GOVERNMENT COLLEGE OF ENGINEERING

TIRUNELVELI – 627 007.

20 - 20

Register No.

CERTIFICATE

This is a Bonafide record of work done by ……...

…… Government College of Engineering, Tirunelveli during the year 20- 20

STATION: TIRUNELVELI – 7.

DATE:

Staff-in-Charge **Head of the Department**

Submitted for the Anna University Practical Examination held at Government College of Engineering, Tirunelveli on ……

……

Internal Examiner **External Examiner**

TABLE OF CONTENTS

Ex No:1 A	INTRODUCTION TO OSI LAYERS

Aim

To study about the OSI layers.

OSI Model

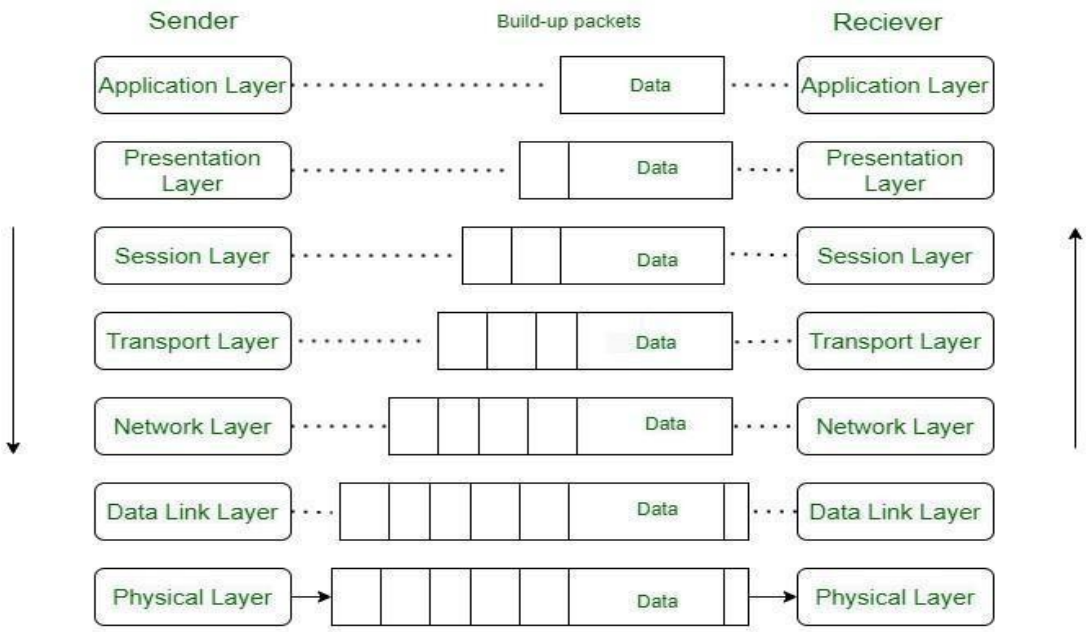
OSI stands for **Open System Interconnection** is a reference model that describes how information from a software application in one computer moves through a physical medium to the software application in another computer. OSI consists of seven layers, and each layer performs a particular network function. OSI model was developed by the International Organization for Standardization (ISO) in 1984, and it is now considered as an architectural model for the inter-computer communications.

OSI model divides the whole task into seven smaller and manageable tasks. Each layer is assigned a particular task.Each layer is self-contained, so that task assigned to each layer can be performed independently

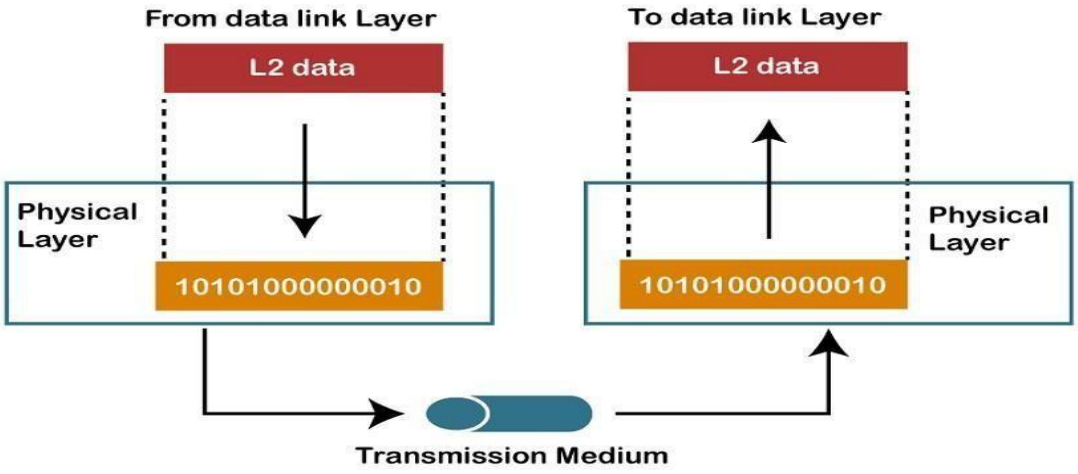
7 Layers of OSI Model

There are the seven OSI layers. Each layer has different functions. A list of seven layers are given below:

1. Physical Layer
2. Data-Link Layer
3. Network Layer
4. Transport Layer
5. Session Layer
6. Presentation Layer
7. Application Layer



1) Physical layer



The main functionality of the physical layer is to transmit the individual bits from one node to another node. It is the lowest layer of the OSI model. It establishes, maintains and deactivates the physical connection. It specifies the mechanical, electrical and procedural network interface specifications.

Functions of a Physical layer

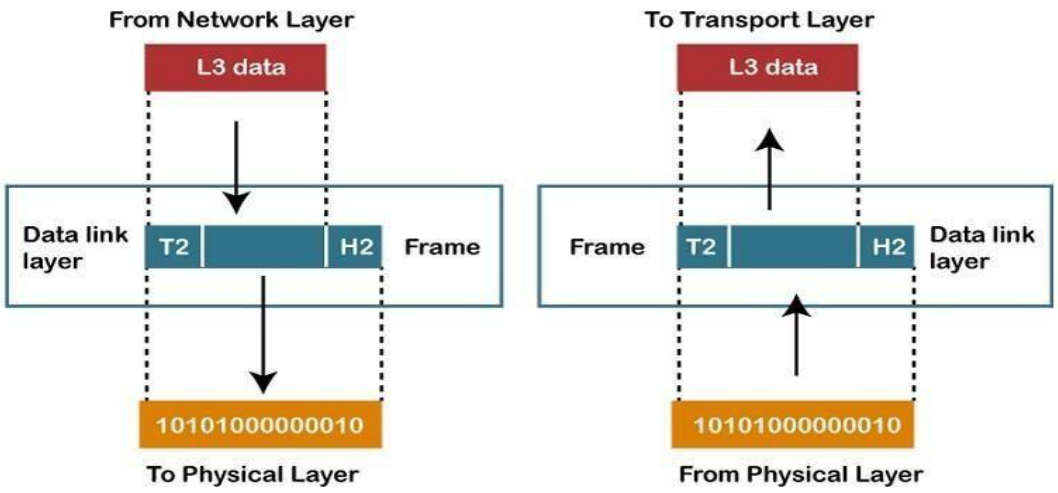
Line Configuration: It defines the way how two or more devices can be connected physically.

Data Transmission: It defines the transmission mode whether it is simplex, half-duplex or fullduplex mode between the two devices on the network.

Topology: It defines the way how network devices are arranged.

Signals: It determines the type of the signal used for transmitting the information.

2) Data-Link Layer



This layer is responsible for the error-free transfer of data frames. It defines the format of the data on the network. It provides a reliable and efficient communication between two or more devices. It is mainly responsible for the unique identification of each device that resides on a local network. It contains two sub-layers: Logical Link Control Layer, Media Access Control Layer.

Functions of the Data-link layer

Framing: The data link layer translates the physical's raw bit stream into packets known as Frames. The Data link layer adds the header and trailer to the frame. The header which is added to the frame contains the hardware destination and source address.

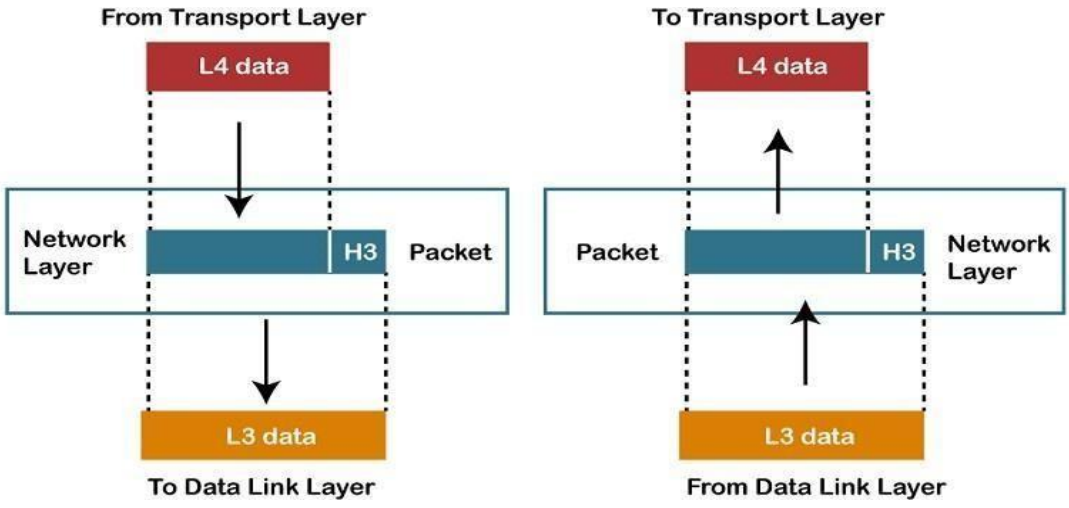
Physical Addressing: The Data link layer adds a header to the frame that contains a destination address. The frame is transmitted to the destination address mentioned in the header.

Flow Control: Flow control is the main functionality of the Data-link layer. It is the technique through which the constant data rate is maintained on both the sides so that no data get corrupted. It ensures that the transmitting station such as a server with higher processing speed does not exceed the receiving station, with lower processing speed.

Error Control: Error control is achieved by adding a calculated value CRC (Cyclic Redundancy Check) that is placed to the Data link layer's trailer which is added to the message frame before it is sent to the physical layer. If any error seems to occurr, then the receiver sends the acknowledgment for the retransmission of the corrupted frames.

Access Control: When two or more devices are connected to the same communication channel, then the data link layer protocols are used to determine which device has control over the link at a given time.

3) Network Layer



It is a layer that manages device addressing, tracks the location of devices on the network. It determines the best path to move data from source to the destination based on the network conditions, the priority of service, and other factors. The Data link layer is responsible for routing and forwarding the packets. Routers are the layer 3 devices, they are specified in this layer and used to provide the routing services within an internetwork. The protocols used to route the network traffic are known as Network layer protocols. Examples of protocols are IP and Ipv6.

Functions of Network Layer

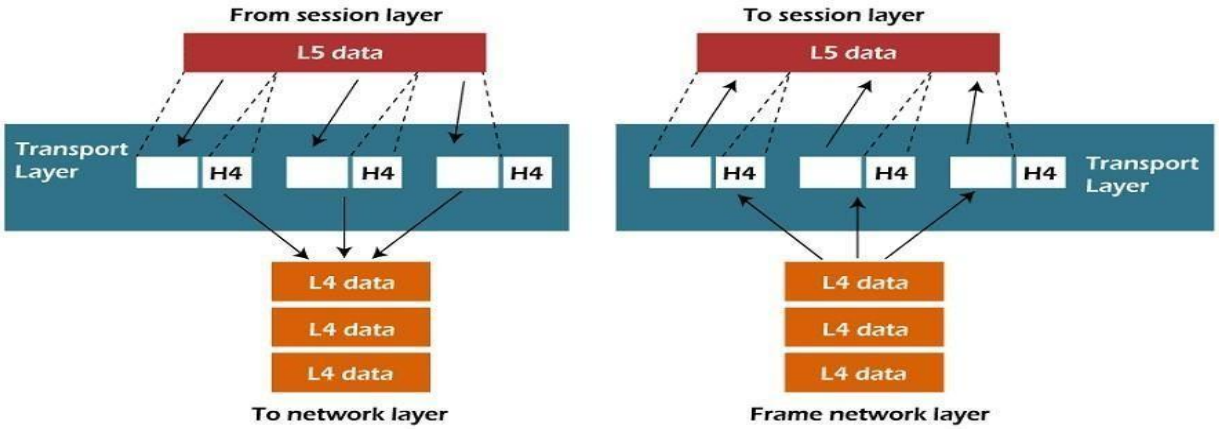
Internetworking: An internetworking is the main responsibility of the network layer. It provides a logical connection between different devices.

Addressing: A Network layer adds the source and destination address to the header of the frame. Addressing is used to identify the device on the internet.

Routing: Routing is the major component of the network layer, and it determines the best optimal path out of the multiple paths from source to the destination.

Packetizing: A Network Layer receives the packets from the upper layer and converts them into packets. This process is known as Packetizing. It is achieved by internet protocol (IP).

4) Transport Layer



The Transport layer is a Layer 4 ensures that messages are transmitted in the order in which they are sent and there is no duplication of data. The main responsibility of the transport layer is to transfer the data completely. It receives the data from the upper layer and converts them into smaller units known as segments. This layer can be termed as an end-to-end layer as it provides a point-to-point connection between source and destination to deliver the data reliably.

The two protocols used in this layer are: Transmission Control Protocol, User Datagram Protocol

Functions of Transport Layer

Service-point addressing: Computers run several programs simultaneously due to this reason, the transmission of data from source to the destination not only from one computer to another computer but also from one process to another process. The transport layer adds the header that contains the address known as a service-point address or port address. The responsibility of the network layer is to transmit the data from one computer to another computer and the responsibility of the transport layer is to transmit the message to the correct process.

