# CB23531 COMPUTER NETWORK TECHNOLOGY LAB MANUAL

**List of Experiments**

1 Learn to use basic commands

2 Configuration of Network in Linux Environment.

3 Assignment of IP Address to computers.

4 Implementation of Subnet mask in IP addressing.

5 Implementation of setup of a Local Area Network (using Switches) – Minimum 3 nodes andInternet

6To capture, save, and analyse network traffic on TCP / UDP / IP / HTTP / ARP /DHCP /ICMP /DNS usingWireshark Tool.

7 Write a socket PING program to test the server connectivity.

8 Study of system administration and network administration

9 Study of socket programming and client server model using TCP and UDP.

10 Programs using TCP Sockets (like date and time server & client, echo server & client, chat etc.)

11 Programs using UDP Sockets (like echo server, chat, simple DNS).

12 Simulation of sliding window.

13 Implementation of ARP.

14 Configuring RIP.

15 Configuring a Cisco Router as a DHCP Server

**Ex.No. 1.** **BASIC LINUX COMMANDS**

**Date:**

**AIM:**

To learn the basic LINUX commands.

**DESCRIPTION:**

**1.1 GENERAL PURPOSE COMMANDS**

1. The 'date' command:

The date command display the current date with day of week, month, day, time (24 hours clock) and the year.

SYNTAX: $ date

The date command can also be used with following format.

| Format | Purpose | Example |
|---|---|---|
| + %m | To display only month | $ date + %m |
| + %h | To display month name | $ date + %h |
| + %d | To display day of month | $ date + %d |
| + %y | To display last two digits of the year | $ date + %y |
| + %H | To display Hours | $ date + %H |
| + %M | To display Minutes | $ date + %M |
| + %S | To display Seconds | $ date + %S |

2. The echo'command:

The echo command is used to print the message on the screen.

SYNTAX: $ echo

EXAMPLE: $ echo "God is Great"

3. The 'cal' command:

The cal command displays the specified month or year calendar.

SYNTAX: $ cal [month] [year]

EXAMPLE: $ cal Jan 2012

4. The 'bc' command:

Unix offers an online calculator and can be invoked by the command bc.

SYNTAX: $ bc

EXAMPLE: bc –l

16/4

5/2

5. The ‘who’ command

The who command is used to display the data about all the users who are currently logged into the system.

SYNTAX: $ who

6. The ‘who am i’ command

The who am i command displays data about login details of the user.

SYNTAX: $ who am i

7. The ‘id’ command

The id command displays the numerical value corresponding to your login.

SYNTAX: $ id

8. The ‘tty’ command

The tty (teletype) command is used to know the terminal name that we are using.

SYNTAX: $ tty

9. The ‘clear’ command

The clear command is used to clear the screen of your terminal.

SYNTAX: $ clear

10. The ‘man’ command

The man command gives you complete access to the Unix commands.

SYNTAX: $ man [command]

11. The ‘ps’ command

The ps command is used to the process currently alive in the machine with the 'ps' (process status) command, which displays information about process that are alive when you run the command. 'ps;' produces a snapshot of machine activity.

SYNTAX: $ ps

EXAMPLE: $ ps

$ ps –e

$ps -aux

12. The ‘uname’ command

The uname command is used to display relevant details about the operating system on the standard output.

-m -> Displays the machine id (i.e., name of the system hardware)

-n -> Displays the name of the network node. (host name)

-r -> Displays the release number of the operating system.

-s -> Displays the name of the operating system (i.e.. system name)

-v -> Displays the version of the operating system.

-a -> Displays the details of all the above five options.

SYNTAX: $ uname [option]

EXAMPLE: $ uname -a

## 1.2 DIRECTORY COMMANDS

1. The 'pwd' command:

The pwd (print working directory) command displays the current working directory.

SYNTAX: $ pwd

2. The 'mkdir' command:

The mkdir is used to create an empty directory in a disk.

SYNTAX: $ mkdir dirname

EXAMPLE: $ mkdir receee

3. The 'rmdir' command:

The rmdir is used to remove a directory from the disk. Before removing a directory, the directory must be empty (no files and directories).

SYNTAX: $ rmdir dirname

EXAMPLE: $ rmdir receee

4. The 'cd' command:

The cd command is used to move from one directory to another.

SYNTAX: $ cd dirname

EXAMPLE: $ cd receee

5. The 'ls' command:

The ls command displays the list of files in the current working directory.

SYNTAX: $ ls

EXAMPLE: $ ls

$ ls –l

$ ls –a

## 1.3 FILE HANDLING COMMANDS

1. The ‘cat’ command:

The cat command is used to create a file.

SYNTAX: $ cat > filename

EXAMPLE: $ cat > rec

2. The ‘Display contents of a file’ command:

The cat command is also used to view the contents of a specified file.

SYNTAX: $ cat filename

3. The ‘cp’ command:

The cp command is used to copy the contents of one file to another and copies the file from one place to another.

SYNTAX: $ cp oldfile newfile

EXAMPLE: $ cp cse ece

4. The ‘rm’ command:

The rm command is used to remove or erase an existing file

SYNTAX: $ rm filename

EXAMPLE: $ rm rec

$ rm –f rec

Use option –fr to delete recursively the contents of the directory and its subdirectories.

5. The ‘mv’ command:

The mv command is used to move a file from one place to another. It removes a specified file from its original location and places it in specified location.

SYNTAX: $ mv oldfile newfile

EXAMPLE: $ mv cse eee

6. The ‘file’ command:

The file command is used to determine the type of file.

SYNTAX: $ file filename

EXAMPLE: $ file receee

7. The ‘wc’ command:

The wc command is used to count the number of words, lines and characters in a file.

SYNTAX: $ wc filename

EXAMPLE: $ wc receee

8. The ‘Directing output to a file’ command:

The ls command lists the files on the terminal (screen). Using the redirection operator '>' we can send the output to file instead of showing it on the screen.

SYNTAX: $ ls > filename

EXAMPLE: $ ls > cseeee

9. The 'pipes' command:

The Unix allows us to connect two commands together using these pipes. A pipe ( | ) is an mechanism by which the output of one command can be channeled into the input of another command.

SYNTAX: $ command1 | command2

EXAMPLE: $ who | wc -l

10. The 'tee' command:

While using pipes, we have not seen any output from a command that gets piped into another command. To save the output, which is produced in the middle of a pipe, the tee command is very useful.

SYNTAX: $ command | tee filename

EXAMPLE: $ who | tee sample | wc -l

11. The 'Metacharacters of unix' command:

Metacharacters are special characters that are at higher and abstract level compared to most of other characters in Unix. The shell understands and interprets these metacharacters in a special way.

* - Specifies number of characters

?- Specifies a single character

[ ]- used to match a whole set of file names at a command line.

! – Used to Specify Not

EXAMPLE:

$ ls r** - Displays all the files whose name begins with 'r'

$ ls ?kkk - Displays the files which are having 'kkk', from the second characters irrespective of the first character.

$ ls [a-m] – Lists the files whose names begins alphabets from 'a' to 'm'

$ ls [!a-m] – Lists all files other than files whose names begins alphabets from 'a' to 'm'

12. The 'File permissions' command:

File permission is the way of controlling the accessibility of file for each of three

users namely Users, Groups and Others.

There are three types of file permissions are available, they are

**r-read**
**w-write**
**x-execute**

The permissions for each file can be divided into three parts of three bits each.

| | |
|---|---|
| First three bits | Owner of the file |
| Next three bits | Group to which owner of the file belongs |
| Last three bits | Others |

EXAMPLE: $ ls college

-rwxr-xr-- 1 Lak std 1525 jan10 12:10 college

Where,

-rwx The file is readable, writable and executable by the owner of the file.

Lak Specifies Owner of the file.

r-x Indicates the absence of the write permission by the Group owner of the file.

Std Is the Group Owner of the file.

r-- Indicates read permissions for others.

13. The 'chmod' command:

The chmod command is used to set the read, write and execute permissions for all categories of users for file.

SYNTAX: $ chmod category operation permission file

| Category | Operation | permission |
|---|---|---|
| u-users | + assign | r-read |
| g-group | -Remove | w-write |
| o-others | = assign absolutely | x-execute |
| a-all | | |

EXAMPLE:

$ chmod u –wx college

Removes write & execute permission for users for 'college' file.

$ chmod u +rw, g+rw college

Assigns read & write permission for users and groups for ‘college’ file.

$ chmod g=wx college

Assigns absolute permission for groups of all read, write and execute permissions for ‘college’ file.

14. The ‘Octal Notations’ command:

The file permissions can be changed using octal notations also. The octal notations for file permission are

| | |
|---|---|
| Read permission | 4 |
| Write permission | 2 |
| Execute permission | 1 |

EXAMPLE:

$ chmod 761 college

Assigns all permission to the owner, read and write permissions to the group and only executable permission to the others for ‘college’ file.

## 1.4 GROUPING COMMANDS

1. The ‘semicolon’ command:

The semicolon(;) command is used to separate multiple commands at the command line.

SYNTAX: $ command1;command2;command3…..... ;commandn

EXAMPLE: $ who;date

2. The ‘&&’ operator:

The ‘&&’ operator signifies the logical AND operation in between two or more valid Unix commands.It means that only if the first command is successfully executed, then the next command will executed.

SYNTAX: $ command1 && command && command3 .....&&commandn

EXAMPLE: $ who && date

3. The ‘||’ operator:

The ‘||’ operator signifies the logical OR operation in between two or more valid Unix commands.It means, that only if the first command will happen to be un successfully,it will continue to execute next commands.

SYNTAX: $ command1 || command || command3… ..... ||commandn

EXAMPLE: $ who || date

1.5 FILTERS

1. The head filter

It displays the first ten lines of a file.

SYNTAX: $ head filename

EXAMPLE: $ head college Display the top ten lines.

$ head -5 college Display the top five lines.

2. The tail filter

It displays ten lines of a file from the end of the file.

SYNTAX: $ tail filename

EXAMPLE: $ tail college Display the last ten lines.

$tail -5 college Display the last five lines.

3. The more filter:

The pg command shows the file page by page.

SYNTAX: $ ls –l | more

4. The 'grep' command:

This command is used to search for a particular pattern from a file or from the standard input and display those lines on the standard output. "Grep" stands for "global search for regular expression."

SYNTAX: $ grep [pattern] [file_name]

EXAMPLE: $ cat > student

Arun cse

Ram ece

Kani cse

$ grep "cse" student

Arun cse

Kani cse

5. The 'sort' command:

The sort command is used to sort the contents of a file. The sort command reports only to the screen, the actual file remains unchanged.

SYNTAX: $ sort filename

EXAMPLE: $ sort college

OPTIONS:

| Command | Purpose |
|---|---|
| Sort –r college | Sorts and displays the file contents in reverse order |
| Sort –c college | Check if the file is sorted |
| Sort –n college | Sorts numerically |
| Sort –m college | Sorts numerically in reverse order |
| Sort –u college | Remove duplicate records |
| Sort –l college | Skip the column with +1 (one) option.Sorts according to second column |

6. The ‘nl’ command:

The nl filter adds lines numbers to a file and it displays the file and not provides access to edit but simply displays the contents on the screen.

SYNTAX: $ nl filename

EXAMPLE: $ nl college

7. The ‘cut’ command:

We can select specified fields from a line of text using cut command.

SYNTAX: $ cut -c filename

EXAMPLE: $ cut -c college

OPTION:

-c – Option cut on the specified character position from each line.

**RESULT:**

Thus the basic LINUX commands have been studied successfully.

## Ex No: 2 CONFIGURATION OF NETWORK IN LINUX ENVIRONMENT

**Date:**

**AIM:**

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

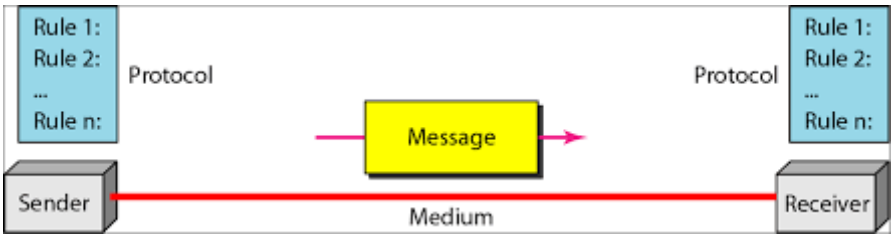
**DESCRIPTION:**

**NETWORKING OVERVIEW:**

**COMPUTER NETWORK**

Computer network is defined as the interconnection of nodes (computers and other devices) connected by a communication channel (wired or wireless) that facilitates communication among users and allows them to share resources.

**Components *of data communication***




- **Sender:** It is the transmitter of data. Some examples are Terminal, Computer, and Mainframe.
- **Medium:** The communication stream through which the data is being transmitted. Some examples are: Cabling, Microwave, Fiber optics, Radio Frequencies (RF), Infrared Wireless
- **Receiver:** The receiver of the data transmitted. Some examples are Printer, Terminal, Mainframe, and Computer.
- **Message:** It is the data that is being transmitted from the Source/Sender to the Destination/Receiver.
- **Protocol:** It is the set of rules and regulations (resides in the form of software and hardware) that are to be followed for communication. If protocol is not present it implies the nodes are connected but they can't communicate.

**CATEGORIES OF NETWORK:**

The three primary categories of network are Local Area Network (LAN), Metropolitan Area Network (MAN), and Wide Area Network (WAN). The category into which a network fall is determined by its size, ownership, the distance it covers and its physical architecture.

**LAN**

- A LAN is usually privately owned and links the devices in a single office, building or campus.
- A LAN can be as simple as two PCs or it can extend throughout a company. LAN size is limited to a few kilometers.
- The most widely used LAN technology is the Ethernet technology developed by the Xerox Corporation.

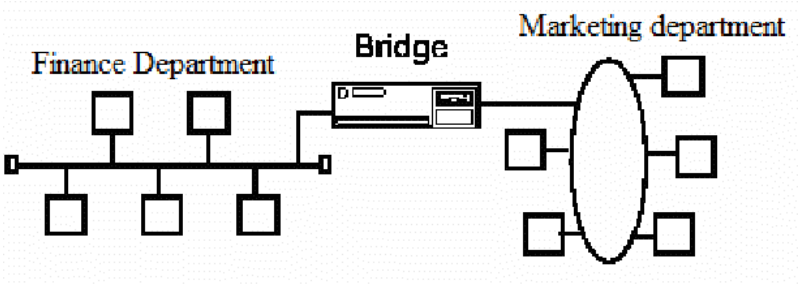



**Figure 1.8: Local Area Network**

**MAN**

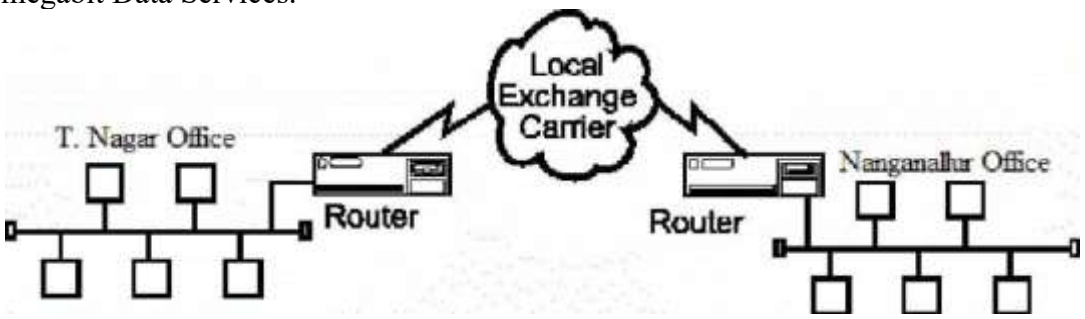
- A MAN is designed to extend over an entire city.
- It could be a single network such as cable TV network or connect a number of LANs into a larger network.
- A MAN can be owned by a private company or it may be a service provided by a public company, such as local telephone company.
- Telephone companies provide a popular MAN service called (SMDS) Switched Multi-megabit Data Services.
- 


Figure: Metropolitan Area Network

**WAN**

- A WAN provides long distance transmission of data, voice, image and video information over large geographic areas.
- WAN utilize public, leased, or private communication equipment usually in combinations and therefore span an unlimited number of miles.
- A WAN that is wholly owned and used by a single company is referred to as an Enterprise Network.

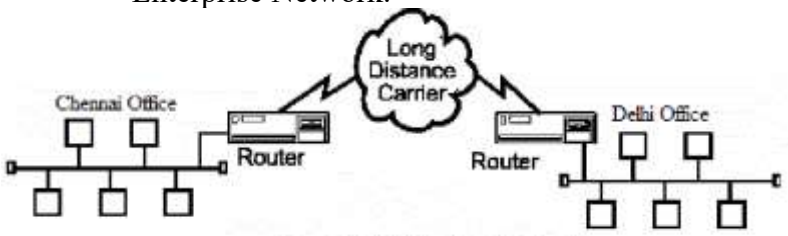



**Figure 1.10: Wide Area Network**

**ADDRESSING IN A NETWORK:**

- Every network device has two types of addresses,

1. **Logical address** -- in most cases this is the IP address- IP address is a number assigned to a connection of a device in a network. It is 32 bit length for IPv4 and 128 bit length for IPv6.

