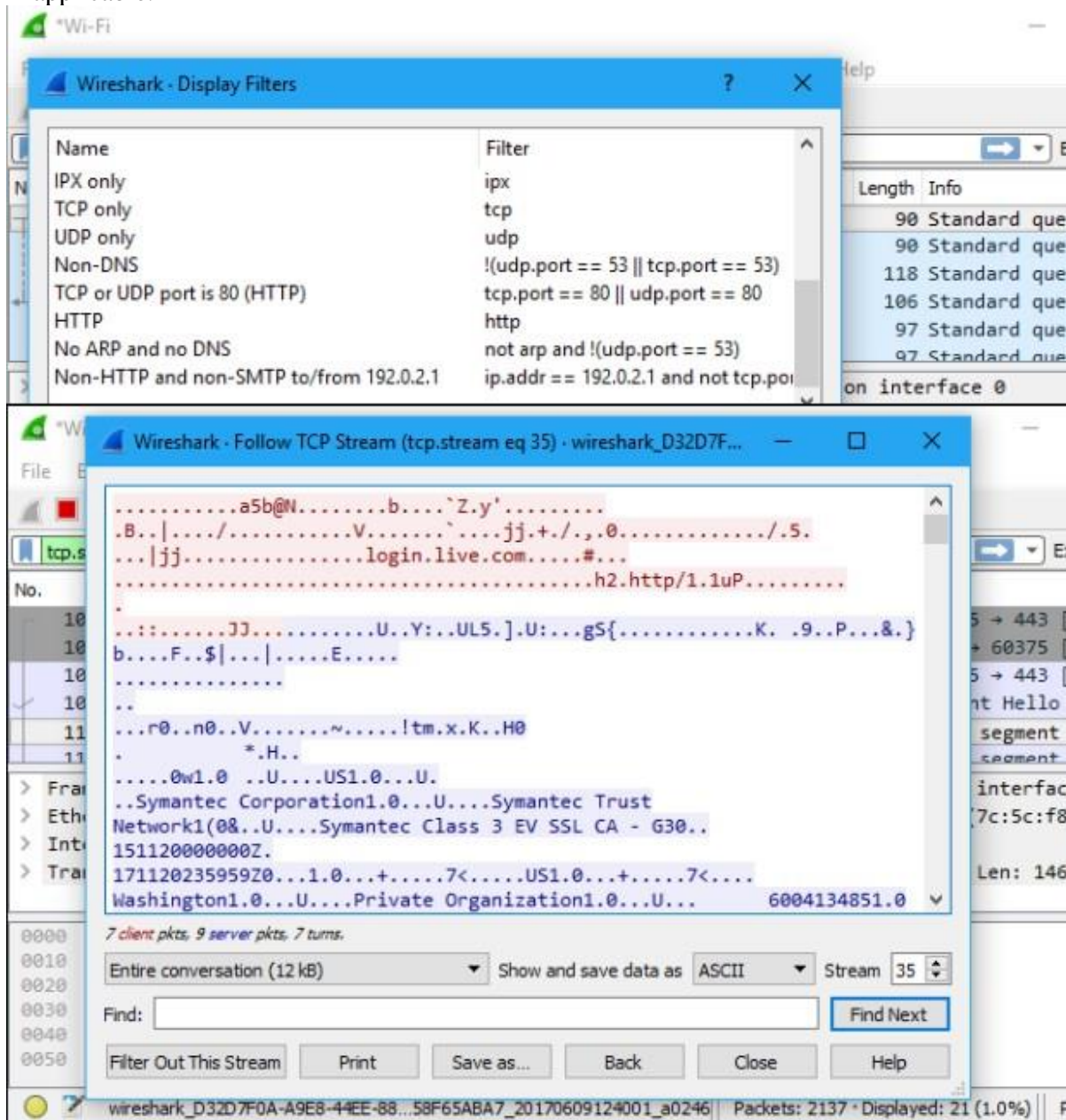
If you're trying to inspect something specific, such as the traffic a program sends when phoning home, it helps to close down all other applications using the network so you can narrow down the traffic. Still, you'll likely have a large amount of packets to sift through. That's where Wireshark's filters come in. The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type "dns" and you'll see only DNS packets. When you start typing, Wireshark will help you autocomplete your filter.



For more information on Wireshark's display filtering language, read the Building display filter expressions page in the official Wireshark documentation.

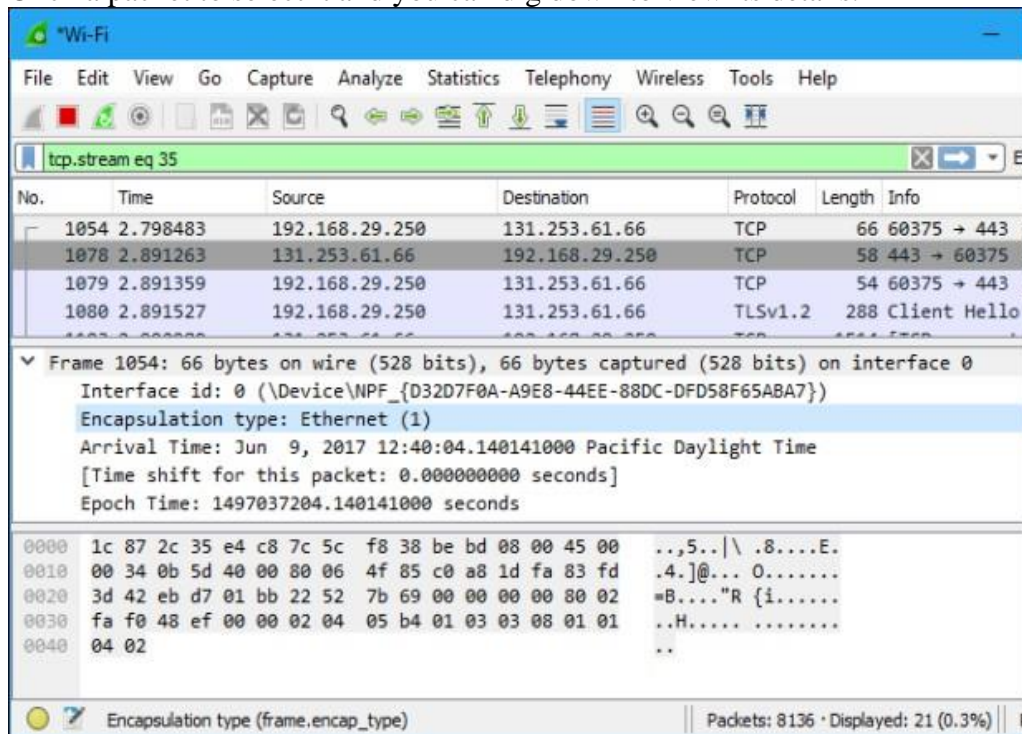
Another interesting thing you can do is right-click a packet and select Follow > TCP Stream.

You'll see the full TCP conversation between the client and the server. You can also click other protocols in the Follow menu to see the full conversations for other protocols, if applicable.