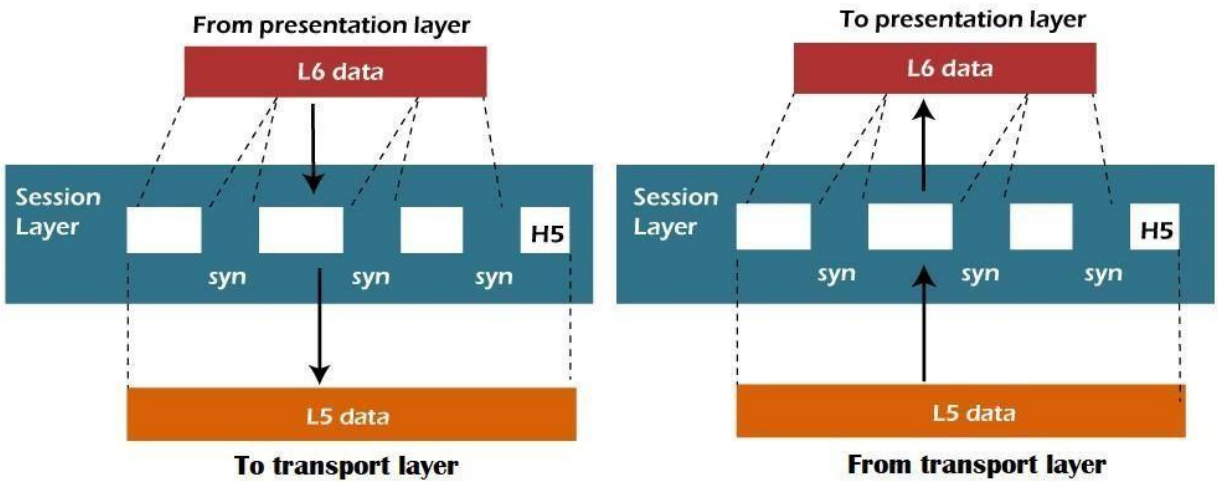
Segmentation and reassembly: When the transport layer receives the message from the upper layer, it divides the message into multiple segments, and each segment is assigned with a sequence number that uniquely identifies each segment. When the message has arrived at the destination, then the transport layer reassembles the message based on their sequence numbers.

Connection control: Transport layer provides two services Connection-oriented service and connectionless service. A connectionless service treats each segment as an individual packet, and they all travel in different routes to reach the destination. A connection-oriented service makes a connection with the transport layer at the destination machine before delivering the packets. In connection-oriented service, all the packets travel in the single route.

Flow control: The transport layer also responsible for flow control but it is performed end-to-end rather than across a single link.

Error control: The transport layer is also responsible for Error control. Error control is performed end-to-end rather than across the single link. The sender transport layer ensures that message reach at the destination without any error.

5) Session Layer



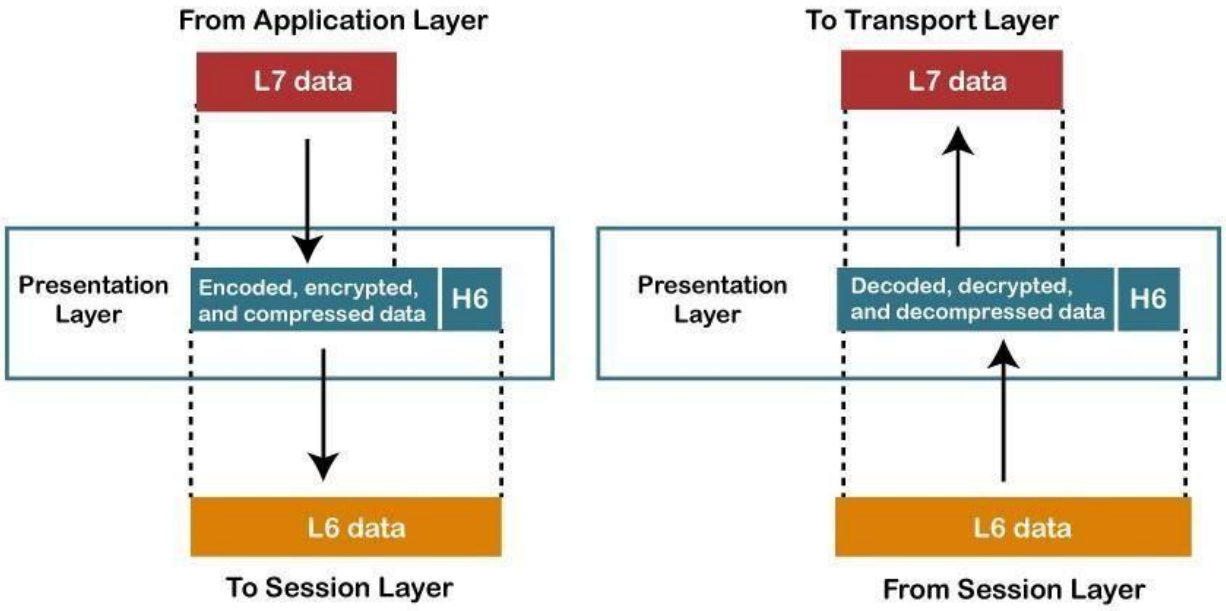
The Session layer is used to establish, maintain and synchronizes the interaction between communicating devices.

Functions of Session layer:

Dialog control: Session layer acts as a dialog controller that creates a dialog between two processes or we can say that it allows the communication between two processes which can be either half-duplex or full-duplex.

Synchronization: Session layer adds some checkpoints when transmitting the data in a sequence. If some error occurs in the middle of the transmission of data, then the transmission will take place again from the checkpoint. This process is known as Synchronization and recovery.

6) Presentation Layer



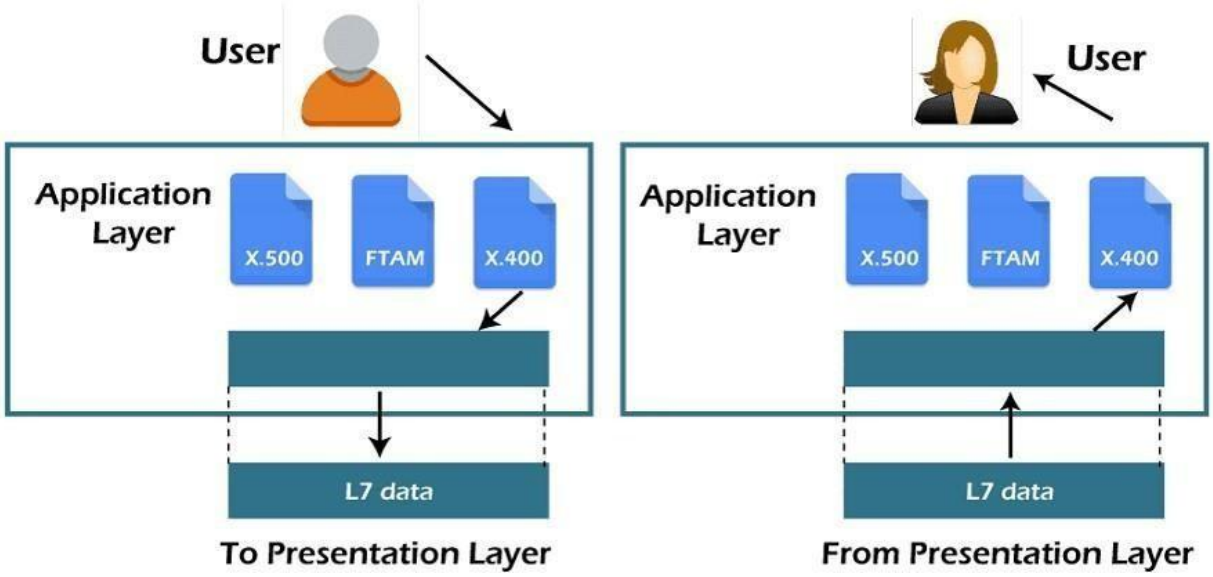
A Presentation layer is mainly concerned with the syntax and semantics of the information exchanged between the two systems.It acts as a data translator for a network.This layer is a part of the operating system that converts the data from one presentation format to another format.The Presentation layer is also known as the syntax layer.

Functions of Presentation layer

Translation: The processes in two systems exchange the information in the form of character strings, numbers and so on. Different computers use different encoding methods, the presentation layer handles the interoperability between the different encoding methods. It converts the data from sender-dependent format into a common format and changes the common format into receiver-dependent format at the receiving end.

Encryption: Encryption is needed to maintain privacy. Encryption is a process of converting the sender-transmitted information into another form and sends the resulting message over the network.

7) Application Layer



An application layer serves as a window for users and application processes to access network service.It handles issues such as network transparency, resource allocation, etc.An application layer is not an application, but it performs the application layer functions.This layer provides the network services to the end-users.

Functions of Application layer

File transfer, access, and management (FTAM): An application layer allows a user to access the files in a remote computer, to retrieve the files from a computer and to manage the files in a remote computer.

Mail services: An application layer provides the facility for email forwarding and storage.

Result

Thus the OSI layers were studied successfully.

Ex No:1 B	COMMANDS

Aim

To explore the commands like tcpdump, netstat, ifconfig, nslookup, traceroute and ping.

1. Tcpdump

The tcpdump utility allows you to capture packets that flow within your network to assist in network troubleshooting. The following are several examples of using tcpdump with different options. Traffic is captured based on a specified filter.

Options	Description
-D	print a list of network interfaces.
-i	specify an interface on which to capture**.**
-c	specify the number of packets to receive.
-w	write captured data to a file.
-r	Read captured data from a file.

Many other options and arguments can be used with tcpdump. The following are some specific examples of the power of the tcpdump utility.

1. Display traffic between 2 hosts

To display all traffic between two hosts (represented by variables host1 and host2):

#tcpdump host host1 and host2

2. Display traffic from a source or destination host only

To display traffic from only a source (src) or destination (dst) host:

#tcpdumpsrc host

#tcpdumpdst host

3. Display traffic for a specific protocol

Provide the protocol as an argument to display only traffic for a specific protocol, for example tcp, udp, icmp, arp:

#tcpdump protocol

For example to display traffic only for the top traffic:

#tcpdumptcp

4. Filtering based on source or destination port To filter based on a source or destination port:

#tcpdumpsrc port ftp

#tcpdumpdst port http

2. Netstat

Netstat is a common command like TCP/IP networking available in most versions of windows,linux,unix and other operating systems. Netstat provides information and statistics about protocols in use and current TCP/IP network connections.

Syntax

Netstat [-a][-b][-e][-f][-n][-o][-p prob][-r][-s][-t][-x][-y][interval]

Parameter

-a **:**Displays all connections and listening ports.

-b **:**Shows the executable involved in creating each connection or listening port.

-e **:** Displays Ethernet statistics.

-f **:** Displays fully qualified domain names for foreign address.

-n **:** Shows addresses and port number in numerical form.

-o **:** Displays the owning process ID associated with each connection.

-p proto : Shows connections for the specified protocol(TCP,UDP,etc).

-r **:** Displays the routing table.

-s **:** Displays per_protocol statistics.

-t **:** Shows the current connection offload state.

-x **:** Displays network direct connections,listeners and shared endpoints.

-y **:** Displays the TCP connection template for all connections.

Interval : The interval option in the netstat command specifies the time in seconds between each display of statictics.When the output every specified number of seconds until you stop it with CTRL+C.

EX:netstat -e 5

Output

```
C:\Users\CSE>netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            DESKTOP-2R4D1F4:0      LISTENING
  TCP    0.0.0.0:445            DESKTOP-2R4D1F4:0      LISTENING
  TCP    0.0.0.0:523            DESKTOP-2R4D1F4:0      LISTENING
  TCP    0.0.0.0:903            DESKTOP-2R4D1F4:0      LISTENING
  TCP    0.0.0.0:913            DESKTOP-2R4D1F4:0      LISTENING
  TCP    0.0.0.0:1521           DESKTOP-2R4D1F4:0      LISTENING
  TCP    0.0.0.0:2323           DESKTOP-2R4D1F4:0      LISTENING
  TCP    0.0.0.0:5040           DESKTOP-2R4D1F4:0      LISTENING
  TCP    0.0.0.0:5432           DESKTOP-2R4D1F4:0      LISTENING
  TCP    0.0.0.0:25000          DESKTOP-2R4D1F4:0      LISTENING
  TCP    0.0.0.0:49664          DESKTOP-2R4D1F4:0      LISTENING
  TCP    0.0.0.0:49665          DESKTOP-2R4D1F4:0      LISTENING
  TCP    0.0.0.0:49666          DESKTOP-2R4D1F4:0      LISTENING
```

```
C:\Users\CSE>netstat -e
Interface Statistics

                           Received            Sent

Bytes                      42467340         6002694
Unicast packets               35235           22435
Non-unicast packets           39990            4094
Discards                          0               0
Errors                            0               0
Unknown protocols                 0
```

```
C:\Users\CSE>netstat -f

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    172.16.16.133:49698    20.198.119.84:https    ESTABLISHED
  TCP    172.16.16.133:49710    a104-90-5-224.deploy.static.akamaitechnologies.com:https  CLOSE_WAIT
  TCP    172.16.16.133:49727    180.149.59.219:https   ESTABLISHED
  TCP    172.16.16.133:49806    na-autoupdate.opera.com:https  CLOSE_WAIT
  TCP    172.16.16.133:49807    na-autoupdate.opera.com:https  CLOSE_WAIT
  TCP    172.16.16.133:49808    maa03s45-in-f3.1e100.net:https  ESTABLISHED
  TCP    172.16.16.133:49813    na-autoupdate.opera.com:https  TIME_WAIT
```

```
C:\Users\CSE>netstat -n

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    172.16.16.133:49698    20.198.119.84:443      ESTABLISHED
  TCP    172.16.16.133:49710    104.90.5.224:443       CLOSE_WAIT
  TCP    172.16.16.133:49727    180.149.59.219:443     ESTABLISHED
  TCP    172.16.16.133:49806    107.167.96.39:443      CLOSE_WAIT
  TCP    172.16.16.133:49807    107.167.96.39:443      CLOSE_WAIT
  TCP    172.16.16.133:49808    142.250.196.35:443     ESTABLISHED
  TCP    172.16.16.133:49815    204.79.197.203:443     TIME_WAIT
  TCP    172.16.16.133:49816    204.79.197.203:443     TIME_WAIT
  TCP    172.16.16.133:49817    204.79.197.203:443     TIME_WAIT
  TCP    172.16.16.133:49837    170.72.245.227:443     ESTABLISHED
  TCP    192.168.153.1:1521     192.168.153.1:49672    ESTABLISHED
  TCP    192.168.153.1:49672    192.168.153.1:1521     ESTABLISHED
```

```
C:\Users\CSE>netstat -o

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    172.16.16.133:49698    20.198.119.84:https    ESTABLISHED     3876
  TCP    172.16.16.133:49710    a104-90-5-224:https    CLOSE_WAIT      7720
  TCP    172.16.16.133:49727    180.149.59.219:https   ESTABLISHED     2888
  TCP    172.16.16.133:49807    na-autoupdate:https    LAST_ACK        8916
  TCP    172.16.16.133:49808    maa03s45-in-f3:https   TIME_WAIT       0
  TCP    172.16.16.133:49815    a-0003:https           TIME_WAIT       0
  TCP    172.16.16.133:49816    a-0003:https           TIME_WAIT       0
  TCP    172.16.16.133:49817    a-0003:https           TIME_WAIT       0
  TCP    172.16.16.133:49837    170.72.245.227:https   ESTABLISHED     4180
  TCP    172.16.16.133:49838    maa03s34-in-f10:https  ESTABLISHED     8916
  TCP    172.16.16.133:49839    a23-35-32-176:https    ESTABLISHED     9864
```

```
C:\Users\CSE>netstat -p

Active Connections

  Proto  Local Address          Foreign Address        State
```

```
C:\Users\CSE>netstat -r
===========================================================================
Interface List
 10...1c 69 7a 43 65 d7 ......Realtek PCIe GbE Family Controller
  4...00 50 56 c0 00 01 ......VMware Virtual Ethernet Adapter for VMnet1
 15...00 50 56 c0 00 08 ......VMware Virtual Ethernet Adapter for VMnet8
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0    172.16.16.16    172.16.16.133     25
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
      172.16.16.0    255.255.255.0         On-link    172.16.16.133    281
    172.16.16.133  255.255.255.255         On-link    172.16.16.133    281
    172.16.16.255  255.255.255.255         On-link    172.16.16.133    281
    192.168.153.0    255.255.255.0         On-link    192.168.153.1    291
    192.168.153.1  255.255.255.255         On-link    192.168.153.1    291
```

```
C:\Users\CSE>netstat -s

IPv4 Statistics

  Packets Received                    = 15442
  Received Header Errors              = 0
  Received Address Errors             = 347
  Datagrams Forwarded                 = 0
  Unknown Protocols Received          = 0
  Received Packets Discarded          = 3663
  Received Packets Delivered          = 14637
  Output Requests                     = 5694
  Routing Discards                    = 0
  Discarded Output Packets            = 0
  Output Packet No Route              = 0
  Reassembly Required                 = 0
  Reassembly Successful               = 0
  Reassembly Failures                 = 0
  Datagrams Successfully Fragmented   = 0
  Datagrams Failing Fragmentation     = 0
  Fragments Created                   = 0
```

```
C:\Users\CSE>netstat -t

Active Connections

  Proto  Local Address          Foreign Address        State           Offload State
  TCP    172.16.16.133:49698    20.198.119.84:https    ESTABLISHED     InHost
  TCP    172.16.16.133:49710    a104-90-5-224:https    CLOSE_WAIT      InHost
  TCP    172.16.16.133:49727    180.149.59.219:https   CLOSE_WAIT      InHost
  TCP    172.16.16.133:49838    maa03s34-in-f10:https  TIME_WAIT       InHost
  TCP    172.16.16.133:49847    na-autoupdate:https    ESTABLISHED     InHost
  TCP    172.16.16.133:49848    na-autoupdate:https    ESTABLISHED     InHost
  TCP    172.16.16.133:49851    na-autoupdate:https    TIME_WAIT       InHost
  TCP    172.16.16.133:49852    na-autoupdate:https    TIME_WAIT       InHost
  TCP    192.168.153.1:1521     DESKTOP-2R4D1F4:49672  ESTABLISHED     InHost
  TCP    192.168.153.1:49672    DESKTOP-2R4D1F4:1521   ESTABLISHED     InHost
```

```
C:\Users\CSE>netstat -x

Active NetworkDirect Connections, Listeners, SharedEndpoints

  Mode   IfIndex Type           Local Address          Foreign Address        PID
```