2. **Physical address** -- also known as the MAC address-A MAC address is a

number assigned to the NIC card by the manufacturer. It length is 48 bits represented in hexadecimal (6 bytes).

- A **network address** is also known as the numerical network part of an IP address. This is used to identify a network that has its own hosts and addresses. For example, in the IP address 192.168.1.1, the network address part is 192.168.1 and the network address of this IP connection is 192.168.1.0.
- A subnetwork or subnet is a logical subdivision of an IP network. The practice of dividing a network into two or more networks is called subnetting.
- **Subnet masks** are expressed in dot-decimal notation like an address. For example, 255.255.255.0 is the subnet mask for the prefix 198.51.100.0/24.

**GATEWAY:**

- A gateway is a hardware device that acts as a "gate" between two networks. It may be a router, firewall, server, or other device that enables traffic to flow in and out of the network. While a gateway protects the nodes within network, it also a node itself.

**DNS:**

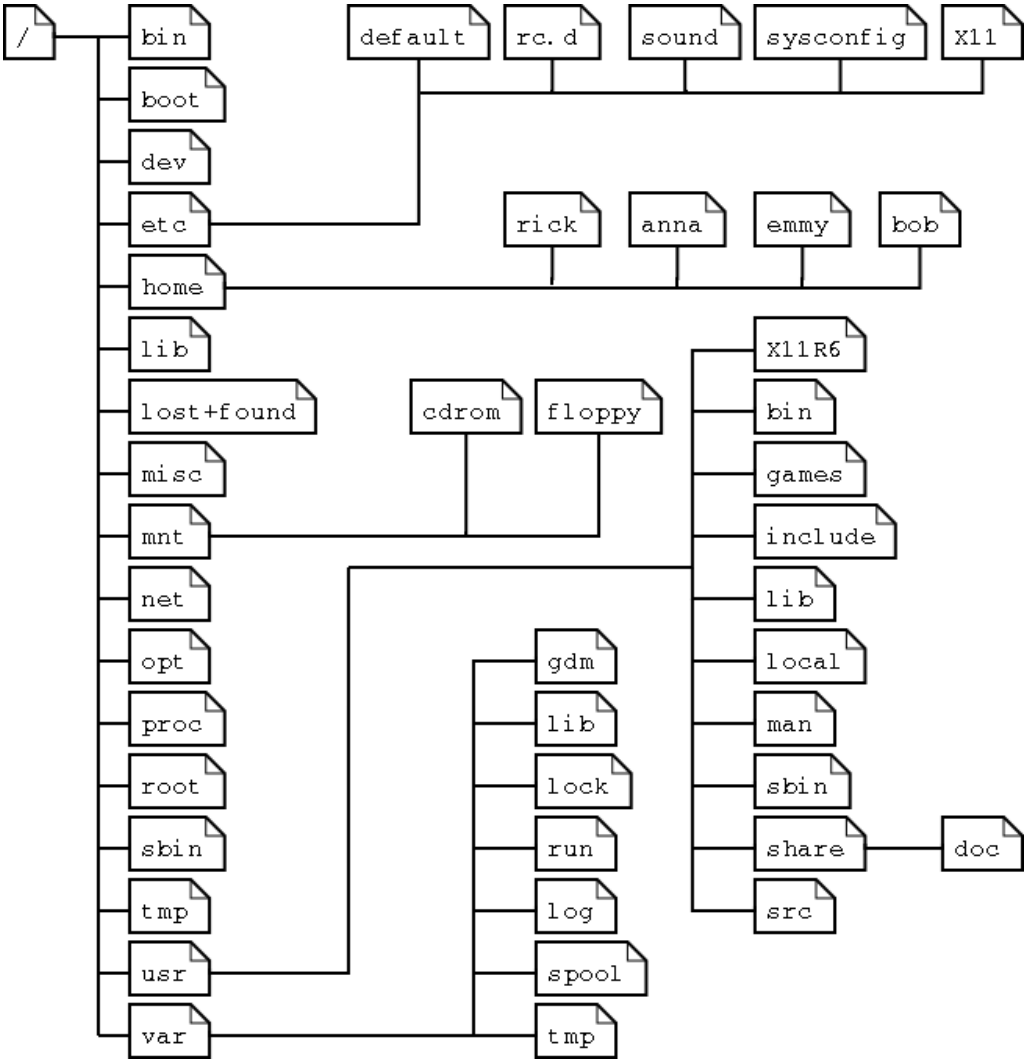
- DNS. (Domain Name System) The Internet's system for converting alphabetic names into numeric IP addresses. For example, when a Web address (URL) is typed into a browser, DNS servers return the IP address of the Web server associated with that name.

## INTRODUCTION TO NETWORKING IN LINUX:

Connecting your Linux computer to a network is pretty straightforward. Linux easily manages multiple network interface adapters.

### *LINUX FILE SYSTEM:*

In a LINUX file system, everything is a file. In order to manage all those files in an orderly fashion, man likes to think of them in an ordered
tree-like structure on the hard disk.
A Linux system, just like UNIX, makes no difference between a file and a directory, since a directory is just a file containing names of other files. Programs, services, texts, images, and so forth, are all files. Input and output devices, and generally all devices, are considered to be files, according to the system.




The tree of the file system starts at the trunk or slash, indicated by a forward slash (/). This directory, containing all underlying directories and files, is also called the root directory or "the root" of the file system.

Directories that are only one level below the root directory are often preceded by a slash, to indicate their position and prevent confusion with other directories that could have the same name. When starting with a new system, it is always a good idea to take a look in the root directory. Let's see what you could run into:

In the above set of directories, /etc directory contains most important system configuration files, this directory contains data similar to those in the Control Panel in Windows

## CONFIGURING A NETWORK INTERFACE:

### /etc directory:

1. This is the nerve center of the Linux file system; it contains all **system related configuration files** in it or in its sub-directories.
2. A "configuration file" is defined as a local file used to control the operation of a program; it must be static and cannot be an executable binary.

**/etc/sysconfig/ sub-directory:**

1. This directory contains configuration files for mouse, keyboard, network, desktop, system clock, power management etc. Its subdirectories contain setup of system configuration specifics, like 'clock', which sets the time zone, or 'keyboard' which controls the keyboard map.
2. The contents may vary drastically depending on which distribution and what utilities you have installed.
3. The sub directory for network configuration is **etc/sysconfig/network-scripts.**

**Ethernet Interfaces files:**

1. Every network interface has its own configuration file in the */etc/sysconfig/network-scripts* directory.
2. This file starts the interface on boot, assigns it a static IP address, defines a domain and network gateway, specifies two DNS servers.
3. It does not allow non-root users to start and stop the interface.
4. The network interface file **/etc/sysconfig/network-scripts/ifcfg-eth0 (eth0-name of the interface card)** controls the first Ethernet ***network interface card*** or NIC in the system.
5. The system reads this network interface files during the boot process to determine which interfaces to bring up and how to configure them.
6. In a system with multiple NICs, there are multiple ifcfg-eth*X* files (where *X* is a unique number corresponding to a specific interface). Because each device has its own configuration file, an administrator can control how each interface functions individually.
7. Sample ifcfg-eth0 file for an interface using a **fixed IP address consists of the following configuration parameters set.**:

- TYPE=Ethernet
- BOOTPROTO=none
- DEFROUTE=yes
- IPV4_FAILURE_FATAL=no
- IPV6INIT=yes
- IPV6_AUTOCONF=yes
- IPV6_DEFROUTE=yes
- IPV4_FAILURE_FATAL=no
- NAME=eth0
- UUID=...
- ONBOOT=yes
- **HWADDR=0e:a5:1a:b6:fc:86**
- **IPADDR0=172.31.24.10**
- **PREFIX0=23**
- **GATEWAY0=172.31.24.1**
- **DNS1=192.168.154.3**
- **DNS2=10.216.106.3**
- DOMAIN=example.com
- IPV6_PEERDNS=yes
- IPV6_PEERROUTES=yes

**Configuration Parameters Description:**

A description of some of these configuration parameters follows:

| **Configuration Parameters** | **Description** |
|---|---|

| | |
|---|---|
| TYPE | The type of network interface device |
| BOOTPROTO | **Protocol is one of the following:**<br>none: No boot-time protocol is used (static)<br>bootp: Use BOOTP (bootstrap protocol)<br>dhcp: Use DHCP (Dynamic Host Configuration Protocol). |
| DEFROUTE\|IPV6_DEFROUTE | **Answer is one of the following:**<br>yes: This interface is set as the default route for IPv4\|IPv6 traffic.<br>no: This interface is not set as the default route. |
| IPV6INIT | **Answer is one of the following:**<br>Enable IPv6 on this interface. If IPV6INIT=yes, the following parameters could also be set in this file:<br>IPV6ADDR=IPv6 address<br>IPV6_DEFAULTGW=The default route through the specified gateway<br>no: Disable IPv6 on this interface. |
| IPV4_FAILURE_FATAL\|<br>IPV6_FAILURE_FATAL | **Answer is one of the following:**<br>yes: This interface is disabled if IPv4 or IPv6 configuration fails.<br>no: This interface is not disabled if configuration fails. |
| ONBOOT | **Answer is one of the following:**<br>yes: This interface is activated at boot time.<br>no: This interface is not activated at boot time |
| HWADDR | The hardware address of the Ethernet device |
| IPADDRN | The IPv4 address assigned to the interface |
| PREFIXN | Length of the IPv4 netmask value |
| GATEWAYN | The IPv4 gateway address assigned to the interface. Because an interface can be associated with several combinations of IP address, network mask prefix length, and gateway address, these are numbered |

| | |
|---|---|
| | starting from 0. |
| DNSN | The address of the Domain Name Servers (DNS) |
| DOMAIN | The DNS search domain |
| USERCTL | **Answer is one of the following:**<br>yes: Non-root users are allowed to control this device.<br>no : Non-root users are not allowed to control this device. |
| NAME | Interface Device name |

## BASIC NETWORKING COMMANDS IN LINUX:

## 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 intrface 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 shows 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
11 74.125.242.145 (74.125.242.145) 121.041 ms 74.125.242.129 (74.125.242.129) 121.181 ms 127.865 ms
12 216.239.42.245 (216.239.42.245) 127.057 ms 216.239.43.77 (216.239.43.77) 125.901 ms *
13 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]#
```

**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
```

**EXERCISES:**

1. What are the two identification addresses given to an device participating in a network.
2. Write down the identification addresses of your device.
3. What is the bit length of IP address and MAC Address.
4. What is purpose of an interconnecting device? Can an interconnecting device have more than one interface card? If so justify your answer.
5. If a device has three interface card activated in it? How many network-interface configuration files are required?
6. How will you define the endpoints of communication?
7. What is the meaning of 'lo' device name when ifconfig command is executed?
8. Who assigns static IP and Dynamic IP to a machine.
9. Open the network interface configuration file of your computer and write the values of the following configuration parameters.
    a) IPADDR
    b) HWADDR
    c) PREFIX/SUBNET MASK
    d) GATEWAY
    e) DNS
    f) NAME
    g) TYPE
10. Display the network configuration details of your network interface.
11. Assign the following IP address to your device 172.16.8.127 and display the change in IP address.
12. Disconnect the network interface and check the dis-connectivity through a command and display the results. Finally activate the interface.
13. Check whether the following hosts are reachable from your device and explain the response.
    a) 172.16.8.2
    b) DNS Server
    c) Gateway
    d) 127.0.0.1
14. Disconnect the network interface and Check whether the following hosts are reachable from your device and explain the response.
    a) 127.0.0.1
    b) Gateway
15. Find the IPv4 and IPv6 addresses of the following urls
    a) www.google.com
    b) www.rajalakshmi.org
    c) www.facebook.com
16. Display the ARP table of your machine.
17. Display the kernel routing table of your machine.
18. List out the hops(IP addresses of the intermediate nodes) taken by a packet to reach its destination
19. Explore the GUI mode of configuring a network in Linux.
20. Assign a new IP address to your machine through GUI Mode.

**Ex No: 3 SETING UP A LOCAL AREA NETWORK USING A SWITCH**
**Date:**

**AIM:**

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

**DESCRIPTION:**

**SETTING UP A LAN:**

Creating and configuring a LAN consists of these steps:

1. **Setting up LAN hardware** — This entails choosing a network topology, purchasing the equipment you need, and installing it (adding cards and connecting wires or using wireless antennas).
2. **Configuring TCP/IP** — To use most of the networking applications and tools that come with Linux, you must have TCP/IP configured. TCP/IP lets you communicate not only with computers on your LAN, but to any computers that you can reach on your LAN, modem, or other network connection (particularly to the Internet).

**Setting up LAN hardware**

To set up a simple LAN, the following decisions need to be done

- Decisions about network topology (that is, how computers are connected) should be done
- Decisions about network equipment (network interface cards, wires, hubs, and so on).

**LAN topologies:**

Most small office and home LANs connect computers together in one of the following topologies:

Star topology — the star topology is by far the most popular LAN topology. In this arrangement, each computer contains a Network Interface Card (NIC) that connects with a cable to a central hub.

Bus topology — Instead of using hubs, the bus topology connects computers in a chain from one to the next. The cabling usually used is referred to as coaxial, or Thin Ethernet cable. A "T" connector attaches to each computer's NIC, then to two adjacent computers in the chain. At the two ends of the chain, the T connectors are terminated.

Ring topology — this is a less popular topology than star and bus topologies. In a ring topology, computers connect to a ring of wires on which tokens are taken and passed by computers that want to send information on the network.

You can configure a wireless Ethernet LAN in several different topologies, depending on how you want to use the LAN. With a wireless LAN, each computer broadcasts in the air rather than across wires. Here are some examples of wireless topologies:

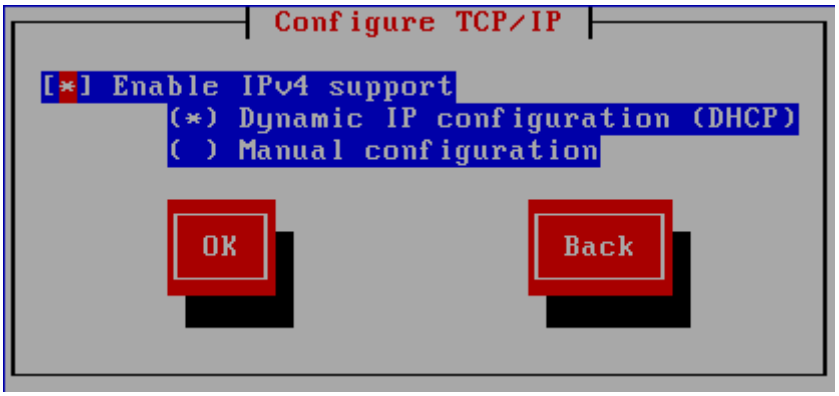
Wireless peer-to-peer — in this topology, frames of data are broadcast to all nodes within range, but are consumed only by the computers for which they are intended. This arrangement is useful if you are sharing file and print services among a group of client computers.

Wireless access point — a wireless interface can act as an access point for one or more wireless clients. Clients can be configured to communicate directly with the access point, instead of with every client that is within range. This arrangement is useful for point-to-point connections between two buildings, where the access point is acting as a gateway to the Internet or, for example, a campus intranet.

**LAN equipment:**

The equipment that you need to connect your LAN can include some or all of the following:
**Network Interface Card (NIC)** — typically, one of these cards goes into a slot in each computer. For wired Ethernet networks, the cards can transmit data at 10 Mbps or 100 Mbps. Gigabit (1000 Mbps) NICs are also now available, but are quite a bit more expensive. An 802.11B wireless NIC card can operate at speeds of up to 11 Mbps, but are more expensive than a wired NIC card.
**Cables** — for star topologies, cables are referred to as twisted-pair. Category 5e wiring, which contains four twisted-pair sets per wire, is the most common type of wiring used for LANs today. A connector at each end of the cable is an RJ-45 plug, similar to those used on telephone cables. Ethernet interfaces are either 10Base-T (10 Mbps speeds) or 100Base-TX (100 Mbps speeds). These cables plug into the computer's NIC at one end and the hub at the other.
**Hubs** — with the star topology, a hub is typically used to connect the computers. Sometimes hubs are also referred to as repeaters or concentrators because they receive signals from the nodes connected to them and send the signals on to other nodes.
**Switches** — a switch can be used instead of a hub. It lets you divide a LAN that is getting too large into segments that are more manageable. A switch can reduce network traffic by directing messages intended for a specific computer directly to that computer. This is as opposed to a hub, which broadcasts all data to all nodes. Because switches have come down so much in price, in most cases you should just pay a few extra dollars and get a switch instead of a hub.

