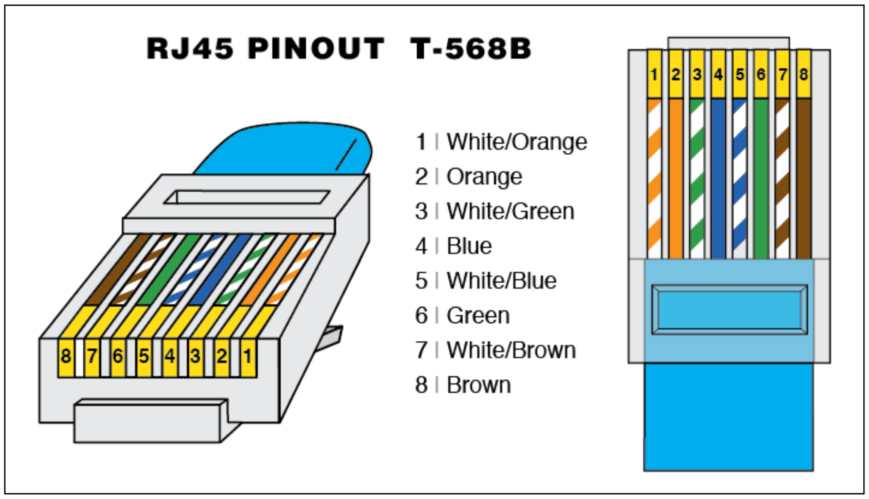
Experiment No 1

Aim: Study of RJ45 and CAT cabling and connection using crimping tool and Study LAN topologies, network devices.

Theory:

RJ45:

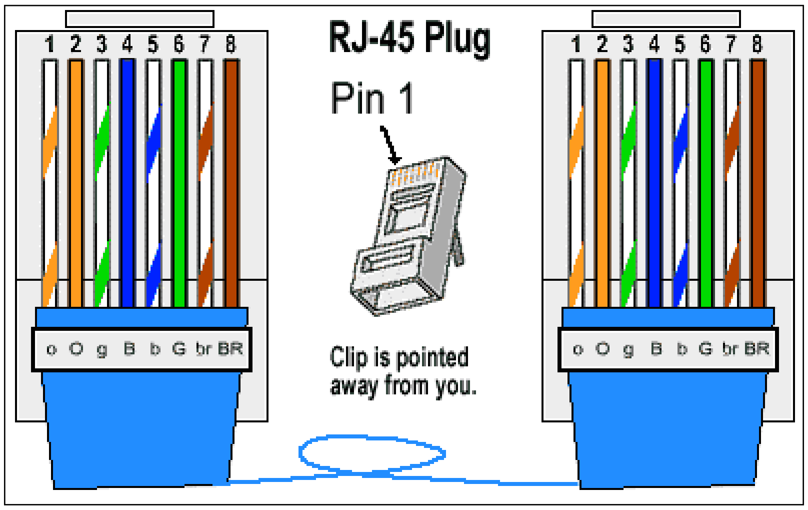
RJ45 cable is used for connect the ALL HMI and engineer station through a switch to communicated each other. It is used to download the any modification and which is made in graphics in engineering station.RJ45 cable also used for communicate the printer with computer. There are four pairs of wires in an Ethernet cable, and an Ethernet connector (8P8C) has eightpin slots. Each pin is identified by a number, starting from left to right, with the clip facing away from you.



There is two kinds of Ethernet cable is used for communication.

1. Straight Through
2. Cross over cable

Straight Through cable: STRAIGHT THROUGH Ethernet cables are the standard cable used for almost all purposes, and are often called "patch cables". It is highly recommend you duplicate the color order as shown on the left. Note how the green pair is not side-by-side as are all the other pairs. This configuration allows for longer wire runs.

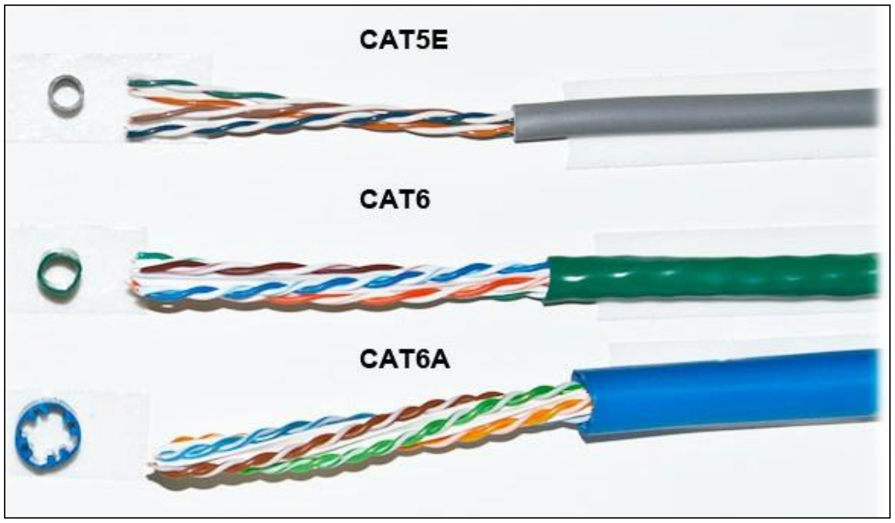


CROSSOVER CABLES: The purpose of a Crossover Ethernet cable is to directly connect one computer to another computer (or device) without going through a router, switch or hub.

CAT Cables:

The Cat, as you might know, is short for **"Category**." The term "Category" refers to the differentlevels of performance in signal bandwidth, attenuation and crosstalk associated with each cable's design. Category 1 cables are a 2-pair copper UTP designed for POTS (plain old telephone systems). **CAT 5e** is currently the most commonly used cable, mainly due to its low production cost and support for speeds faster than Cat 5 cables.

A Cat6 cable has a bandwidth capacity of 250 MHz, for example, and it offers you speeds of up to 10 Gbps. It's also compatible with both Cat5 and Cat5E cables. "A Cat6 cable is used mainly for computer networks reaching a GB, 1000 Mbps or one Gbps of data transfer speed (DTR) or higher,"



Crimping Tool:

Crimpers are tools used to make cold weld joints between two wires or a wireand aconnector, such as lugs. Ideally, the electrical and mechanical properties of the weld joint are as strong as the parent materials. Crimping tools are sized according to the wire gauges (using AWG - American Wire Gauge) they can accept. Some come with interchangeable die heads thatallow for a wider range of wire sizes and connectors.

How to use: First you will need to strip the length of wire that you want to crimp. Then, attach the connector. For crimping tools with interchangeable dies, you will need to select the right die head for the connector by matching wire gauge ratings. For dieless crimpers, you will need to match to the proper groove. Finally, apply pressure, take out the newly crimped connector,and give a few tugs to make sure you have a solid and secure connection.

LAN Topologies:

Bus

A bus topology is a local area network (LAN) where each of the networked devices are attached to a single cable or link as shown in figure 1. In a bus topology, stations are attached to a linear multiport medium where only half-duplex operations exist between a station and a bus.

NOTE

Half-duplex is where communication occurs bi-directionally on one cable. This means that a device sending data cannot receive data at the same time. Full-duplex uses two individual cables, one to send and another to receive. This allows a device to send at the same time it receives data.

Frames that are transmitted to the bus provide the address of the frame's destination. If the frame gets to the end of the link and the frame has not found its intended destination, then the frame is lost. In a bus topology, there is no security; every node attached to the line can see the conversations of theother nodes on the link.

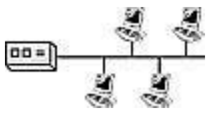


Figure 1, A bus topology LAN. Notice that all the workstations are connected by a single cable.

Ring

In a ring topology LAN (shown in Figure 2), as in a bus topology LAN, all the nodes or devices in the network are attached to the network on the same cable or link. The difference is that a ring topology makes a complete circle. Both Token Ring/IEEE and Fiber Distributed Data Interface (FDDI) use a ring topology. FDDI is an American National Standard Institute (ANSI) X3T9.5 standard cable, which now supports up to Gigabit speeds using fiber-optic cabling. It can use a single ring for half-duplex operations or a dual–ring architecture for full-duplex operations.

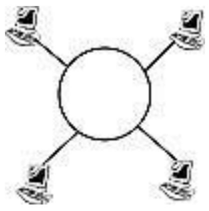


Figure 2 A. ring topology LAN. Notice that all of the nodes connect to the ring. Data for most implementations travels in one direction on the ring. However, many technologies, including Token Ring, allow for a second ring which allows for full-duplex operations.

When a break in the ring occurs, such as a cut cable or other cabling problem, it affects all the stations. This means that none of the stations connected can receive or transmit data. The longer the cable or link and the more attached stations, the more repeaters that are needed. However, due to timing distortions within signals, a limited number of repeaters can be used in the same network. In a ring-topology network, centralized access means that faults are easy to detect and isolate. Multiple rings are sometimes used to make a very robust and reliable network.

Star

The star topology is the most common topology in today's networks, and includes Ethernet, Fast Ethernet, and Gigabit Ethernet. Each node in a star topology connects to a dedicated link where the other end connects to a switch or hub. In the star-topology network shown in Figure 3., multiple devices are connected to a switch or hub.

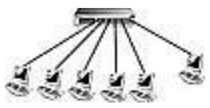


Figure 3. A Star topology LAN. Notice that each workstation is directly connected to a hub or switch.

One of the best reasons to use a star topology is that a loss of any node will not disrupt network operations. It is also easy to add or remove a node from the network. From wiring to installation, it is particularly easy to set up a star topology network.

Network Devices:

- Repeater – A repeater operates at the physical layer. Its job is to regenerate the signal over the same network before the signal becomes too weak or corrupted so as to extend the length to which the signal can be transmitted over the same network. An important point to be noted about repeaters is that they do not amplify the signal. When the signal becomes weak, theycopy the signal bit by bit and regenerate it at the original strength. It is a 2 port device.
- Hub – A hub is basically a multiport repeater. A hub connects multiple wires coming from different branches, for example, the connector in star topology which connects different stations. Hubs cannot filter data, so data packets are sent to all connected devices. In other words, collision domain of all hosts connected through Hub remains one. Also, they do not have intelligence to find out best path for data packets which leads to inefficiencies and wastage.
  Types of Hub
  Active Hub:- These are the hubs which have their own power supply and can clean , boost and relay the signal along the network. It serves both as a repeater as well as wiring center. These are used to extend maximum distance between nodes.
  Passive Hub:- These are the hubs which collect wiring from nodes and power supply from active hub. These hubs relay signals onto the network without cleaning and boosting them and can’t be used to extend distance between nodes.
- Bridge – A bridge operates at data link layer. A bridge is a repeater; with add on functionality of filtering content by reading the MAC addresses of source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2 port device.

Types of Bridges-

Transparent Bridges:- These are the bridge in which the stations are completely unaware of the bridge's existence i.e. whether or not a bridge is added or deleted from the network reconfiguration of the stations is unnecessary. These bridges make use of two processes i.e. bridge forwarding and bridge learning.

Source Routing Bridges:- In these bridges, routing operation is performed by source station and the frame specifies which route to follow. The hot can discover frame by sending a special frame called discovery frame, which spreads through the entire network using all possible paths to destination.

- Switch – A switch is a multiport bridge with a buffer and a design that can boost its efficiency (large number of ports imply less traffic) and performance. Switch is data link layer device. Switch can perform error checking before forwarding data that makes it very efficient as it does not forward packets that have errors and forward good packets selectively to correct port only. In other words, switch divides collision domain of hosts, but broadcast domain remains same.
- Routers – A router is a device like a switch that routes data packets based on their IP addresses. Router is mainly a Network Layer device. Routers normally connect LANs and WANs together and have a dynamically updating routing table based on which they make decisions on routing the data packets. Router divide broadcast domains of hosts connected through it.
- Gateway – A gateway, as the name suggests, is a passage to connect two networks together that may work upon different networking models. They basically works as the messenger agents that take data from one system, interpret it, and transfer it to another system. Gateways are also called protocol converters and can operate at any network layer. Gateways are generally more complex than switch or router.
- Brouter – It is also known as bridging router is a device which combines features of both bridge and router. It can work either at data link layer or at network layer. Working as router, it is capable of routing packets across networks and working as bridge, it is capable of filtering local area network traffic.

Conclusion: Thus we have Studied RJ45 and CAT cabling and connection using crimping tool and Study LAN topologies, network devices.

Experiment No 2

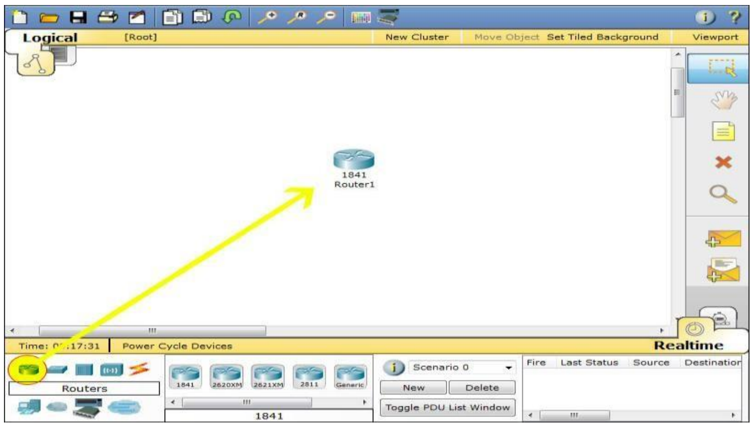
Aim: Setup a network and configure IP addressing, subnetting, Masking. (Eg. CISCO Packet Tracer)

Theory:

Configuring ip address:

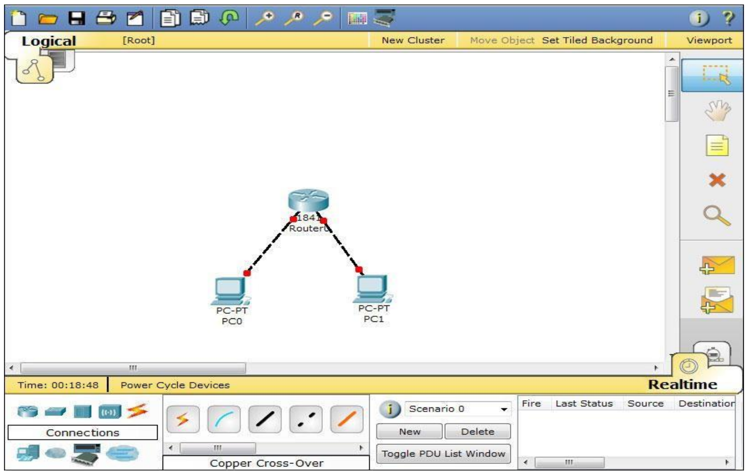
Setting the topology

1. Drag and drop the router from the bottom of the screen



2. Select end devices from the bottom left-hand corner and drag it to the sandbox screen. Do this twice to make two computers appear below the router.