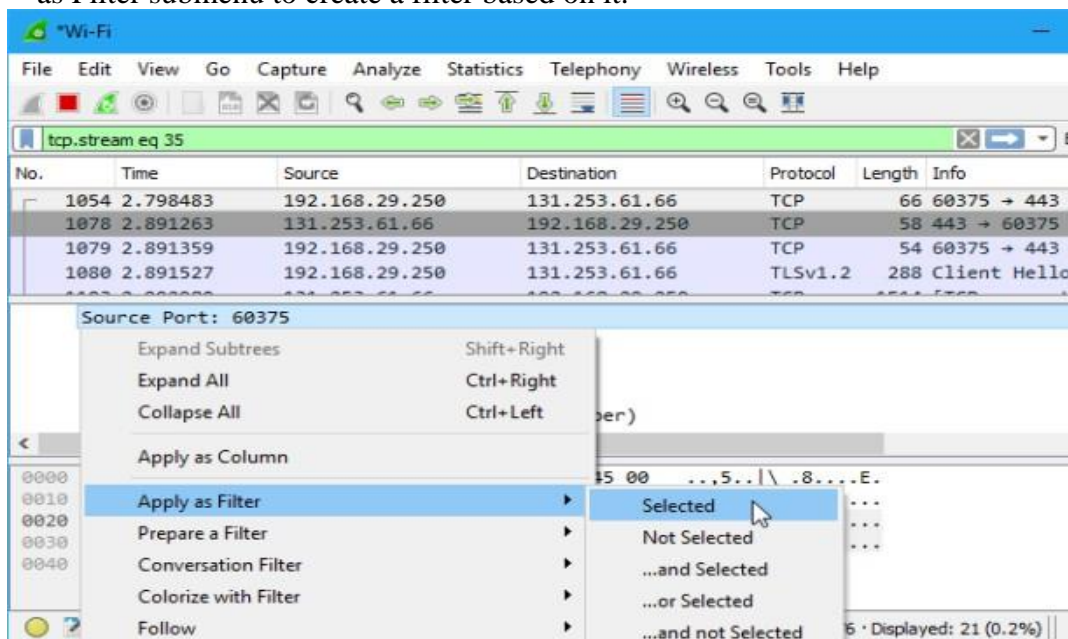


Inspecting Packets

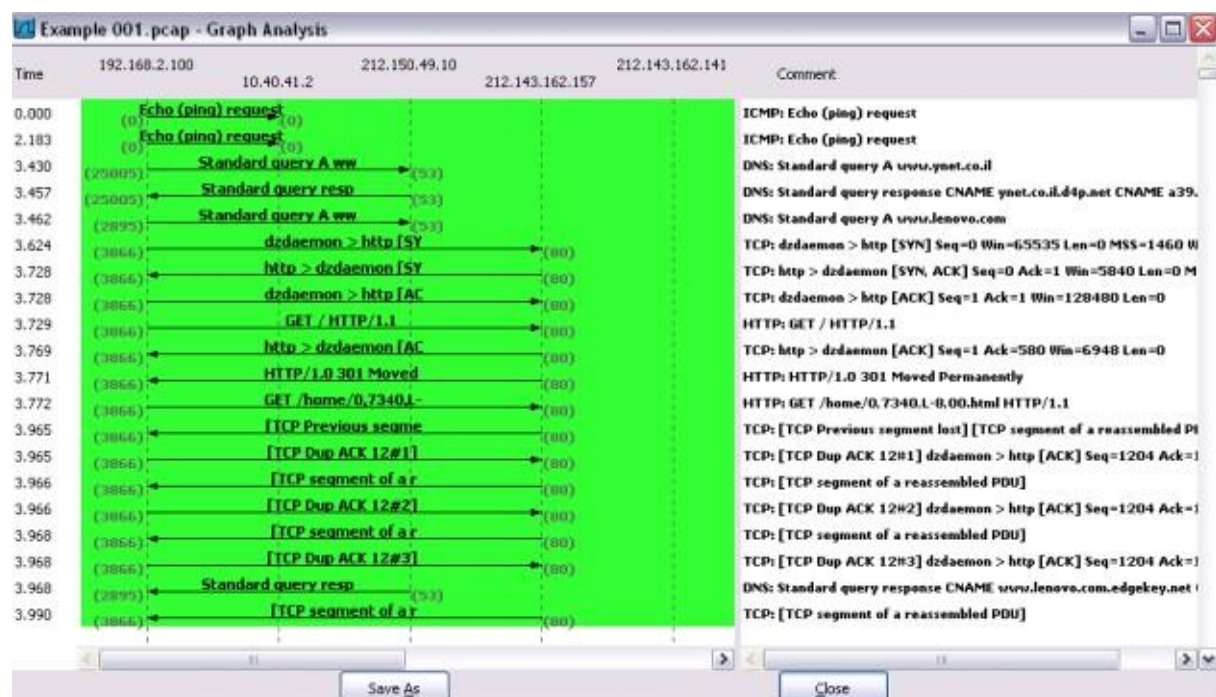
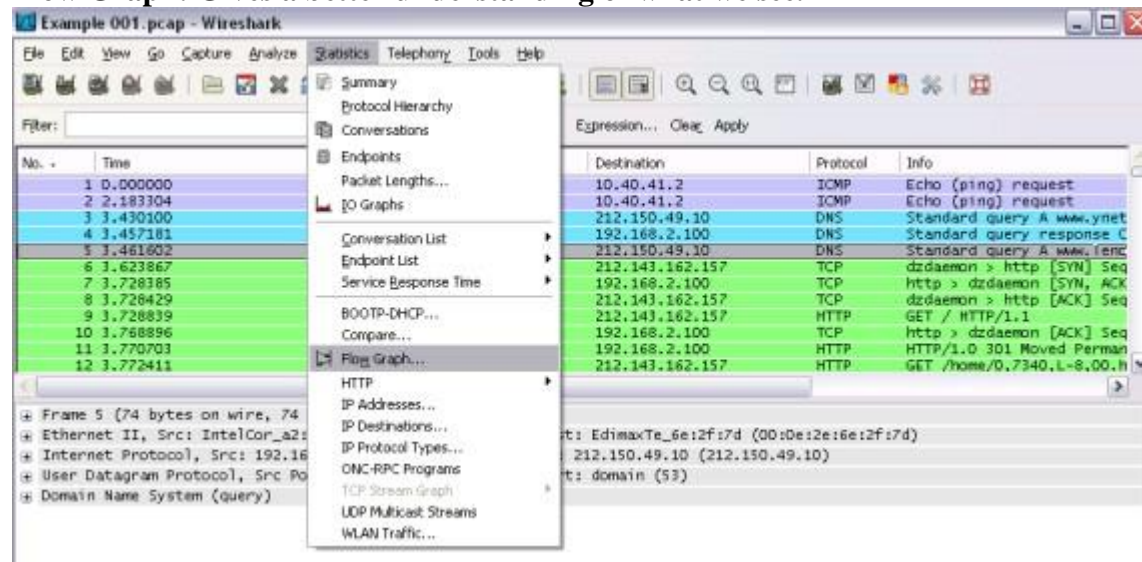
Click a packet to select it and you can dig down to view its details.



You can also create filters from here — just right-click one of the details and use the Apply as Filter submenu to create a filter based on it.



Flow Graph: Gives a better understanding of what we see.



Result:

Thus, the Study of Wireshark is completed.

Ex.No:10b

Date:

PACKET SNIFFING USING WIRESHARK

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	Realtek5_55:2c:b8	Broadcast	ARP	60	who has 172.16.10.36? Tell 172.16.10.50
3	0.000294	Realtek5_55:2c:b8	Broadcast	ARP	60	who has 172.16.11.36? Tell 172.16.10.50
4	0.000295	Realtek5_55:2c:b8	Broadcast	ARP	60	who has 172.16.8.37? Tell 172.16.10.50
5	0.000296	Realtek5_55:2c:b8	Broadcast	ARP	60	who has 172.16.9.37? Tell 172.16.10.50
6	0.000296	Realtek5_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
11	0.045051	fe80::4968:12a7:5e3...	ff02::1:3	LLMNR	95	Standard query 0xae2b A 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	LS-----
0010	00 00 00 29 11 01 fe 00	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 4e 4c 33	----		-----TLFL3
0050	2d 48 44 43 31 38 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 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 TCP packets in search bar.
- To see flow graph click Statistics ☐ Flow graph.
- Save the packets.