**Configuring TCP/IP for your LAN(during Installation of OS):**
During OS Installation, in the part of network installation, the **Configure TCP/IP** dialog appears.




This dialog asks for your IP and other network addresses.

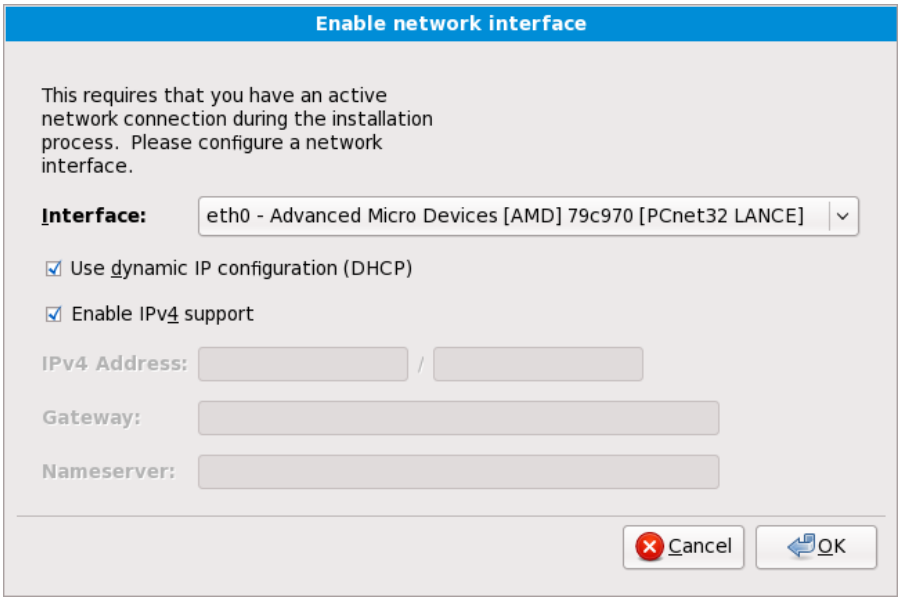
1. You can choose to configure the IP address and Netmask of the device via DHCP or manually.
2. By default, the installation program uses DHCP to automatically provide network settings.
3. If you use a cable or DSL modem, router, firewall, or other network hardware to communicate with the Internet,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

**Network Configuration**

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

1. During Network Configuration, a Setup prompts you to supply a host name and domain name for the computer, in the format **hostname.domainname**.
2. Many networks have a DHCP (Dynamic Host Configuration Protocol) service that automatically supplies connected systems with a domain name, leaving the user to enter a hostname.
3. Unless you have a specific need to customize the host name and domain name, the default setting **localhost.localdomain** is a good choice for most users.
4. To set up a network that is behind an Internet firewall or router, you may want to use hostname.localdomain for your Fedora system.
5. If you have more than one computer on this network, you should give each one a separate host name in this domain.
6. In some networks, the DHCP provider also provides the name of the computer, or hostname. The complete hostname includes both the name of the machine and the name of the domain of which it is a member, such as machine1.example.com. The machine name (or "short hostname") is machine1, and the domain name is example.com.


**Configuring the network interface:**




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

**LAN setup:**

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

**Steps for setting up an Ethernet LAN are:**

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

**EXERCISE:**

1. Setup a LAN with three devices connected using a switch. Test the connectivity of the LAN.

**Ex No:4 STUDY OF PACKET TRACER-Installation and User Interface Overview**

**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).

Windows

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

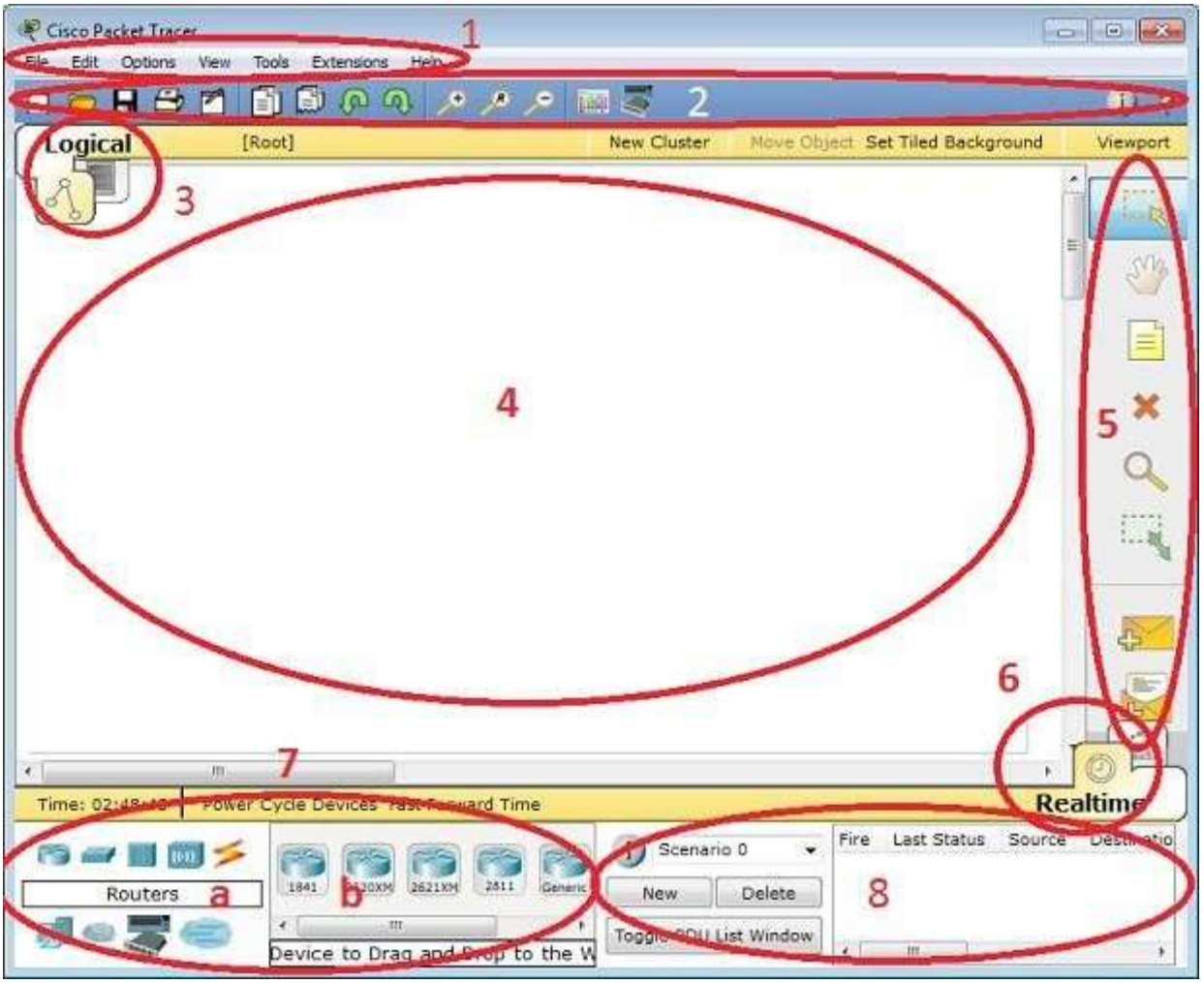
Linux

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

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

USER INTERFACE OVERVIEW:

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




1. Menu bar – This is a common menu found in all software applications; it is used to open, save, print, change preferences, and so on.
2. Main toolbar – This bar provides shortcut icons to menu options that are commonly accessed, such as open, save, zoom, undo, and redo, and on the right-hand side is an icon for entering network information for the current network.
3. Logical/Physical workspace tabs – These tabs allow you to toggle between the Logical and Physical work areas.
4. Workspace – This is the area where topologies are created and simulations are displayed.
5. Common tools bar – This toolbar provides controls for manipulating topologies, such as select, move layout, place note, delete, inspect, resize shape, and add simple/complex PDU.
6. Real-time/Simulation tabs – These tabs are used to toggle between the real and simulation modes. Buttons are also provided to control the time, and to capture the packets.
7. Network component box – This component contains all of the network and end devices available with Packet Tracer, and is further divided into two areas: Area 7a: Device-type selection box – This area contains device categories Area 7b: Device- specific selection box – When a device category is selected, this selection box displays the different device models within that category
8. User-created packet box – Users can create highly-customized packets to test their topology from this area, and the results are displayed as a list.

## Ex No: 5 ASSIGNMENT OF IP ADDRESS AND IMPLEMENTATION OF SUBNET MASK IN IP ADDRESSING TO COMPUTERS

**Date:**

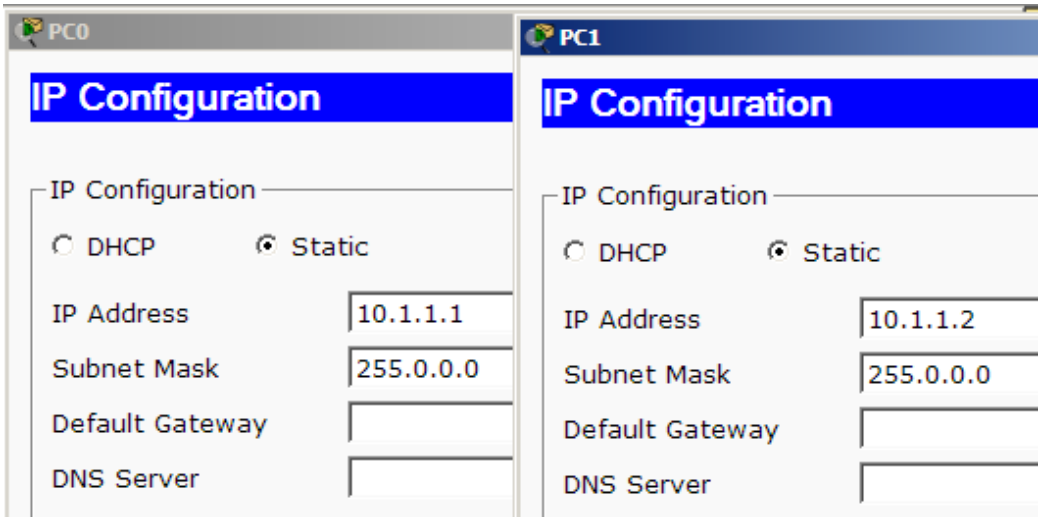
**AIM:**

To using assignment of IP address and implementation of subnet mask in IP Addressing to computers 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.