3. select connections from the same bottom left-hand corner. When you connect like-devices(Such as a router and computer) you use a crossover cable, so you should select copper cross-over cable from the second menu to the immediate right. Click on Router0, and connect the cable via FastEthernet0/0 as seen below:



4. click the PC0 and select FastEthernet. You will notice that although a link is established, it is not functional. You can tell by the red dots that are present on both ends of the connection. Once the router is configured correctly, the red dots will turn green to indicate the devices are able to communicate. Do the same operation to PC1, only this time connect the cable to FastEthernet0/1 since FastEthernet0/0 is already taken by PC0.

Configure the Router:

1. Type config terminal (or config t for short) to access the configuration menu.

2. Type interface fastethernet0/0 to access Ethernet0/0

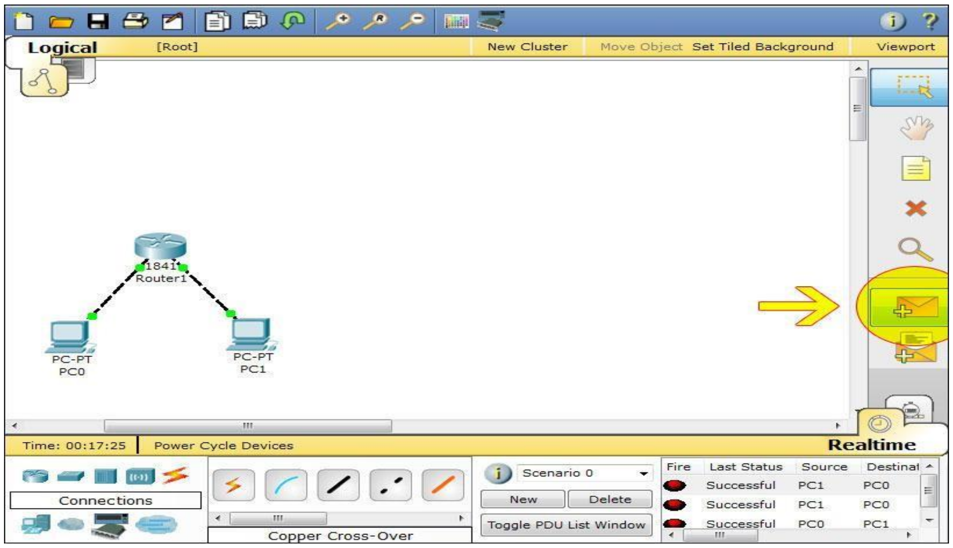
3. Type ip address 192.168.10.1 255.255.255.0 to assign an IP address and subnet mask to the interface.

4. Type no shutdown to open the interface up for business.

Configuring the gateway in packet tracer:

1. Click on PC0 to bring up the configuration menu. Under global settings you will find a field for the gateway. Enter the corresponding IP address of the router's interface, which is 192.168.10.1. Then click the FastEthernet tab on the left column to set the actual computer's IP address to be on the network. Use 192.168.10.2 for the IP address, and 255.255.255.0 for the subnet mask.

2. Do the same thing for PC1, only use 192.168.20.1 for the gateway address, 192.168.20.2 for the IP address, and 255.255.255.0 for the subnet mask. You can confirm that your network works by sending out a packet of information from PC0 to PC1, and vice versa. Click the packet icon on the right menu as seen below:



Fire	Last Status	Source	Destinat
●	Successful	PC1	PC0
●	Successful	PC1	PC0
●	Successful	PC0	PC1

Click on PC0 and then click PC1. On the lower right of the screen you will see a message box that says "Successful." If it doesn't, you may have had a syntax error when putting in an IP address or router configuration command.

Subnetting a network:

A sub network or subnet is a logical subdivision of an IP network. The practice of dividing a network into two or more networks is called sub netting. Subnetting offers many advantages. Some of them are:

1. It provides security to the network.

2. Speeds up the network thus improving the performance of the network.

3. It allows for better organization of the resources.

Procedure:

Since we require only three subnets, we will create the first three subnets. The first subnet (192.168.10.1 to 192.168.10.63)

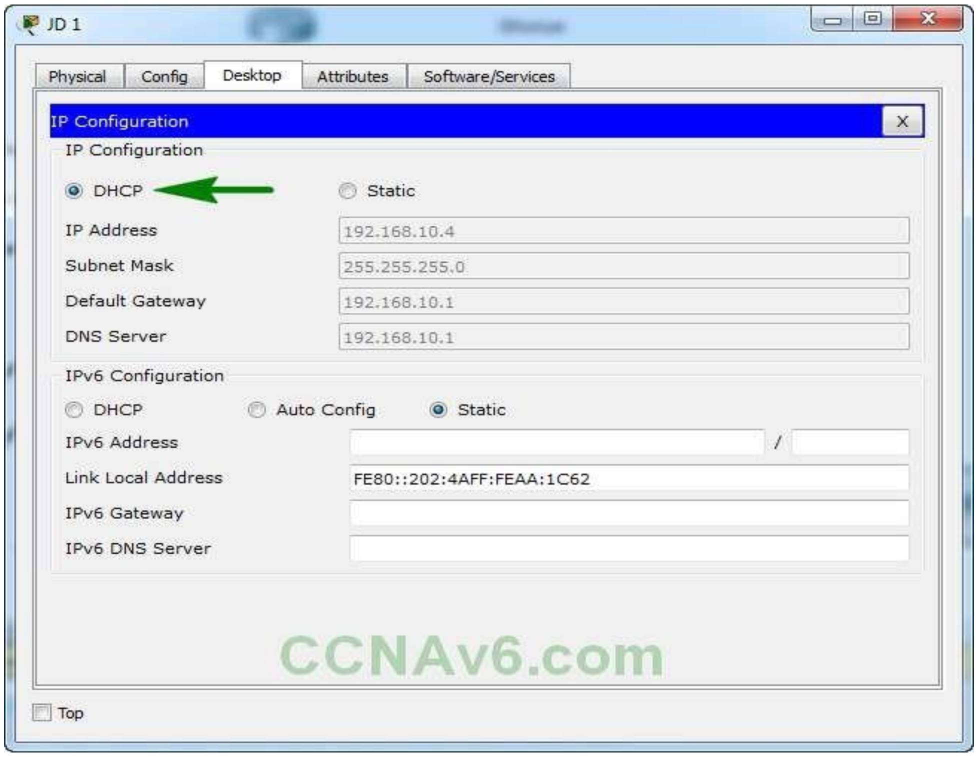
The commands are as shown below:

```
1. Router(config)#int fa 0/0

Router(config-if)#ip add 192.168.10.1 255.255.255.192

Router(config-if)#no shut

Router(config-if)#ip dhcp pool net1

Router(dhcp-config)#network 192.168.10.0 255.255.255.192

Router(dhcp-config)#dns-server 192.168.10.1

Router(dhcp-config)#default-router 192.168.10.1

Router(dhcp-config)#exit


2. Router(config)#int fa 1/0

Router(config-if)#ip add 192.168.10.129 255.255.255.192

Router(config-if)#no shut

Router(config-if)#ip dhcp pool net2

Router(dhcp-config)#network 192.168.10.128 255.255.255.192

Router(dhcp-config)#dns-server 192.168.10.129

Router(dhcp-config)#default-router 192.168.10.129
Router(dhcp-config)#exit
```

```
3. Router(config)#int fa 0/1

Router(config-if)#ip add 192.168.10.65 255.255.255.192

Router(config-if)#no shut

Router(config-if)#ip dhcp pool net3

Router(dhcp-config)#network 192.168.10.64 255.255.255.192

Router(dhcp-config)#dns-server 192.168.10.65

Router(dhcp-config)#default-router 192.168.10.65

Router(dhcp-config)#exit
```

Enabling DHCP on all pcs:

1. Click on a pc, for instance pc0
2. Go to “Desktop” tab
3. Choose DHCP in ip config



JD 1

Physical | Config | Desktop | Attributes | Software/Services

IP Configuration

IP Configuration

[✓ DHCP] Static

IP Address	192.168.10.4
Subnet Mask	255.255.255.0
Default Gateway	192.168.10.1
DNS Server	192.168.10.1

IPv6 Configuration

DHCP Auto Config [✓ Static]

IPv6 Address	/
Link Local Address	FE80::202:4AFF:FEAA:1C62
IPv6 Gateway	
IPv6 DNS Server	

Top

Conclusion :

Thus we have Setup a network and configure IP addressing, subnetting, Masking. (Eg. CISCO Packet Tracer**)**

Experiment No 3

Aim: Learn and Use basic networking commands in Linux (ping, tracert, nslookup, netstat, ARP, RARP, ip, ifconfig, dig, route) Using Command Prompt.

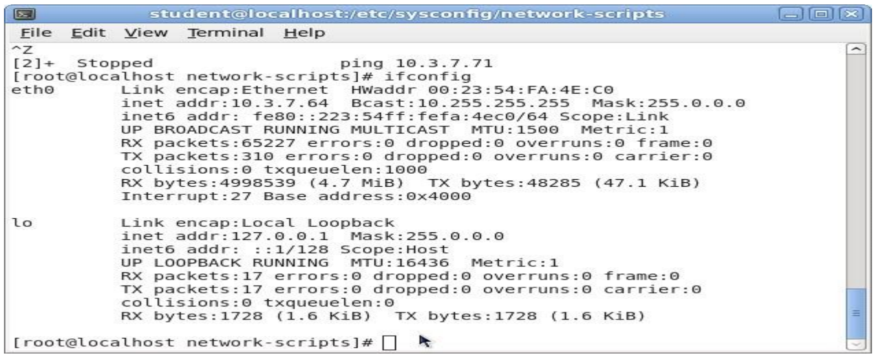
Theory:

1. ifconfig

ifconfig is used to configure the system's kernel-resident network interfaces. It is used at boot time to set up interfaces as necessary. After that, it is usually only needed when debugging or when system tuning is needed. If no arguments are given, ifconfig displays the status of the system's active interfaces. If a single interface argument is given, it displays the status of the given interface only.

Eg: ifconfig

Running ifconfig with no options will display the configuration of all active interfaces.


student@localhost:/etc/sysconfig/network-scripts
File Edit View Terminal Help

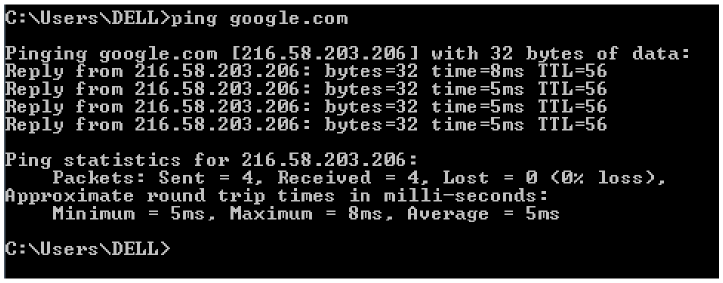
```
^Z
[2]+  Stopped                 ping 10.3.7.71
[root@localhost network-scripts]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:23:54:FA:4E:C0
          inet addr:10.3.7.64  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::223:54ff:fefa:4ec0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:65227 errors:0 dropped:0 overruns:0 frame:0
          TX packets:310 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4998539 (4.7 MiB)  TX bytes:48285 (47.1 KiB)
          Interrupt:27 Base address:0x4000

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:17 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1728 (1.6 KiB)  TX bytes:1728 (1.6 KiB)

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

2. ping

ping is a simple way to send network data to, and receive network data from, another computer on a network. It is frequently used to test, at the most basic level, whether another system is reachable over a network, and if so, how much time it takes for that data to be exchanged.

Eg: ping google.com

Ping the host google.com to see if it is alive.

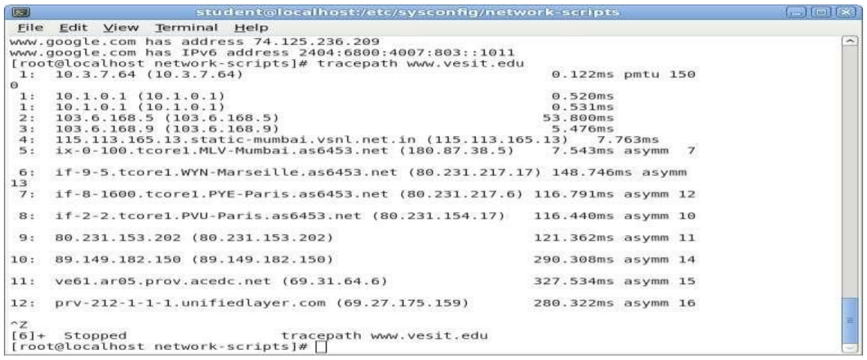