3. Ipconfig

The ipconfig command is not available on windows however you can use the ipconfigcommand, which serves a similar purpose for managing and viewing network configurationin windows.

Syntax

ipconfig [/allcompartments] [/? | /all | /renew [adapter] | /release [adapter] | /renew6[adapter] | /release6 [adapter] | /flushdns | /displaydns | /registerdns | /showclassid adapter |/setclassid adapter [classid] | /showclassid6 adapter | /setclassid6 adapter [classid]]

ipconfig /? : Display this help message **ipconfig /all**

:Display full configuration information.

ipconfig /release : Release the IPv4 address for the specified adapter.

ipconfig /release6 : Release the IPv6 address for the specified adapter.

ipconfig /renew : Renew the IPv4 address for the specified adapter. **ipconfig**

/renew6 : Renew the IPv6 address for the specified adapter. **ipconfig**

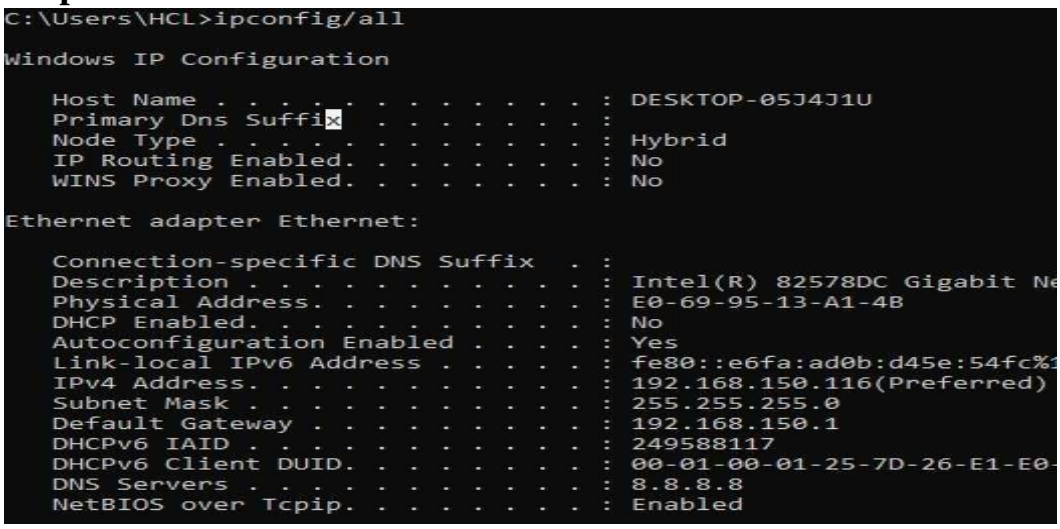
/flushdns : Purges the DNS Resolver cache.

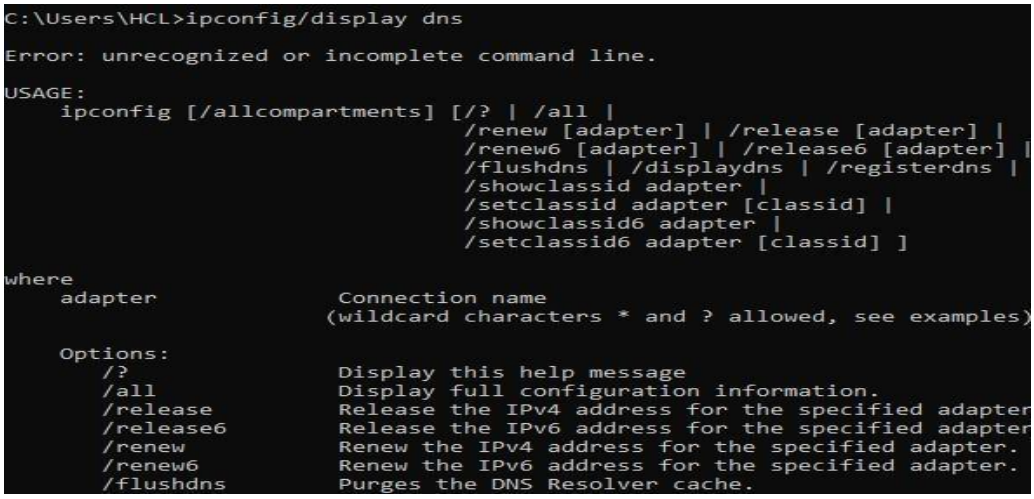
ipconfig /registerdns : Refreshes all DHCP leases and re-registers DNSnames **ipconfig**

/displaydns : Display the contents of the DNS Resolver Cache. **ipconfig /showclassid**

:Displays all the dhcp class IDs allowed for adapter. **ipconfig /setclassid :**Modifies the dhcp class id. **ipconfig /showclassid6 :**Displays all the IPv6 DHCP class IDs allowed for adapter. **ipconfig /setclassid6 :**Modifies the IPv6 DHCP class id.

Output





```
C:\Users\HCL>ipconfig/flush dns

Error: unrecognized or incomplete command line.

USAGE:
    ipconfig [/allcompartments] [/? | /all |
                                 /renew [adapter] | /release [adapter] |
                                 /renew6 [adapter] | /release6 [adapter] |
                                 /flushdns | /displaydns | /registerdns |
                                 /showclassid adapter |
                                 /setclassid adapter [classid] |
                                 /showclassid6 adapter |
                                 /setclassid6 adapter [classid] ]

where
    adapter             Connection name
                       (wildcard characters * and ? allowed, see examples)

    Options:
       /?               Display this help message
       /all             Display full configuration information.
       /release         Release the IPv4 address for the specified adapter.
       /release6        Release the IPv6 address for the specified adapter.
       /renew           Renew the IPv4 address for the specified adapter.
       /renew6          Renew the IPv6 address for the specified adapter.
       /flushdns        Purges the DNS Resolver cache.
       /registerdns     Refreshes all DHCP leases and re-registers DNS names
       /displaydns      Display the contents of the DNS Resolver Cache.
       /showclassid     Displays all the dhcp class IDs allowed for adapter.
       /setclassid      Modifies the dhcp class id.
```

```
C:\Users\CSE>ipconfig/renew

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b69b:b6dd:1118:6f60%10
   IPv4 Address. . . . . . . . . . . : 172.16.16.133
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 172.16.16.16

Ethernet adapter VMware Network Adapter VMnet1:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::277d:3d3b:125f:9533%4
   IPv4 Address. . . . . . . . . . . : 192.168.206.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Ethernet adapter VMware Network Adapter VMnet8:
```

```
C:\Users\CSE>ipconfig/renew

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b69b:b6dd:1118:6f60%10
   IPv4 Address. . . . . . . . . . . : 172.16.16.133
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 172.16.16.16

Ethernet adapter VMware Network Adapter VMnet1:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::277d:3d3b:125f:9533%4
   IPv4 Address. . . . . . . . . . . : 192.168.206.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Ethernet adapter VMware Network Adapter VMnet8:
```

4. Nslookup

The nslookup (Which stands for name server lookup) command is a network utilityprogram used to obtain information about internet servers. Its finds name server informationfor domains by queryibg the Domain Name System .

Syntax

>nslookup

>www.google.com

Output

```
C:\Users\ACER>nslookup
Default Server:  dns.google
Address:  8.8.8.8

> www.google.com
Server:  dns.google
Address:  8.8.8.8

Non-authoritative answer:
Name:    www.google.com
Addresses:  2404:6800:4007:809::2004
          172.217.31.196

> _
```

5. Traceroute

In Windows, the tracert command is known as tracert. It is used to trace the. path packets take from your computer to a destination host, showing each hop along the way.

Syntax:> tracert www.google.com

Output

```
C:\Users\ACER>tracert www.google.com

Tracing route to www.google.com [142.250.193.132]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  172.16.16.16
  2    <1 ms    <1 ms    <1 ms  10.119.248.117
  3    39 ms    38 ms    39 ms  10.163.255.165
  4    38 ms    39 ms    40 ms  10.119.73.122
  5    42 ms    41 ms    42 ms  72.14.213.20
  6    41 ms    42 ms     *     142.251.227.211
  7    40 ms    46 ms    39 ms  142.251.55.227
  8    39 ms    39 ms    39 ms  maa05s25-in-f4.1e100.net [142.250.193.132]

Trace complete.
```

6. Ping

The ping command in Windows is used to test the connectivity between your computer and another device by sending ICMP Echo Requests and waiting for a response. It's a simple and effective tool for network troubleshooting.

Commands

-Ping a specific number of times:
> ping -n 10 google.com -Ping
until stopped: > ping -t
google.com -Specify packet size:
> ping -l 1000 google.com
-Don't resolve IP addresses to hostnames:
> ping -a google.com
-Set a Timeout
>ping -w 5000 google.com
-Force IPv4 or IPv6 ping
> ping -4 google.com
> ping -6 google.com

Output

```
C:\Users\cse>ping -l 1000 google.com

Pinging google.com [142.250.192.78] with 1000 bytes of data:
Reply from 142.250.192.78: bytes=1000 time=70ms TTL=59
Reply from 142.250.192.78: bytes=1000 time=71ms TTL=59
Reply from 142.250.192.78: bytes=1000 time=72ms TTL=59
Reply from 142.250.192.78: bytes=1000 time=71ms TTL=59

Ping statistics for 142.250.192.78:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 70ms, Maximum = 72ms, Average = 71ms
```

```
C:\Users\cse>ping -a google.com

Pinging google.com [142.250.196.78] with 32 bytes of data:
Reply from 142.250.196.78: bytes=32 time=40ms TTL=59
Reply from 142.250.196.78: bytes=32 time=40ms TTL=59
Request timed out.
Request timed out.

Ping statistics for 142.250.196.78:
    Packets: Sent = 4, Received = 2, Lost = 2 (50% loss),
Approximate round trip times in milli-seconds:
    Minimum = 40ms, Maximum = 40ms, Average = 40ms
```

```
C:\Users\cse>ping -w 5000 google.com

Pinging google.com [142.251.42.46] with 32 bytes of data:
Reply from 142.251.42.46: bytes=32 time=63ms TTL=59
Reply from 142.251.42.46: bytes=32 time=63ms TTL=59
Reply from 142.251.42.46: bytes=32 time=63ms TTL=59
Request timed out.

Ping statistics for 142.251.42.46:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 63ms, Maximum = 63ms, Average = 63ms
```

Result

Thus the commands were explored.

Ex No:2	HTTP WEB CLIENT PROGRAM TO DOWNLOAD A WEB PAGE USING TCP SOCKET

Aim

To download a webpage using TCP Socket.

Client side

1) Start the program.

2) Create a socket which binds the Ip address of server and the port address to acquire service.

3) After establishing connection send the URL to server.

4) Open a file and store the received data into the file.

5) Close the socket.

6) End the program.

Server side

1) Start the program.

2) Create a server socket to activate the port address.

3) Create a socket for the server socket which accepts the connection.

4) After establishing connection receive url from client.

5) Download the content of the url received and send the data to client.