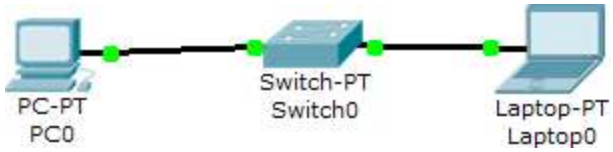
5. Close the IP Configuration box, open the command prompt, and ping the IP address of the device at the end to check connectivity. (in desktop tab, command prompt is also there)

```
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:

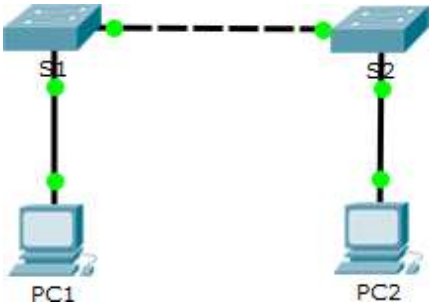
1. Click on Switches from the device-type selection box and insert any switch (except Switch-PT-Empty) into the workspace.
2. From the network component box, click on End Devices and drag-and-drop a Generic PC icon and a Generic laptop icon into the Workspace.
3. Choose the Copper Straight-Through cable 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.

**Exercise 1:**

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



**Ex No: 6 IMPLEMENTATION OF SETUP OF A LAN USING SWITCHES**

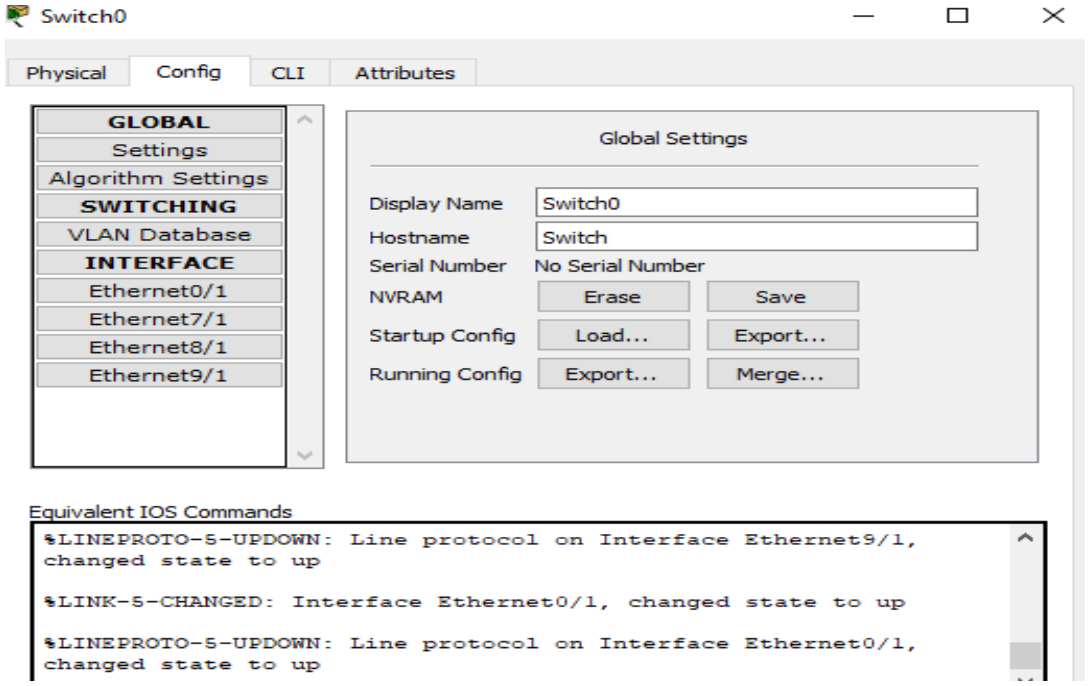
**Date:**

**AIM:**

To implement a setup of a LAN using switches in Packet tracer.

**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

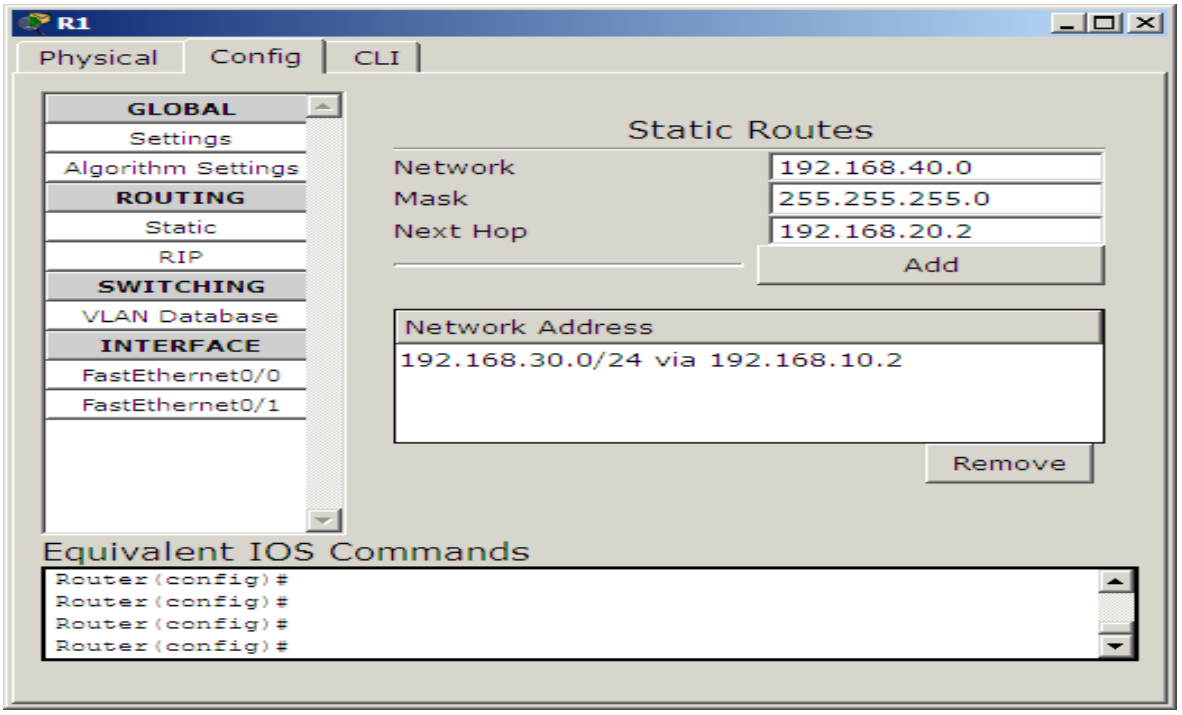
This section slightly differs from the switch and the router. Switches have options for modifying the speed and duplex setting and for assigning a port to VLAN. On routers, the VLAN section is replaced by the IP address configuration.
Configuring a Router is done by performing the following steps:

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

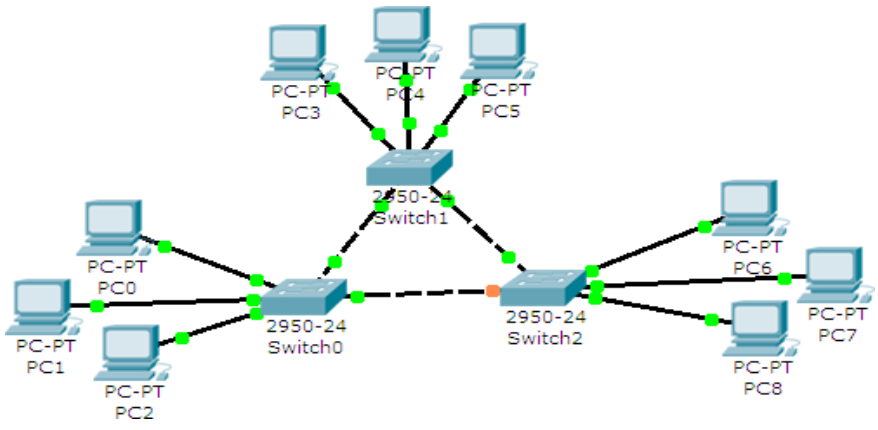
This section has options for configuring Static and dynamic routing (RIP). To configure static routing, enter the network address, netmask, and its next hop address, and then click on Add.



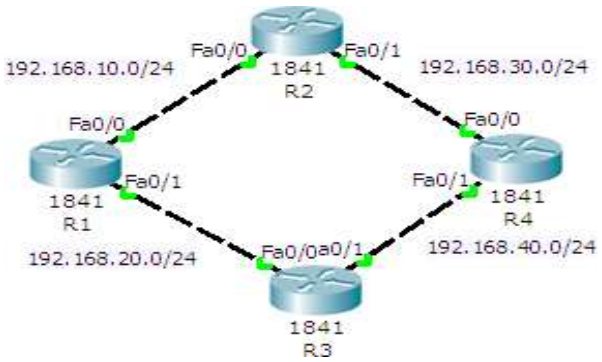

To configure Routing Information Protocol (RIP), it is enough to add only the network IP.

**Exercises:**

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




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


192.168.10.0/24
Fa0/0
1841
R2
Fa0/1
192.168.30.0/24
Fa0/0
Fa0/0
1841
R1
Fa0/1
Fa0/1
1841
R4
192.168.20.0/24
Fa0/0a0/1
192.168.40.0/24
1841
R3

**EX NO 7: STUDY OF WIRESHARK TOOL**
**Date:**

**AIM:**

To study packet sniffing concept using Wireshark Tool.

**DESCRIPTION:**

**WIRESHARK**

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:**

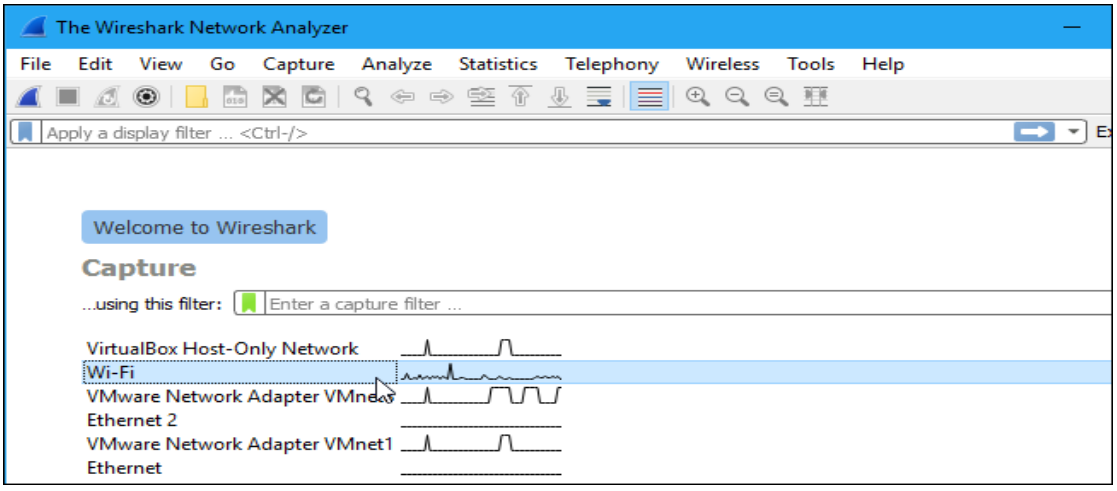
- Network administrators: troubleshoot network problems
- Network security engineers: examine security problems
- Developers: debug protocol implementations
- People: learn **network protocol internals**

**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.

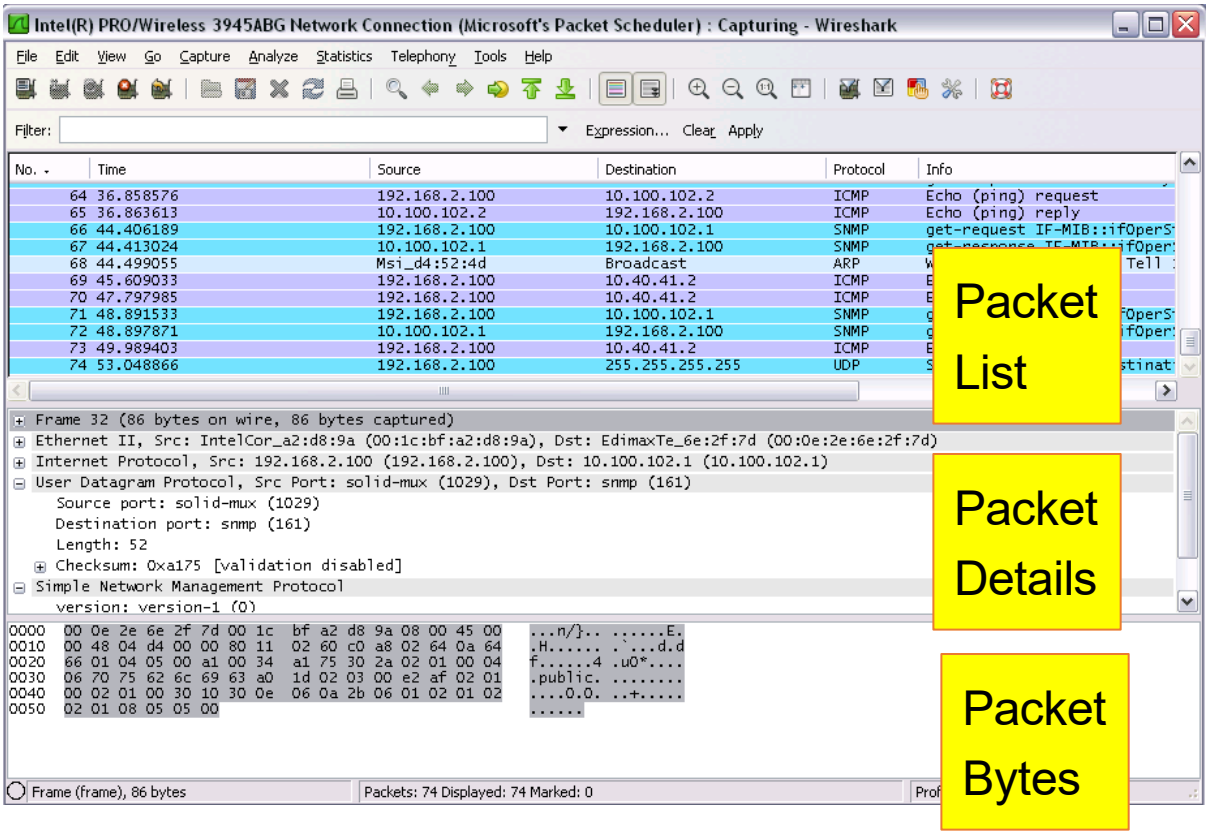
**Capturing Packets**

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.




Click the red "Stop" button near the top left corner of the window when you want to stop capturing traffic.