Outputs:

No.	Time	Source	Destination	Protocol	Length	Info
122	4.557032	fe80::4968:12a7:5e3...	ff02::1:3	TCP	74	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
123	4.557091	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
124	4.557172	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
125	4.557254	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
126	4.557335	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
127	4.557417	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
128	4.557498	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
129	4.557579	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
130	4.557660	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
131	4.557741	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
132	4.557822	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
133	4.557903	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
134	4.557984	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
135	4.558065	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
136	4.558146	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
137	4.558227	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
138	4.558308	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
139	4.558389	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
140	4.558470	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
141	4.558551	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
142	4.558632	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
143	4.558713	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
144	4.558794	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
145	4.558875	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
146	4.558956	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
147	4.559037	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
148	4.559118	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
149	4.559199	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
150	4.559280	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
151	4.559361	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
152	4.559442	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
153	4.559523	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
154	4.559604	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
155	4.559685	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
156	4.559766	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
157	4.559847	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
158	4.559928	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
159	4.559999	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
160	4.560070	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
161	4.560151	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
162	4.560232	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
163	4.560313	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
164	4.560394	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
165	4.560475	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
166	4.560556	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
167	4.560637	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
168	4.560718	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
169	4.560799	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
170	4.560880	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
171	4.560961	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
172	4.561042	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
173	4.561123	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
174	4.561204	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
175	4.561285	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
176	4.561366	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
177	4.561447	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
178	4.561528	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
179	4.561609	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
180	4.561690	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
181	4.561771	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
182	4.561852	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
183	4.561933	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
184	4.562014	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
185	4.562095	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
186	4.562176	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
187	4.562257	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
188	4.562338	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
189	4.562419	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
190	4.562500	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
191	4.562581	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
192	4.562662	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
193	4.562743	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
194	4.562824	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
195	4.562905	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
196	4.562986	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
197	4.563067	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
198	4.563148	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
199	4.563229	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
200	4.563310	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
201	4.563391	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
202	4.563472	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
203	4.563553	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
204	4.563634	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
205	4.563715	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
206	4.563796	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
207	4.563877	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
208	4.563958	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
209	4.564039	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
210	4.564120	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
211	4.564201	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
212	4.564282	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
213	4.564363	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
214	4.564444	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
215	4.564525	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
216	4.564606	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
217	4.564687	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
218	4.564768	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
219	4.564849	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
220	4.564930	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
221	4.565011	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
222	4.565092	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
223	4.565173	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
224	4.565254	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
225	4.565335	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
226	4.565416	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
227	4.565497	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
228	4.565578	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
229	4.565659	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
230	4.565740	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
231	4.565821	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
232	4.565902	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
233	4.565983	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
234	4.566064	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
235	4.566145	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
236	4.566226	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
237	4.566307	172.16.8.96	172.16.8.96	TCP	60	1980 → 2888 [RST, ACK] Seq=1431610200 Len=0
238	4.566388	172.16.8.96				

3. 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 search bar.
- Save the packets.

Output:

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 172.16.10.20
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
71	3.420652	Dell_35:11:11	Broadcast	ARP	60	who has 172.16.8.118? Tell 172.16.10.195

▶ 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 0c 10 09 60	-----	----
0020	00 e0 4c b2 60 90 ac 10	09 6a	--L- - - - -	----

4. 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 search bar.
- To see flow graph click Statistics□Flow graph.
- Save the packets.

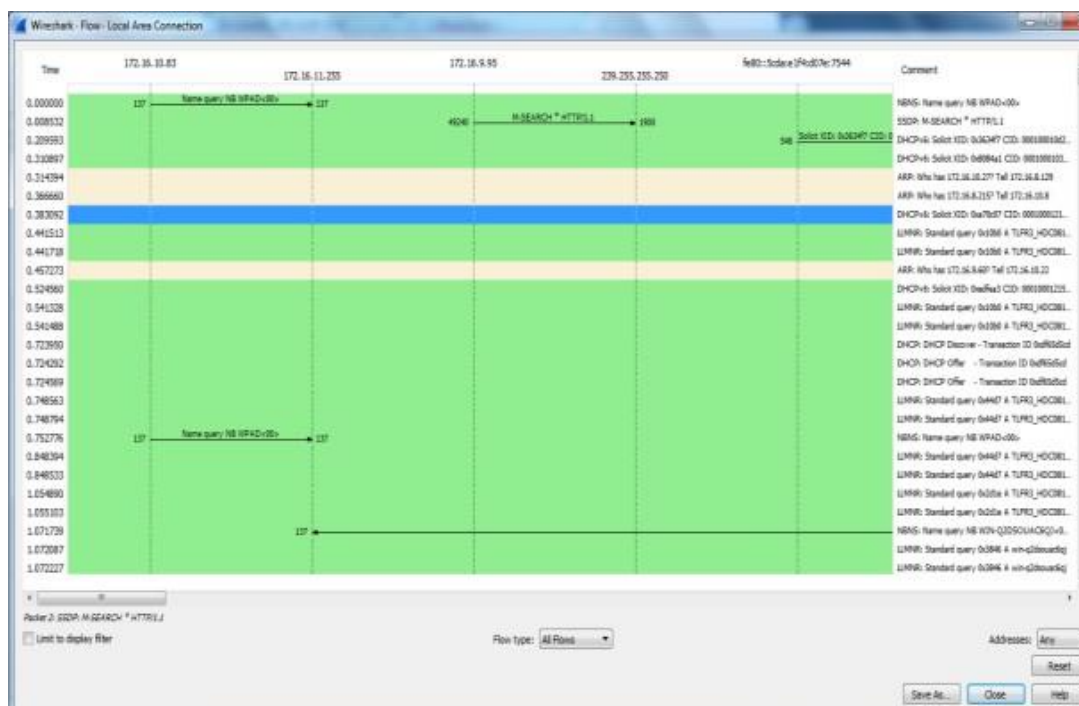
Output:

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 0xb5f4 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 0xb5f4 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 gstaticads1.l.google.com A 172.217.168.131
1738	36.875963	172.16.9.96	172.16.8.1	DNS	80	Standard query 0x7a09 A accounts.youtube.com
1739	36.875294	172.16.8.1	172.16.9.96	DNS	124	Standard query response 0x7a09 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 (00:27:9e: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  00 35 71 f2 b4 a1 00 27 0e 13 ed 7c 00 00 45 00  -Sq-----E-
0010  00 3c 37 1b 00 00 00 11 00 00 ac 10 09 00 ac 10  -<Z-----T-
0020  00 02 f3 46 00 35 00 28 69 bb 0e 40 01 00 00 01  -...F.5 { i : @ ...
0030  00 00 00 00 00 00 00 00 03 77 77 77 06 67 6f 64  -...w ww googl
0040  65 83 63 6f 6d 00 00 01 00 01                  e .com.....
  
```

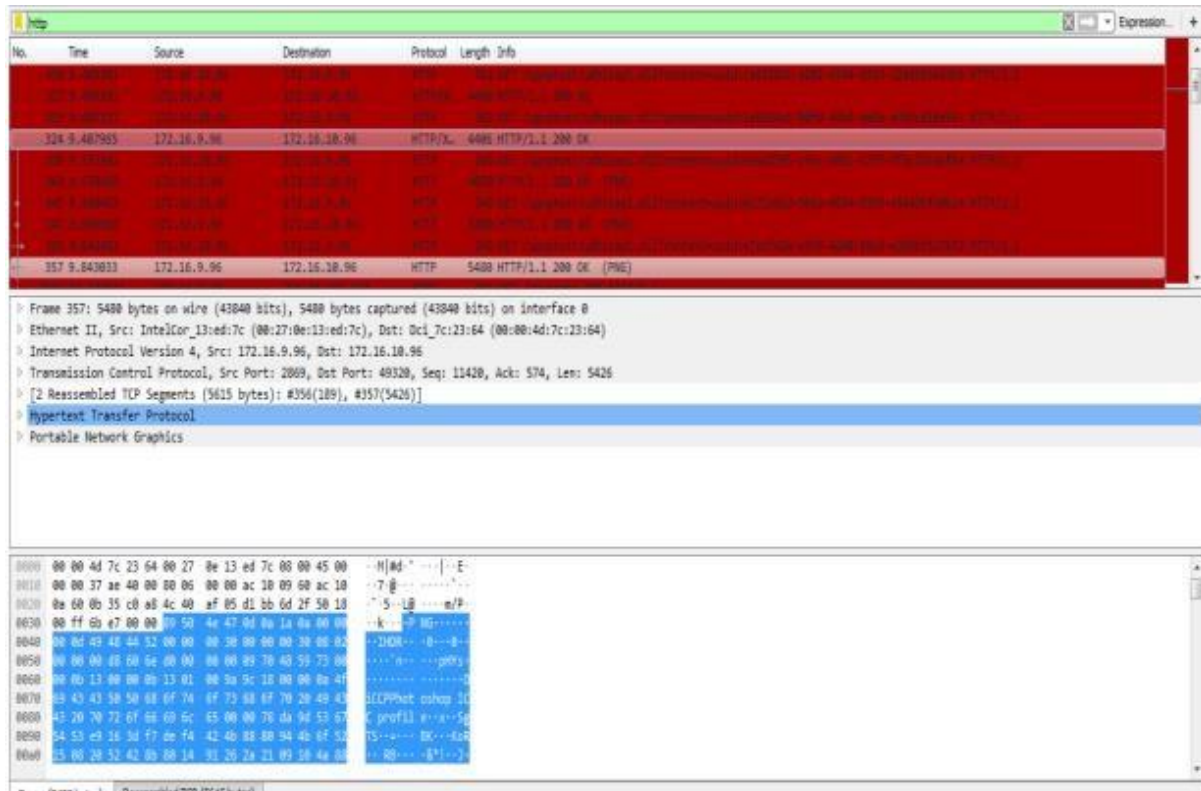


5.Create a Filter to display only HTTP packets and inspect the packets

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 search bar.
- Save the packets.

Output:



6. 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 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 IPAD<00>
2	0.000532	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 0x1800 A TLFR3_HDC081307
13	0.541488	172.16.9.91	224.0.0.252	LLMNR	75	Standard query 0x1800 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	0.740794	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 IPAD<00>
21	0.848533	172.16.9.91	224.0.0.252	LLMNR	75	Standard query 0x44d7 A TLFR3_HDC081307
22	0.865102	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 00 00 4d 7c 23 ad 00 00 45 00N #...E
0010	00 4e 53 79 00 00 00 11 78 b3 ac 10 0a 53 ac 10	..NSy....X....S..
0020	0b ff 00 89 00 09 00 3a 56 90 f7 f6 01 10 00 01:V.....
0030	00 00 00 00 00 00 00 46 48 46 41 45 42 45 45 43F HFAEBEE
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 00 00 20 00 01	ACACAAA ...

7. 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 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: 000100010d2a7d3000004d7c6d3e
4	0.310097	fe80::d40:4448:5018...	ff02::1:2	DHCPv6	153	Solicit XID: 0x0004a1 CID: 0001000103c2673c00e04c552cb8
7	0.383092	fe80::5147:5612:a48...	ff02::1:2	DHCPv6	158	Solicit XID: 0xa70c87 CID: 00010001215e5349d43d7ec887b3
11	0.524560	fe80::f021:1809:190...	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: 000100010d2a7d3000004d7c6d3e
85	3.220007	fe80::5cda:e1f4:d07...	ff02::1:2	DHCPv6	150	Solicit XID: 0x3634f7 CID: 000100010d2a7d3000004d7c6d3e
96	3.649834	fe80::c159:d7a6:a74...	ff02::1:2	DHCPv6	156	Solicit XID: 0x8236e6 CID: 00010001210226a5001f40e2a391
109	4.006915	fe80::c405:5e2c:922...	ff02::1:2	DHCPv6	156	Solicit XID: 0xd802548 CID: 0001000121d5ad5200270e13ee4
118	4.311356	fe80::d40:4448:5018...	ff02::1:2	DHCPv6	153	Solicit XID: 0x0004a1 CID: 0001000103c2673c00e04c552cb8
211	7.240392	fe80::5cda:e1f4:d07...	ff02::1:2	DHCPv6	150	Solicit XID: 0x3634f7 CID: 000100010d2a7d3000004d7c6d3e
314	7.302503	fe80::302a:66c3:8c2...	ff02::1:2	DHCPv6	150	Solicit XID: 0x6355 CID: 000100012153c05f00270e13f6d9

> 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: IPv6multicast_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 06 dd 60 00  33.....H(mo...
0010  00 00 00 60 11 01 fe 00 00 00 00 00 00 5c da    ....
0020  e1 f4 d0 7e 75 44 ff 02 00 00 00 00 00 00 00    ...ad.....
0030  00 00 00 01 00 02 02 22 02 23 00 00 7d b0 01 36  ....*#]...6
0040  34 f7 00 00 00 02 00 00 00 01 00 0e 00 01 00 01  4.....
0050  00 2a 7d 38 00 00 4d 7c 6d 3e 00 03 00 0c 0e 00  -*]8-N|ms....
0060  00 4d 00 00 00 00 00 00 00 00 27 00 0a 00 00    -H.....
0070  61 64 6d 69 6e 2d 50 43 00 10 00 0e 00 00 01 37  admin-PC.....7
0080  00 00 4d 53 46 54 20 35 2e 30 00 06 00 00 00 18  -HSFT 5 .0....
0090  00 17 00 11 00 27    .....
  
```

Result:

Thus, the Packet Sniffing using wireshark is executed successfully.

AIM:

Ex.No:11	STUDY OF DIFFERENT CABLES AND CRIMPING OF CABLE WITH RJ45 CONNECTOR
Date:	

To study the different kinds of cables used in networks and do perform crimping of network cable with RJ4 connector.

DESCRIPTION:

Some of the types of cable are:

1. Unshielded Twisted Pair (UTP) Cable
2. Shielded Twisted Pair (STP) Cable
3. Coaxial Cable
4. Fiber Optic Cable
5. Wireless LANs
6. USB Cables
7. Crossover Cables

SPECIFICATION:**1. UNSHIELDED TWISTED PAIR (UTP) CABLE:**

Category of Cable	Cable Type	Maximum Data Transmission
Category 3	UTP	10 Mbps
Category 5	UTP	10/100 Mbps
Category 5 e	UTP	1000 Mbps
Category 6	UTP or STP	1000 Mbps
Category 6a, Category 7	STP, SSTP	10,000 Mbps

2. SHIELDED TWISTED PAIR (STP) CABLE:

a) What speeds for data transmission can be achieved using a particular type of cable?