6) Close the socket.

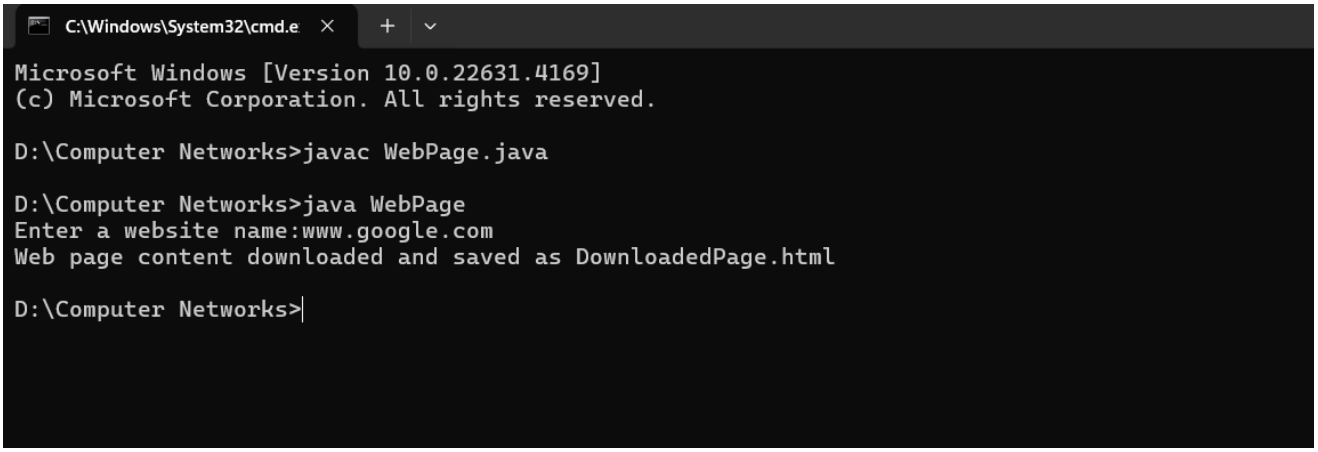
7) End the program.

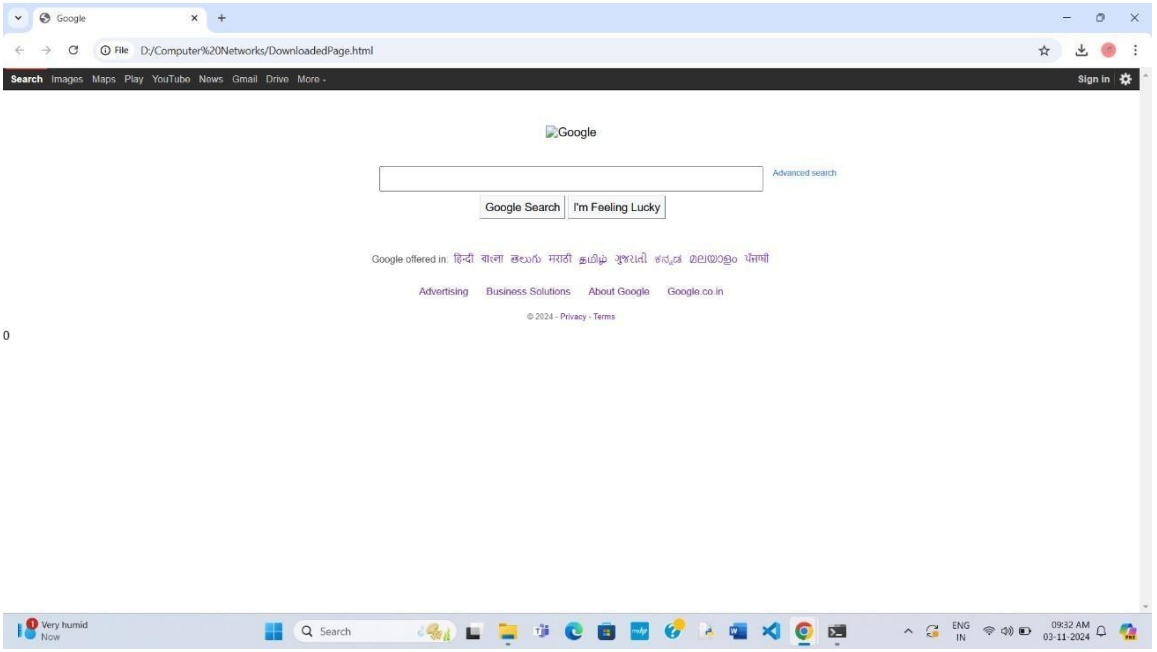
Program

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
public class WebPage{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter a website name:");
      String host = sc.nextLine();
      int port = 80; // HTTP default port
      String path = "/"; // Path of the web page
      try {
         Socket socket = new Socket(host, port);
         // Send HTTP request
         PrintWriter out = new PrintWriter(socket.getOutputStream());
         out.println("GET " + path + " HTTP/1.1");
         out.println("Host: " + host);
         out.println("Connection: close");
         out.println(); // Empty line to indicate end of headers
         out.flush();
         // Receive and save HTTP response content to a file
         BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
         String line;
         boolean foundDoctype = false;
         StringBuilder content = new StringBuilder();
         while ((line = in.readLine()) != null){
            if (line.contains("<!doctype html>")){
               foundDoctype = true;
            }
            if (foundDoctype) {
               content.append(line);
            }
         }
```

```
            // Close the streams and socket
            in.close();
            out.close();
            socket.close();
           // Save content to a file
            BufferedWriter writer = new BufferedWriter(new FileWriter("DownloadedPage.html"));
            writer.write(content.toString());
            writer.close();
            System.out.println("Web page content downloaded and saved as DownloadedPage.html");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT





Result

Hence downloading the web page using TCP socket was executed successfully.

Ex No:3 A	ECHO CLIENT AND ECHO SERVER USING TCP SOCKETS

Aim

To write a TCP socket program for implementation of echo.

Algorithm

Client side

1. Start the program.

2. Create a socket which binds the IP address of server and the port address to acquire service.

3. After establishing connection send a data to server.

4. Receive and print the same data from server.

5. Close the socket.

6. End the program.

Server side

1. Start the program.

2. Create a server socket to activate the port address.

3. Create a socket for the server socket which accepts the connection.

4. After establishing connection receive the data from client.

5. Print and send the same data to client.

6. Close the socket.

7. End the program.

Program

Server

```
import java.io.*;
import java.net.*;
public class MyServer {
public static void main(String[] args){
try{
ServerSocket ss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection
DataInputStream dis=new DataInputStream(s.getInputStream());
String str=(String)dis.readUTF();
System.out.println("message= "+str);
ss.close();
}
catch(Exception e){
System.out.println(e);
}
}
}
```

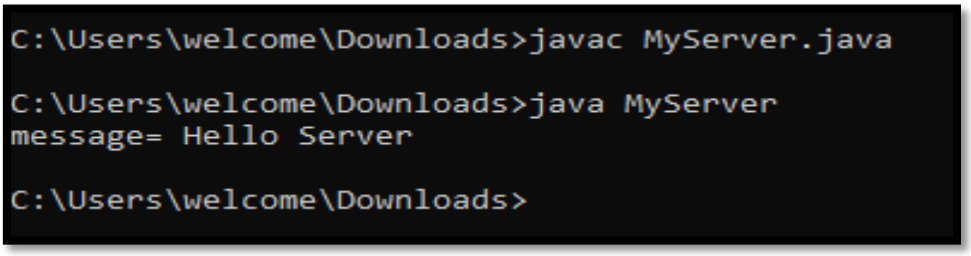
Client

```
import  java.io.*;
import java.net.*;
public class MyClient {
public static void main(String[] args) {
try{
Socket s=new Socket("localhost",6666);
DataOutputStream dout=new DataOutputStream
(s.getOutputStream());
dout.writeUTF("Hello Server");
```

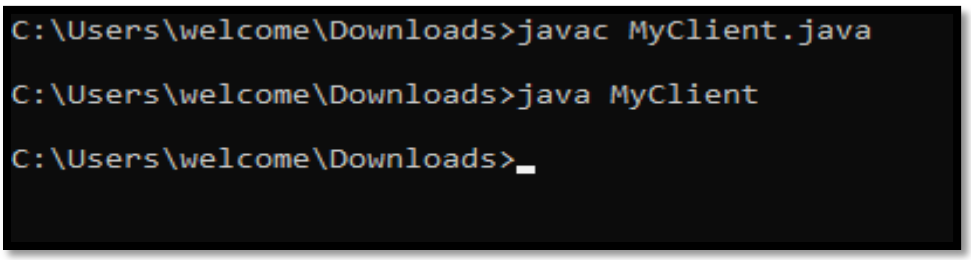
```
dout.flush();
dout.close();
s.close();
}
catch(Exception e){
System.out.println(e);
}
}
}
```

Output

Server



Client



Result

Thus the Echo Client and Echo Server using TCP Sockets in java has written and executed successfully.

Ex No:3 B	CHAT APPLICATION USING TCP SOCKETS

Aim

To write a client-server application for chat using TCP SOCKETS.

Algorithm

Client

1. Start the program
2. Include necessary package in java
3. To create a socket in client to server.
4. The client establishes a connection to the server.
5. The client accept the connection and to send the data from client to server.
6. The client communicates the server to send the end of the message
7. Stop the program.

Server

1. Start the program
2. Include necessary package in java
3. To create a socket in server to client
4. The server establishes a connection to the client.
5. The server accept the connection and to send the data from server to client
6. The server communicates the client to send the end of the message.
7. Stop the program

Program

Server

```
import java.util.*;
import java.net.*;
import java.io.*;
public class Server1 {
 public static void main(String[] args) {
 try
 {
 ServerSocket ss=new ServerSocket(3333);
 Socket s=ss.accept();
 String str,str2;
 DataInputStream din=new DataInputStream(s.getInputStream());
 DataOutputStream dout=new DataOutputStream(s.getOutputStream());
 Scanner m=new Scanner(System.in);
 System.out.println("Connection established");
 System.out.println("enter any data(enter stop to terminate)");
 do
 {
 str2=din.readUTF();
 System.out.println("Data from client:"+str2);
 str=m.next();
 dout.writeUTF(str);
 }while((str2.equals("stop"))==false);
 din.close();
 dout.close();
 s.close();
 ss.close();
 System.out.println("Connection closed.");
 }
```

```
 catch(Exception e)
 {
 System.out.println(e);
 }
 }
}
```

Client

```
import java.util.*;
import java.net.*;
import java.io.*;
public class Client1 {
 public static void main(String[] args) {
 try
 {
 Socket s=new Socket("localhost",3333);
 String str,str2;
 DataInputStream din=new DataInputStream(s.getInputStream());
 DataOutputStream dout=new DataOutputStream(s.getOutputStream());
 Scanner m=new Scanner(System.in);
 System.out.println("enter any data(enter stop to terminate)");
 do
 {
 str=m.next();
 dout.writeUTF(str);
 str2=din.readUTF();
 System.out.println("Data from server:"+str2+"\n");
 }while((str.equals("stop"))==false);
 din.close();
 dout.close();
 s.close();
```

```
}
catch(Exception e)
{
System.out.println(e);
}
}
}
```

Output

Server

```
C:\Users\welcome\Downloads>javac Server1.java

C:\Users\welcome\Downloads>java Server1
Connection established
enter any data(enter stop to terminate)
Data from client:Hello
Hiii
Data from client:Networks
TCP
Data from client:UDP
Sockets
Data from client:stop
stop
Connection closed.

C:\Users\welcome\Downloads>
```

Client

```
C:\Users\welcome\Downloads>javac Client1.java

C:\Users\welcome\Downloads>java Client1
enter any data(enter stop to terminate)
Hello
Data from server:Hiii

Networks
Data from server:TCP

UDP
Data from server:Sockets

stop
Data from server:stop


C:\Users\welcome\Downloads>
```

Result

Hence the chat application using TCP Sockets in java has written and executed successfully.

Ex No : 4	SIMULATION OF DNS USING UDP SOCKETS

Aim

To write a java program to simulate DNS using UDP sockets.

Algorithm

Server

i. Start the server program.
ii. Create a Datagram socket and a buffer (byte array).
iii. Initialize position to 0.
iv. Loop through the data:
    a. If the character at the current position is not null or newline.
v. Create a Datagram Packet to send the character to the client. Send the packet.
vi. Increment the position.
vii. Close the Datagram socket and stop the server.

Client

i. Start the client program.
ii. Create a Datagram socket and a buffer (byte any) for receiving data.
Step3 Loop to receive data.
    1. Create a Datagram Socket to receive data from the server
    2. Receive the packet from the server.
    3. Print the received data.
iii. Close the Datagram socket and stop the client.