**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.

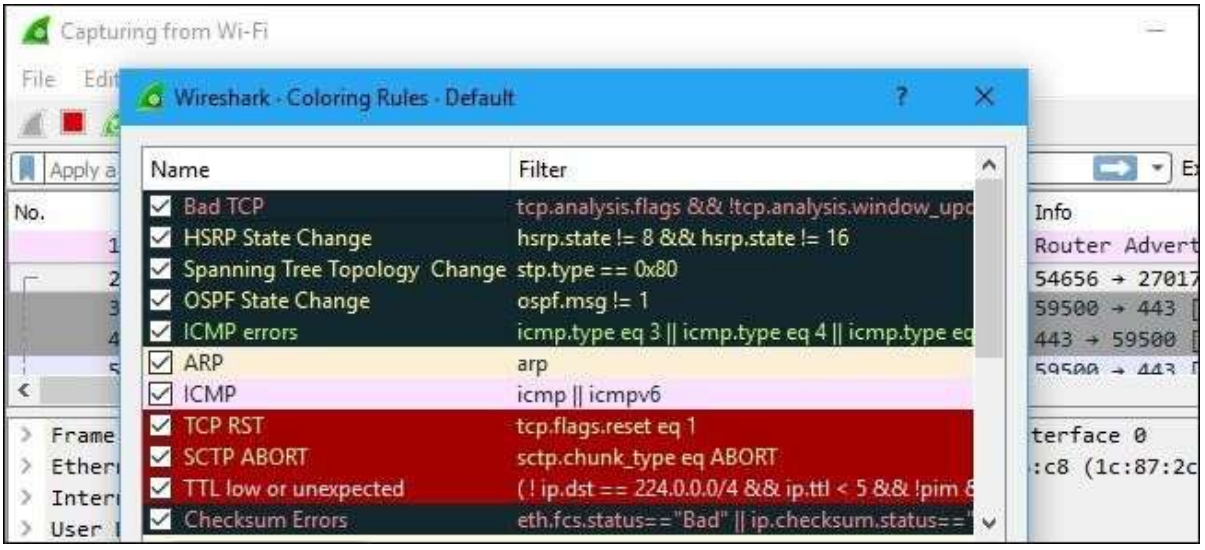
**The "Packet Bytes" Pane**

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.

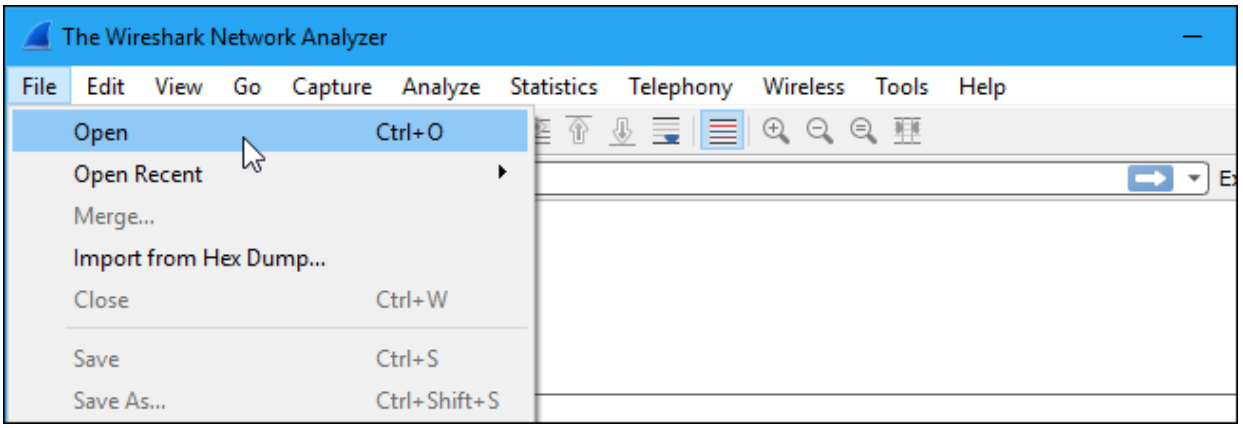
To view exactly what the color codes mean, click View > Coloring Rules. You can also customize and modify the coloring rules from here, if you like.




**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.

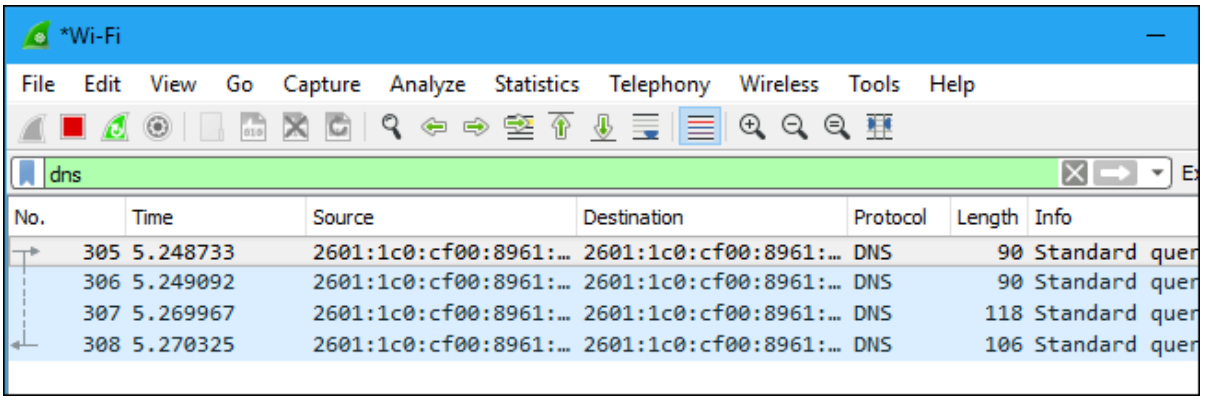
You can also save your own captures in Wireshark and open them later. Click File > Save to save your captured packets.




**Filtering Packets**

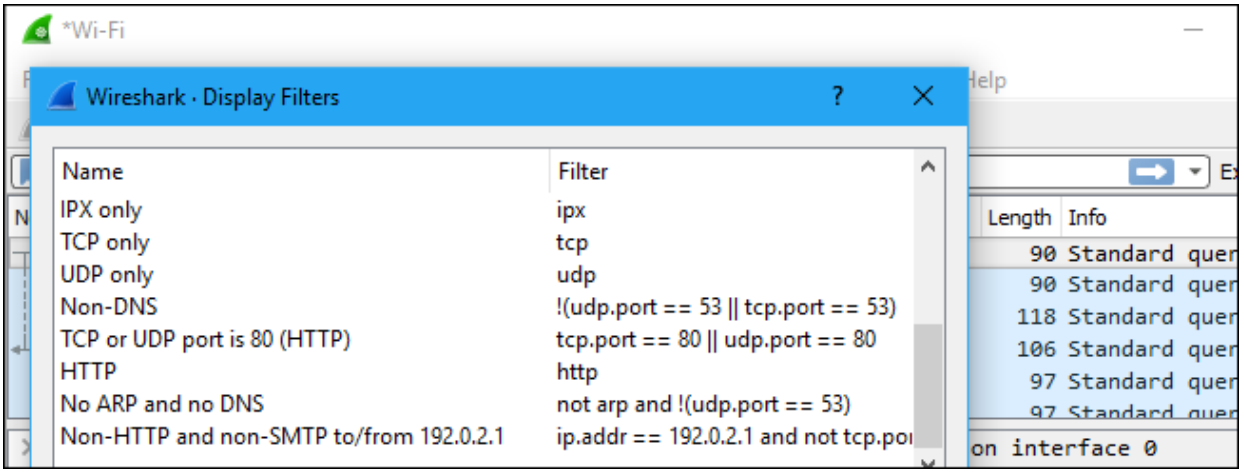
If you're trying to inspect something specific, such as the traffic a program sends when phoning home, it helps to close down all other applications using the network so you can narrow down the traffic. Still, you'll likely have a large amount of packets to sift through. That's where Wireshark's filters 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.



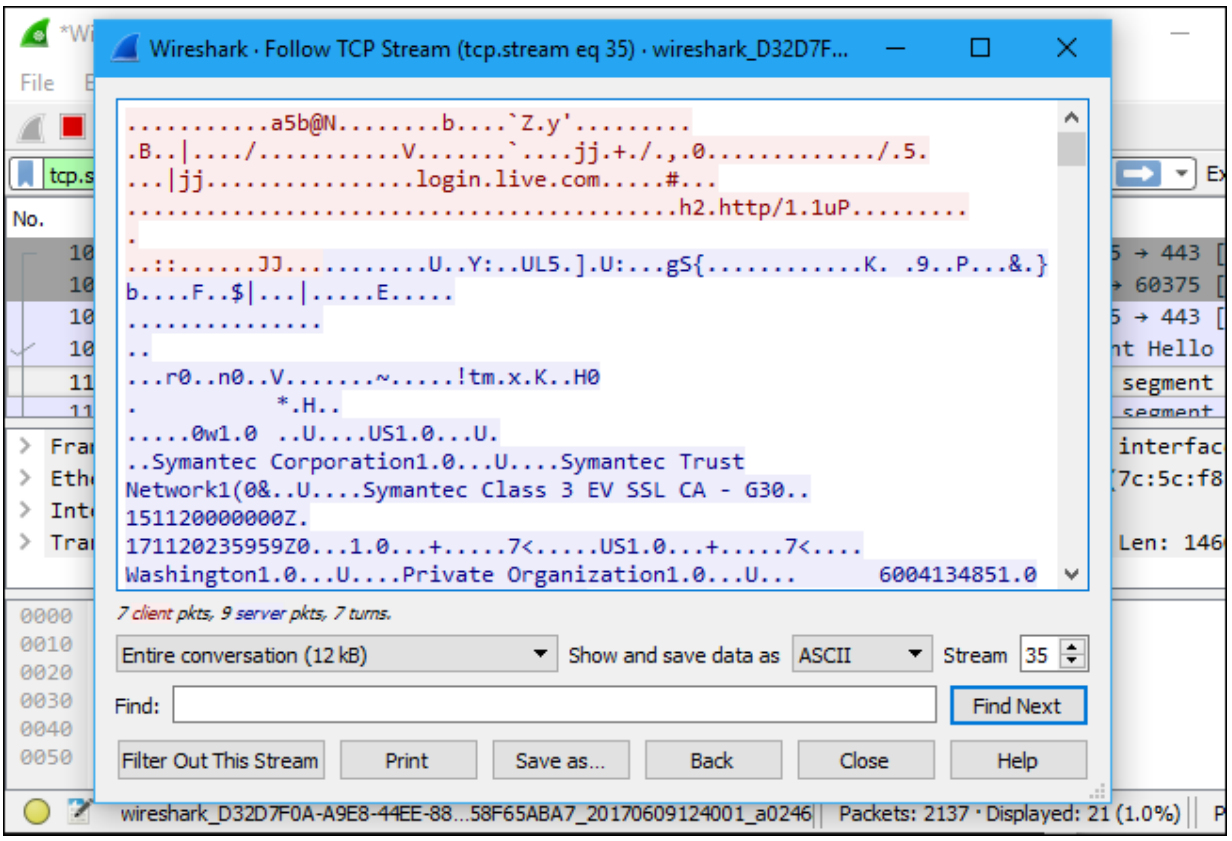

You can also click Analyze > Display Filters to choose a filter from among the default filters included in Wireshark. From here, you can add your own custom filters and save them to easily access them in the future.

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

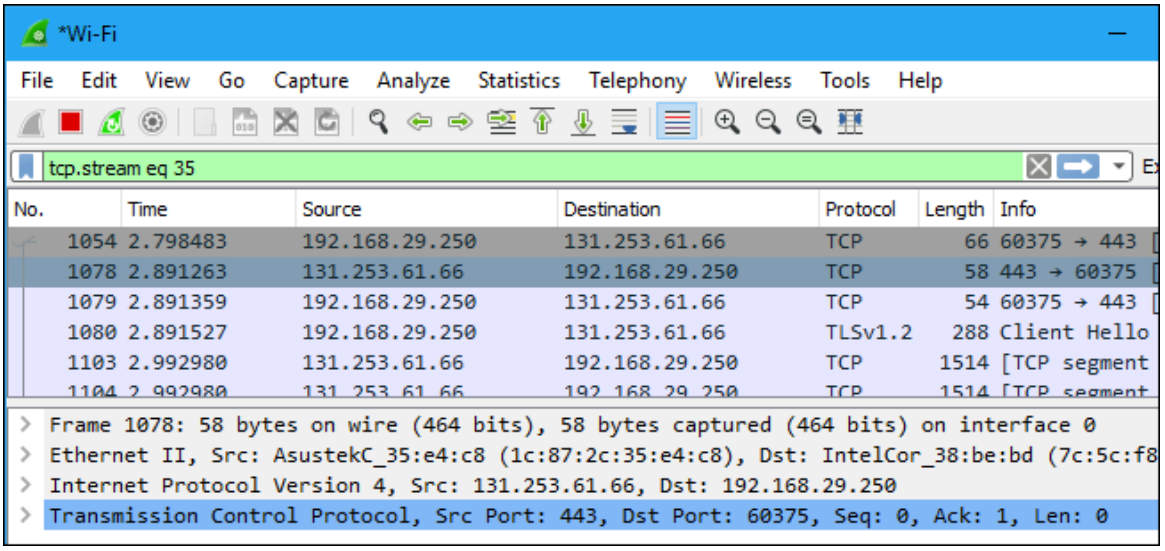



Another interesting thing you can do is right-click a packet and select Follow > TCP Stream.

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.

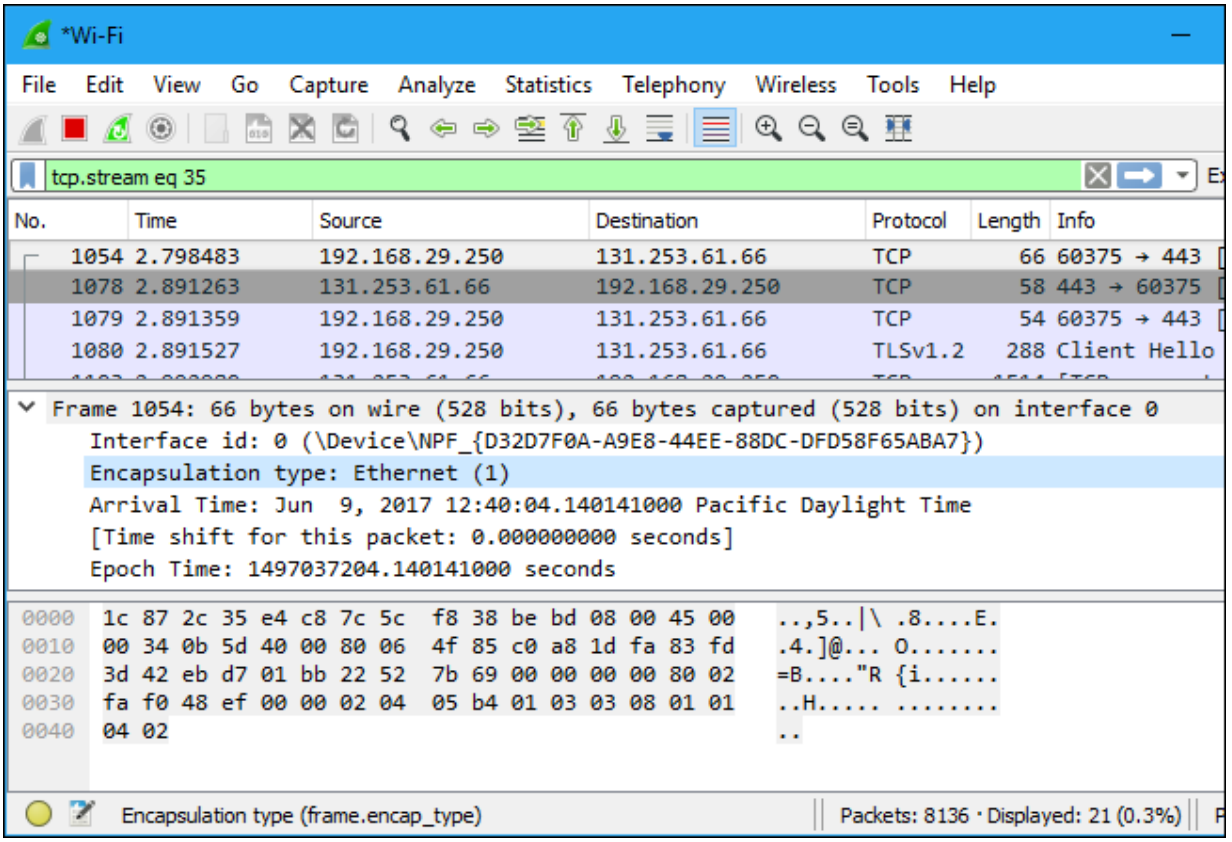



Close the window and you'll find a filter has been applied automatically. Wireshark is showing you the packets that make up the conversation.

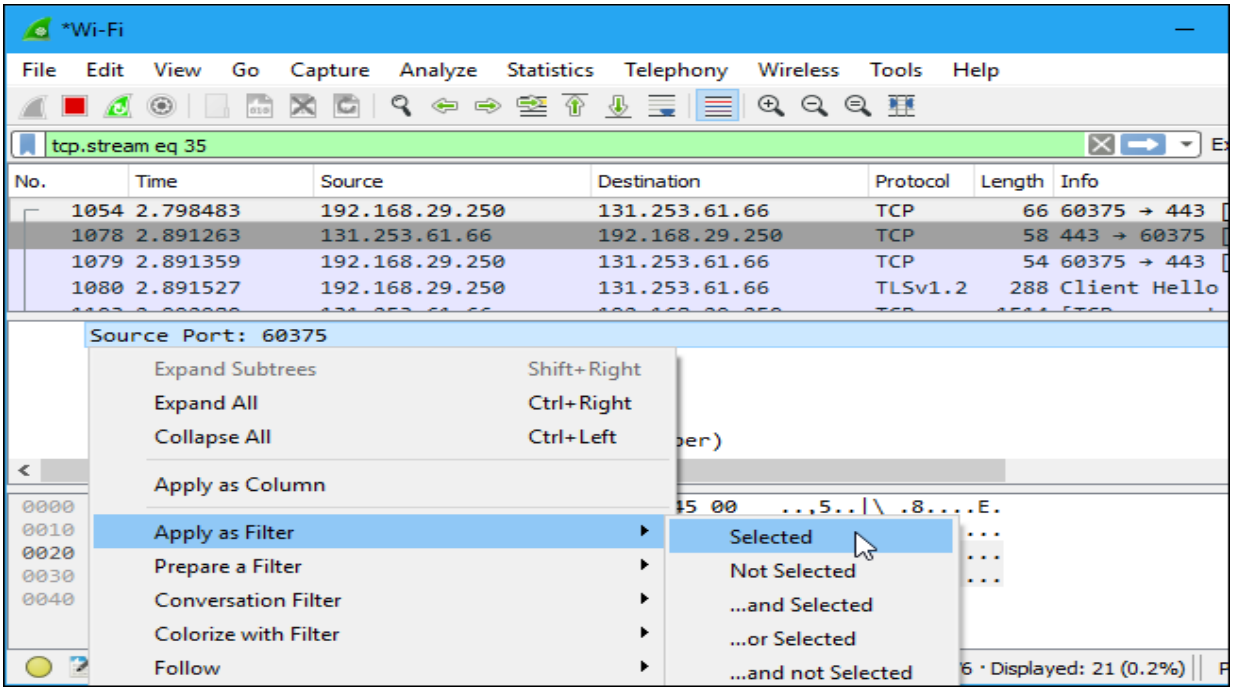



**Inspecting Packets**

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



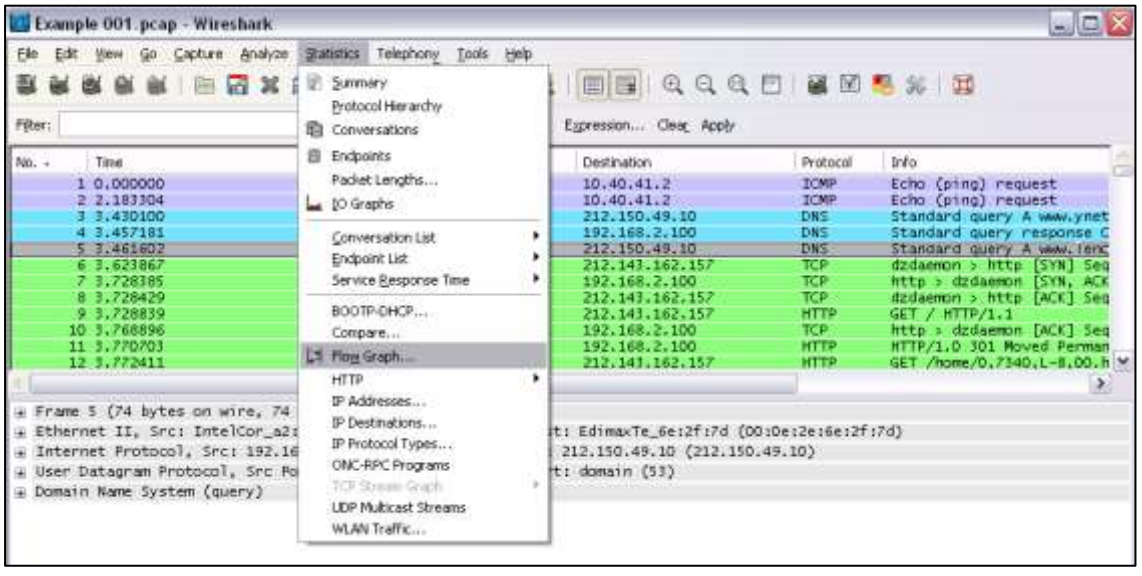

You can also create filters from here — just right-click one of the details and use the Apply as Filter submenu to create a filter based on it.

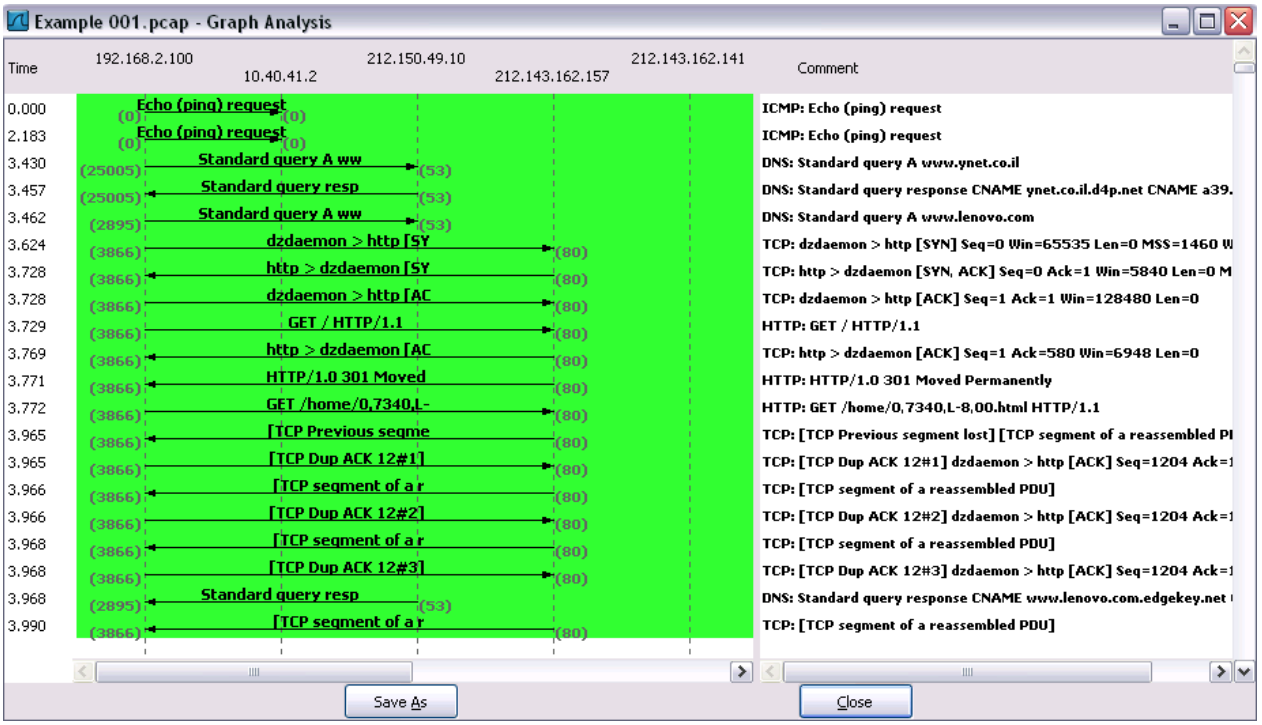



---

Wireshark is an extremely powerful tool, and this tutorial is just scratching the surface of what you can do with it. Professionals use it to debug network protocol implementations, examine security problems and inspect network protocol internals.

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






**EXERCISES:**

1. Capture 100 packets from the Ethernet: IEEE 802.3 LAN Interface and save it.
2. Create a Filter to display only TCP/UDP packets, inspect the packets and provide the flow graph.
3. Create a Filter to display only ARP packets and inspect the packets.
4. Create a Filter to display only DNS packets and provide the flow graph.
5. Create a Filter to display only HTTP packets and inspect the packets.
6. Create a Filter to display only IP/ICMP packets and inspect the packets.