```
C:\Users\DELL>ping google.com

Pinging google.com [216.58.203.206] with 32 bytes of data:
Reply from 216.58.203.206: bytes=32 time=8ms TTL=56
Reply from 216.58.203.206: bytes=32 time=5ms TTL=56
Reply from 216.58.203.206: bytes=32 time=5ms TTL=56
Reply from 216.58.203.206: bytes=32 time=5ms TTL=56

Ping statistics for 216.58.203.206:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 5ms, Maximum = 8ms, Average = 5ms

C:\Users\DELL>
```

3. tracepath:

It traces the complete path to a networking host discovering the MTU along the path. It uses UDP port or some random port. It is similar to traceroute, only it does not require superuser privileges and has no fancy options.

Syntax: tracepath destination [port]


student@localhost:/etc/sysconfig/network-scripts
File Edit View Terminal Help

```
www.google.com has address 74.125.236.209
www.google.com has IPv6 address 2404:6800:4007:803::1011
[root@localhost network-scripts]# tracepath www.vesit.edu
 1:  10.3.7.64 (10.3.7.64)                               0.122ms pmtu 150
0
 1:  10.1.0.1 (10.1.0.1)                                 0.520ms
 1:  10.1.0.1 (10.1.0.1)                                 0.531ms
 2:  103.6.168.5 (103.6.168.5)                          53.800ms
 3:  103.6.168.9 (103.6.168.9)                           5.476ms
 4:  115.113.165.13.static-mumbai.vsnl.net.in (115.113.165.13)   7.763ms
 5:  ix-0-100.tcore1.MLV-Mumbai.as6453.net (180.87.38.5)    7.543ms asymm  7

 6:  if-9-5.tcore1.WYN-Marseille.as6453.net (80.231.217.17) 148.746ms asymm
13
 7:  if-8-1600.tcore1.PYE-Paris.as6453.net (80.231.217.6) 116.791ms asymm 12

 8:  if-2-2.tcore1.PVU-Paris.as6453.net (80.231.154.17)   116.440ms asymm 10

 9:  80.231.153.202 (80.231.153.202)                    121.362ms asymm 11

10:  89.149.182.150 (89.149.182.150)                    290.308ms asymm 14

11:  ve61.ar05.prov.acedc.net (69.31.64.6)              327.534ms asymm 15

12:  prv-212-1-1-1.unifiedlayer.com (69.27.175.159)      280.322ms asymm 16

^Z
[6]+  Stopped                 tracepath www.vesit.edu
[root@localhost network-scripts]# 
```

4. traceroute www.google.com:

traceroute prints the route that packets take to a network host. It is used to find network path from machine to server. The server name above is destination name or IP address.

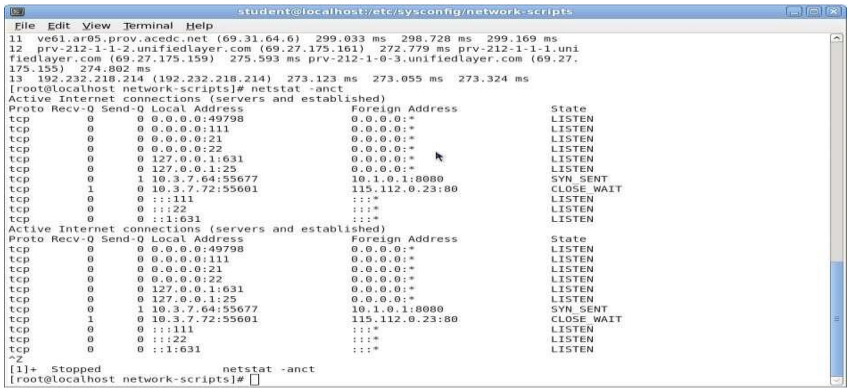
Syntax: traceroute <server name>

```
$ traceroute google.com

traceroute to google.com (74.125.236.132), 30 hops max, 60 byte packets

1  220.224.141.129 (220.224.141.129)  89.174 ms  89.094 ms  89.054 ms

2  115.255.239.65 (115.255.239.65)  109.037 ms  108.994 ms  108.963 ms

3  124.124.251.245 (124.124.251.245)  108.937 ms  121.322 ms  121.300 ms

4  * 115.255.239.45 (115.255.239.45)  113.754 ms  113.692 ms

5  72.14.212.118 (72.14.212.118)  123.585 ms  123.558 ms  123.527 ms

6  72.14.232.202 (72.14.232.202)  123.499 ms  123.475 ms  143.523 ms

7  216.239.48.179 (216.239.48.179)  143.503 ms  95.106 ms  95.026 ms

8  bom03s02-in-f4.1e100.net (74.125.236.132)  94.980 ms  104.989 ms  104.954 ms
```

5. netstat

The netstat command is used to print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships. It is used for finding problems in the network and to determine the amount of traffic on the network as a performance measurement. It Shows information about all active connections to the server, including the source and destination IP addressesand ports, if you have proper permissions.

Eg: netstat –an


student@localhost:/etc/sysconfig/network-scripts
File Edit View Terminal Help

```
11  ve61.ar05.prov.acedc.net (69.31.64.6)  299.033 ms  298.728 ms  299.169 ms
12  prv-212-1-1-2.unifiedlayer.com (69.27.175.161)  272.779 ms prv-212-1-1-1.uni
fiedlayer.com (69.27.175.159)  275.593 ms prv-212-1-0-3.unifiedlayer.com (69.27.
175.155)  274.802 ms
13  192.232.218.214 (192.232.218.214)  273.123 ms  273.055 ms  273.324 ms
[root@localhost network-scripts]# netstat -anct
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 0.0.0.0:49798               0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:21                  0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:631               0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:25                0.0.0.0:*                   LISTEN
tcp        0      1 10.3.7.64:55677             10.1.0.1:8080               SYN_SENT
tcp        1      0 10.3.7.72:55601             115.112.0.23:80             CLOSE_WAIT
tcp        0      0 :::111                      :::*                        LISTEN
tcp        0      0 :::22                       :::*                        LISTEN
tcp        0      0 ::1:631                     :::*                        LISTEN
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 0.0.0.0:49798               0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:21                  0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:631               0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:25                0.0.0.0:*                   LISTEN
tcp        0      1 10.3.7.64:55677             10.1.0.1:8080               SYN_SENT
tcp        1      0 10.3.7.72:55601             115.112.0.23:80             CLOSE_WAIT
tcp        0      0 :::111                      :::*                        LISTEN
tcp        0      0 :::22                       :::*                        LISTEN
tcp        0      0 ::1:631                     :::*                        LISTEN
^Z
[1]+  Stopped                 netstat -anct
[root@localhost network-scripts]# 
```

Conclusion :
Thus we have Learnt and Use basic networking commands in Linux (ping, tracert, nslookup, netstat, ARP, RARP, ip, ifconfig, dig, route) Using Command Prompt.

Experiment No 4

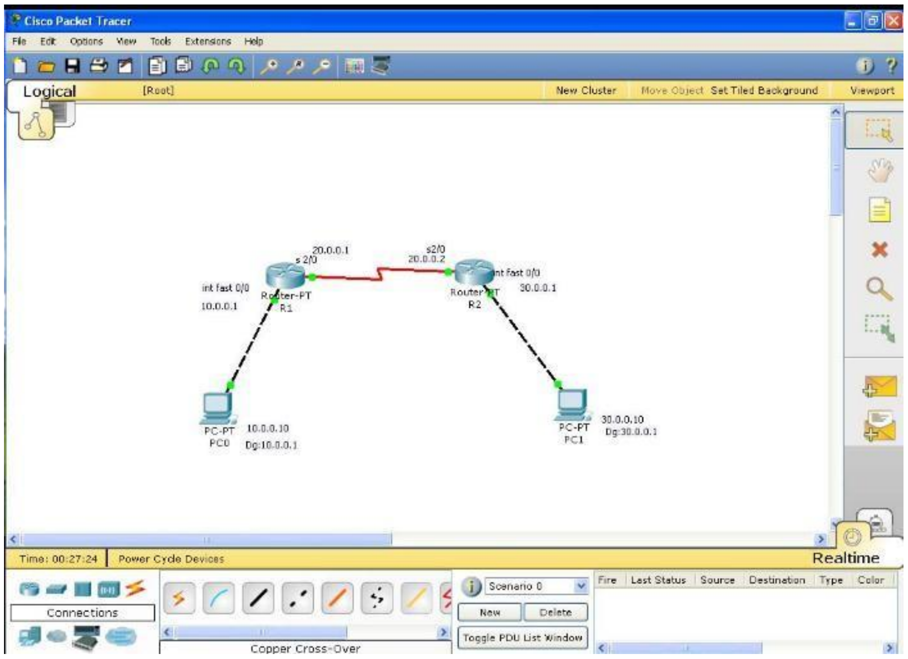
Aim: Build a simple network topology and configure it for static routing protocol using packet tracer.

Theory:

Static routing is a form of routing that occurs when a router uses a manually-configured routing entry, rather than information from a dynamic routing traffic.

Procedure:

Step 1:First Create a topology shown in below figure:



Step 2: Configure ip address to routers go to global configuration mode in R1 and R2 configure connected interfaces

In Router 1

Interface Fastethernet0/0 in global configuration mode

```
R1(config)#interface fastethernet 0/0
R1(config-if)#ip address 10.0.0.1 255.0.0.0
R1(config-if)#no shutdown
R1(config-if)#exit
```

Interface Serial 2/0

```
R1(config)#interface serial 2/0
```

```
R1(config-if)#ip address 20.0.0.1 255.0.0.0
R1(config-if)#clock rate 64000
R1(config-if)#encapsulation ppp
R1(config-if)#no shutdown
R1(config-if)#exit
```

In Router 2

Interface Fastethernet 0/0

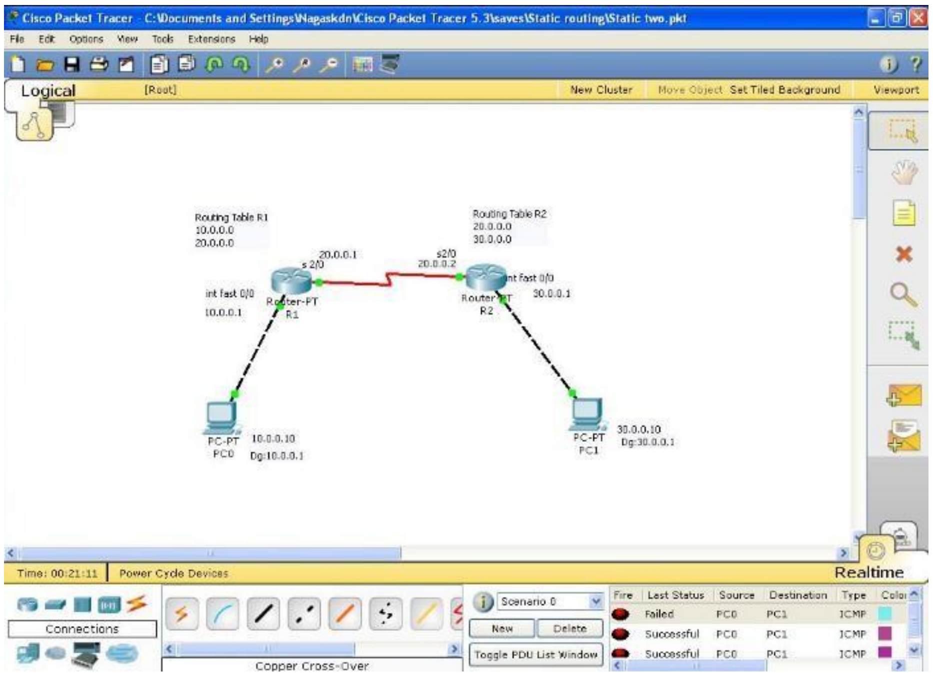
```
R2(config)#interface fastethernet 0/0
R2(config-if)#ip address 30.0.0.1 255.0.0.0
R2(config-if)#no shutdown
R2(config-if)#exit
```

Interface Serial 2/0