The speed of the data transmission is 10 to 100 Mbps can be achieved using Shielded Twisted Pair Cable (STP).

b) What kind of transmission is being considered? Will the transmissions be digital or will they be analog-based?

Guided Media is a kind of transmission is being considered in STP. For Analog Signals, Amplifiers are required about every 5 to 6 Km. For Digital Signals, Repeaters are required about every 2 to 3 Km.

c) How far can a signal travel through a particular type of cable before attenuation of that signal becomes a concern? In other words, will the signal become so degraded that the recipient device might not be able to accurately receive and interpret the signal by the time the signal reaches that device?

Signal travel through Shielded Twisted Pair Cable is 100 meters.

d) (I) Advantage:

- Shielded.
- Faster than UTP.
- STP is less susceptible to noise as compared to UTP and therefore reduces the cross talk and interference.

(II) Disadvantage:

- More expensive than UTP.
- High attenuation rate.
- It is difficult to terminate and It must be properly grounded.

3. COAXIAL CABLE:

a) What speeds for data transmission can be achieved using a particular type of cable?

The speeds of the data transmission are 10 to 100 Mbps can achieved using a Coaxial Cable.

b) What kind of transmission is being considered? Will the transmissions be digital or will they be analog-based?

Guided Media is kind of transmission is being considered in Coaxial Cable. The transmission is occurred in both ways. During digital transmission, the Repeater every 1Km closer for higher data rates. In Analog transmission, Amplifier every few Km closer if high frequency up to 500MHz

c) How far can a signal travel through a particular type of cable before attenuation of that signal becomes a concern? In other words, will the signal become so degraded that the recipient device might not be able to accurately receive and interpret the signal by the time the signal reaches that device?

Performance limited by attenuation and noise. Speed of signal travel in Coaxial Cable are 500 meters.

d) (I) Advantage:

- Higher bandwidth
- Much less susceptible to interference than twisted pair.

(II) Disadvantage:

- High attenuation rate makes it expensive over long distance.
- Bulky.

4. FIBER OPTIC CABLE:

a) What speeds for data transmission can be achieved using a particular type of cable?

The speed of the data transmission is 100 Mbit/s or Gbit/s can be achieved using a Fiber Optic Cable.

b) What kind of transmission is being considered? Will the transmissions be digital or will they be analog-based?

Guided Media is a kind of transmission is being considered in Fiber Optic Cable. It is mostly digital signal. However, you can also transmit an analog signal over fiber optic, such as a video.

c) How far can a signal travel through a particular type of cable before attenuation of that signal becomes a concern? In other words, will the signal become so degraded that the recipient device might not be able to accurately receive and interpret the signal by the time the signal reaches that device?

Speed of Signal travel in Fiber Optic Cable are 80 to 100 Kilometers.

d) (I) Advantage:

- Greater capacity (bandwidth of up to 2Gbps)
- Smaller size and lighter weight
- Lower attenuation

(II) Disadvantage:

- Expensive over short distance
- Requires highly skilled installers
- Adding additional nodes is difficult

5. WIRELESS LAN'S:

a) What speeds for data transmission can be achieved using a particular type of cable?

The speeds of the data transmission are 100 Megabyte per second can be achieved using

a Wireless LAN's.

b) What kind of transmission is being considered? Will the transmissions be digital or will they be analog-based?

Unguided Media is a kind of transmission and reception are achieved by means of an antenna in Wireless Lan's. If it refers to the medium, it is analog. If it refers to the data, it is digital.

c) How far can a signal travel through a particular type of cable before attenuation of that signal becomes a concern? In other words, will the signal become so degraded that the recipient device might not be able to accurately receive and interpret the signal by the time the signal reaches that device?

Speed of Signal travel in Wireless LAN'S are 2 Kilometers.

d) (I) Advantage:

- Installation flexibility.
- Reduced cost-of-ownership.
- Mobility.

(II) Disadvantage:

- Cost.
- Environmental Conditions.
- Less capacity.

6. USB CABLES:

a) What speeds for data transmission can be achieved using a particular type of cable?

The speeds of the data transmission which transfers data at a maximum rate of 1.5 Mbit per second and with a 480 Mbit per second data transfer rate.

b) What kind of transmission is being considered? Will the transmissions be digital or will they be analog-based?

The USB Cable is kind of asynchronous serial data transmission between a computer and attached devices. Clock and Data Recovery circuits etc., and they are in analog domain. Once the data is sampled, the processing is done in the digital domain.

c) How far can a signal travel through a particular type of cable before attenuation of that signal becomes a concern? In other words, will the signal become so degraded that the recipient device might not be able to accurately receive and interpret the signal by the time the signal reaches that device?

Speed of Signal travel in USB Cable are 30 meters.

d) (I) Advantage:

- It has low cost
- It has variety of connector types and size available
- It has true plug and play nature
- It has low power consumption

(II) Disadvantage:

- It has limited capability and overall performance.
- Universal Serial Bus does not provide the broadcasting feature, only individual messages can be communicated between host and peripheral.
- The data transfer not as fast as some other systems.

7. CROSSOVER CABLES:

a) What speeds for data transmission can be achieved using a particular type of cable?

The speeds of the data transmission are 10BASE-T and 100BASE-TX are being achieved using Crossover Cable.

b) What kind of transmission is being considered? Will the transmissions be digital or will they be analog-based?

The transmission involved here both analog and digital transmission.

c) How far can a signal travel through a particular type of cable before attenuation of that signal becomes a concern? In other words, will the signal become so degraded that the recipient device might not be able to accurately receive and interpret the signal by the time the signal reaches that device?

Speed of Signal travel in Crossover cable is 100 meters.

d) (I) Advantage:

- Easy to set
- Inexpensive to set up

(II) Disadvantage:

- Difficult to Expand
- Not Trusted
- Gigabit Incompatibility
- Alternatives to Crossover Connections

STEPS FOR CRIMPING CABLE WITH RJ4 CONNECTOR:

1. Run the full length of ethernet cable in place, from endpoint to endpoint, making sure to leave excess.
2. At one end, cut the wire to length leaving enough length to work, but not too much excess.
3. Strip off about 2 inches of the ethernet cable sheath.
4. Align each of the colored wires according to the layout of the jack.
5. Use the punch down tool to insert each wire into the jack.
6. Repeat the above steps for the second RJ45 jack.

Result:

Thus , the Study Of Different Cables And Crimping Of Cable With Rj45 Connector is completed successfully.