7. Create a Filter to display only DHCP packets and inspect the packets.

## EX NO 8: CAPTURING AND ANALYSING PACKETS USING WIRESHARK TOOL
**Date:**

### Aim

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

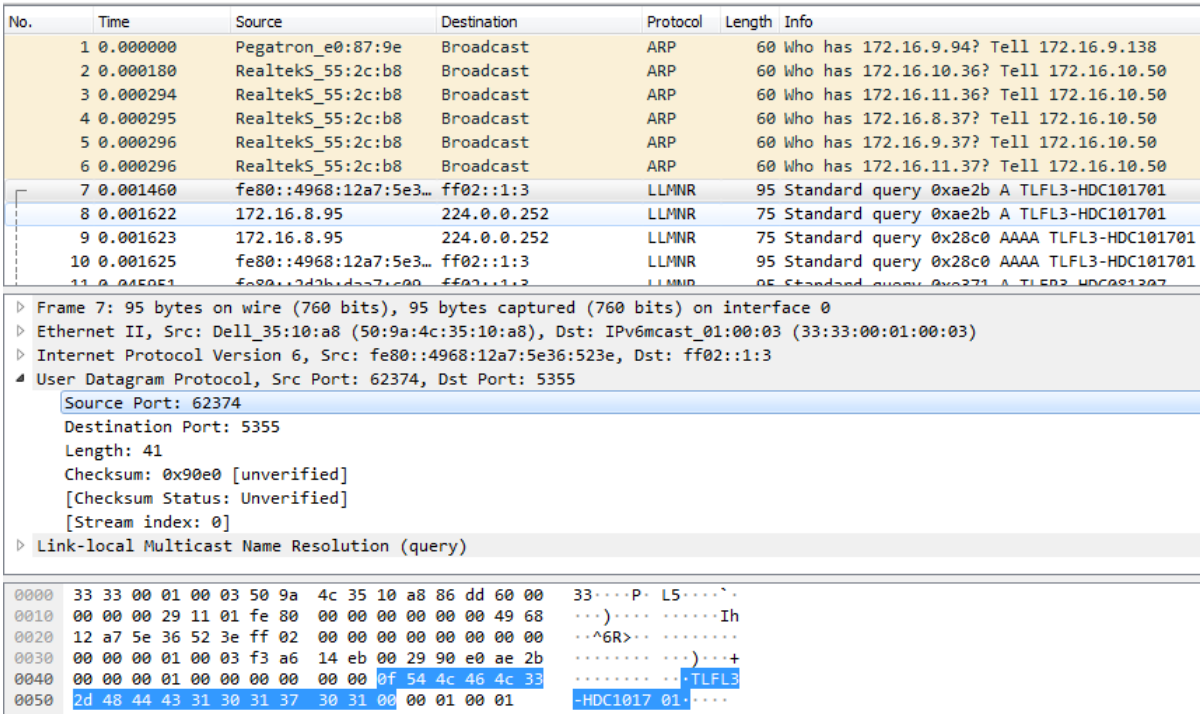
### Exercises

### 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.

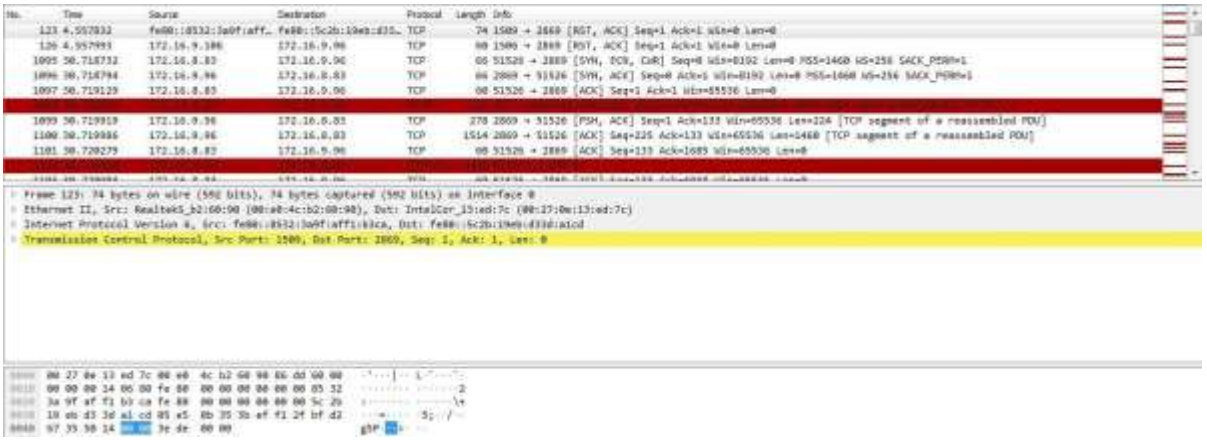
### Output




### 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.



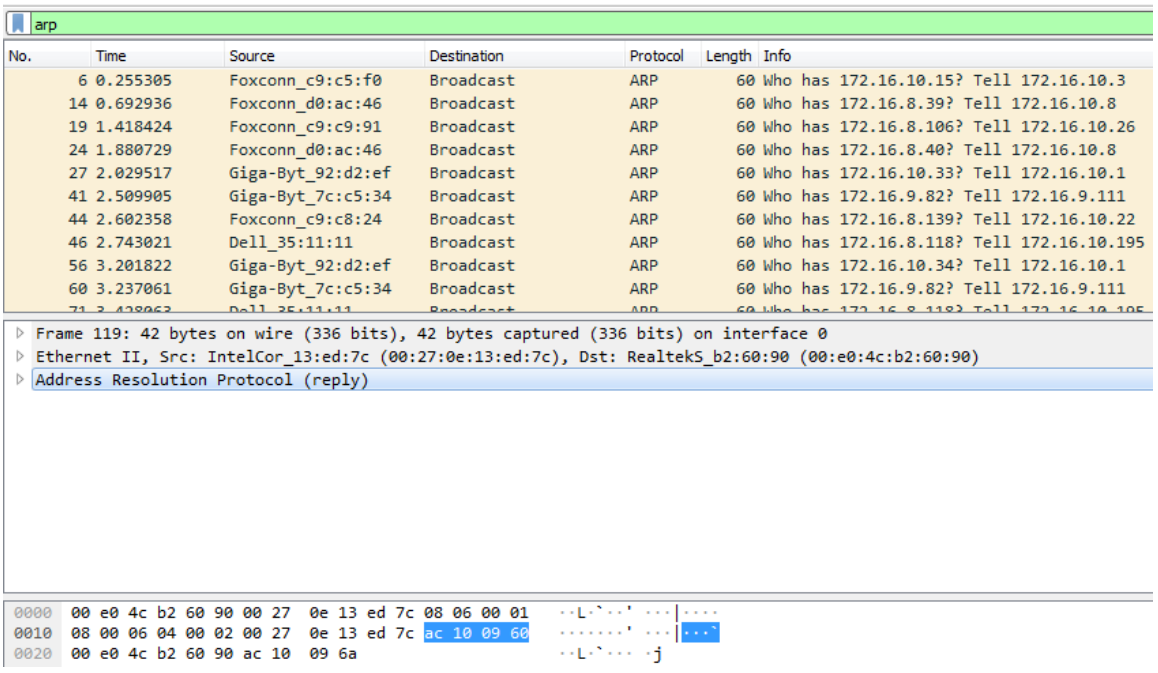
**Flow Graph**

**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.

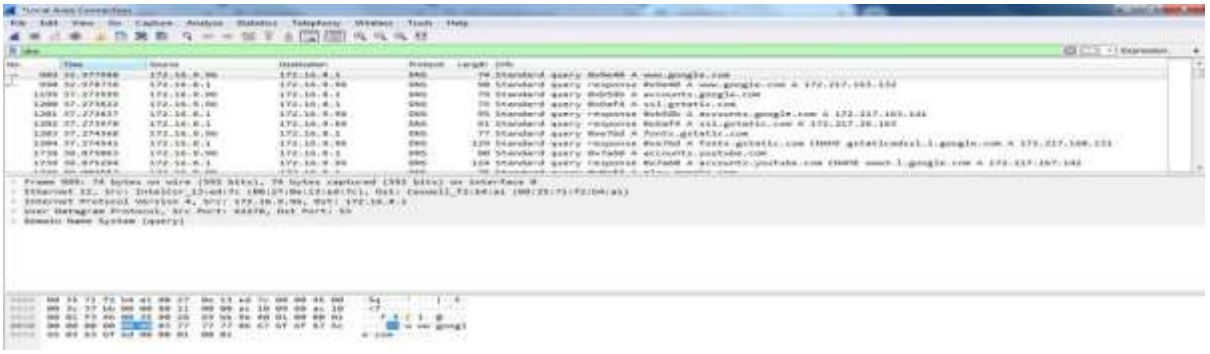
**Output**

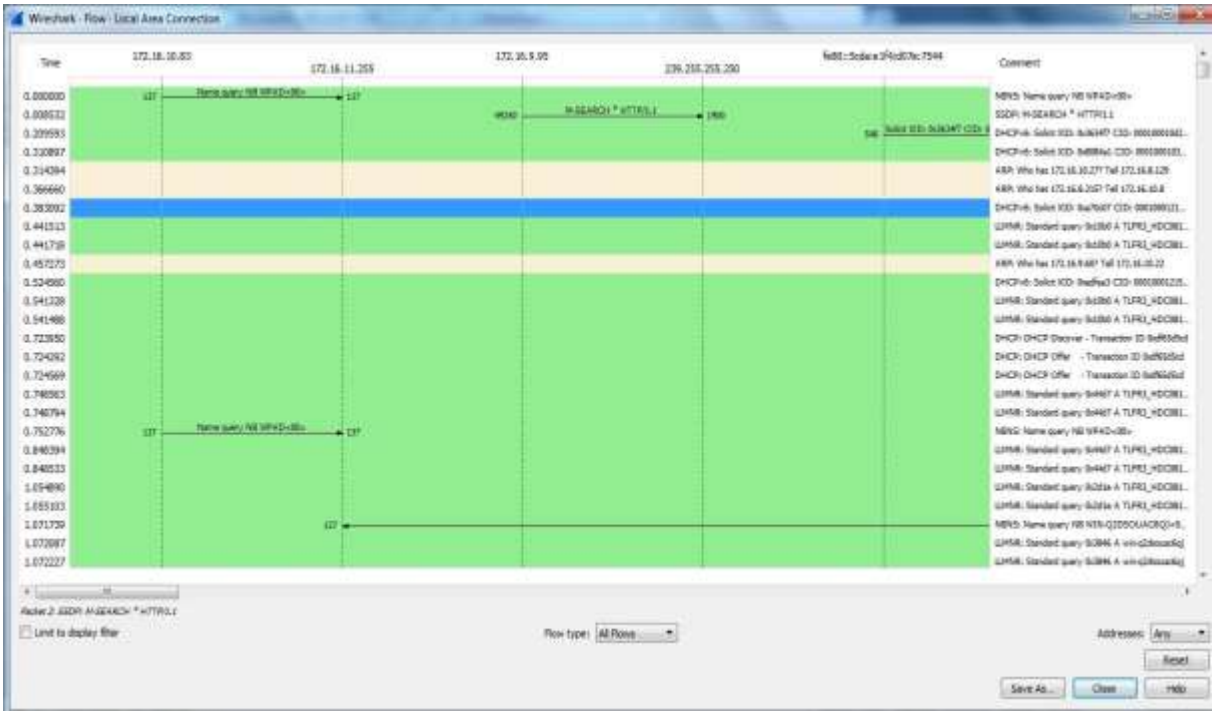



**4. Create a Filter to display only DNS packets and provide the flow graph.**

**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.

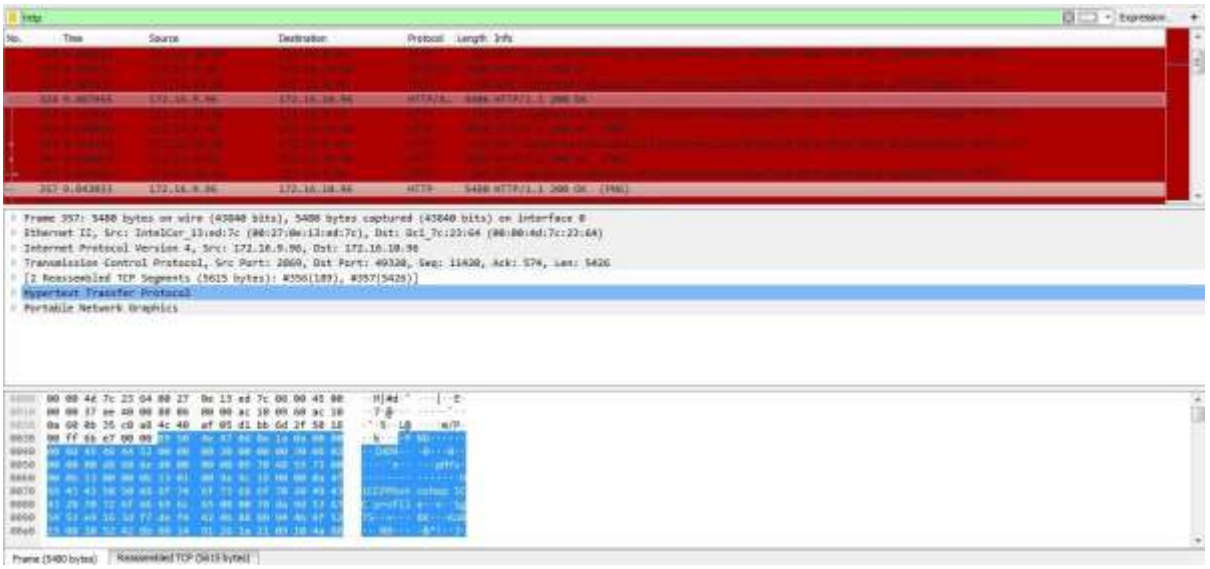




**5. Create a Filter to display only HTTP packets and inspect the packets**

**Procedure**

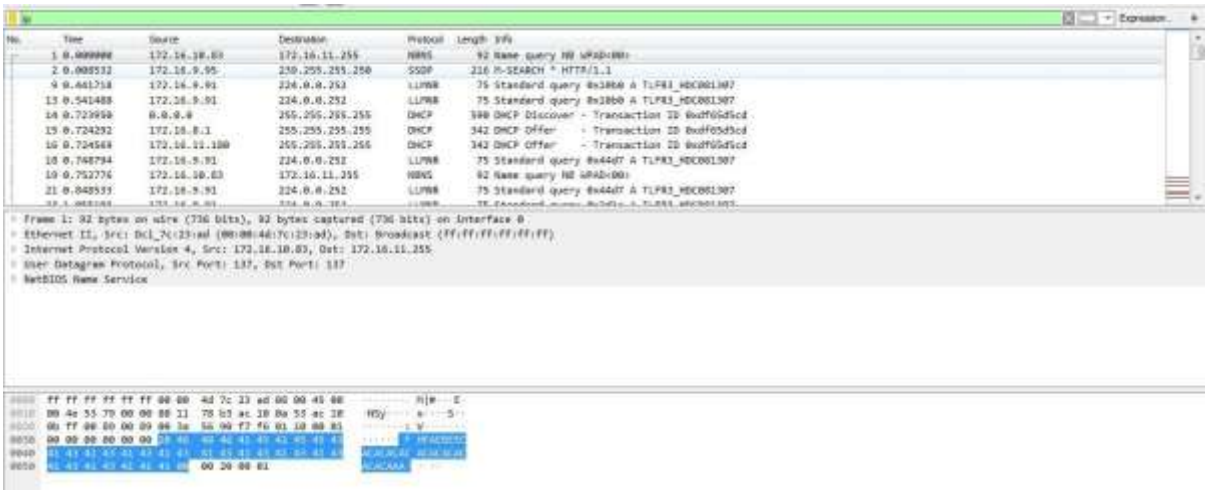
- Select Local Area Connection in Wireshark.
- Go to capture → option
- Select stop capture automatically after 100 packets.
- Then click Start capture.
- Search HTTP packets in search bar.
- Save the packets.



**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



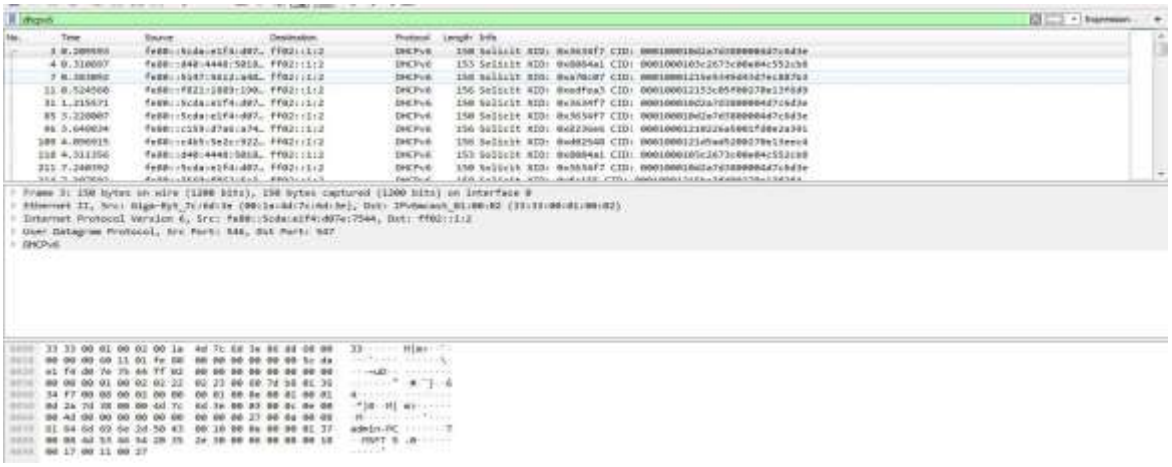
**7. Create a Filter to display only DHCP packets and inspect the packets.**

**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**



**EX NO :9 STUDY OF SOCKET PROGRAMMING AND CLIENT-SERVER MODEL USING TCP AND UDP.**

**Date:**

**AIM:**

To study the concepts of socket programming using python.

**DESCRIPTION:**

## 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:**

| Istance 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 |

| Istance method | Description |
|---|---|
| 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:**

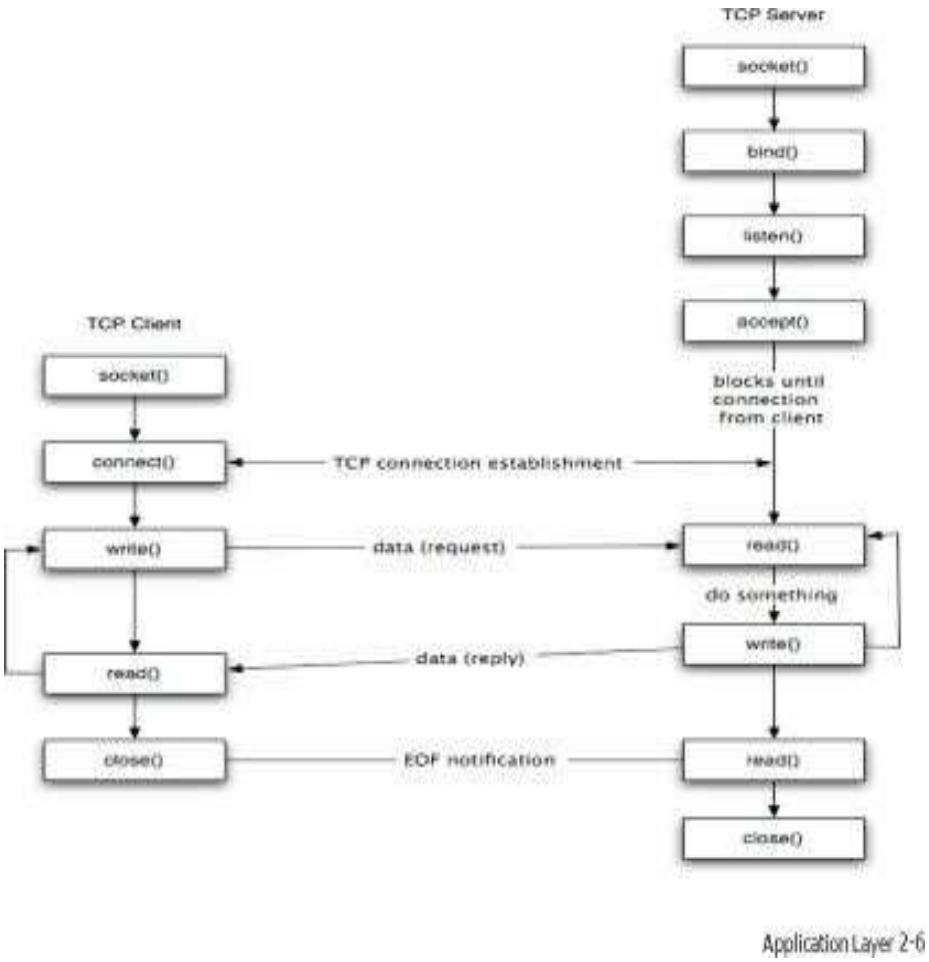
- creating TCP socket, specifying IP address, port number of server process
- client socket is now bound to that specific server

**server accepts connect by:**

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

**application viewpoint:**

TCP provides reliable, in-order byte-stream transfer ("pipe") between client and server




Application Layer 2-6

**HOW TO WORK WITH UDP SOCKETS IN PYTHON:**

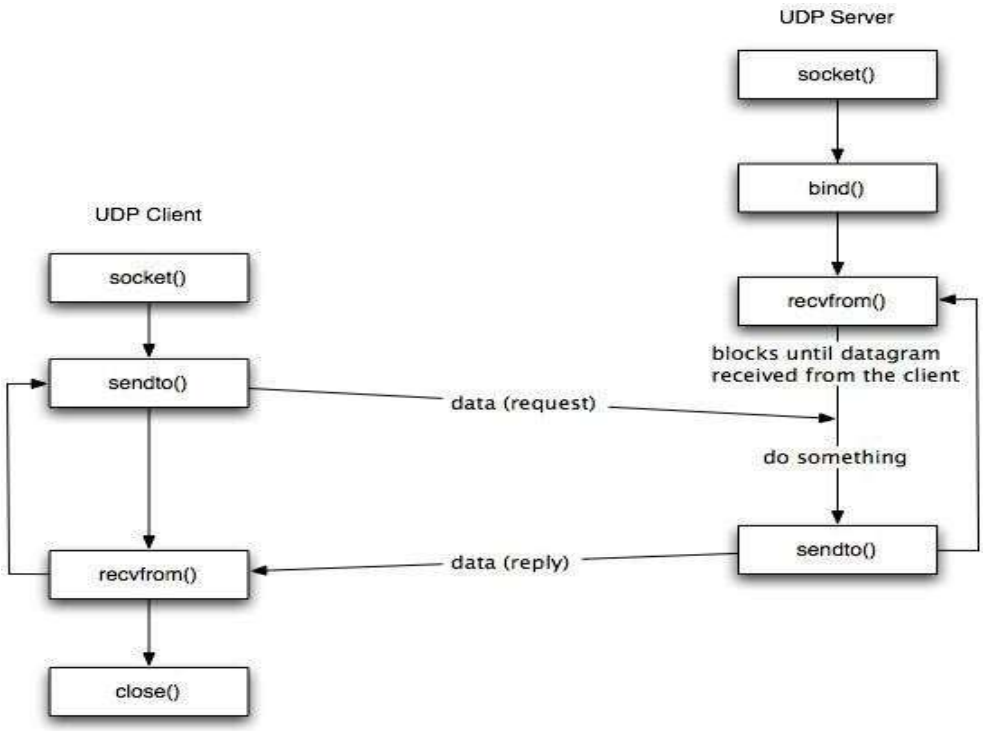
**UDP: no "connection" between client & server**

- 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


UDP Server
socket()
bind()
recvfrom()
blocks until datagram
received from the client
do something
sendto()
UDP Client
socket()
sendto()
data (request)
recvfrom()
data (reply)
close()

**EX.NO.10 SOCKET PING PROGRAM TO TEST THE SERVER CONNECTIVITY**

**AIM:**

Write a socket PING program to test the server connectivity using python.

**ALGORITHM:**

**PROGRAM:**