```
R2(config)#interface serial 2/0
R2(config-if)#ip address 20.0.0.2 255.0.0.0
R2(config-if)#encapsulation ppp
R2(config-if)#no shutdown
R2(config-if)#exit
```

Step 3 : Assign ip address for both Pc's with appropriate ip and subnetmask and default gateway.

Step 4: Now configure both router with static route.



Fire	Last Status	Source	Destination	Type	Color
●	Failed	PC0	PC1	ICMP	🟦
●	Successful	PC0	PC1	ICMP	🟪
●	Successful	PC0	PC1	ICMP	🟪

By default, Routers Know only directed connected networks here Router 1 know only 10.0.0.0 and 20.0.0.0 it doesn't know the 30.0.0.0 like this R2 doesn't know about 10.0.0.0.So We are going to add Static route to this both router.

R1(config)#ip route Destination Network| Destination N/W SubnetMask |Next Hop Address

In Router R1,give this command, In this case Destination is 30.0.0.0 and its subnet mask is 255.0.0.0 next hop address is 20.0.0.2

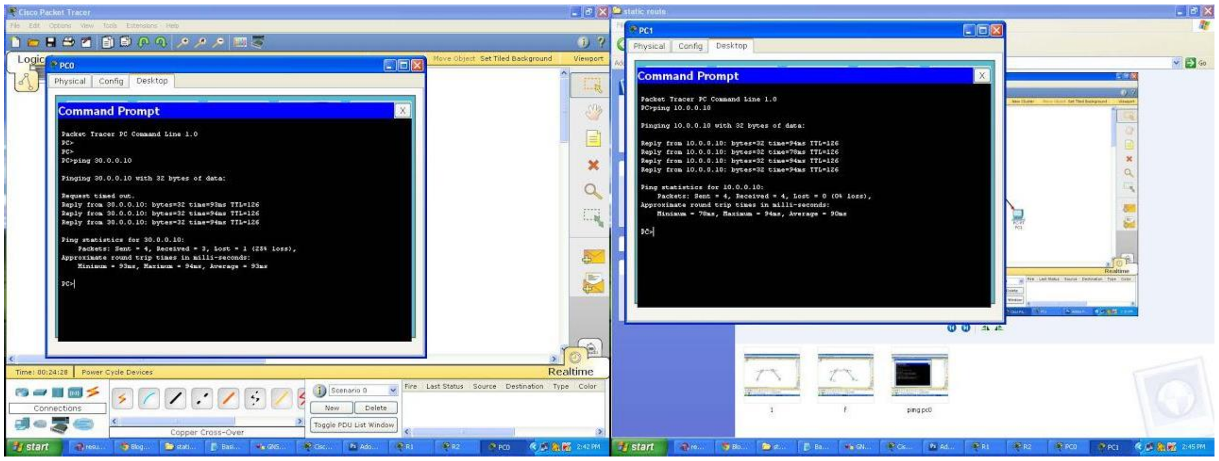
R1(config)#ip route 30.0.0.0 255.0.0.0 20.0.0.2

In Router R2

R2(config)#ip route 10.0.0.0 255.0.0.0 20.0.0.1

Now both routers know all networks, check by ping ip address of host

Step 5:Double click PC move to desktop then command prompt give the command ping 30.0.0.10 in PC 0 you will get reply from 30.0.0.10 like this:



```
Packet Tracer PC Command Line 1.0
PC>
PC>
PC>ping 30.0.0.10

Pinging 30.0.0.10 with 32 bytes of data:

Request timed out.
Reply from 30.0.0.10: bytes=32 time=93ms TTL=126
Reply from 30.0.0.10: bytes=32 time=94ms TTL=126
Reply from 30.0.0.10: bytes=32 time=94ms TTL=126

Ping statistics for 30.0.0.10:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 93ms, Maximum = 94ms, Average = 93ms

PC>
```

```
Packet Tracer PC Command Line 1.0
PC>ping 10.0.0.10

Pinging 10.0.0.10 with 32 bytes of data:

Reply from 10.0.0.10: bytes=32 time=94ms TTL=126
Reply from 10.0.0.10: bytes=32 time=78ms TTL=126
Reply from 10.0.0.10: bytes=32 time=94ms TTL=126
Reply from 10.0.0.10: bytes=32 time=94ms TTL=126

Ping statistics for 10.0.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 78ms, Maximum = 94ms, Average = 90ms

PC>
```

Conclusion :
Thus we have Built a simple network topology and configure it for static routing protocol using packet tracer.

Experiment No 5

Aim: Use Wireshark to understand the operation of TCP/IP layers.

- **Ethernet Layer : Frame header, Frame size etc.**
- **Data Link Layer : MAC address, ARP (IP and MAC address binding)**
- **Network Layer : IP Packet (header, fragmentation), ICMP (Query and Echo)**
- **Transport Layer: TCP Ports, TCP handshake segments etc.**
- **Application Layer: DHCP, FTP, HTTP header formats**

Theory:

Wireshark is a free application that allows you to capture and view the data traveling back and forth on your network, providing the ability to drill down and read the contents of each packet – filtered to meet your specific needs. It is commonly utilized to troubleshoot network problems as well as to develop and test software. Originally known as Ethereal, Wireshark features a user-friendly interface that can display data from hundreds of different protocols on all major network types. These data packets can be viewed in real-time or analyzed offline, with dozens of capture/trace file formats supported including CAP and ERF. Integrated decryption tools allow you to view encrypted packets for several popular protocols such as WEP and WPA/WPA2.

Procedure:

Wireshark Installation steps:

Step 1: Enter admin mode by following command. If not entered in admin mode then packets will not get captured in wireshark.

sudo su

Step 2: Command to install wireshark on ubuntu

sudo apt-get install wireshark

Step 3: Double click on the wireshark icon. We get an open window as given below.

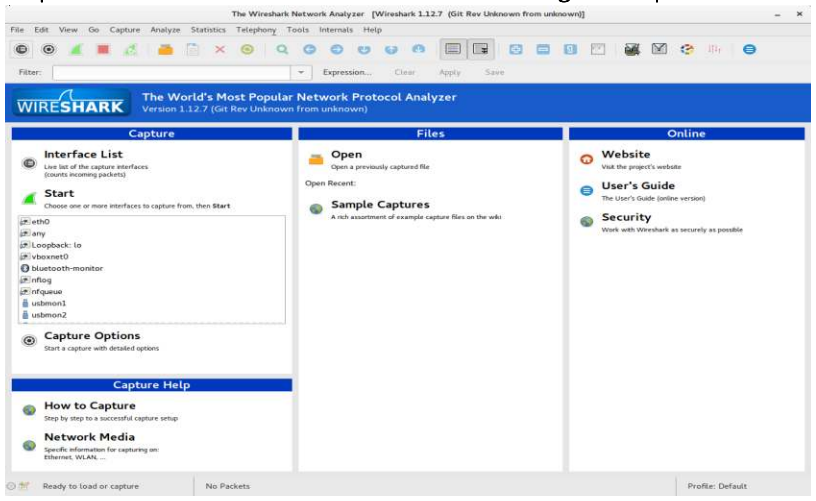
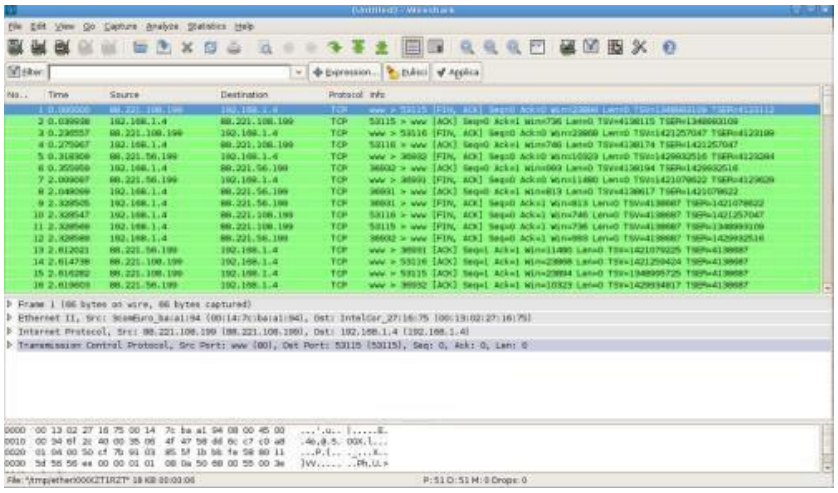
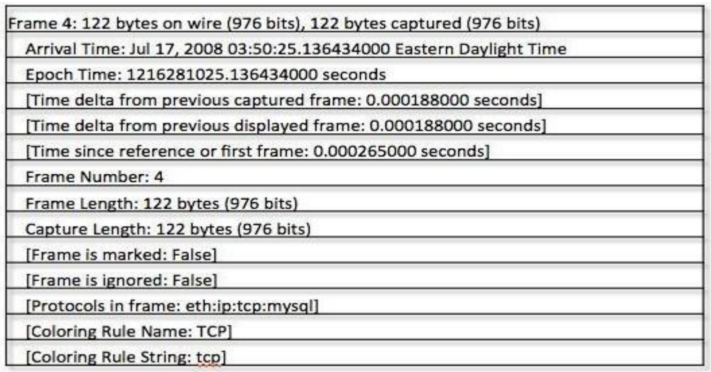

The Wireshark Network Analyzer [Wireshark 1.12.7 (Git Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
WIRESHARK
The World's Most Popular Network Protocol Analyzer
Version 1.12.7 (Git Rev Unknown from unknown)
Capture
Interface List
Live list of the capture interfaces (counts incoming packets)
Start
Choose one or more interfaces to capture from, then Start
eth0
any
Loopback: lo
vboxnet0
bluetooth-monitor
nflog
nfqueue
usbmon1
usbmon2
Capture Options
Start a capture with detailed options
Capture Help
How to Capture
Step by step to a successful capture setup
Network Media
Specific information for capturing on: Ethernet, WLAN, ...
Files
Open
Open a previously captured file
Open Recent:
Sample Captures
A rich assortment of example capture files on the wiki
Online
Website
Visit the project's website
User's Guide
The User's Guide (online version)
Security
Work with Wireshark as securely as possible
Ready to load or capture
No Packets
Profile: Default

Figure 1.1: Wireshark initial showing interfaces (sudo mode)



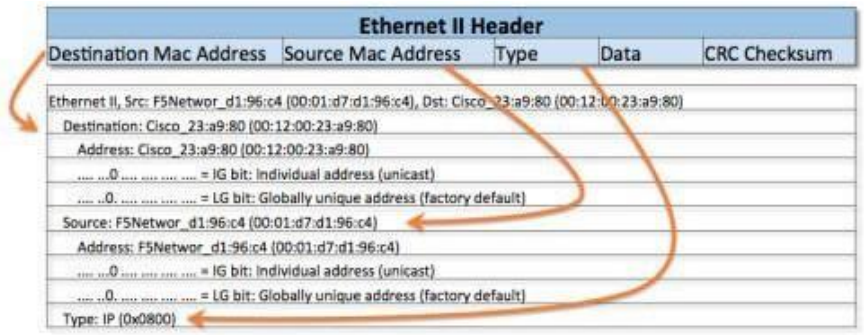
```
No..  Time        Source           Destination      Protocol Info
   1 0.000000    88.221.108.199   192.168.1.4      TCP      www > 53115 [FIN, ACK] Seq=0 Ack=0 Win=23884 Len=0 TSV=1348993109 TSER=4123112
   2 0.039938    192.168.1.4      88.221.108.199   TCP      53115 > www [ACK] Seq=0 Ack=1 Win=736 Len=0 TSV=4138115 TSER=1348993109
   3 0.236557    88.221.108.199   192.168.1.4      TCP      www > 53116 [FIN, ACK] Seq=0 Ack=0 Win=23888 Len=0 TSV=1421257047 TSER=4123189
   4 0.275067    192.168.1.4      88.221.108.199   TCP      53116 > www [ACK] Seq=0 Ack=1 Win=746 Len=0 TSV=4138174 TSER=1421257047
   5 0.318309    88.221.56.199    192.168.1.4      TCP      www > 36932 [FIN, ACK] Seq=0 Ack=0 Win=10323 Len=0 TSV=1429932516 TSER=4123284
   6 0.355959    192.168.1.4      88.221.56.199    TCP      36932 > www [ACK] Seq=0 Ack=1 Win=993 Len=0 TSV=4138194 TSER=1429932516
   7 2.009087    88.221.56.199    192.168.1.4      TCP      www > 36931 [FIN, ACK] Seq=0 Ack=0 Win=11480 Len=0 TSV=1421078622 TSER=4129629
   8 2.049099    192.168.1.4      88.221.56.199    TCP      36931 > www [ACK] Seq=0 Ack=1 Win=813 Len=0 TSV=4138617 TSER=1421078622
   9 2.328505    192.168.1.4      88.221.56.199    TCP      36931 > www [FIN, ACK] Seq=0 Ack=1 Win=813 Len=0 TSV=4138687 TSER=1421078622
  10 2.328547    192.168.1.4      88.221.108.199   TCP      53116 > www [FIN, ACK] Seq=0 Ack=1 Win=746 Len=0 TSV=4138687 TSER=1421257047
  11 2.328569    192.168.1.4      88.221.108.199   TCP      53115 > www [FIN, ACK] Seq=0 Ack=1 Win=736 Len=0 TSV=4138687 TSER=1348993109
  12 2.328589    192.168.1.4      88.221.56.199    TCP      36932 > www [FIN, ACK] Seq=0 Ack=1 Win=993 Len=0 TSV=4138687 TSER=1429932516
  13 2.612021    88.221.56.199    192.168.1.4      TCP      www > 36931 [ACK] Seq=1 Ack=1 Win=11480 Len=0 TSV=1421079225 TSER=4138687
  14 2.614738    88.221.108.199   192.168.1.4      TCP      www > 53116 [ACK] Seq=1 Ack=1 Win=23888 Len=0 TSV=1421259424 TSER=4138687
  15 2.616282    88.221.108.199   192.168.1.4      TCP      www > 53115 [ACK] Seq=1 Ack=1 Win=23884 Len=0 TSV=1348995725 TSER=4138687
  16 2.619609    88.221.56.199    192.168.1.4      TCP      www > 36932 [ACK] Seq=1 Ack=1 Win=10323 Len=0 TSV=1429934817 TSER=4138687

▷ Frame 1 (66 bytes on wire, 66 bytes captured)
▷ Ethernet II, Src: ScomEuro_ba:a1:94 (00:14:7c:ba:a1:94), Dst: IntelCor_27:16:75 (00:13:02:27:16:75)
▷ Internet Protocol, Src: 88.221.108.199 (88.221.108.199), Dst: 192.168.1.4 (192.168.1.4)
▷ Transmission Control Protocol, Src Port: www (80), Dst Port: 53115 (53115), Seq: 0, Ack: 0, Len: 0

0000  00 13 02 27 16 75 00 14  7c ba a1 94 08 00 45 00   ...'.u.. |.....E.
0010  00 34 6f 2c 40 00 35 06  4f 47 58 dd 6c c7 c0 a8   .4o,@.5. OGX.l...
0020  01 04 00 50 cf 7b 91 03  85 5f 1b bb fe 58 80 11   ...P.{.. ._...X..
0030  5d 56 56 ea 00 00 01 01  08 0a 50 68 00 55 00 3e   ]VV..... ..Ph.U.>