Program

Server

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class Server {
	static String[] hosts = {
"zoho.com", "gmail.com", "netflix.com", "google.com", "facebook.com",
"amazon.com", "leetcode.com", "geeksforgeeks.com", "youtube.com",
"hope3.org", "stackoverflow.com", "hackerrank.com", "codeforces.com",
"codechef.com"
};
	static String[] ipAddresses = {
"192.168.1.1", "192.168.1.2", "192.168.1.3", "192.168.1.4", "192.168.1.5",
"192.168.1.6", "192.168.1.7", "192.168.1.8", "192.168.1.9", "192.168.1.10",
"192.168.1.11", "192.168.1.12", "192.168.1.13", "192.168.1.14",
"192.168.1.15"
};
public static int search(String key) {
	for (int i = 0; i < hosts.length; i++) {
		if (key.equals(hosts[i])) {
			return i;
		}
	} return -1;
}
```

```
    public static void main(String[] args) {
        try{
            DatagramSocket ds = new DatagramSocket();
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter host name: ");
            String str = sc.nextLine();
            int strIndex = search(str);
            InetAddress ip = InetAddress.getLocalHost();
            String message = strIndex != -1 ? ipAddresses[strIndex]
        :"Host_name_not_found";
            DatagramPacket dp = new
        DatagramPacket(message.getBytes(), message.length(), ip, 3000);
            ds.send(dp);
            ds.close();
        }
         catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

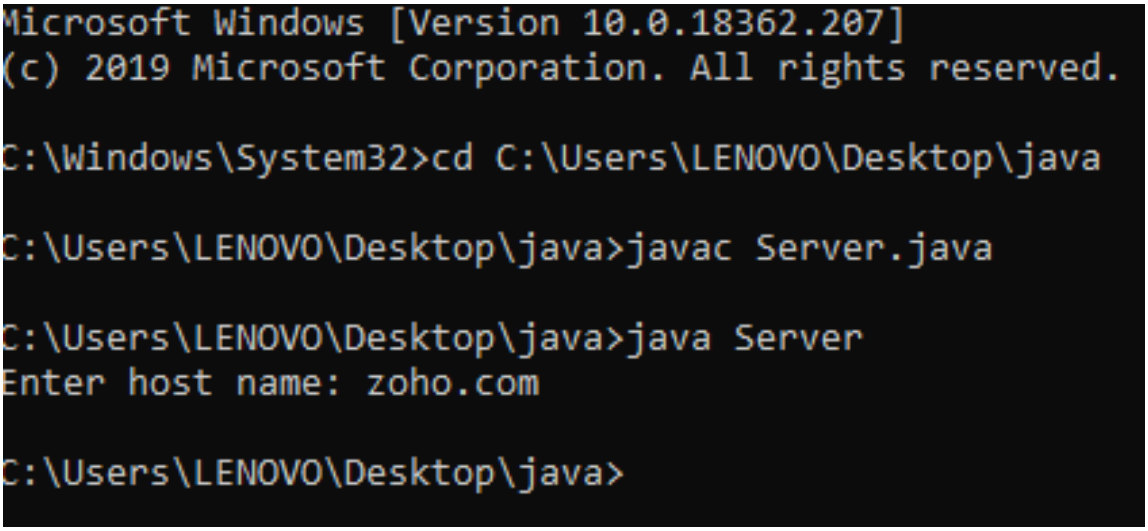
Client

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
public class Client {
    public static void main(String[] args) {
```

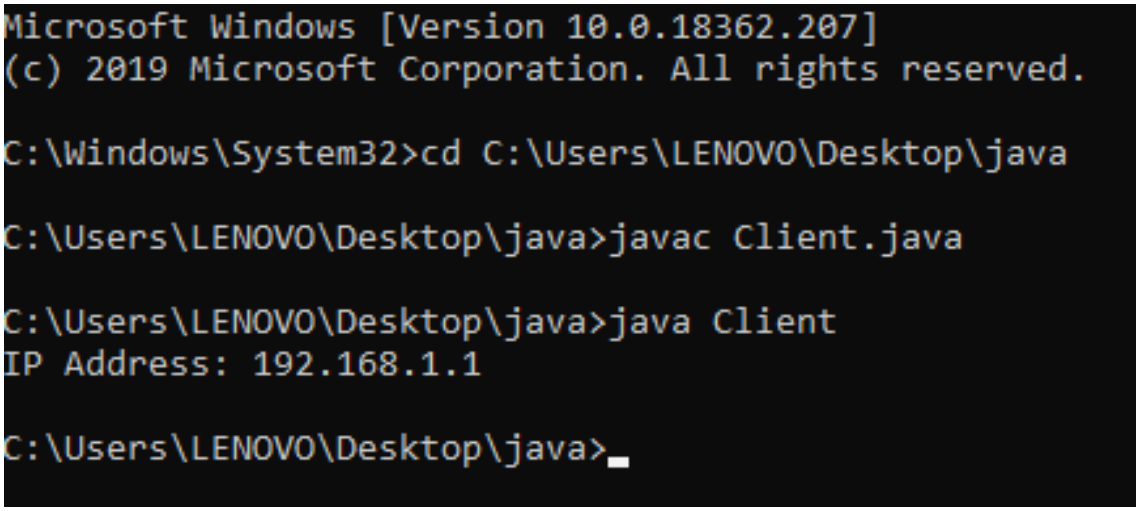
```
            try{
                    DatagramSocket ds = new DatagramSocket(3000);
                    byte[] arr = new byte[30];
                    DatagramPacket dp = new DatagramPacket(arr, arr.length);
                    ds.receive(dp);
                    String str = new String(dp.getData()).trim();
                    System.out.println("IP Address: " + str);
                    ds.close();
            }
             catch (Exception e) {
                    System.out.println(e);
            }
        }
}
```

Output

Server



Client



Result

Thus a java program to simulate DNS using UDP sockets is executed successfully.

Ex No : 5	USING WIRESHARK TO CAPTURE AND EXAMINE NETWORK PACKETS

Introduction

Wireshark is a powerful open-source network protocol analyzer used to capture and analyze network traffic in real-time. By capturing packets, network administrators and security professionals can gain insights into network behavior, troubleshoot issues, and detect potential security threats.

Installation Steps

1. Download Wireshark :

- Visit the official Wireshark website at https://www.wireshark.org/.
- Navigate to the "Download" section.
- Choose the appropriate installer for your operating system (Windows, macOS, or Linux).
- Follow the on-screen instructions to download the installer.

2. Install Wireshark :

- Locate the downloaded installer and run it.
- Follow the installation wizard's prompts:
  - Choose installation options (e.g., desktop shortcuts).
  - Accept the license agreement.
  - Select installation location.
  - Complete the installation process.

Capturing Packets

1. Open Wireshark

- Launch Wireshark from the desktop shortcut or start menu.

2. Select Network Interface

- In the main window, you'll see a list of available network interfaces (e.g., Ethernet, Wi-Fi).
- Choose the relevant interface for capturing packets.

3. Start Packet Capture

- Click on the chosen interface.
- Click the "Start" button or press "Ctrl + E" to begin capturing packets on that interface.

Examining Packets

1. Packet List

- The main window displays a list of captured packets.
- Columns show source and destination IPs, protocols, packet lengths, and more.

2. Packet Details

- Select a packet in the list to see detailed information.
- The packet details area dissects the packet's protocol layers.
- Inspect the layers for protocol-specific information and payload data.

3. Filtering Packets

- Use display filters to focus on specific types of packets.
- Enter a filter expression in the display filter field at the top of the window and press Enter.

Analyzing Packets

1. Identifying Anomalies

- Look for unusual traffic patterns, such as sudden spikes or frequent connections.
- Pay attention to communication between unfamiliar IP addresses.

2. Protocol Behavior

- Analyze protocol behavior for deviations from normal patterns.
- Investigate unexpected responses, errors, or repeated requests.

3. Malware Indicators

- Examine packet payloads for strings or patterns that may indicate malware activity.
- Look for suspicious commands, URLs, or file transfers.

Uses of Capturing Packets

Packet capture with Wireshark has several important uses, including:

1. Network Troubleshooting and Diagnostics

- Identify and resolve network performance issues, latency problems, and dropped connections.
- Analyze network behavior to pinpoint the source of problems.

2. Security Analysis and Intrusion Detection

- Detect unauthorized access attempts and security breaches.
- Monitor for suspicious traffic patterns and potential threats.

3. Vulnerability Assessment

- Identify vulnerabilities in network systems and applications

- Analyze traffic to uncover potential weaknesses that can be exploited

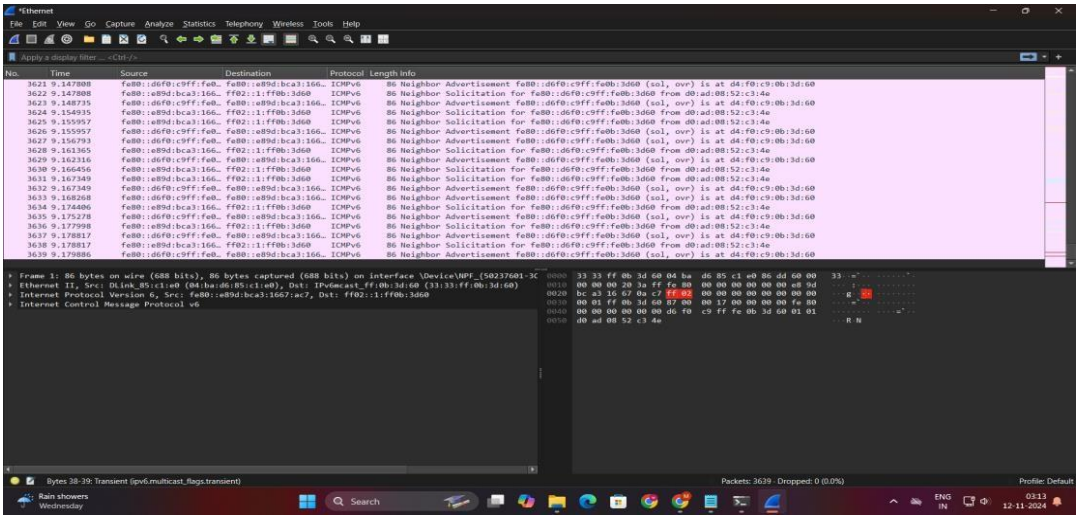
4. Quality of Service (QoS) Monitoring

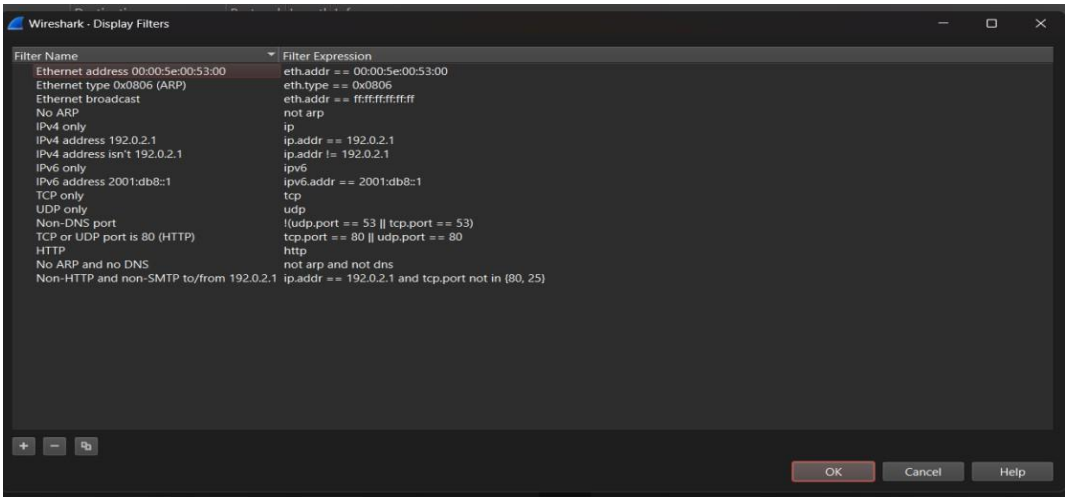
- Monitor bandwidth usage, packet loss, and latency to maintain optimal network performance.
- Ensure that critical applications receive the necessary resources.

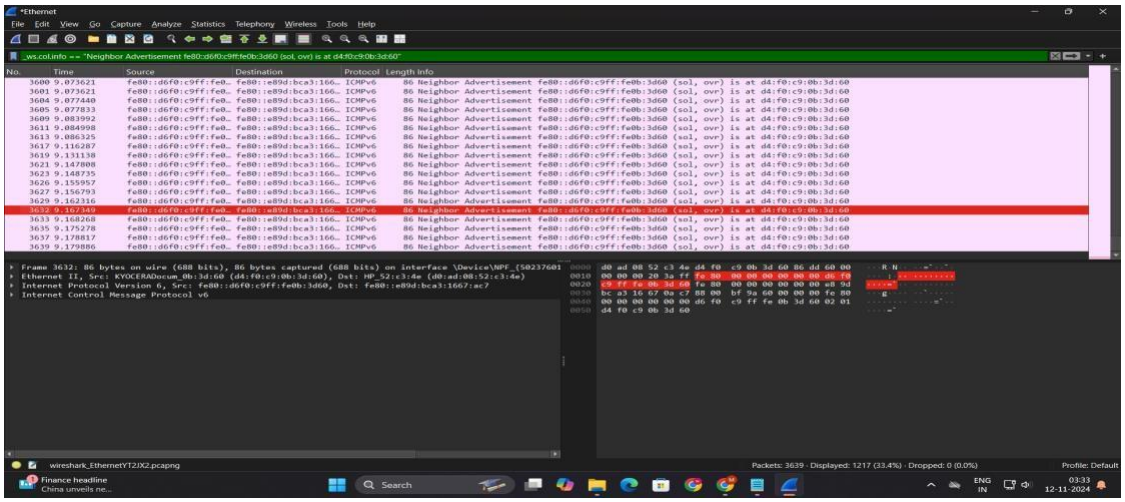
5. Forensics and Incident Response

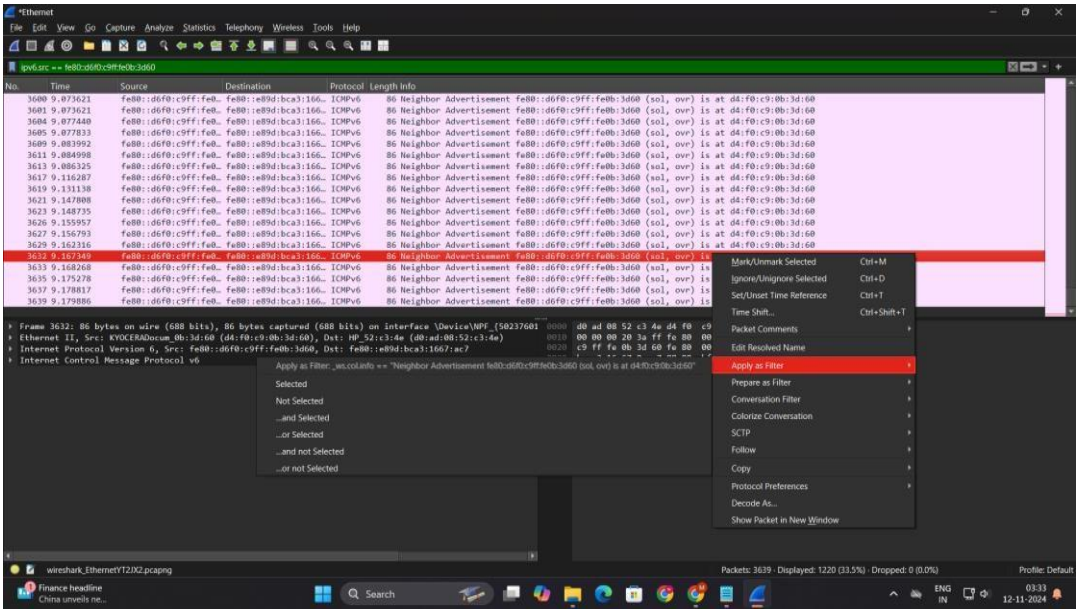
- Use captured packet data as evidence during incident investigations.
- Reconstruct events leading up to security incidents.

Output









Conclusion

Wireshark is an invaluable tool for capturing and analyzing network packets. It aids in diagnosing network issues, ensuring security, and identifying potential threats. Responsible and ethical use of Wireshark is essential to maintain privacy and comply with legal considerations.

Ex No:6A	SIMULATING ADDRESS RESOLUTION PROTOCOL

Aim

To write a java program for simulating ARP protocol.

Algorithm

Server

1. Start the program.
2. Accept the socket which is created by the client.
3. Server maintains the table in which IP and corresponding MAC addresses are stored.
4. Read the IP address which is send by the client.
5. Map the IP address with its Mac address and return the MAC address to client.

Client

1. Start the program.
2. Using socket connection is established between client and server.
3. Get the IP address to be converted into MAC address.
4. Send this IP address to server.
5. Server returns the MAC address to client.

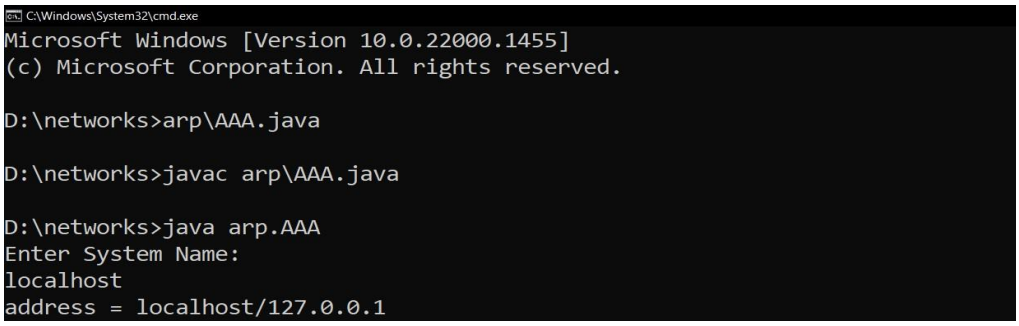
Program