Ex.No:12

Date:

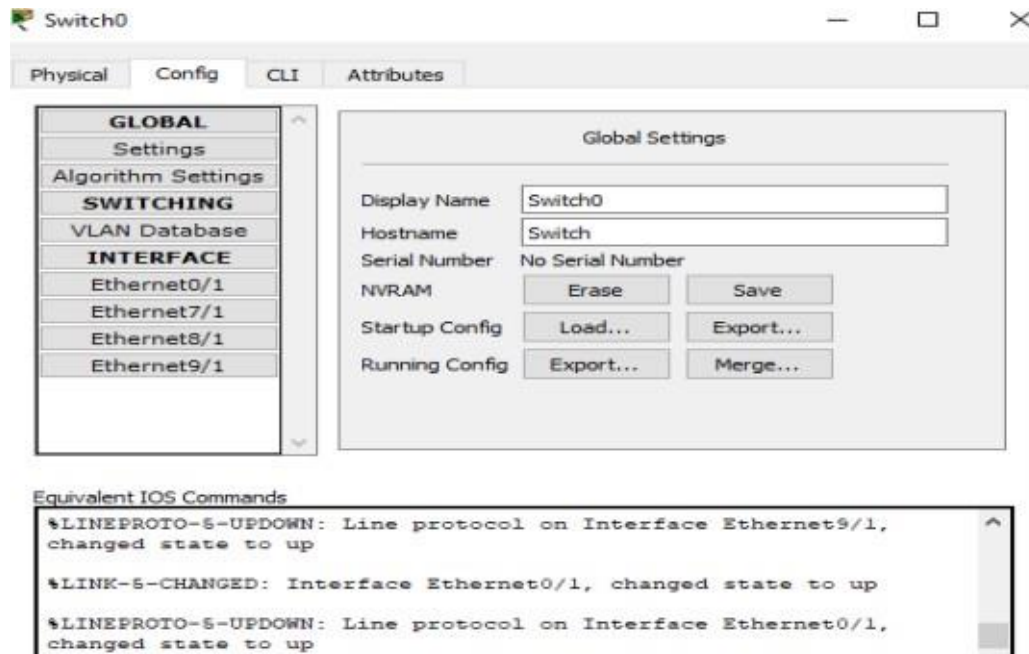
CONFIGURE NETWORK DEVICES USING PACKET TRACER

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.



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

The first part of Global settings allows you to change the Display name and Hostname of the device. The display name can also be changed by clicking on the name below the device icon. The configuration file for the device can also be saved, erased, or exported for later use. The Algorithm Settings section contains settings meant for advanced users who want to minutely tweak their device to see how it responds to certain situations. These settings can also be globally set for all network devices by navigating to Options | Algorithm Settings, or by using the shortcut Ctrl + Shift + M.

Interface 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.

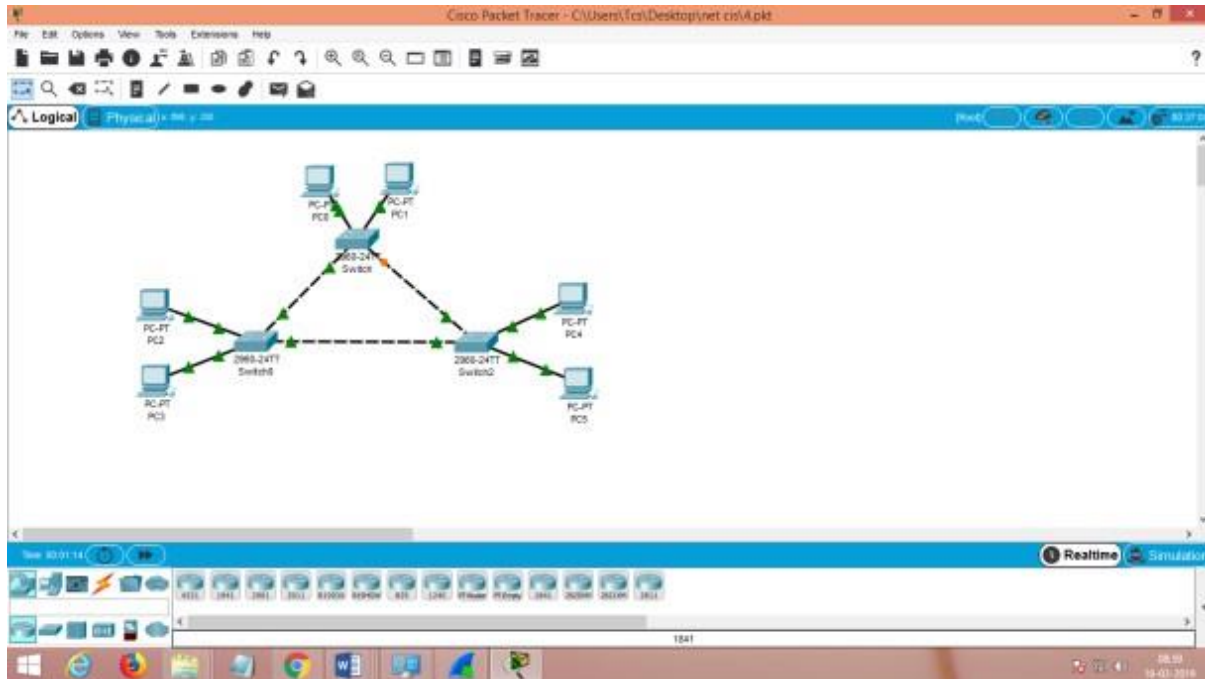
The screenshot shows the configuration window for router R1, specifically the 'Static Routes' section. On the left, a navigation pane lists various configuration categories: GLOBAL (Settings, Algorithm Settings), ROUTING (Static, RIP), SWITCHING (VLAN Database), and INTERFACE (FastEthernet0/0, FastEthernet0/1). The 'Static' option under ROUTING is selected. The main area is titled 'Static Routes' and contains three input fields: 'Network' with the value '192.168.40.0', 'Mask' with '255.255.255.0', and 'Next Hop' with '192.168.20.2'. Below these fields is an 'Add' button. A summary box below the inputs displays 'Network Address' as '192.168.30.0/24 via 192.168.10.2' and includes a 'Remove' button. At the bottom, a section titled 'Equivalent IOS Commands' shows a list of commands: 'Router(config)#', 'Router(config)#', 'Router(config)#', and '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)

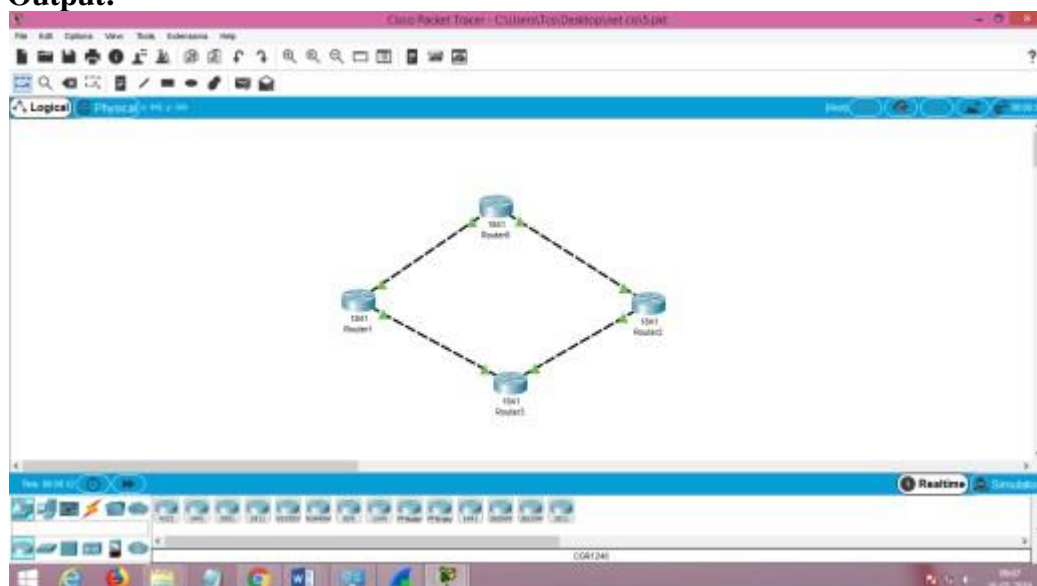
OUTPUT:



1. The topology is created as shown above.
2. The PC'S are configured.
3. The switches are configured.
4. Using the Config tab, the following can be configured:
5. Global settings
6. Routing (on a router and a layer 3 switch)
7. VLAN database (on a switch)
8. Interface setting

2. Perform 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 Network devices using Packet Tracer is executed successfully.