File: "/tmp/etherXXXXZT1RZT" 18 KB 00:00:06        P: 51 D: 51 M: 0 Drops: 0
```

Figure 1.2. An example of a Wireshark capture.



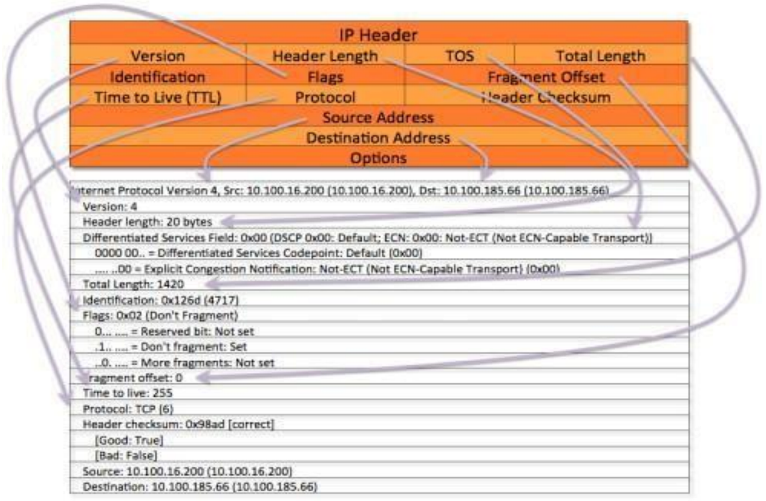
Frame 4: 122 bytes on wire (976 bits), 122 bytes captured (976 bits)
Arrival Time: Jul 17, 2008 03:50:25.136434000 Eastern Daylight Time
Epoch Time: 1216281025.136434000 seconds
[Time delta from previous captured frame: 0.000188000 seconds]
[Time delta from previous displayed frame: 0.000188000 seconds]
[Time since reference or first frame: 0.000265000 seconds]
Frame Number: 4
Frame Length: 122 bytes (976 bits)
Capture Length: 122 bytes (976 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ip:tcp:mysql]
[Coloring Rule Name: TCP]
[Coloring Rule String: tcp]

Figure 2. The summary before the protocols in a Wireshark packet. Information about the packet characteristic.



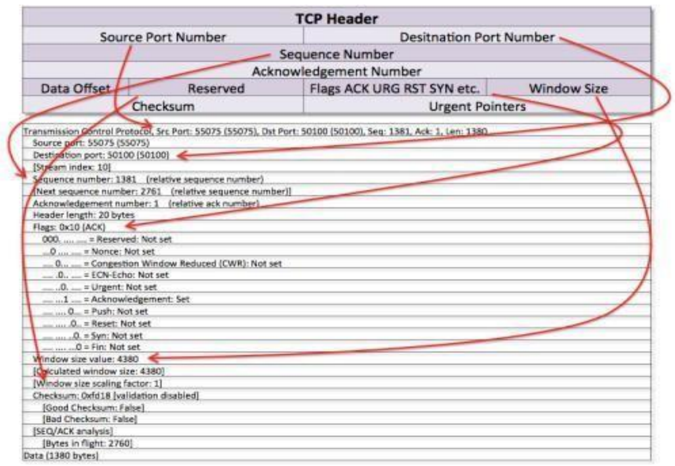
```
Ethernet II, Src: F5Networ_d1:96:c4 (00:01:d7:d1:96:c4), Dst: Cisco_23:a9:80 (00:12:00:23:a9:80)
  Destination: Cisco_23:a9:80 (00:12:00:23:a9:80)
    Address: Cisco_23:a9:80 (00:12:00:23:a9:80)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
  Source: F5Networ_d1:96:c4 (00:01:d7:d1:96:c4)
    Address: F5Networ_d1:96:c4 (00:01:d7:d1:96:c4)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
  Type: IP (0x0800)
```

Figure 3. Ethernet II (Layer 2) header along with the Wireshark



```
Internet Protocol Version 4, Src: 10.100.16.200 (10.100.16.200), Dst: 10.100.185.66 (10.100.185.66)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    0000 00.. = Differentiated Services Codepoint: Default (0x00)
    .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport) (0x00)
  Total Length: 1420
  Identification: 0x126d (4717)
  Flags: 0x02 (Don't Fragment)
    0... .... = Reserved bit: Not set
    .1.. .... = Don't fragment: Set
    ..0. .... = More fragments: Not set
  Fragment offset: 0
  Time to live: 255
  Protocol: TCP (6)
  Header checksum: 0x98ad [correct]
    [Good: True]
    [Bad: False]
  Source: 10.100.16.200 (10.100.16.200)
  Destination: 10.100.185.66 (10.100.185.66)
```

Figure 4. IP Header (Layer-3)

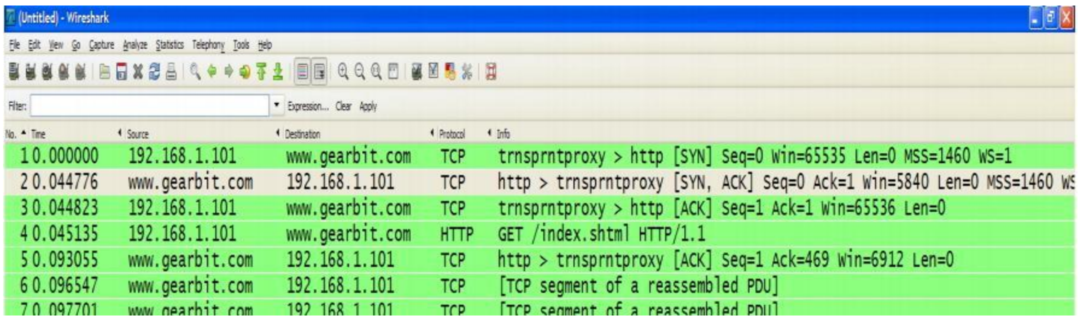


```
Transmission Control Protocol, Src Port: 55075 (55075), Dst Port: 50100 (50100), Seq: 1381, Ack: 1, Len: 1380
  Source port: 55075 (55075)
  Destination port: 50100 (50100)
  [Stream index: 10]
  Sequence number: 1381    (relative sequence number)
  [Next sequence number: 2761    (relative sequence number)]
  Acknowledgement number: 1    (relative ack number)
  Header length: 20 bytes
  Flags: 0x10 (ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Nonce: Not set
    .... 0... .... = Congestion Window Reduced (CWR): Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...1 .... = Acknowledgement: Set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
  Window size value: 4380
  [Calculated window size: 4380]
  [Window size scaling factor: 1]
  Checksum: 0xfd18 [validation disabled]
    [Good Checksum: False]
    [Bad Checksum: False]
  [SEQ/ACK analysis]
    [Bytes in flight: 2760]
Data (1380 bytes)
```

Figure 5. TCP headers.

TCP Three-way Handshake

The delta value between frames 1 and 2 can be used as a TCP transport connect baseline value. Other important information gathered from this handshake: • Window Size • SACK • Maximum Segment Size • Window Scale Option value


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply

```
No. Time      Source          Destination     Protocol Info
 1 0.000000   192.168.1.101   www.gearbit.com TCP   trnsprntproxy > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=1
 2 0.044776   www.gearbit.com 192.168.1.101   TCP   http > trnsprntproxy [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 WS
 3 0.044823   192.168.1.101   www.gearbit.com TCP   trnsprntproxy > http [ACK] Seq=1 Ack=1 Win=65536 Len=0
 4 0.045135   192.168.1.101   www.gearbit.com HTTP  GET /index.shtml HTTP/1.1
 5 0.093055   www.gearbit.com 192.168.1.101   TCP   http > trnsprntproxy [ACK] Seq=1 Ack=469 Win=6912 Len=0
 6 0.096547   www.gearbit.com 192.168.1.101   TCP   [TCP segment of a reassembled PDU]
 7 0.097701   www.gearbit.com 192.168.1.101   TCP   [TCP segment of a reassembled PDU]
```

Conclusion :

Thus we have Used Wireshark to understand the operation of TCP/IP layers.

Experiment No 6

Aim: CRC / Hamming Code Implementation in java.

Theory:

Cyclic Redundancy Check:

CRC or Cyclic Redundancy Check is a method of detecting accidental changes/errors in communication channel.

CRC uses Generator Polynomial which is available on both sender and receiver side. An example generator polynomial is of the form like $x^3 + x + 1$. This generator polynomial represents key 1011. Another example is $x^2 + 1$ that represents key 101.

Sender Side (Generation of Encoded Data from Data and Generator Polynomial (or Key)):

Step 1: The binary data is first augmented by adding k-1 zeros in the end of the data

Step 2: Use modulo-2 binary division to divide binary data by the key and store remainder of division.

Step 3: Append the remainder at the end of the data to form the encoded data and send the same.

Receiver Side (Check if there are errors introduced in transmission):

Step 1: Perform modulo-2 division again and if remainder is 0, then there are no errors.

Program:

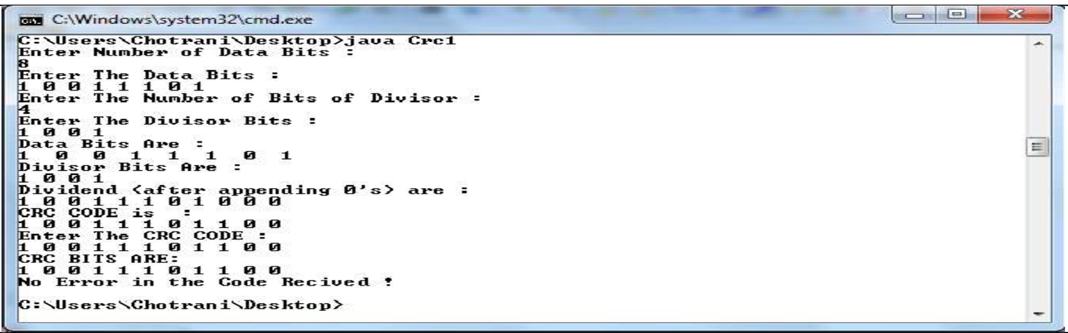
```
import java.util.*;
class Crc1
{
  public static void main(String[] args)
  {
    int[] data;
    int[] div;
    int[] divisor;
    int[] rem;
    int[] crc;
    int data_bits,divisor_bits,tot_bits;
    Scanner s = new Scanner(System.in);
    //Sender Side
    System.out.println("Enter Number of Data Bits
:");
    data_bits= s.nextInt();
    data= new int[data_bits];
    System.out.println("Enter The Data Bits : ");
    for(int i=0;i<data_bits;i++)
```

```
for(int i=0;i<div.length;i++)
      crc[i] = (div[i] ^ rem[i]);
    System.out.println("CRC CODE is  :");
    for(int i=0;i<crc.length;i++)
      System.out.print(crc[i] + " ");

    //Reciver Side
    System.out.println("\nEnter The CRC CODE :");
    for(int i=0;i<crc.length;i++)
    crc[i] = s.nextInt();
    System.out.println("CRC BITS ARE:");
    for(int i =0; i<crc.length ; i++)
      System.out.print(crc[i] + " ");
    for(int j=0; j<crc.length; j++)
      rem[j] = crc[j];
    rem = divide(crc,divisor,rem);
    for(int i=0;i<rem.length; i++)
    {
      if(rem[i]!=0)
```

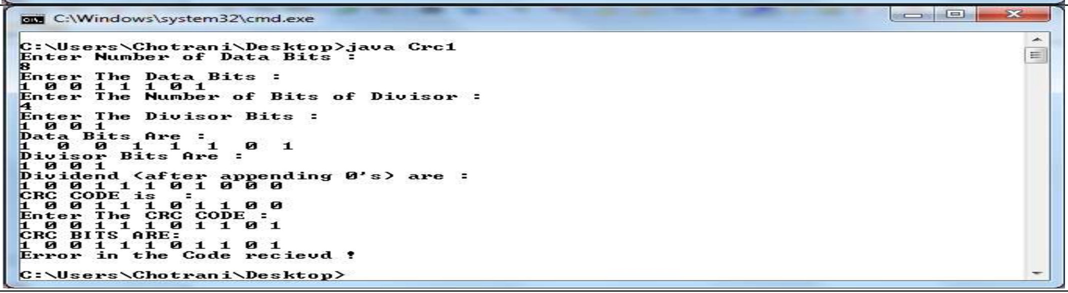
```
            data[i] = s.nextInt();
System.out.println("Enter The Number of Bits of Divisor");
        divisor_bits= s.nextInt();
        divisor = new int[divisor_bits];
        System.out.println("Enter The Divisor Bits : ");
        for(int i=0; i<divisor_bits ; i++)
            divisor[i] = s.nextInt();
        System.out.println("Data Bits Are :");
        for(int i=0; i<data_bits ; i++)
            System.out.print(data[i] + " ");
        System.out.println("\nDivisor Bits Are :");
        for(int i=0; i<divisor_bits; i++)
            System.out.print(divisor[i] + " ");
        tot_bits = data_bits+divisor_bits-1;
        div = new int[tot_bits];
        rem = new int[tot_bits];
        crc = new int[tot_bits];
        for(int i=0;i<data.length;i++)
            div[i] = data[i];
System.out.println("\nDividend (after appending 0's) are”);
        for(int     i=0;i<div.length;i++)
            System.out.print(div[i]+" ");
        System.out.print("\n");
        for(int j=0;j<div.length ;j++)
            rem[j] = div[j];
        rem = divide(div,divisor,rem);
```

```
            {
System.out.println("\nError in the Code recievd!");
                break;
            }
            if(i==rem.length-1)
System.out.print("\nNo Error in the Code recived!");
        }
    }
    static int[] divide(int div[], int divisor[], int rem[])
    {
        int cur=0;
        while(true)
        {
            for(int i=0; i<divisor.length;i++)
                rem[cur+i] = (rem[cur+i] ^ divisor[i] );
            while(rem[cur] == 0 && cur!=rem.length-1)
                cur++;
            if((rem.length-cur)<divisor.length)
                break;
        }
        return rem;
    }
}
```

Output:



```
C:\Users\Chotrani\Desktop>java Crc1
Enter Number of Data Bits :
8
Enter The Data Bits :
1 0 0 1 1 1 0 1
Enter The Number of Bits of Divisor :
4
Enter The Divisor Bits :
1 0 0 1
Data Bits Are :
1  0  0  1  1  1  0  1
Divisor Bits Are :
1 0 0 1
Dividend (after appending 0's) are :
1 0 0 1 1 1 0 1 0 0 0
CRC CODE is   :
1 0 0 1 1 1 0 1 1 0 0
Enter The CRC CODE :
1 0 0 1 1 1 0 1 1 0 0
CRC BITS ARE:
1 0 0 1 1 1 0 1 1 0 0
No Error in the Code Recived !