```
package arp;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.Scanner;
public class AAA {
 public static void main(String[] args)
 {
```

```
 try
{
 Scanner console = new Scanner(System.in);
 System.out.println("Enter System Name: ");
 String ipaddr = console.nextLine();
 InetAddress address = InetAddress.getByName(ipaddr);
 System.out.println("address = "+address);
 NetworkInterface ni = NetworkInterface.getByInetAddress(address);
 if (ni!=null)
 {
 byte[] mac = ni.getHardwareAddress();
 if (mac != null)
 {
 System.out.print("MAC Address : ");
 for (int i=0; i < mac.length; i++)
 {
 System.out.format("%02X%s", mac[i], (i < mac.length - 1) ? "-" :"");
 }
 }
else
 {
 System.out.println("Address doesn't exist or is not accessible/");

 }
 }
 else
 {
 System.out.println("Network Interface for the specified address is not found");
 }
 }
 catch(UnknownHostException | SocketException e)
 {
 }
```

}

Output



Result

A java program for simulating ARP protocol was written,executed and the output was verified successfully.

Ex No:6 B	SIMULATING REVERSE ADDRESS RESOLUTION PROTOCOL

Aim

To write a java program for simulationg RARP protocol.

Algorithm

Server

1. Start the program.
2. Server maintains the table in which IP and corresponding MAC addreddes are stored.
3. Read the MAC address which is send by the client.
4. Map the IP address with its MAC address and return the IP address to client.
5. Stop the program.

Client

1. Start the program.
2. Using datagram sockets UDP function is established.
3. Get the MAC address to be converted into IP address.
4. Sernd this MAC address to server.
5. Server returns the IP address to client.
6. Stop the program.

Program

Server

```
import java.io.*;
import java.net.*;
import java.util.*;
class Serverrarp{
```

```
public static void main(String args[]){
try{
DatagramSocket server=new DatagramSocket(1309);
while(true){
byte[] sendbyte=new byte[1024];
byte[] receivebyte=new byte[1024];
DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);
server.receive(receiver);
String str=new String(receiver.getData());
String s=str.trim();
InetAddress addr=receiver.getAddress();
int port=receiver.getPort();
String ip[]={"165.165.80.80","165.165.79.1"};
String mac[]={"6A:08:AA:C2","8A:BC:E3:FA"};
for(int i=0;i<ip.length;i++){
if(s.equals(mac[i])){
sendbyte=ip[i].getBytes();
DatagramPacket sender=new
DatagramPacket(sendbyte,sendbyte.length,addr,port);
server.send(sender);
break;}}
break;}}
catch(Exception e){
System.out.println(e);
}
}
}
```

Client

```
import java.io.*;
import java.net.*;
import java.util.*;
class Clientrarp{
```

```
public static void main(String args[]){try{
DatagramSocket client=new DatagramSocket();
InetAddress addr=InetAddress.getByName("127.0.0.1");
byte[] sendbyte=new byte[1024];
byte[] receivebyte=new byte[1024];
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the Physical address (MAC):");
String str=in.readLine(); sendbyte=str.getBytes();
DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,1309);
client.send(sender);
DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);
client.receive(receiver);
String s=new String(receiver.getData());
System.out.println("The Logical Address is(IP): "+s.trim());
client.close();
}
catch(Exception e){
System.out.println(e);
}
}
}
```

Output

Server

Client

Result

A java program for simulating RARP protocol was written,executed and the output was verified successfully.

Ex No:7A	STUDY OF NETWORK SIMULATOR (NS)

Aim

To explore Cisco Packet Tracer for network simulation and lab setup.

Introduction to Cisco Packet Tracer

A. Overview and Significance

- **Definition:** Cisco Packet Tracer is a robust network simulation tool developed by Cisco Systems, designed for teaching and learning complex networking concepts.
- **Importance in Lab Settings:** Provides a virtual, risk-free environment for hands-on learning, allowing students to experiment with network configurations and troubleshoot issues.

B. Key Features of Cisco Packet Tracer

1. **Wide Device Support:** Packet Tracer supports a variety of devices, including routers, switches, hubs, PCs, servers, and more.
2. **Realistic Simulation:** Offers real-time simulation of network behaviors, allowing students to observe the effects of their configurations instantly.
3. **Protocols and Applications:** Simulates protocols like TCP/IP, DHCP, and DNS, along with applications like web browsers and email clients.
4. **Multi-User Collaboration:** Supports multi-user mode, enabling collaborative learning and group projects.
5. **Assessment and Grading:** Allows instructors to create assessments, track student progress, and grade their performance.
6. **Customization:** Users can create custom devices, enhancing the range of network scenarios that can be simulated.

Setting Up Cisco Packet Tracer Lab Environment

Installation and Configuration

- Go to the Cisco Networking Academy website (https://www.netacad.com/).
- Sign in with your account or create a new account.
- Find the "Packet Tracer" section.
- Click on "Download" or "Get Packet Tracer."
- Choose your operating system (Windows, macOS, Linux).

• Accept the license agreement.

• Wait for the download to complete.

• Locate the downloaded file and install Packet Tracer.

• Launch the application.

• Sign in with your Cisco Networking Academy account if prompted.

Exploring Packet Tracer Interface Main Components

i. Device Palette:

• **Routers:** Represented with icons resembling actual Cisco routers.

• **Switches:** Icons vary based on the type of switch (Layer 2 or Layer 3).

• **Hubs:** Depicted as circular icons.

• **End Devices:** PCs, laptops, servers, and phones, represented with appropriate icons.

ii. Physical Workspace:

• **Grid Layout:** Devices can be placed on a grid representing the physical workspace.

• **Drag-and-Drop Functionality:** Devices can be dragged from the Device Palette and dropped onto the workspace.

iii. Toolbar:

• **Select Tool:** Allows selection of devices and connections for configuration.

• **Place Tool:** Used to place devices onto the workspace.

• **Connection Types:** Various connection types (straight-through, crossover, etc.) can be selected based on the devices being connected.

iv. Menu Bar:

• **File:** Options for creating, opening, and saving Packet Tracer files.

• **Edit:** Functions for undo, redo, and managing selections.

• **Options:** Customization options for Packet Tracer's behavior and appearance.

• **Help:** Access to Packet Tracer tutorials, guides, and support resources.

Basic Networking Configurations and Troubleshooting

Configuring Basic Network Device

i. Connecting Switches:

• **Step 1:** Drag a switch from the Device Palette and drop it onto the workspace.

• **Step 2:** Use straight - through Ethernet cables to connect multiple switches.

- **Step 3:** Click on the switch, select a port, click on another switch, and select a port to establish a connection.

ii. Connecting Routers and End Devices:

- **Step 1:** Drag a router and end devices (PCs, laptops) onto the workspace.
- **Step 2:** Use appropriate cables to connect PCs to router ports and configure IP addresses on router interfaces.
- **Step 3:** Configure IP addresses on end devices within the same subnet as the router interface for connectivity.

iii. Connecting Hubs and End Devices:

- **Step 1:** Drag a hub and end devices onto the workspace.
- **Step 2:** Use straight-through Ethernet cables to connect end devices to hub ports.
- **Step 3:** Configure IP addresses on end devices within the same subnet for communication through the hub.

Troubleshooting Basics

- **Ping and Traceroute:** Hands-on practice on using ping and traceroute commands to troubleshoot network issues.
- **Debugging Tools:** Introduction to Packet Tracer's built-in debugging tools for diagnosing problems.

Advanced Networking Configurations

VLAN and Inter-VLAN Configurations

- **VLAN Setup:** Creating and configuring Virtual LANs to segment a network.
- **Inter-VLAN Routing:** Setting up routers to enable communication between different VLANs.

DHCP and NAT Configurations

- **DHCP Setup:** Configuring routers or servers to provide IP addresses dynamically.
- **NAT Configuration:** Implementing Network Address Translation for internet connectivity.

Access Control and Security Implementations

- **Access Control Lists (ACLs):** Implementing ACLs to control traffic flow based on source and destination IP addresses.
- **Firewall Configuration:** Configuring firewalls for network security.

Real-World Simulations and Projects

Multi-Site Network Setup

- **Branch Office Connections:** Configuring VPNs and secure communication channels between branch offices.
- **Data Center Design:** Designing a data center network with redundancy and high availability.

Performance Optimization and QoS

- **Traffic Analysis:** Analyzing network traffic patterns and optimizing network performance.
- **Quality of Service (QoS):** Implementing QoS to prioritize critical network traffic.

Advanced Security Scenarios

- **Intrusion Detection:** Configuring intrusion detection systems for network security.
- **Security Best Practices:** Implementing security best practices to safeguard the network infrastructure.

Conclusion

Cisco Packet Tracer is an invaluable tool for networking education. It provides hands-on learning experience, allowing users to design and simulate intricate network setups. With its user-friendly interface and powerful features, it equips learners with essential skills, making complex networking concepts accessible. By fostering practical understanding and problem-solving abilities, Packet Tracer empowers users to excel in real-world networking scenarios, making it an essential resource for students and professionals alike.

Ex No 7B	SIMULATION OF CONGESTION CONTROL ALGORITHMS

Aim

To write a program to simulate congestion control algorithm-stop and wait protocol.

Algorithm

Step 1: Read the number of frames to be sent.

Step 2: Sender establish connection with the receiver.

Step 3: After the connection is established, the sender send the frames one by one.

Step 4: The next frame is sent only when the acknowledgement for the previous frame is received.

Step5: Disconnect the connection after all the frames are received by the receiver.

Program

//Sender

```
import java.io.*;
import java.net.*;
public class Sender{
Socket sender;
ObjectOutputStream out;
ObjectInputStream in;
String packet, ack, str, msg;
int n, i=0, sequence=0;
Sender(){}
public void run(){
```

```
try{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Waiting  for Connection. ..");
Sender = new Socket("localhost" ,2004);
sequence=0;
out = new ObjectOutputStream(sender.getOutputStream());
out.flush();
in=new ObjectInputStream(sender.getInputStream());
str=(String)in.readObject();
System.out.println("receiver  >"+str);
System.out.println("Enter  the data to send...");
Packet=br.readLine();
n=packet.length();
do{
try{
if(i<n){
msg=String.valueOf(sequence);
msg=msg.concat(packet, substring(i,,i+1));
}
else if (i==n){
msg="end";out.writeObject(msg);break;
}
out.writeObject(msg);
```

```
 sequence=(sequence==0)?1:0;
out.flush();
System.out.println("data  sent>"+msg);
ack=(String)in.readObject();
System.out.println("waiting  for ack…\n\n");
if(ack.equals(String.valueOf(sequence))){
i++;
System.out.println("receiver  >"+"packet received\n\n");
}
else{
System.out.println("Time  out resending data…\n\n");
sequence=(sequence==0)?1:0;
}
}catch(Exception e){}
}{ while(i<n+1);
System.out.println("All  data sent. exting.");
}catch(Exception e){}
finally{
try{
in.close();
out.close();
sender.close()
}
```