Ex.No:13

EXAMINE THE ARP TABLE USING PACKET TRACER

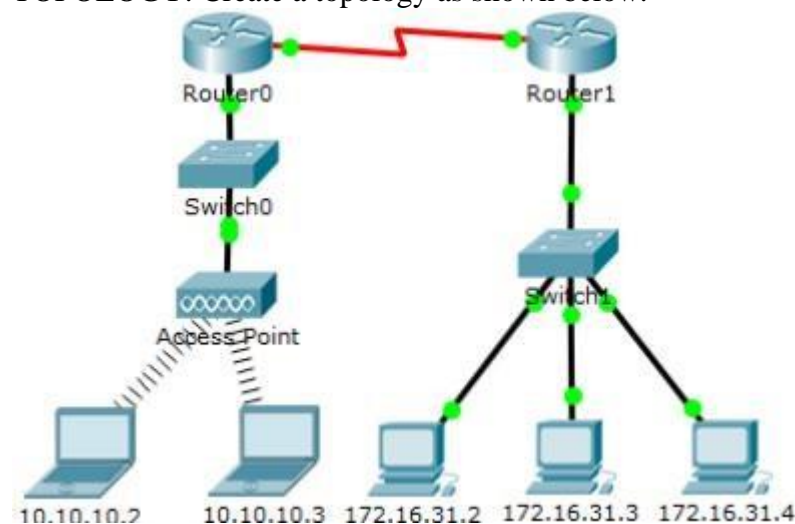
Date:

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

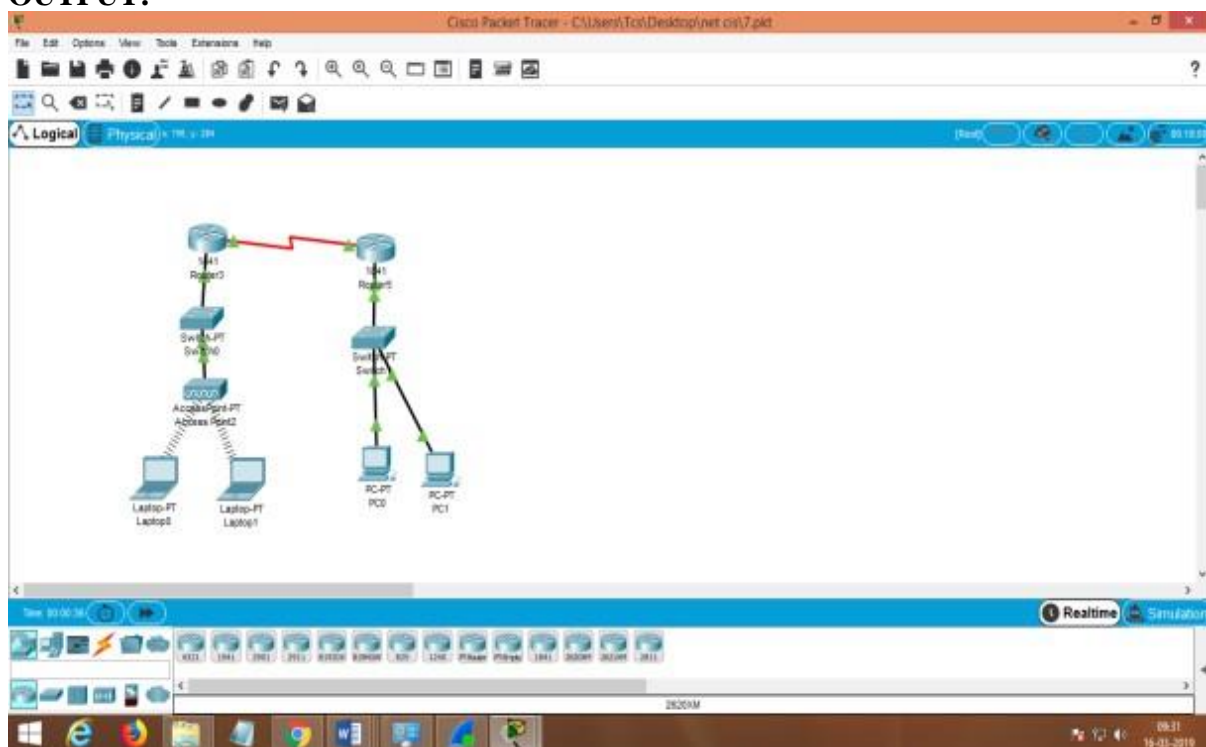
- a. Enter Simulation mode.
- b. Click 172.16.31.2 and open the Command Prompt.
- c. Enter the arp -d command to clear the ARP table.
- d. Enter the command ping 172.16.31.3.
- e. Two PDUs will be generated.
- f. The ping command cannot complete the ICMP packet without knowing the MAC address of the 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, waiting for 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 Switch1 and 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, the ARP table using Packet Tracer is examined successfully.

Ex.No:14

CREATING A SIMPLE NETWORK TOPOLOGY

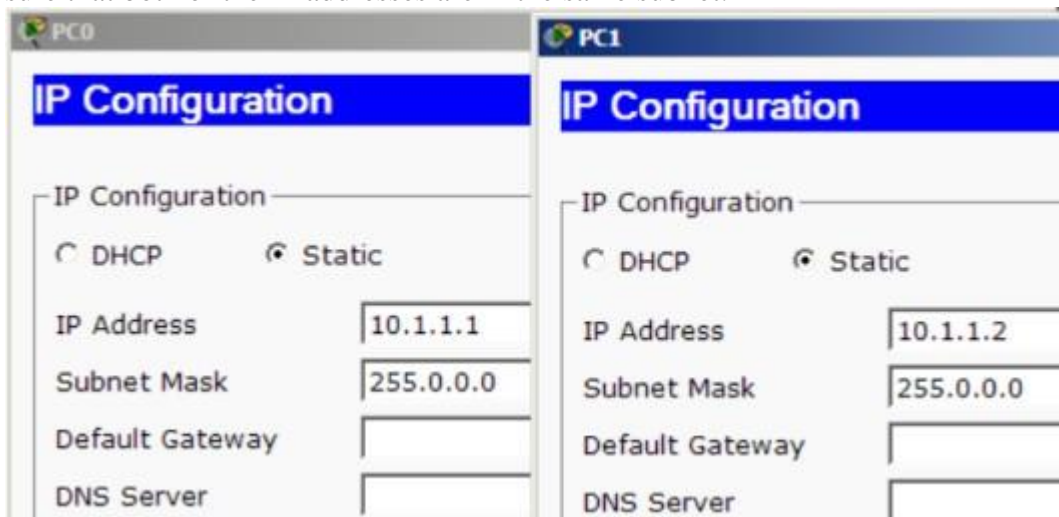
Date:

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 of the IP addresses are in the same subnet.