```
import os
ip_list = ['8.8.8.8']
for ip in ip_list:
    response = os.popen(f"ping {ip}").read()
    if "Received = 4" in response:
        print(f"UP {ip} Ping Successful")
    else:
        print(f"DOWN {ip} Ping Unsuccessful")
```

**Result:**

**Ex.No. 11 PROGRAMS USING TCP SOCKETS**

**a) IMPLEMENTATION OF ECHO CLIENT/SERVER APPLICATION USING TCP:**

**AIM:**

**ALGORITHM:**

**PROGRAM:**

**Client code:**

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

**Server Code:**

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

**Result:**

**Ex.No. 11 b) CHAT APPLICATION USING TCP SOCKET**

**AIM:**

**ALGORITHM:**

**PROGRAM:**

**Source 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 Code**

```
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:**

**Ex.No.12 PROGRAM USING UDP SOCKETS**

**a) IMPLEMENTATION OF ECHO CLIENT/SERVER APPLICATION USING UDP**

**AIM:**

**ALGORITHM:**

**PROGRAM:**

**Client code:**

```
import socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
message = 'test'
addr = ("127.0.0.1", 12000)
client_socket.sendto(message.encode('utf-8'),  addr)
data, server = client_socket.recvfrom(1024).decode()
print(data)
```

**Server Code:**

```
import socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind(('', 12000))
while True:
    message, address = server_socket.recvfrom(1024).decode()
    server_socket.sendto(message.encode('utf-8'),, address)
```

**RESULT:**

**EX. NO. 12 b) DICTIONARY APPLICATION USING UDP SOCKET**

**Aim:**

**Algorithm:**

**BPROGRAM:**

**Source Code:**

**Server Side:**

```
import socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind(('', 12000))
while True:
    m, address = server_socket.recvfrom(1024)

a={'use':'utilize','destroy':'demolish','fall':'drop','decide':'choose','help':'serve','plan':'blueprint','
show':'display','break':'smash','make':'create','hurry':'rush'}
    g=a[m]
    server_socket.sendto(g, address)
```

**Client Side**

```
import socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
while True:
    print("Enter the Word")
    w=input("-->",)
    m=str.encode(w)
    addr = ("127.0.0.1", 12000)
    client_socket.sendto(m, addr)
    data, server = client_socket.recvfrom(1024)
    print("Meaning is →",data)
```

**OUTPUT:**

**Client Side:**

Enter the Word→show

Meaning is→ display

**Result :**

**EXERCISE:**

1. Develop a Chat application using TCP Socket.
2. Write a service to convert uppercase letters to lowercase and the respective client program using UDP Socket.
3. Develop a calculator application using TCP Socket.
4. Write a dictionary service and the respective client program using UDP Socket

**Ex. No. 13** **SIMULATION OF SLIDING WINDOW**

**AIM:**

To write a C program to implement simulation of sliding window protocol.

**ALGORITHM:**

**PROGRAM:**

```
#include<stdio.h>

int main()
{
   int w,i,f,frames[50];

   printf("Enter window size: ");
   scanf("%d",&w);

   printf("\nEnter number of frames to transmit: ");
   scanf("%d",&f);

   printf("\nEnter %d frames: ",f);

   for(i=1;i<=f;i++)
      scanf("%d",&frames[i]);

   printf("\nWith sliding window protocol the frames will be sent in the following manner
(assuming no corruption of frames)\n\n");
   printf("After sending %d frames at each stage sender waits for acknowledgement sent by
the receiver\n\n",w);

   for(i=1;i<=f;i++)
   {
      if(i%w==0)
      {
         printf("%d\n",frames[i]);
         printf("Acknowledgement of above frames sent is received by sender\n\n");
      }
      else
         printf("%d ",frames[i]);
   }

   if(f%w!=0)
      printf("\nAcknowledgement of above frames sent is received by sender\n");

   return 0;
}
```

**OUTPUT:**

Enter window size: 3

Enter number of frames to transmit: 5

Enter 5 frames: 12 5 89 4 6

With sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)

After sending 3 frames at each stage sender waits for acknowledgement sent by the receiver

12 5 89
Acknowledgement of above frames sent is received by sender

4 6
Acknowledgement of above frames sent is received by sender

**RESULT:**

**Ex. No. 14 IMPLEMENTATION OF ARP**

**AIM:**

To write a C program to get the MAC address from the system using Address Resolution Protocol.

**ALGORITHM:**

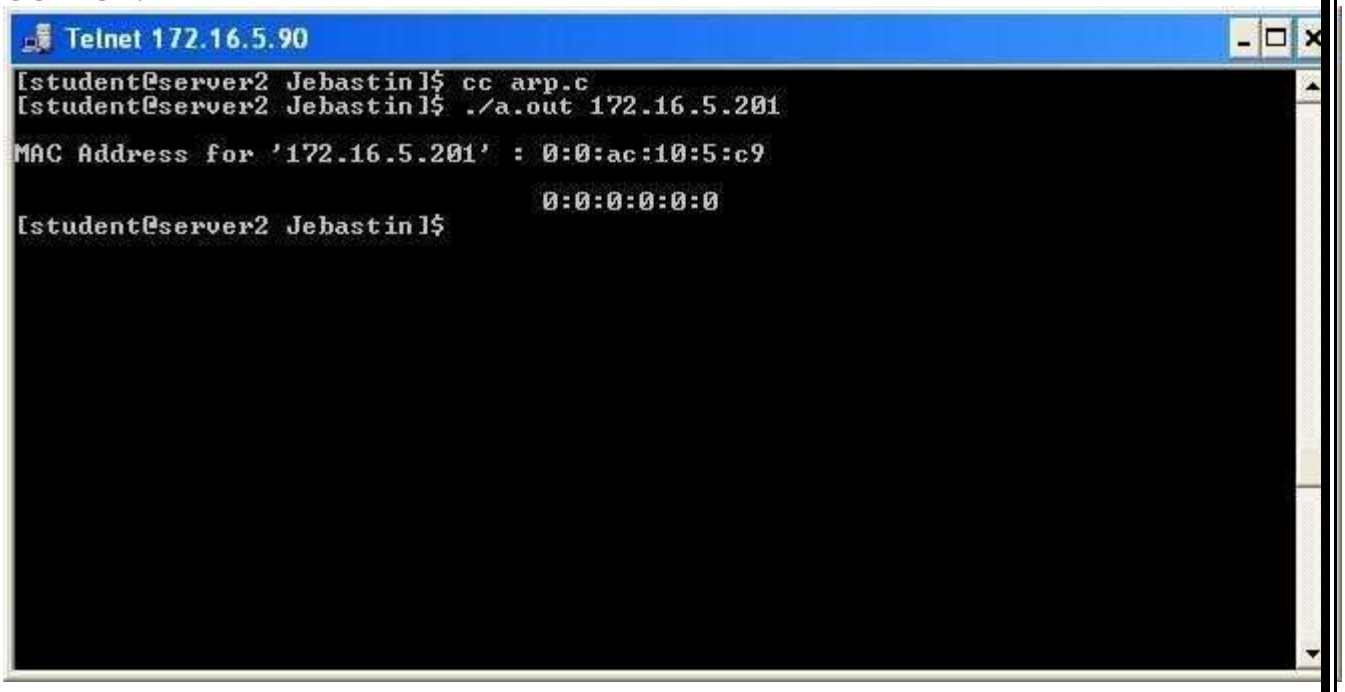
STEP 1: Start
STEP 2: Declare the variables and structure for the socket
STEP 3: Specify the family, protocol, IP address and port number
STEP 4: Create a socket using socket() function
STEP 5: Call memcpy() and strcpy functions
STEP 6: Display the MAC address
STEP 7: Stop

**SOURCE CODE:**

```
#include<sys/types.h>
#include<sys/socket.h>
#include<net/if_arp.h>
#include<sys/ioctl.h>
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<math.h>
#include<complex.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<netinet/if_ether.h>
#include<net/ethernet.h>
#include<stdlib.h>
int main(int argc,char *argv[])
{
      struct sockaddr_in sin={0};
      struct arpreq myarp={{0}};
      unsigned char *ptr;
      int                        sd;
      sin.sin_family=AF_INET;
      if(inet_aton(argv[1],&sin.sin_addr)==0)
      {
            printf("IP address Entered '%s' is not valid \n",argv[1]);
            exit(0);
      }
      memcpy(&myarp.arp_pa,&sin,sizeof(myarp.arp_pa));
      strcpy(myarp.arp_dev,"echo");
      sd=socket(AF_INET,SOCK_DGRAM,0);

     if(ioctl(sd,SIOCGARP,&myarp)==1)
      {
            printf("No Entry in ATP cache for '%s'\n",argv[1]);
            exit(0);
```

```
    }
    ptr=&myarp.arp_pa.sa_data[0];
    printf("\nMAC Address for '%s' : ",argv[1]);
    printf("%x:%x:%x:%x:%x:%x\n",*ptr,*(ptr+1),*(ptr+2),*(ptr+3),*(ptr+4),*(ptr+5));
    printf("\n\t\t\t\t%x:%x:%x:%x:%x:%x\n", myarp.arp_ha.sa_data[0],
            myarp.arp_ha.sa_data[1], myarp.arp_ha.sa_data[2],
            myarp.arp_ha.sa_data[3], myarp.arp_ha.sa_data[4], myarp.arp_ha.sa_data[5]);
    return 0;
}
```

**OUTPUT:**




**RESULT:**

**Ex. No. 15 IMPLEMENTATION OF ARP**

**Date:**

**AIM:**

Simulate the RIP (Routing Information Protocol) algorithm where routers share and update routing tables dynamically.