```
catch(Exception e){}
}
}
public static void main(String args[]){
Senders s= new Sender();
s.run();
}
}
```

//Receiver

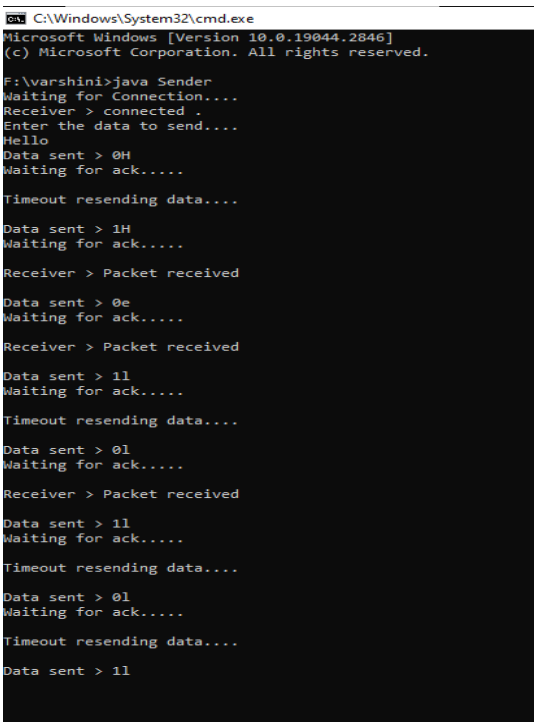
```
import java.io.*;
 import java.net.*;
 public class Receiver{
 ServerSocket receiver;
Socket connection=null;
ObjectOutputStream out;
ObjectInputStream  in;
String packet,ack,data="";
int i=0,sequence=0;
Receiver(){
public void run(){
try{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
receiver = new ServerSocket(2004,10);
```

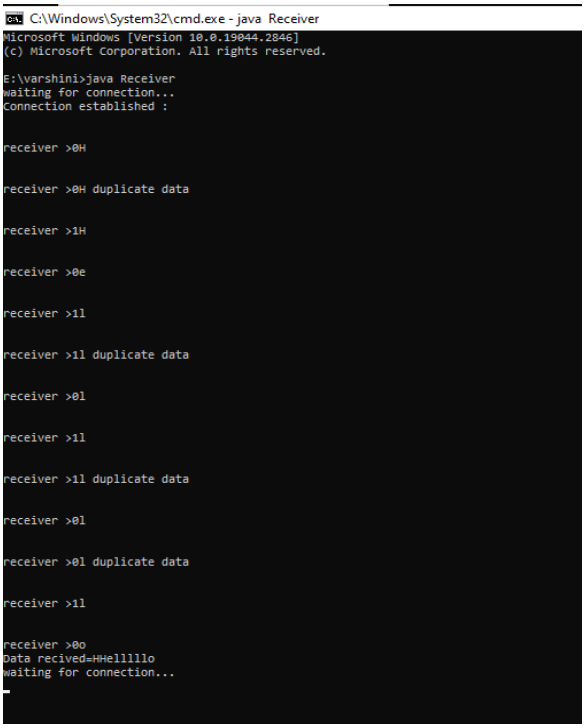
```
System.out.println("waiting  for connection...");
Connection=receiver.accept();
sequence=0;
System.out.println("Connection  established:");
out=new ObjectOutputStream(connection.getOutputStream());
out.flush();
in=new ObjectInputStream(connection.getInputStream());
out.writeObject("connected  .");
do{
try{
packet=(String)in.readObject();
if (Integer.valueOf(packet.substring(0,1))==sequence)  {
data+=packet.substring(1);
sequence= (sequence==0)?1:0;
System.out.println("\n\nreceiver  >"+packet);
}
else
{
System.out.println("\n\nreceiver 1 >"+packet+" duplicate data");
}
if(i<3){
out.writeObject(String.valueOf(sequence));i++;
}
```

```
else{
out.writeObject(String.valueOf(sequence+1)%2));
i=0;
}
}
catch(Exception e){}
}while(!packet.equals("end"));
System.out.println("Data recived="+data);
out.writeObject("connection ended .");
}
catch(Exception e){}
finally{
try{
in.close();
out.close();
receiver.close();
}
catch(Exception e){}
}
}
public static void main(String args[]){
Receiver s=new Receiver();
while(true){
```

```
s.run();

}

}

}
```

Output





Result

Thus the program to simulate congestion control algorithm – stop and wait protocol is executed successfully.

Ex No: 7C	AIMD – CONGESTION CONTROL ALGORITHM

Aim

To write a java program to perform the Congestion Control Algorithm

Algorithm

Step 1: Start.

Step 2: Read the initial congestion window size , probability of receiving an acknowledgment and probability of packet loss from the user.

Step3: Set the initial threshold ssthresh to Integer.MAX_VALUE ,Set the acknowledgment count ackCount to 0.

Step 4: Send packets up to the current congestion window size cwnd.

Step 5: Simulate acknowledgment reception:

Step 5.1: If acknowledgment is received based on ackProbability, increment ackCount.

Step 5.2: If acknowledgment is not received, simulate packet loss and perform the following:

Set ssthresh to half of cwnd, but at least 1.

reset cwnd to 1.

Reset `ackCount` to 0.

Step 6: If ackCount is greater than or equal to cwnd then,Increment cwnd by 1 and reset ackCount to 0.

Step 7: Check if cwnd has reached or exceeded the threshold ssthresh:

Step 7.1: If cwnd $\geq$ ssthresh, enter the Congestion Avoidance phase.

Step 7.2: In Congestion Avoidance, if packet loss occurs based on lossProbability, reduce cwnd by half multiplicative decrease.

Step 8: If cwnd is below ssthresh, continue the Slow Start phase , increment cwnd by 1 for each acknowledgment received until ssthresh is reached.

Step 9: Simulate a time interval sleep for 1 second and Repeat from Step 4.

Step 10: Stop.

Program

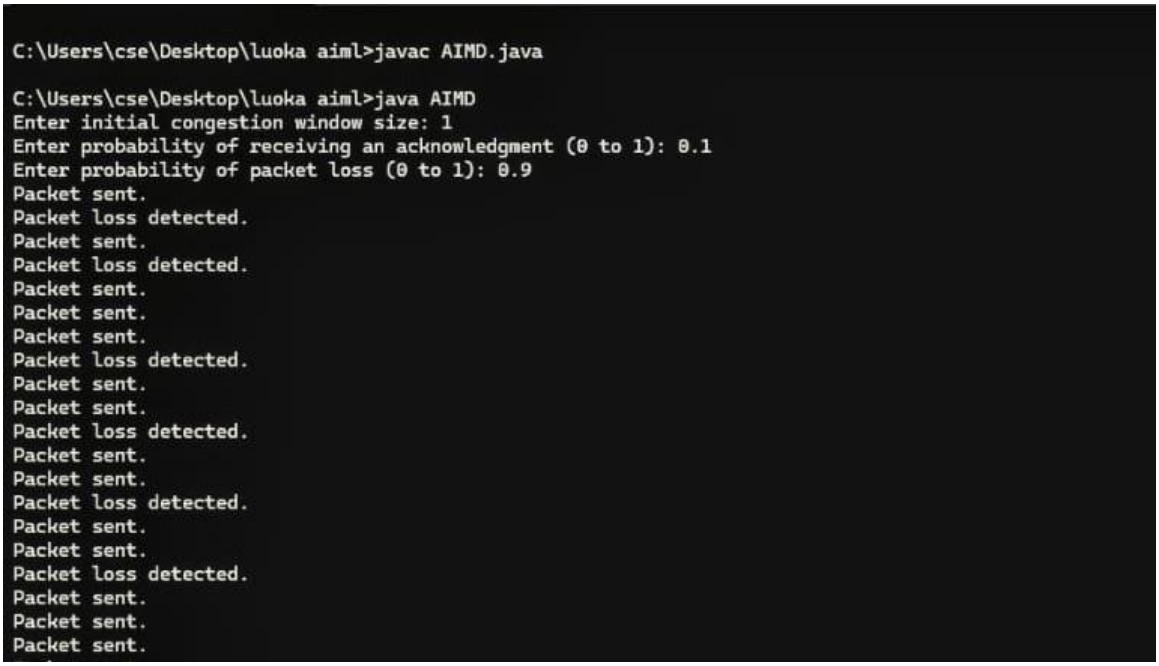
```
import java.util.Random;

import java.util.Scanner;

public class AIMD {

   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);

      // Get initial congestion window size from user

      System.out.print("Enter initial congestion window size: ");

      int cwnd = scanner.nextInt();

      System.out.print("Enter probability of receiving an acknowledgment (0 to 1): ");

      double ackProbability = scanner.nextDouble();

      System.out.print("Enter probability of packet loss (0 to 1): ");

      double lossProbability = scanner.nextDouble();

      int ssthresh = Integer.MAX_VALUE; // Initial threshold set to maximum value

      int ackCount = 0; // Number of acknowledgments received

      Random random = new Random();

      System.out.println("\nSimulation started. Press Ctrl+C to terminate.\n");

      // Simulate AIMD

      while (true) {
```

```
System.out.println("Current cwnd: " + cwnd + ", ssthresh: " + ssthresh);
// Send packets up to the current congestion window size
for (int i = 0; i < cwnd; i++) {
    sendPacket();
}
// Simulate acknowledgment reception
if (random.nextDouble() <= ackProbability) {
    System.out.println("Acknowledgment received.");
    ackCount++;
    // Additive increase
    if (ackCount >= cwnd) {
        cwnd++;
        ackCount = 0;
    }
}
// Simulate packet loss
if (random.nextDouble() <= lossProbability) {
    handlePacketLoss(cwnd, ssthresh);
    ssthresh = cwnd / 2; // Set new threshold
    cwnd = Math.max(1, cwnd / 2); // Multiplicative decrease
    ackCount = 0; // Reset acknowledgment count
}
// Slow Start and Congestion Avoidance
```

```
        if (cwnd < ssthresh) {
            // Slow Start Phase: Additive increase
            continueAdditiveIncrease(cwnd);
        } else {
            // Congestion Avoidance Phase
            System.out.println("Congestion Avoidance Phase.");
        }
        // Simulate time passing
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            System.out.println("Simulation interrupted.");
            break;
        }
    }
    scanner.close();
}
private static void sendPacket() {
    System.out.println("Packet sent.");
}
private static void handlePacketLoss(int cwnd, int ssthresh) {
    System.out.println("Packet loss detected! Reducing cwnd and updating ssthresh.");
}
```