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
```

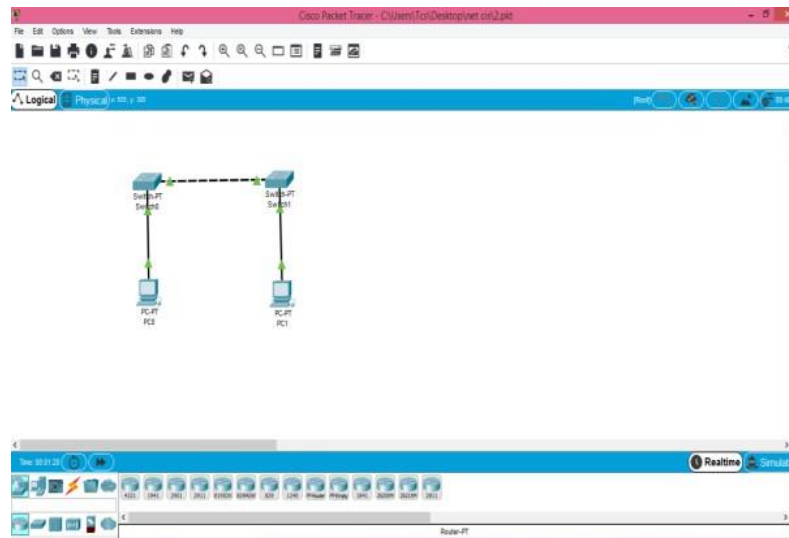
SAMPLE 2: TOPOLOGY WITH NETWORK DEVICE AND END DEVICES

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.

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



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, the Network Topology is created successfully.

Ex.No:15	STUDY OF CISCO PACKET TRACER
Date:	

AIM:

To study the Packet tracer tool Installation and User Interface Overview.

INTRODUCTION:

A simulator, as the name suggests, simulates network devices and its environment. Packet Tracer is an exciting network design, simulation and modelling tool.

1. It allows you to model complex systems without the need for dedicated equipment.
2. It helps you to practice your network configuration and troubleshooting skills via computer or an Android or iOS based mobile device.
3. It is available for both the Linux and Windows desktop environments.
4. Protocols in Packet Tracer are coded to work and behave in the same way as they would on real hardware.

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).

a) Windows

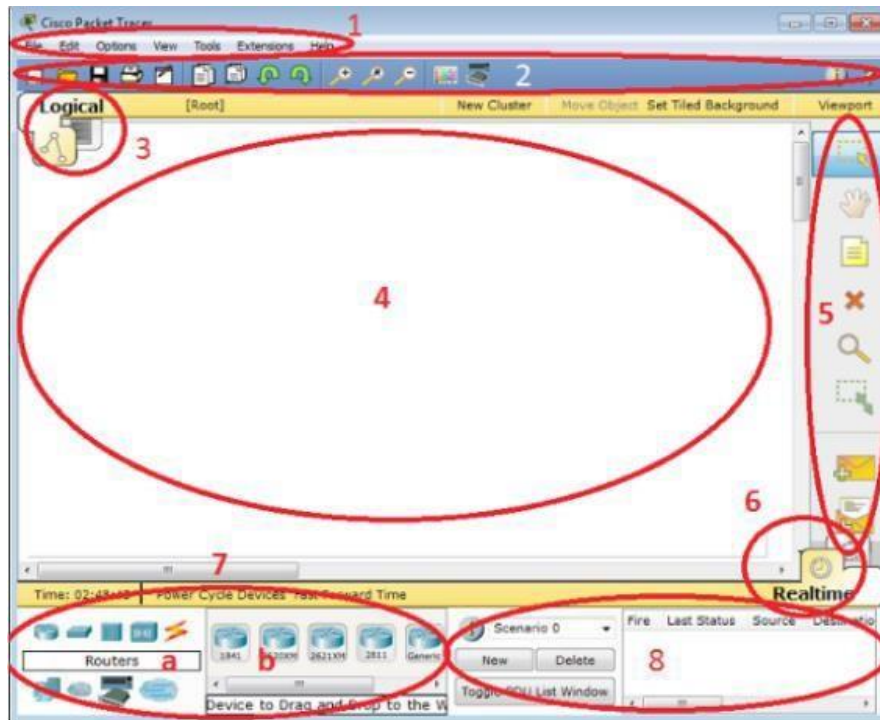
Installation in Windows is pretty simple and straightforward; the setup comes in a single file named Packettracer_Setup6.0.1.exe. Open this file to begin the setup wizard, accept the license agreement, choose a location, and start the installation.

b) 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, the Study of Cisco Packet Transfer is completed.

Ex.No:16	INSTALL, CONFIGURE AND RUN WEBALIZER
Date:	

Aim:

To install, configure and run webalizer on Fedora.

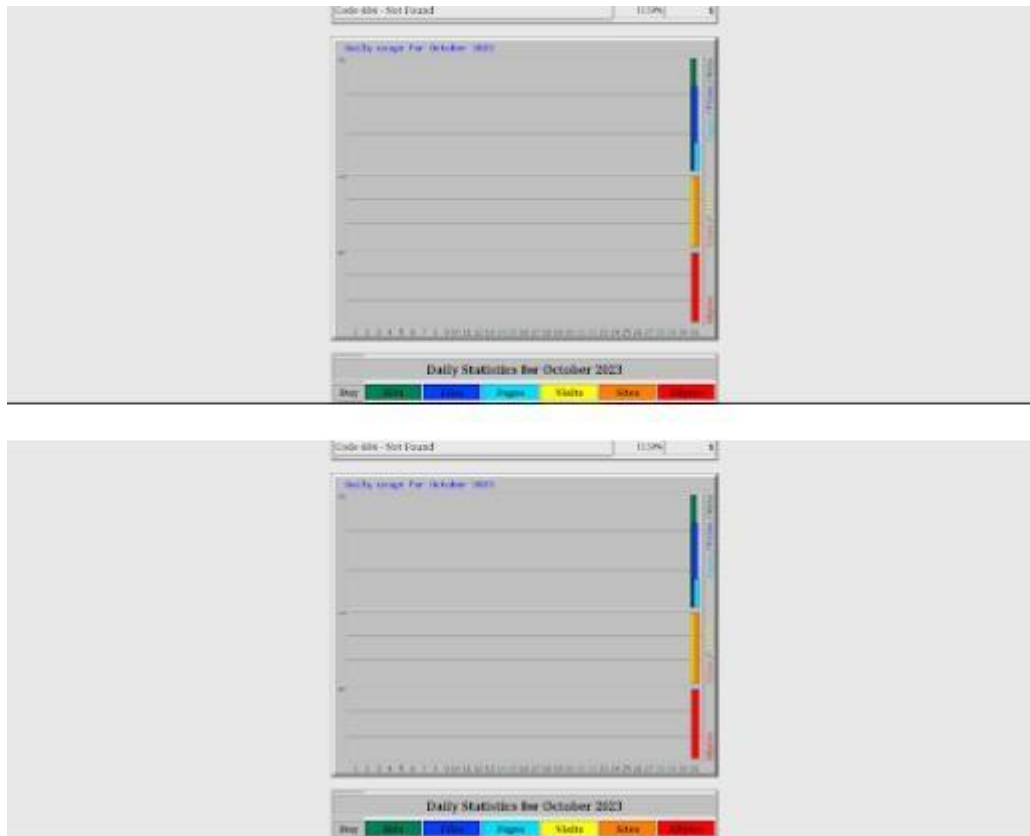
Algorithm:

1. Install apache web server
2. Set SELinux to permissive
3. Install and configure webalizer
4. Start webalizer

Program Code:

```
su
dnf install httpd
vi /etc/sysconfig/selinux
Put SELINUX=permissive instead of enforcing
systemctl enable httpd.service
systemctl start httpd.service
mkdir /var/www/html/webalizer
dnf install webalizer
cp * /var/www/usage /var/www/html
vi /etc/webalizer.conf
LogFile /var/log/httpd/access_log
LogType clf
Hostname localhost
OutputDir /var/www/html/webalizer
```

Output:



Result:

Thus , the webalizer is installed, configured and executed successfull