**ALGORITHM / STEPS:**

1. Each router maintains a routing table with destination, next hop, and ho count.
2. Routers periodically exchange their routing tables with neighbours.
3. When a router receives a neighbour's route:
   - If the destination is not in its table → add it.
   - If it exists but a better route (lower hop count) is found → update it.
4. Apply RIP rules:
   - Max hop count = 15 (infinity).
   - Increment hop count by 1 when receiving from a neighbour.
5. Repeat until convergence or max iterations reached.

**PYTHON SCRIPT**

```
class Router:
    def __init__(self, name):
        self.name = name
        self.routing_table = {}  # {dest: (next_hop, hops)}
        self.neighbors = []

    def add_direct_route(self, dest, hops=0):
        self.routing_table[dest] = (self.name, hops)

    def add_neighbor(self, neighbor):
        self.neighbors.append(neighbor)

    def send_routing_table(self):
        return {dest: hops for dest, (_, hops) in self.routing_table.items()}

    def receive_table(self, neighbor_name, table):
        updated = False
        for dest, neighbor_hops in table.items():
            if dest in self.routing_table:
                current_next_hop, current_hops = self.routing_table[dest]
            else:
```

```
            current_next_hop, current_hops = None, float('inf')

        received_hops = neighbor_hops + 1  # Add one hop from neighbor

        if received_hops < current_hops and received_hops <= 15:
            self.routing_table[dest] = (neighbor_name, received_hops)
            updated = True
            print(f"{self.name}: Updated route to {dest} via {neighbor_name},
hops={received_hops}")
        return updated

    def show_table(self):
        print(f"\nRouting Table for {self.name}:")
        print("Destination\tNext Hop\tHops")
        print("----------------------------------------")
        for dest, (nh, hops) in sorted(self.routing_table.items()):
            print(f"{dest}\t{nh}\t\t{hops}")


# === Simulation Setup ===
router_a = Router("A")
router_b = Router("B")
router_c = Router("C")

# Directly connected networks
router_a.add_direct_route("Net1")
router_b.add_direct_route("Net2")
router_c.add_direct_route("Net3")

# Connect routers
router_a.add_neighbor(router_b)
router_b.add_neighbor(router_a)
router_b.add_neighbor(router_c)
router_c.add_neighbor(router_b)

# Initial tables before exchange
print("=== Initial Routing Tables ===")
router_a.show_table()
router_b.show_table()
router_c.show_table()

# Simulate RIP updates
print("\n=== Starting RIP Updates ===")
for i in range(5):  # Simulate up to 5 rounds
    print(f"\n--- Round {i+1} ---")
```

```
    updated = False
    for src in [router_a, router_b, router_c]:
        for neighbor in src.neighbors:
            print(f"{src.name} sending table to {neighbor.name}")
            updated |= neighbor.receive_table(src.name, src.send_routing_table())
    if not updated:
        print("No more updates. Converged.")
        break

# Final routing tables
print("\n=== Final Routing Tables ===")
router_a.show_table()
router_b.show_table()
router_c.show_table()
```

**Output:**

```
=== Initial Routing Tables ===

Routing Table for A:
Destination    Next Hop     Hops
----------------------------------------
Net1     A              0

Routing Table for B:
Destination    Next Hop     Hops
----------------------------------------
Net2     B              0

Routing Table for C:
Destination    Next Hop     Hops
----------------------------------------
Net3     C              0

=== Starting RIP Updates ===

--- Round 1 ---
A sending table to B
B: Updated route to Net1 via A, hops=1
B sending table to A
A: Updated route to Net2 via B, hops=1
B sending table to C
C: Updated route to Net2 via B, hops=1
C: Updated route to Net1 via B, hops=2
C sending table to B
```

B: Updated route to Net3 via C, hops=1

--- Round 2 ---
A sending table to B
B sending table to A
A: Updated route to Net3 via B, hops=2
B sending table to C
C sending table to B

--- Round 3 ---
A sending table to B
B sending table to A
B sending table to C
C sending table to B
No more updates. Converged.

=== Final Routing Tables ===

Routing Table for A:

| Destination | Next Hop | Hops |
|---|---|---|
| Net1 | A | 0 |
| Net2 | B | 1 |
| Net3 | B | 2 |

Routing Table for B:

| Destination | Next Hop | Hops |
|---|---|---|
| Net1 | A | 1 |
| Net2 | B | 0 |
| Net3 | C | 1 |

Routing Table for C:

| Destination | Next Hop | Hops |
|---|---|---|
| Net1 | B | 2 |
| Net2 | B | 1 |
| Net3 | C | 0 |

**Result:**

**Ex. No. 16 CONFIGURING A CISCO ROUTER AS A DHCP SERVER**
**Date:**

**Aim**
To configure a Cisco router to act as a DHCP server , dynamically assigning IP addresses, default gateway, DNS server, and other network parameters to clients in the local network.

**ALGORITHM / CONFIGURATION STEPS**

1. Enter Privileged EXEC Mode .
2. Enter Global Configuration Mode .
3. Define a DHCP pool using ip dhcp pool <pool-name>.
4. Specify the network range and subnet mask .
5. Set the default gateway with default-router.
6. Assign a DNS server using dns-server.
7. Optionally set the domain name .
8. Exclude static IP addresses from the DHCP pool using ip dhcp excluded-address.
9. Save the configuration.
10. Verify the configuration using appropriate show commands.

PROGRAM (Cisco IOS CLI Commands)

**CONFIGURE THE ROUTER AS A DHCP SERVER**

Router> enable
Router# configure terminal
Step 1: Exclude Static IP Addresses (e.g., .1 to .10)

Router(config)# ip dhcp excluded-address 192.168.1.1 192.168.1.10
Step 2: Create a DHCP Pool
Router(config)# ip dhcp pool OFFICE-LAN
Router(dhcp-config)# network 192.168.1.0 255.255.255.0
Router(dhcp-config)# default-router 192.168.1.1
Router(dhcp-config)# dns-server 8.8.8.8
Router(dhcp-config)# domain-name example.com
Router(dhcp-config)# exit
Step 3: Exit and Save Configuration
Router(config)# exit
Router# write memory

OUTPUT / VERIFICATION COMMANDS
Use these commands to verify that the DHCP server is working correctly.

Show DHCP Pool Information:
Router# show ip dhcp pool
Sample Output:

Pool OFFICE-LAN :
Utilization mark (high/low) : 100 / 0
Subnet size (first/last) : 255.255.255.0 (/24)
Total addresses : 254
Leased addresses : 0
Excluded addresses : 10
Pending event : none

1 subnet is currently in the pool :
Current subnet's utilization : 0.00%
Subnet 192.168.1.0 netmask 255.255.255.0
Show DHCP Binding Table:
Router# show ip dhcp binding
Sample Output:

| IP address | Client-ID/ Hardware addr | Lease expiration | Type |
|---|---|---|---|
| 192.168.1.11 | 0100.1a2b.3c4d5e | Infinite | Automatic |

Show DHCP Server Statistics:
Router# show ip dhcp server statistics
Sample Output:

Memory usage 23456
Address allocation fail count 0

**CLIENT SIDE VERIFICATION** (on PC connected to router):
On a Windows PC connected via Ethernet:

cmd

C:\> ipconfig /release
C:\> ipconfig /renew
Expected Output:
Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix . : example.com
Link-local IPv6 Address . . . . . : fe80::1c2d:3e5f:ac12:3456%11
IPv4 Address. . . . . : 192.168.1.11
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.1.1
DNS Servers . . . . . : 8.8.8.8


**RESULT:**

## Mini Project

| sl.no | Project Idea | Tools Used | Description |
|---|---|---|---|
| 1 | Network Traffic Analyzer Dashboard | Python, Flask, Wireshark, Pandas | Build a dashboard that reads pcap files and visualizes traffic patterns. |
| 2 | Automated Network Health Checker | Python, Ping, Traceroute, SMTP | Create a script that checks device reachability and sends alerts via email. |
| 3 | Simple CLI-Based Network Emulator | Python, Scapy | Emulate sending/receiving packets, simulate ping, ARP, etc., using Scapy. |
| 4 | Lab Automation Tool using Ansible | Ansible, Linux, Cisco IOS | Automate repetitive tasks across multiple devices using Ansible playbooks. |
| 5 | IoT Network Simulation with MQTT Broker | Mosquitto, Raspberry Pi/VM | Simulate an IoT network with sensors publishing data over MQTT. |

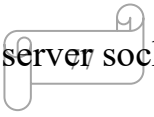
1. What is the ls command used for?
   It lists the contents of a directory.
   Example: ls -l shows detailed file information.
2. How do you create a new directory in Linux?
   Use the mkdir command.
   Example: mkdir folder_name creates a folder.
3. What does the pwd command do?
   It prints the current working directory path.
   Useful to know your location in the file system.
4. How do you view the contents of a file?
   Use cat filename.
   Also can use less or nano for editing/viewing.
5. What is the function of the man command?
   Displays manual pages for any command.
   Example: man ls gives help on ls.
6. How do you check IP address in Linux?
   Use ip addr show or ifconfig.
   ifconfig may be deprecated in newer systems.
7. Which file is used to configure static IP in Linux?
   /etc/network/interfaces or netplan in Ubuntu.
   Or nmcli in some distributions.
8. How do you restart the network service in Linux?
   Use sudo systemctl restart networking or networkmanager.
   Depends on distribution and setup.
9. What is the purpose of the ping command?
   Tests connectivity between two hosts.
   Sends ICMP echo requests and waits for replies.
10. What is the use of traceroute command?
    Shows the route packets take to reach a destination.
    Used to debug routing issues.
11. What is an IP address?
    A unique identifier assigned to devices on a network.
    Enables communication over the network.
12. What is the difference between IPv4 and IPv6?
    IPv4 uses 32-bit addresses; IPv6 uses 128-bit.
    IPv6 provides more addresses and better security.
13. How do you assign a static IP address in Linux?
    Edit network config files or use nmcli.
    Example: sudo ip addr add 192.168.1.10 dev eth0
14. What is a loopback address?
    127.0.0.1 used to refer to the local machine.
    Used to test network applications locally.
15. What is DHCP?
    Dynamic Host Configuration Protocol.

Automatically assigns IP addresses to clients.

16. What is a subnet mask?
    Defines which part of an IP is network and host.
    Helps in dividing networks into subnets.
17. What is CIDR notation?
    Classless Inter-Domain Routing.
    Example: 192.168.1.0/24 means 24 bits for network.
18. What is the default subnet mask for Class C?
    255.255.255.0
    Allows 254 usable hosts per network.
19. Why is subnetting used?
    To reduce broadcast domains and improve performance.
    Helps manage large networks efficiently.
20. What is a broadcast address?
    The last address in a subnet used to send messages to all hosts.
    Example: 192.168.1.255
21. What is a LAN?
    Local Area Network connects devices within a limited area.
    Usually within a building or campus.
22. What is the role of a switch in a LAN?
    Connects multiple devices within the same network.
    Uses MAC addresses to forward frames.
23. What is a VLAN?
    Virtual LAN allows logical grouping of devices.
    Segments a physical LAN into multiple virtual networks.
24. What is CSMA/CD?
    Carrier Sense Multiple Access with Collision Detection.
    Prevents data collisions in Ethernet networks.
25. What is the purpose of a hub?
    Hubs broadcast data to all connected devices.
    Not commonly used now due to switches being more efficient.
26. What is Wireshark?
    A packet analyzer tool that captures and inspects network traffic.
    Used for troubleshooting and learning protocols.
27. How do you start capturing packets in Wireshark?
    Select interface and click "Start".
    Packets are captured in real-time.
28. What is a filter in Wireshark?
    Filters allow you to see only relevant packets.
    Example: tcp.port == 80 filters HTTP traffic.
29. What is promiscuous mode?
    Allows a network card to capture all traffic on the network.
    Required for full packet capture.
30. What is the difference between TCP and UDP packets in Wireshark?
    TCP has handshake and sequence numbers.

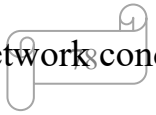
UDP is connectionless and faster but unreliable.

31. What is a socket?
    An endpoint for communication between machines.
    Used in client-server architecture.
32. What protocol is used in a ping program?
    ICMP (Internet Control Message Protocol).
    Used to check if a host is reachable.
33. How is a socket identified?
    By IP address and port number.
    Example: 192.168.1.1:80
34. What is socket() function in Python?
    Creates a new socket.
    Takes domain, type, and protocol as arguments.
35. What is the purpose of checksum in ICMP packets?
    Ensures data integrity during transmission.
    Calculated before sending and verified at receiver.
36. What is system administration?
    Managing and maintaining computer systems and servers.
    Includes user management, backups, and updates.
37. What is a daemon process?
    Background process that runs without user interaction.
    Example: Apache web server.
38. How do you check running processes in Linux?
    Use ps or top command.
    htop is also available for enhanced view.
39. What is cron?
    Scheduler to run tasks automatically at fixed times.
    Defined in crontab files.
40. What is log rotation?
    Process of compressing and archiving old logs.
    Managed by logrotate utility in Linux.
41. What is a client-server model?
    Architecture where client requests services from a server.
    Commonly used in web, email, and file transfer.
42. What is the difference between TCP and UDP sockets?
    TCP is connection-oriented and reliable.
    UDP is connectionless and fast.
43. What is the bind() function used for?
    Associates a socket with a specific IP and port.
    Needed before listening or connecting.
44. What is the listen() function used for?
    Puts the server socket into passive mode.
    Waits for incoming connections.
45. What is the accept() function?
46. Accepts an incoming connection on a server socket.



Returns a new socket descriptor for communication.

46. How do you create a TCP server in Python?
    Use socket.socket(socket.AF_INET, socket.SOCK_STREAM).
    Then bind, listen, and accept connections.
47. How do you send data using TCP sockets?
    Use send() or sendall() methods.
    Data must be in bytes format.
48. How do you receive data from a TCP socket?
    Use recv(buffer_size) method.
    Buffer size usually set to 1024 bytes.
49. What is multithreading in socket programming?
    Allows handling multiple clients simultaneously.
    Each client handled in a separate thread.
50. What is the close() function used for?
    Closes the socket connection.
    Frees up resources after communication ends.
51. How do you create a UDP socket in Python?
    Use socket.socket(socket.AF_INET, socket.SOCK_DGRAM).
    No need to connect or accept.
52. What is the sendto() function used for?
    Sends data to a specific address in UDP.
    Requires target address as argument.
53. How do you receive data in UDP?
    Use recvfrom(buffer_size) method.
    Returns data and sender's address.
54. Is UDP connectionless or connection-oriented?
    Connectionless.
    No handshake required before sending data.
55. Can UDP guarantee delivery of packets?
    No, it’s unreliable.
    But faster than TCP due to no acknowledgments.
56. What is the sliding window protocol?
    Flow control mechanism in TCP.
    Allows sending multiple packets before ACK.
57. What is window size?
    Number of packets that can be sent without acknowledgment.
    Adjusted dynamically based on network conditions.
58. What is flow control?
    Prevents sender from overwhelming the receiver.
    Done using window size negotiation.
59. What is congestion control?
    Prevents network overload by adjusting window size.
    Algorithms like slow start and congestion avoidance used.
60. How is window size calculated?
    Based on receiver buffer space and network conditions.



Max(DataWindow, CongestionWindow).

61. What is ARP?
    Address Resolution Protocol maps IP to MAC address.
    Essential for local network communication.
62. How does ARP work?
    Host broadcasts request asking for MAC of IP.
    Owner responds with its MAC address.
63. What is ARP cache?
    Stores recently resolved IP-MAC mappings.
    View with arp -a or arp -n.
64. What is Gratuitous ARP?
    Sent by a device to announce its presence.
    Used to detect IP conflicts.
65. What is Proxy ARP?
    Router responds to ARP requests on behalf of others.
    Used in legacy network environments.
66. What is RIP?
    Routing Information Protocol, a distance-vector protocol.
    Uses hop count as metric.
67. What is the maximum hop count in RIP?
    15 hops. Beyond that, network is unreachable.
    Hop count 16 means infinity.
68. How do you enable RIP version 2 in Cisco router?
    Use router rip, then version 2.
    Advertise networks using network <ip>.
69. What is split horizon?
    Prevents routing loops by not advertising routes back.
    Enabled by default in many routers.
70. What is the update interval in RIP?
    Every 30 seconds.
    Full routing table sent periodically.
71. What is DHCP?
    Dynamic Host Configuration Protocol.
    Assigns IP addresses automatically to clients.
72. How do you define a DHCP pool in Cisco?
    Use ip dhcp pool <name>.
    Then specify network, default-router, DNS, etc.
73. How do you exclude static IPs from DHCP?
    Use ip dhcp excluded-address <start> <end>.
    Prevents conflict with manually assigned IPs.
74. What is a DHCP lease?
    Time duration for which an IP is assigned.
    Can be infinite or time-limited.
75. What are the four steps in DHCP handshake?
    Discover, Offer, Request, Acknowledge (DORA).

Ensures correct assignment of IP addresses.