C:\Users\Chotrani\Desktop>
```



```
C:\Users\Chotrani\Desktop>java Crc1
Enter Number of Data Bits :
8
Enter The Data Bits :
1 0 0 1 1 1 0 1
Enter The Number of Bits of Divisor :
4
Enter The Divisor Bits :
1 0 0 1
Data Bits Are :
1  0  0  1  1  1  0  1
Divisor Bits Are :
1 0 0 1
Dividend (after appending 0's) are :
1 0 0 1 1 1 0 1 0 0 0
CRC CODE is   :
1 0 0 1 1 1 0 1 1 0 0
Enter The CRC CODE :
1 0 0 1 1 1 0 1 1 0 1
CRC BITS ARE:
1 0 0 1 1 1 0 1 1 0 1
Error in the Code recievd !

C:\Users\Chotrani\Desktop>
```

Hamming Code:

Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is moved or stored from the sender to the receiver. It is technique developed by R.W. Hamming for error correction.

Program:

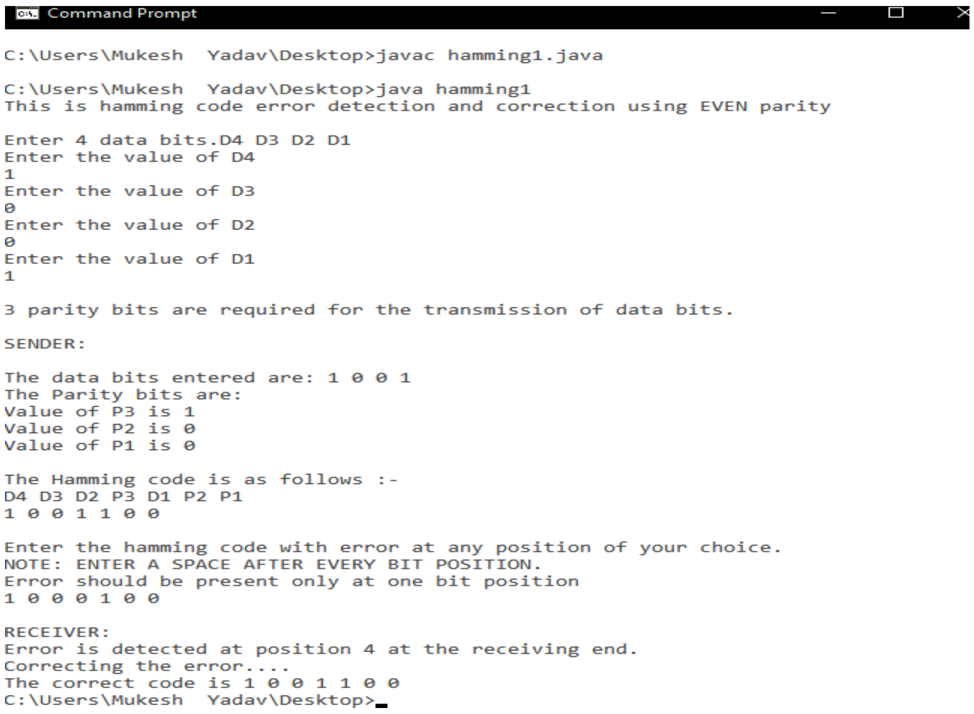
```
import java.util.*;
class hamming
{
  public static void main(String args[])throws
Exception
  {
    Scanner read=new Scanner(System.in);
    int i,value;
    int a[] = new int[7];
    int r[] = new int[7];
    int v[] = new int[3];
    a[0] = -1;
    a[1] = -1;
    a[3] = -1;
    value = 0;
    System.out.println("Enter data in bits:");
    for(i=6;i>=0;i--)
      if(i!=3 && i!=1 && i!=0)
        a[i]=read.nextInt();
    int c;

    //calculating parity 1
    c=0;
    if(a[2]==1)
      c++;
    if(a[4]==1)
      c++;
    if(a[6]==1)
      c++;
    if(c%2==0)
      a[0]=0;
    else
      a[0]=1;

    //calculating parity 2
    c=0;
    if(a[2]==1)
      c++;
    if(a[5]==1)
      c++;
```

```
System.out.println("The encoded message is:");
    for(i=6;i>=0;i--)
      System.out.println(a[i]);
    System.out.println();
    System.out.println("Received message:");
    for(i=6;i>=0;i--)
      r[i]=read.nextInt();
    c=0;
    if(r[0]==1)
      c++;
    if(r[2]==1)
      c++;
    if(r[4]==1)
      c++;
    if(r[6]==1)
      c++;
    if(c%2==0)
      v[0]=0;
    else
      v[0]=1;

    c=0;
    if(r[1]==1)
      c++;
    if(r[2]==1)
      c++;
    if(r[5]==1)
      c++;
    if(r[6]==1)
      c++;
    if(c%2==0)
      v[1]=0;
    else
      v[1]=1;
      c=0;
    if(r[3]==1)
      c++;
    if(r[4]==1)
      c++;
    if(r[5]==1)
```

```
        if(a[6]==1)
           c++;
        if(c%2==0)
           a[1]=0;
        else
           a[1]=1;
//calculating parity 3
        c=0;
        if(a[4]==1)
           c++;
        if(a[5]==1)
           c++;
        if(a[6]==1)
           c++;
        if(c%2==0)
           a[3]=0;
        else
           a[3]=1;
```

```
           c++;
        if(r[6]==1)
           c++;
        if(c%2==0)
           v[2]=0;
        else
           v[2]=1;
        value=v[0]+v[1]*2+v[2]*4;
        if(value==0)
           System.out.println("NO ERROR!!");
        else
        {
System.out.println("Error detected at: " +value+
"th position");
           if(r[value-1]==0)
              r[value-1]=1;
           else
              r[value-1]=0;
           System.out.println("Corrected message:");
           for(i=6;i>=0;i--)
              System.out.println(r[i]);
        }
     }
}
```

Output:



```
C:\Users\Mukesh  Yadav\Desktop>javac hamming1.java

C:\Users\Mukesh  Yadav\Desktop>java hamming1
This is hamming code error detection and correction using EVEN parity

Enter 4 data bits.D4 D3 D2 D1
Enter the value of D4
1
Enter the value of D3
0
Enter the value of D2
0
Enter the value of D1
1

3 parity bits are required for the transmission of data bits.

SENDER:

The data bits entered are: 1 0 0 1
The Parity bits are:
Value of P3 is 1
Value of P2 is 0
Value of P1 is 0

The Hamming code is as follows :-
D4 D3 D2 P3 D1 P2 P1
1 0 0 1 1 0 0

Enter the hamming code with error at any position of your choice.
NOTE: ENTER A SPACE AFTER EVERY BIT POSITION.
Error should be present only at one bit position
1 0 0 0 1 0 0

RECEIVER:
Error is detected at position 4 at the receiving end.
Correcting the error....
The correct code is 1 0 0 1 1 0 0
C:\Users\Mukesh  Yadav\Desktop>
```

Conclusion ; Thus we have implemented CRC / Hamming Code in java

Experiment No 7

Aim: Use simulator (Eg. Netsim) to understand functioning of ALOHA, CSMA/CD.

Theory:

(Using Netsim)

ALOHA provides a wireless data network. It is a multiple access protocol (this protocol is for allocating a multiple access channel). There are two main versions of ALOHA: pure and slotted. They differ with respect to whether or not time is divided up into discrete slots into which all frames must fit.

Slotted ALOHA

- In slotted Aloha, time is divided up into discrete intervals, each interval corresponding to one frame. In Slotted ALOHA, a computer is required to wait for the beginning of the next slot in order to send the next packet. The probability of no other traffic being initiated during the entire vulnerable period is given by e^{-G}which leads to $S = Ge^{-G}$.
- Where, S (frames per frame time) is the mean of the Poisson distribution with which frames are being generated. For reasonable throughput S should lie between 0 and 1. G is the mean of the Poisson distribution followed by the transmission attempts per frame time, old and new combined. Old frames mean those frames that have previously suffered collisions.
- It is easy to note that Slotted ALOHA peaks at G=1, with a throughput of or about 0.368. It means that if the system is operating at G=1, the probability of an empty an slot is 0.368

Calculations used in NetSim to obtain the plot between S and G:
Using NetSim, the attempts per packet time (G) can be calculated as follows:

$$G = \frac{TA * PT}{ST * 1000}$$

Where, G = Attempts per packet time, TA = Total Attempt, PT = Packet time (in milli seconds), ST = Simulation time (in seconds).
The throughput (mbps) per frame time can be obtained as follows:

$$S = \frac{Throughp(Mbps) * 1000 * PT}{PS * 8}$$

Where, S = Throughput per frame time, PT = Packet time (in milli seconds), PS = Packet size (in bytes)

Calculations for the packet time:

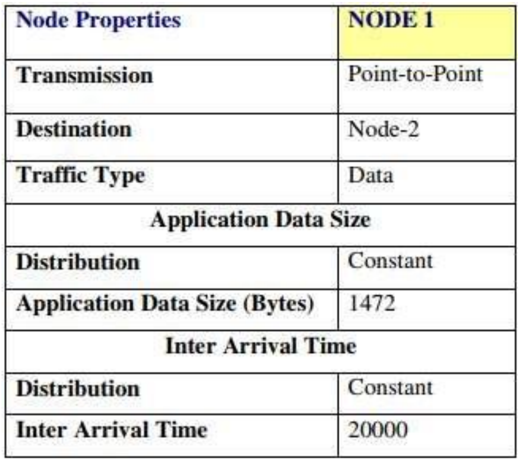
$$PT = \frac{Packetbits}{Bandwidth(Mbps)}$$

In the following experiment we have taken packet size=1498 bytes (1472 + 26 Overheads) Bandwidth is 10 Mbps and hence, packet time comes as 1.198 ms.

Steps:
1. Create Scenario: “Help àNetSim Help àRunning Simulation via GUIàLegacy Networks àCreate Scenario”.
2. Obtain the values of Throughput and Total Attempts from the statistics of NetSim simulation for various numbers of traffic generators.

Sample Inputs:

Input for Sample 1: Node 1 transmits data to Node 2.

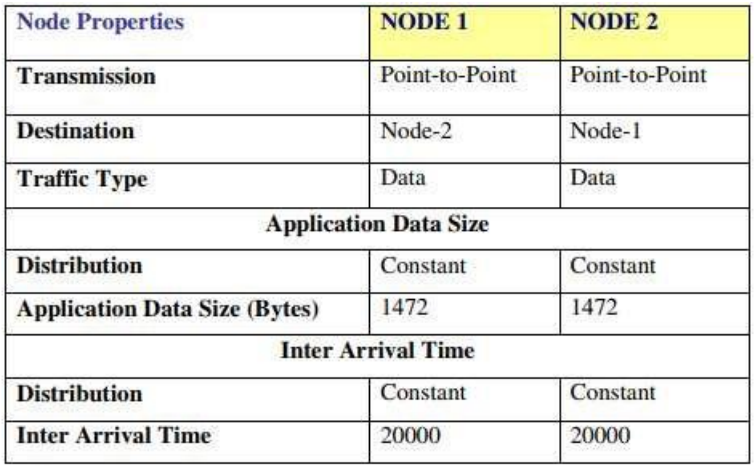


Node Properties	NODE 1
Transmission	Point-to-Point
Destination	Node-2
Traffic Type	Data
Application Data Size	
Distribution	Constant
Application Data Size (Bytes)	1472
Inter Arrival Time	
Distribution	Constant
Inter Arrival Time	20000

Simulation Time - 10 Seconds

(Note: The Simulation Time can be selected only after doing the following two tasks: Set the properties of Nodes and then click on the Simulate button).

Input for Sample 2: Node 1 transmits data to Node 2, Node 2 transmits data to Node 1.



Node Properties	NODE 1	NODE 2
Transmission	Point-to-Point	Point-to-Point
Destination	Node-2	Node-1
Traffic Type	Data	Data
Application Data Size		
Distribution	Constant	Constant
Application Data Size (Bytes)	1472	1472
Inter Arrival Time		
Distribution	Constant	Constant
Inter Arrival Time	20000	20000

Simulation Time - 10 Seconds

(Note: The Simulation Time can be selected only after doing the following two tasks: Set the properties of Nodes and Then click on the Simulate button).

Experiment 1: Node 1 transmits data to Node 2.
Experiment 2: Node 1 transmits data to Node 2, and Node 2 transmits data to Node 1.
Experiment 3: Node 1 transmits data to Node 2, and Node 2 transmits data to Node 3, and Node 3 transmits data to Node 1.
And so on do the experiment by increasing the number of nodes generating traffic as 4, 5, 7, 9, 10, 15, 20 22 and 24 nodes.

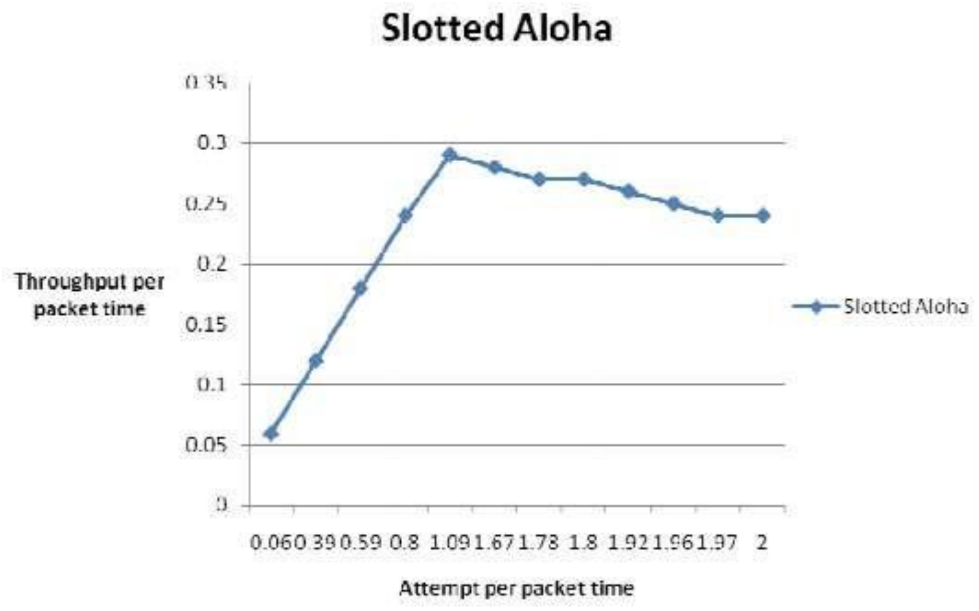
Simulation Time - 10 Seconds

(Note: The Simulation Time can be selected only after doing the following two tasks: Set the properties of Nodes and then click on the Simulate button).

Comparison Table:

Number of nodes generating traffic	Throughput (mbps)	Total attempts	Throughput per packet time G	Attempts per packet time S
1	0.59	499	0.06	0.06
2	1.2	3308	0.12	0.39
3	1.8	4953	0.18	0.59
4	2.4	6691	0.24	0.80
5	2.9	9180	0.29	1.09
7	2.8	14012	0.28	1.67
9	2.7	14868	0.27	1.78
10	2.7	15078	0.27	1.80
15	2.6	16037	0.26	1.92
20	2.5	16437	0.25	1.96
22	2.4	16496	0.24	1.97
24	2.4	16755	0.24	2.00

We have obtained the following characteristic plot for the Slotted ALOHA, which matches the theoretical result.



Note: The optimum value is slightly less than the theoretical maximum of 0.368 because NetSim's simulation is per real-world and includes overheads, inter-frame gaps etc.

Conclusion : Thus we have Used simulator to understand functioning of ALOHA, CSMA/CD.

Experiment No 8

Aim: Perform File Transfer and Access using FTP. Use Command Prompt.

Theory:

File Transfer Protocol (FTP):

File Transfer Protocol(FTP) is an application layer protocol which moves files between local and remote file systems. It runs on the top of TCP, like HTTP. To transfer a file, 2 TCP connections are used by FTP in parallel: control connection and data connection.

FTP Session:

When a FTP session is started between a client and a server, the client initiates a control TCP connection with the server side. The client sends the control information over this. When the server receives this, it initiates a data connection to the client side. Only one file can be sent over one data connection. But the control connection remains active throughout the user session. As we know HTTP is stateless i.e. it does not have to keep track of any user state. But FTP needs to maintain a state about its user throughout the session.

******************** File Transfer and Access using FTP ********************************

Install ProFTPD Server:

```
root@www:~# apt-get -y install proftpd //Standalone

root@www:~# vi /etc/proftpd/proftpd.conf

# line 11: turn off if not needed

UseIPv6 off

# line 15: change to your hostname

ServerName "www.srv.world"

# line 34: uncomment ( specify root directory for chroot )

DefaultRoot ~

root@www:~# vi /etc/ftpusers

# add users you prohibit FTP connection

test

root@www:~# systemctl restart proftpd
```

Install FTP Client.

```
root@client:~# apt-get -y install lftp

# lftp [option] [hostname]

xerus@client:~$ lftp -u ubuntu www.srv.world // lftp -u pc_user server_ip

Password:

# password of the user

lftp ubuntu@www.srv.world:~>

# show current directory on FTP server
lftp ubuntu@www.srv.world:~> pwd
ftp://ubuntu@www.srv.world

# show current directory on local server
lftp ubuntu@www.srv.world:~> !pwd
/home/ubuntu

# show files in current directory on FTP server
lftp ubuntu@www.srv.world:~> ls
drwxr-xr-x    2 1000     1000          23 Jul 19 01:33 public_html
-rw-r--r--    1 1000     1000         399 Jul 20 16:32 test.py


# show files in current directory on local server
lftp ubuntu@www.srv.world:~> !ls -l


total 12
-rw-rw-r-- 1 ubuntu ubuntu 10 Jul 20 14:30 ubuntu.txt
-rw-rw-r-- 1 ubuntu ubuntu 10 Jul 20 14:59 test2.txt
-rw-rw-r-- 1 ubuntu ubuntu 10 Jul 20 14:59 test.txt

# change directory
lftp ubuntu@www.srv.world:~> cd public_html
lftp ubuntu@www.srv.world:~/public_html> pwd
ftp://ubuntu@www.srv.world/%2Fhome/ubuntu/public_html

# upload a file to FTP server
# "-a" means ascii mode ( default is binary mode )
lftp ubuntu@www.srv.world:~> put -a ubuntu.txt test.txt
22 bytes transferred
Total 2 files transferred
lftp ubuntu@www.srv.world:~> ls
```

```
drwxr-xr-x    2 1000     1000          23 Jul 19 01:33 public_html
-rw-r--r--    1 1000     1000          10 Jul 20 17:01 ubuntu.txt
-rw-r--r--    1 1000     1000         399 Jul 20 16:32 test.py
-rw-r--r--    1 1000     1000          10 Jul 20 17:01 test.txt

# upload some files to FTP server

lftp ubuntu@www.srv.world:~> mput -a test.txt test2.txt

22 bytes transferred
Total 2 files transferred
lftp ubuntu@www.srv.world:~> ls
drwxr-xr-x    2 1000     1000          23 Jul 19 01:33 public_html
-rw-r--r--    1 1000     1000         399 Jul 20 16:32 test.py
-rw-r--r--    1 1000     1000          10 Jul 20 17:06 test.txt
-rw-r--r--    1 1000     1000          10 Jul 20 17:06 test2.txt

# download a file from FTP server

# "-a" means ascii mode ( default is binary mode )

lftp ubuntu@www.srv.world:~> get -a test.py

416 bytes transferred
# download some files from FTP server

lftp ubuntu@www.srv.world:~> mget -a test.txt test2.txt

20 bytes transferred
Total 2 files transferred
# create a directory in current directory on FTP Server

lftp ubuntu@www.srv.world:~> mkdir testdir
mkdir ok, `testdir' created
lftp ubuntu@www.srv.world:~> ls
drwxr-xr-x    2 1000     1000          23 Jul 19 01:33 public_html
-rw-r--r--    1 1000     1000         399 Jul 20 16:32 test.py
-rw-r--r--    1 1000     1000          10 Jul 20 17:06 test.txt
-rw-r--r--    1 1000     1000          10 Jul 20 17:06 test2.txt
drwxr-xr-x    2 1000     1000           6 Jul 20 17:16 testdir
226 Directory send OK.

# delete a direcroty in current directory on FTP Server

lftp ubuntu@www.srv.world:~> rmdir testdir
rmdir ok, `testdir' removed

lftp ubuntu@www.srv.world:~> ls
```

```
drwxr-xr-x    2 1000     1000          23 Jul 19 01:33 public_html
-rw-r--r--    1 1000     1000         399 Jul 20 16:32 test.py
-rw-r--r--    1 1000     1000          10 Jul 20 17:06 test.txt
-rw-r--r--    1 1000     1000          10 Jul 20 17:06 test2.txt

# delete a file in current directory on FTP Server

lftp ubuntu@www.srv.world:~> rm test2.txt
rm ok, `test2.txt' removed

lftp ubuntu@www.srv.world:~> ls
drwxr-xr-x    2 1000     1000          23 Jul 19 01:33 public_html
-rw-r--r--    1 1000     1000         399 Jul 20 16:32 test.py
-rw-r--r--    1 1000     1000          10 Jul 20 17:06 test.txt

# delete some files in current directory on FTP Server

lftp ubuntu@www.srv.world:~> mrm ubuntu.txt test.txt

rm ok, 2 files removed
lftp ubuntu@www.srv.world:~> ls
drwxr-xr-x    2 1000     1000          23 Jul 19 01:33 public_html

# execute commands with "![command]"

lftp ubuntu@www.srv.world:~> !cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
...
...
ubuntu:x:1001:1001::/home/ubuntu:/bin/bash

# exit

lftp ubuntu@www.srv.world:~> quit

221 Goodbye.
```

Conclusion :-
Thus we have Performed File Transfer and Access using FTP.

Experiment No 9

Aim: Implement Socket programming using TCP in Java.

Theory:

Socket:

Sockets allow communication between two different processes on the same or different machines. To be more precise, it's a way to talk to other computers using standard Unix file descriptors.

Socket Types:

There are four types of sockets available to the users. The first two are most commonly used and the last two are rarely used.

Processes are presumed to communicate only between sockets of the same type but there is no restriction that prevents communication between sockets of different types.

- **Stream Sockets** − Delivery in a networked environment is guaranteed. If you send through the stream socket three items "A, B, C", they will arrive in the same order − "A, B, C". These sockets use **TCP** (Transmission Control Protocol) for data transmission. If delivery is impossible, the sender receives an error indicator. Data records do not have any boundaries.
- **Datagram Sockets** − Delivery in a networked environment is not guaranteed. They're connectionless because you don't need to have an open connection as in Stream Sockets − you build a packet with the destination information and send it out. They use **UDP** (User Datagram Protocol).
- **Raw Sockets** − These provide users access to the underlying communication protocols, which support socket abstractions. These sockets are normally datagram oriented, though their exact characteristics are dependent on the interface provided by the protocol.
- **Sequenced Packet Sockets** − They are similar to a stream socket, with the exception that record boundaries are preserved. This interface is provided only as a part of the Network Systems (NS) socket abstraction, and is very important in most serious NS applications. Sequenced-packet sockets allow the user to manipulate the Sequence Packet Protocol (SPP) or Internet Datagram Protocol (IDP) headers on a packet or a group of packets, either by writing a prototype header along with whatever data is to be sent, or by specifying a default header to be used with all outgoing data, and allows the user to receive the headers on incoming packets.

********************** Socket Programming using TCP ******************************

Program:

Server Side:

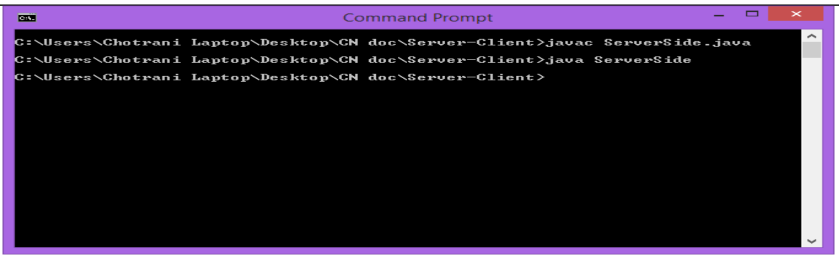
```
import java.net.*;
import java.io.*;
class ServerSide
{
    public static void main(String[] args) throws
Exception
   {
       int choice,a,b,c=0;
       ServerSocket ss = new ServerSocket(1024);
       Socket  s = ss.accept();
       BufferedReader br = new BufferedReader(new
InputStreamReader (s.getInputStream() ) );
       choice =Integer.parseInt(br.readLine());
       a =Integer.parseInt(br.readLine());
       b = Integer.parseInt(br.readLine());
       switch(choice)
       {
          case 1 : c = a+b;
          break;
          case 2 : c = a-b;
          break;
          case 3 : c = a*b;
          break;
          case 4 : c = a/b;
          break;
          case 5 : c = (a%b);
          break;
       }
                   PrintStream pr = new
PrintStream(s.getOutputStream()) ;
       pr.println(c);
       ss.close();
       s.close();
   }}
```

Client Side:

```
import java.net.*;
import java.io.*;
class ClientSide
{
    public static void main(String[] args) throws
Exception
   {
       int ch=0,a,b,c;
       Socket s = new Socket("localhost",1024);
       BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
                           PrintStream ps=new
PrintStream(s.getOutputStream());
       System.out.println("Please Enter Number 1:");
       a = Integer.parseInt(br.readLine());
       System.out.println("Please Enter Number 2:");
       b = Integer.parseInt(br.readLine());
             System.out.println("Please Enter The
Operation to Be
Performed\n");
       System.out.println("1.Addition 2.Subtraction
            3.Multiplication 4.Divison 5.Modulo
0.Exit"); ch = Integer.parseInt(br.readLine());
       ps.println(ch);
       ps.println(a);
       ps.println(b);
                  BufferedReader br1 = new
BufferedReader(new
     InputStreamReader(s.getInputStream()));
       c=Integer.parseInt(br1.readLine());
       System.out.println("Answer: "+c);
       s.close();
   }
}
```

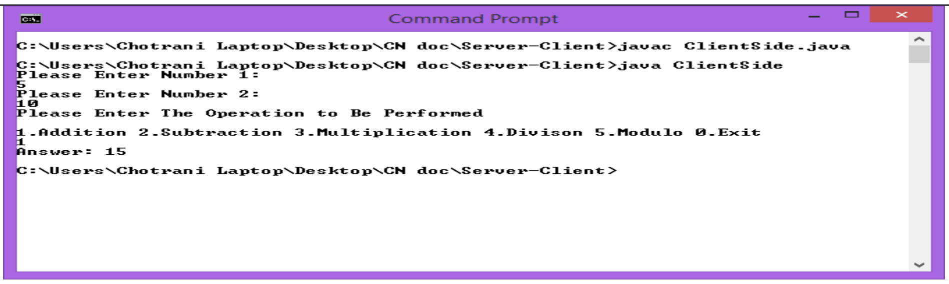
Output:

Server Side:



```
C:\Users\Chotrani Laptop\Desktop\CN doc\Server-Client>javac ServerSide.java
C:\Users\Chotrani Laptop\Desktop\CN doc\Server-Client>java ServerSide
C:\Users\Chotrani Laptop\Desktop\CN doc\Server-Client>
```

Client Side:



```
C:\Users\Chotrani Laptop\Desktop\CN doc\Server-Client>javac ClientSide.java

C:\Users\Chotrani Laptop\Desktop\CN doc\Server-Client>java ClientSide
Please Enter Number 1:
5
Please Enter Number 2:
10
Please Enter The Operation to Be Performed

1.Addition 2.Subtraction 3.Multiplication 4.Divison 5.Modulo 0.Exit
1
Answer: 15

C:\Users\Chotrani Laptop\Desktop\CN doc\Server-Client>
```

Conclusion :-

Thus we have Implemented Socket programming using TCP in Java.

Experiment No 10

Aim: Perform Remote login using Telnet server. Use Command Prompt Only.

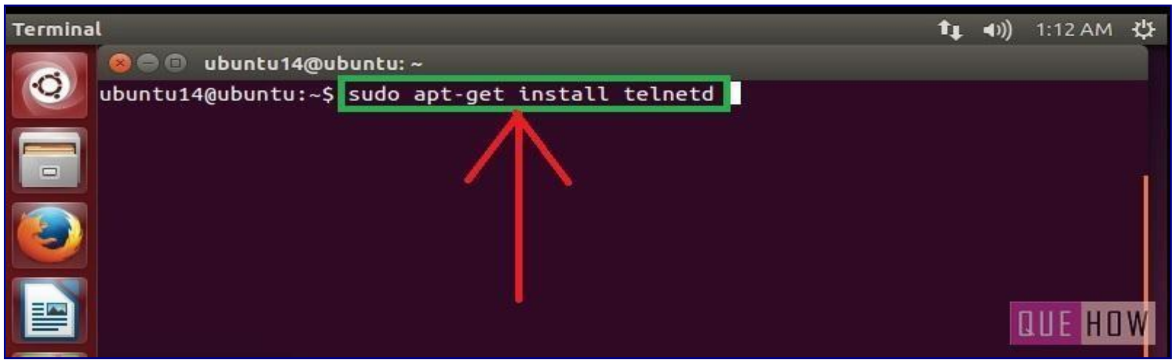
Theory:

Telnet is a protocol used on the Internet or local area network to provide a bidirectional interactive text-oriented communication facility using a virtual terminal connection. User data is interspersed in-band with Telnet control information in an 8-bit byte oriented data connection over the Transmission Control Protocol (TCP).

Procedure:

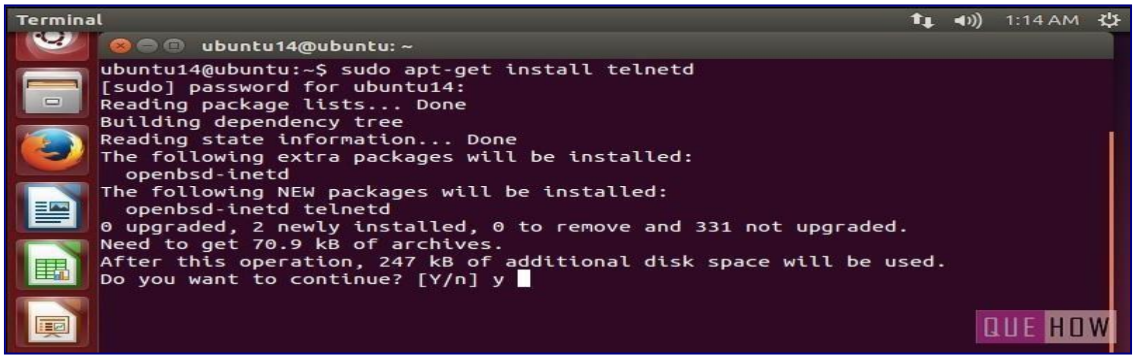
Steps to Install and Use Telnet in Ubuntu:

Step 1: First write **"sudo apt-get install telnetd"** and press enter. **"telnetd"** is a daemon that gets invoked by *"inetd" or its extension "xinetd"*, both are the internet servers.



```
ubuntu14@ubuntu:~$ sudo apt-get install telnetd
```

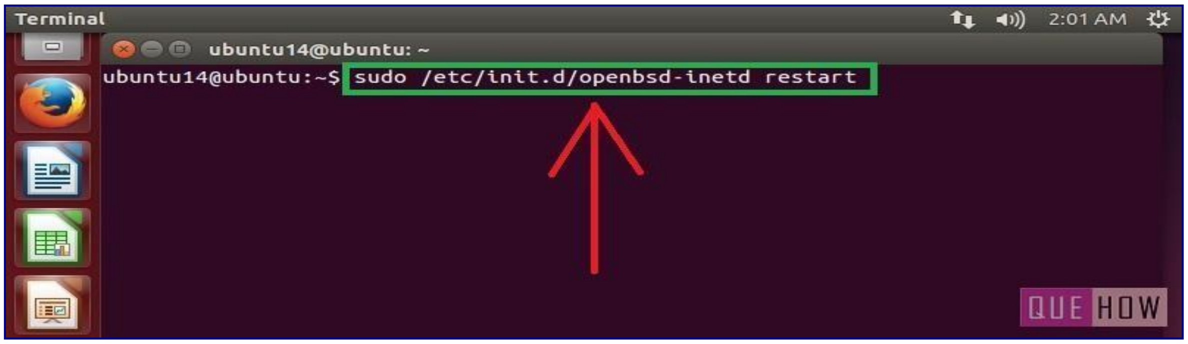
Step 2: write **"y"** and then press enter to continue.



```
ubuntu14@ubuntu:~$ sudo apt-get install telnetd
[sudo] password for ubuntu14:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  openbsd-inetd
The following NEW packages will be installed:
  openbsd-inetd telnetd
0 upgraded, 2 newly installed, 0 to remove and 331 not upgraded.
Need to get 70.9 kB of archives.
After this operation, 247 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

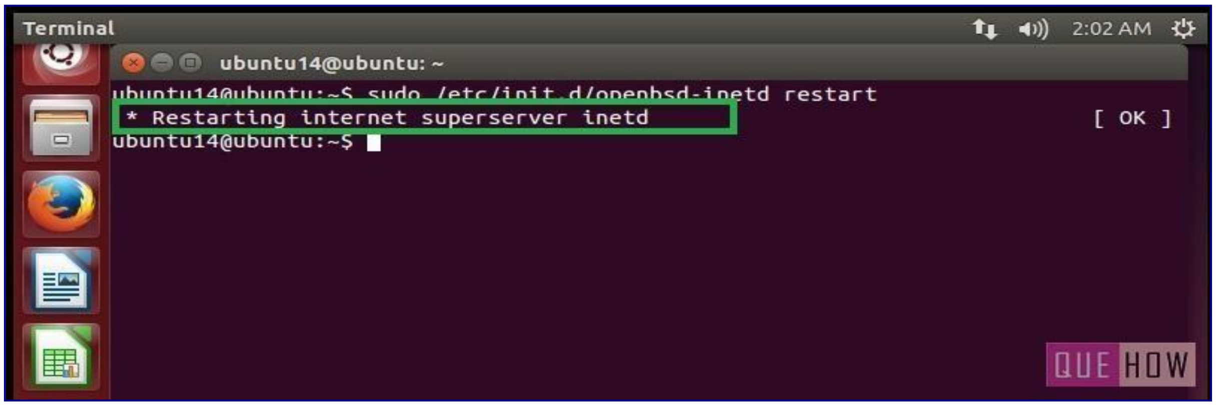
Step 3: Now **restart "inetd"**. Type **"sudo /etc/init.d.open-bsd-inetd restart"**.

"inetd" is daemon used for *dealing with incoming network* and it is responsible for deciding which program to run when a request comes.



```
ubuntu14@ubuntu:~$ sudo /etc/init.d/openbsd-inetd restart
```

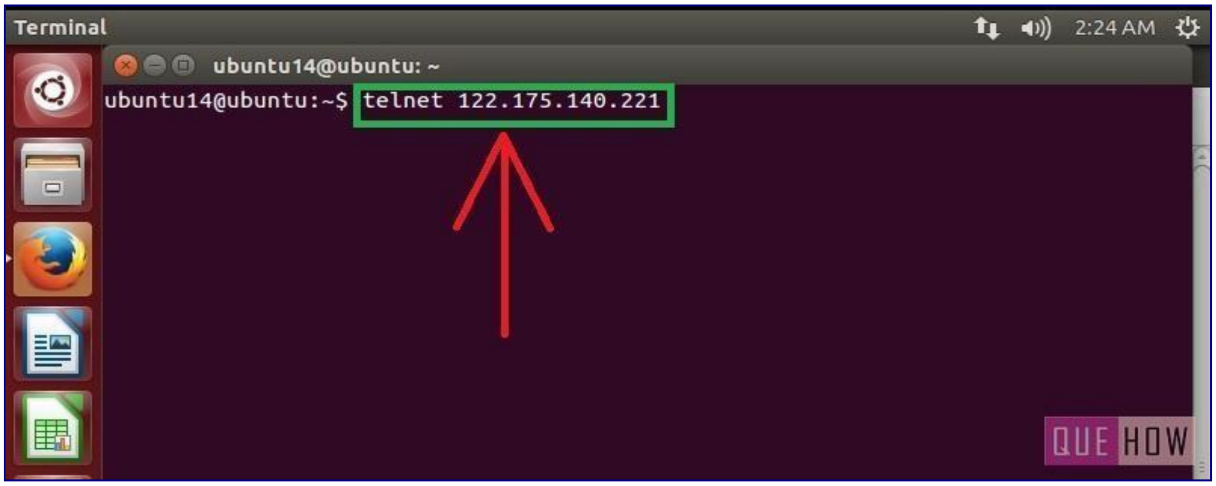
Step 4: To ensure "inetd" is started, press enter after writing the above command.



```
ubuntu14@ubuntu:~$ sudo /etc/init.d/openbsd-inetd restart
 * Restarting internet superserver inetd                          [ OK ]
ubuntu14@ubuntu:~$
```

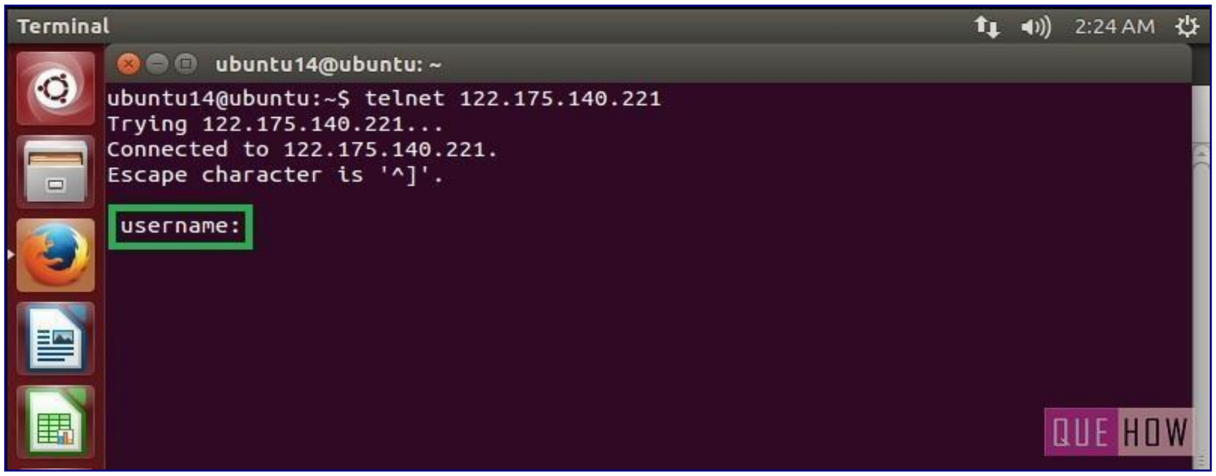
To connect with any remote client:

Step 5: Just type:**"telnet hostipaddress"**. For an example: "telnet 122.175.140.221" and press enter.



```
ubuntu14@ubuntu:~$ telnet 122.175.140.221
```

Step 6: Then it is connected to **"host ip address"**. For security reasons, it is required to provide "username" and "password" as well.



```
ubuntu14@ubuntu:~$ telnet 122.175.140.221
Trying 122.175.140.221...
Connected to 122.175.140.221.
Escape character is '^]'.

username:
```

Conclusion :-

Thus we have Performed Remote login using Telnet server.