```
    private static void continueAdditiveIncrease(int cwnd) {
        System.out.println("Slow Start Phase: Increasing cwnd additively.");
    }
}
```

Output



Result

Thus the java program to implement AIMD congestion control algorithm was written, executed and the output is verified.

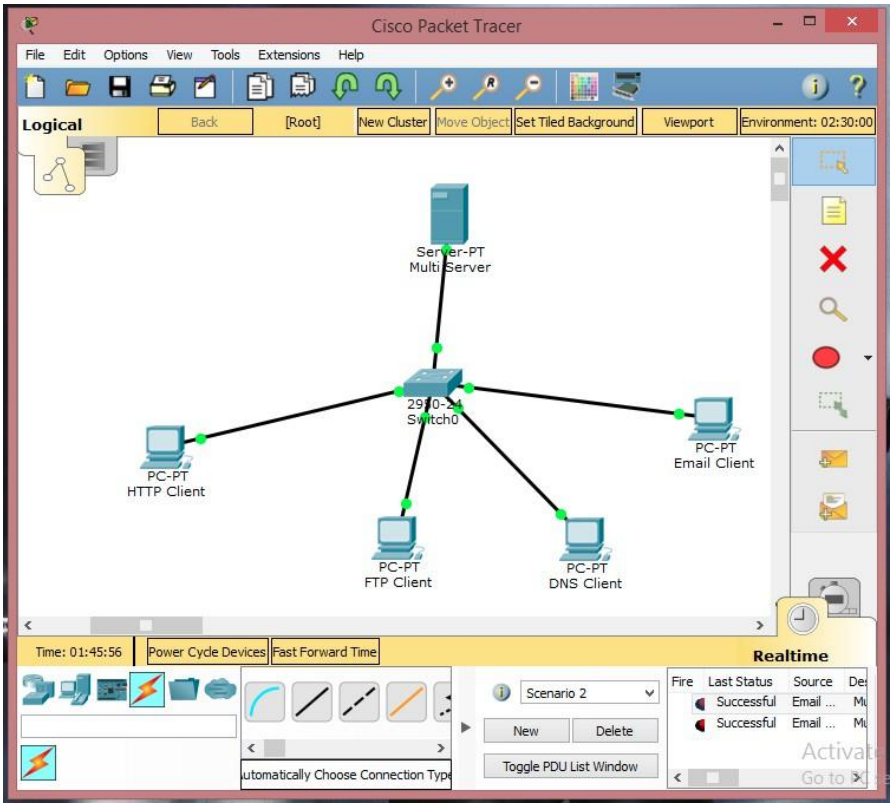
Ex No:8 A	STUDY OF TCP/UDP PERFORMANCE USING SIMULATION TOOL

Aim

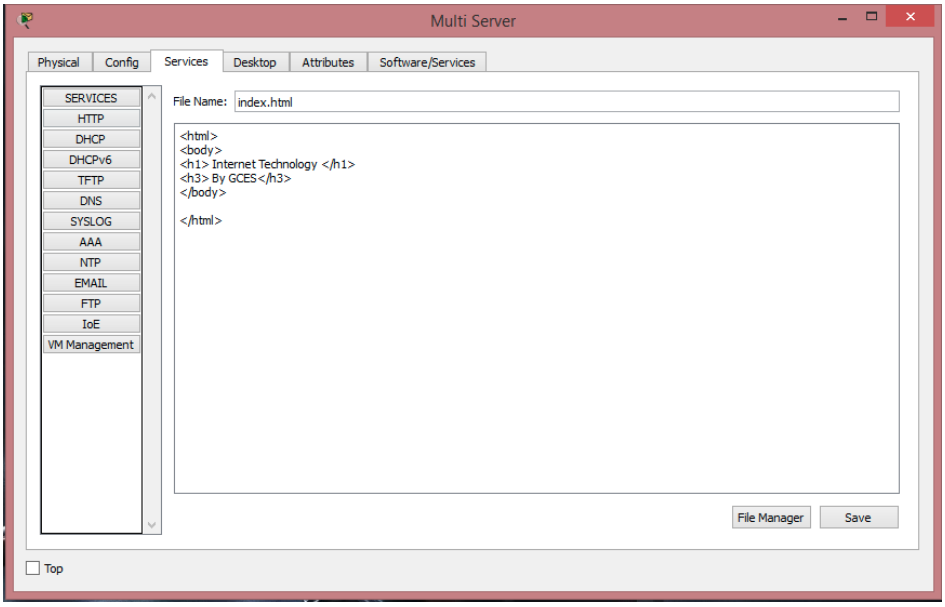
To Study of TCP/UDP performance using Simulation tool .

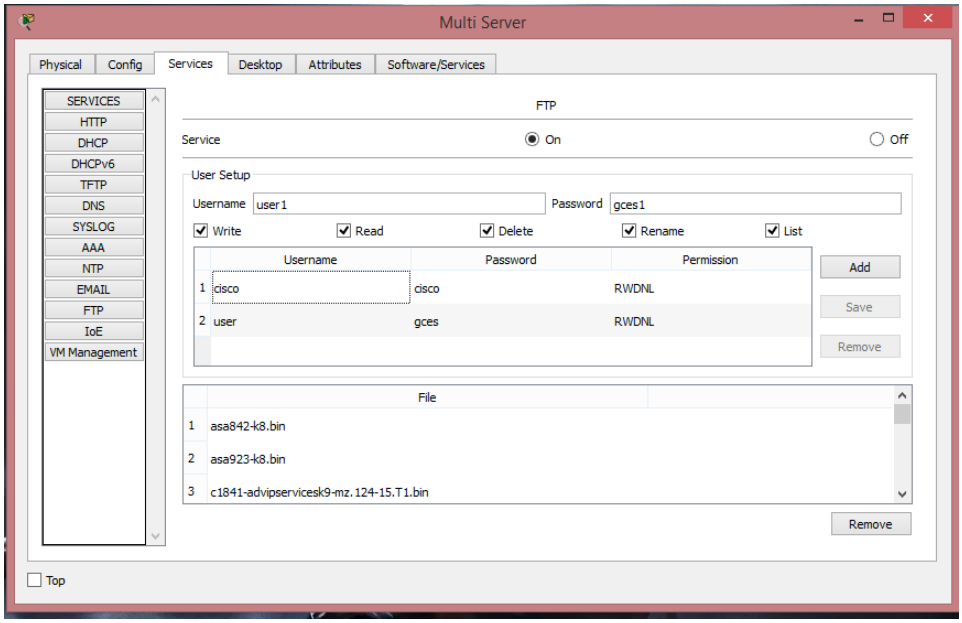
Procedure

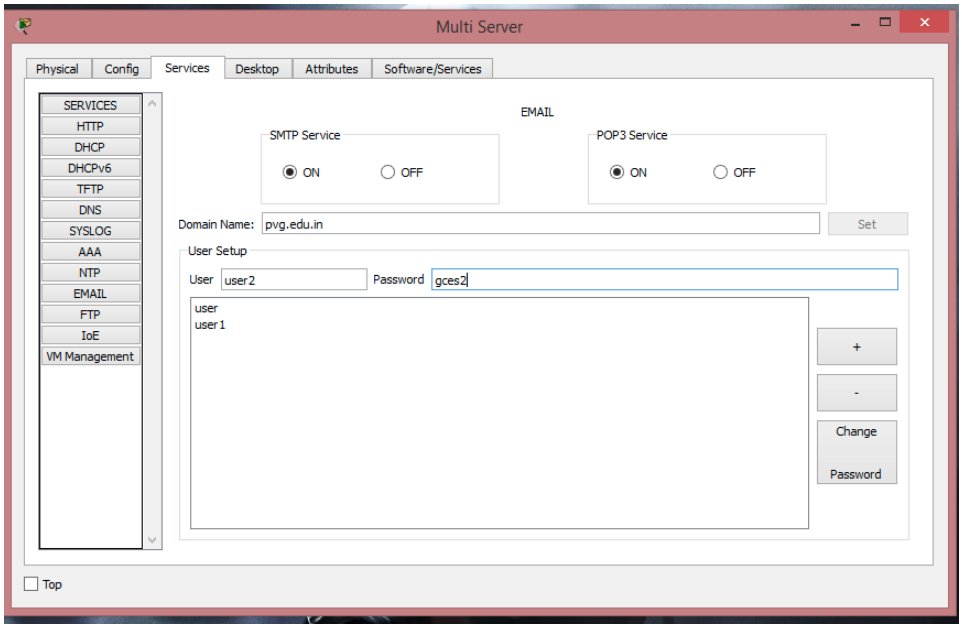
1. Develop a Topology shown in figure given below.
2. Add HTTP Client, FTP Client, DNS Client, and Email Client connected to a multi-server through the switch.
3. Configure the network.
4. Add users to the multi-server and create domain name.

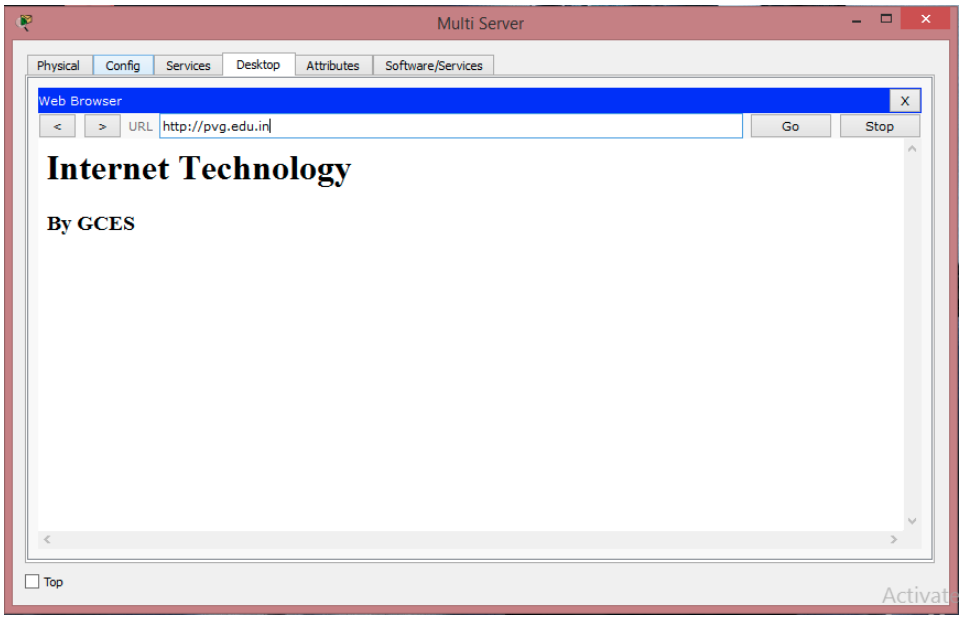


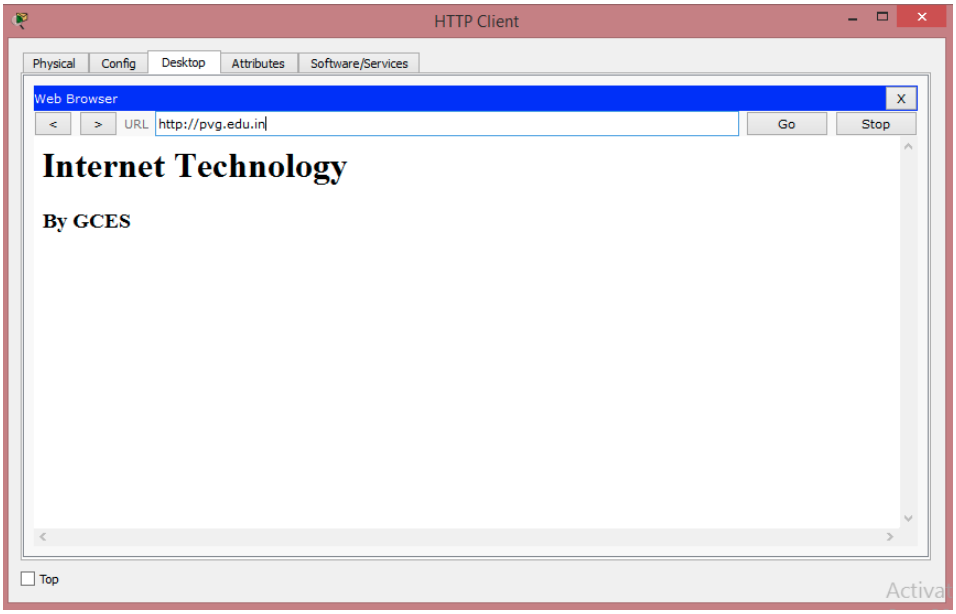
5. Edit index.html file using the code shown in figure. Run the Code by giving the domain in multi- server and HTTP client.
6. Connect the pvg.edu.in domain using ftp server.
7. Compose, send and receive mail between the clients connected to the server.

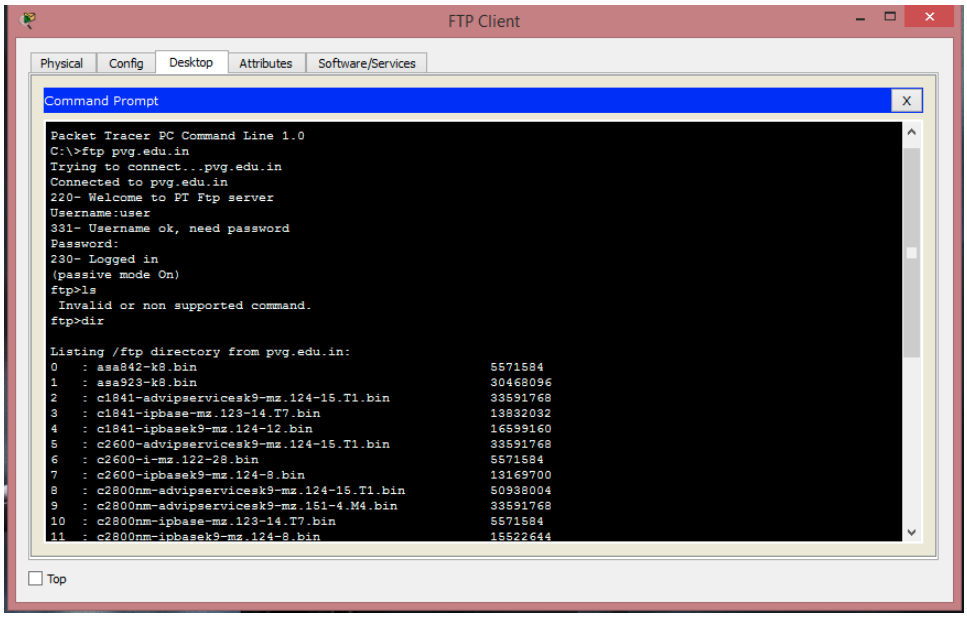

Multi Server
Physical
Config
Services
Desktop
Attributes
Software/Services
SERVICES
HTTP
DHCP
DHCPv6
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
IoE
VM Management
File Name: index.html
<html>
<body>
<h1> Internet Technology </h1>
<h3> By GCES</h3>
</body>
</html>
File Manager
Save
Top

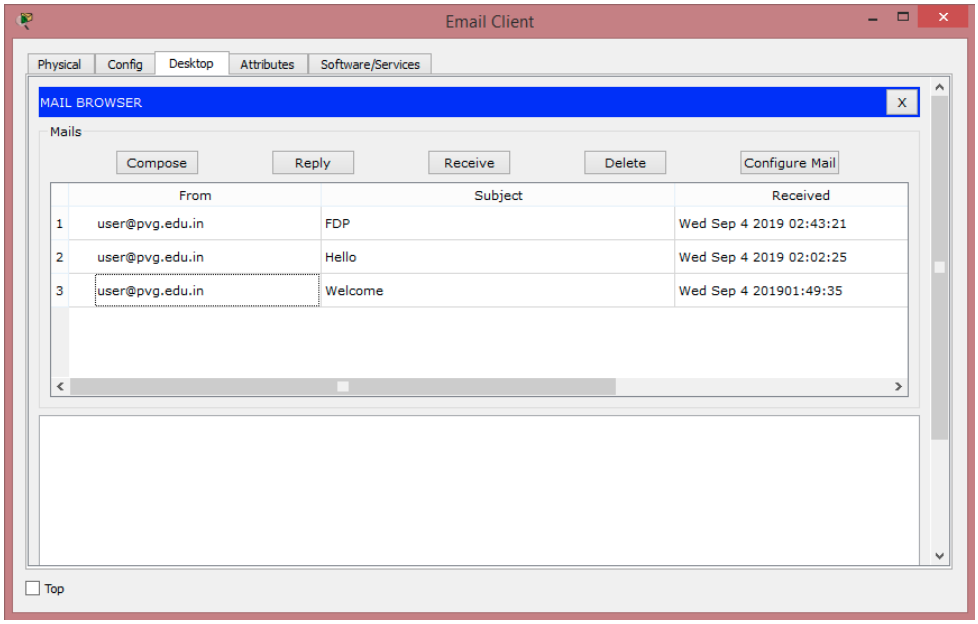

Multi Server
Physical
Config
Services
Desktop
Attributes
Software/Services
SERVICES
HTTP
DHCP
DHCPv6
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
IoE
VM Management
FTP
Service
On
Off
User Setup
Username user1
Password gces1
Write
Read
Delete
Rename
List
Username
Password
Permission
1 cisco cisco RWDNL
2 user gces RWDNL
Add
Save
Remove
File
1 asa842-k8.bin
2 asa923-k8.bin
3 c1841-advipservicesk9-mz.124-15.T1.bin
Remove
Top

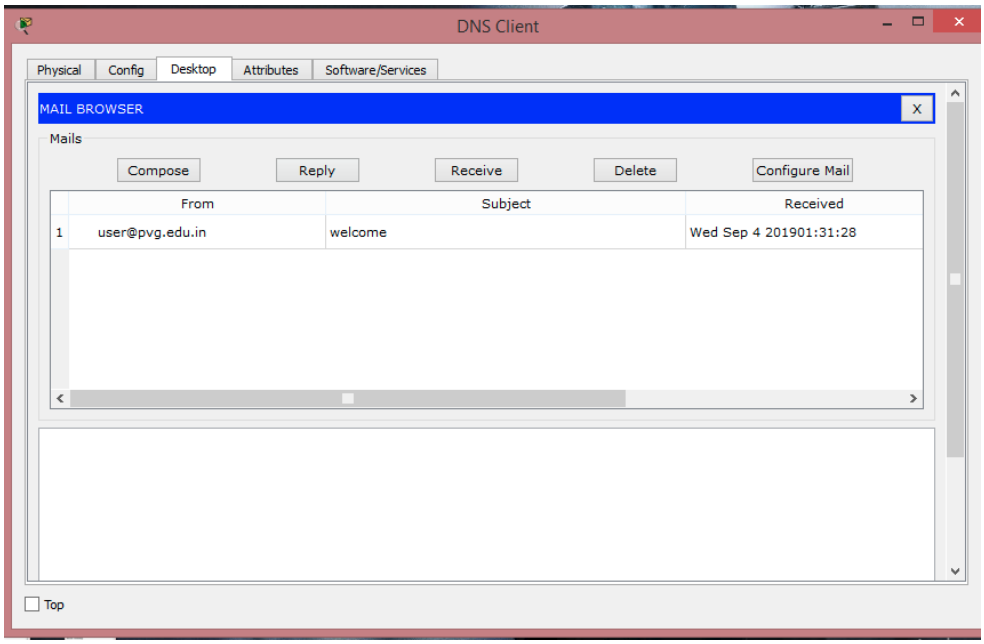

Multi Server
Physical
Config
Services
Desktop
Attributes
Software/Services
SERVICES
HTTP
DHCP
DHCPv6
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
IoE
VM Management
EMAIL
SMTP Service
ON
OFF
POP3 Service
ON
OFF
Domain Name: pvg.edu.in
Set
User Setup
User user2
Password gces2
user
user1
+
-
Change
Password
Top


Multi Server
Physical
Config
Services
Desktop
Attributes
Software/Services
Web Browser
URL
http://pvg.edu.in
Go
Stop
Internet Technology
By GCES
Top


HTTP Client
Physical
Config
Desktop
Attributes
Software/Services
Web Browser
URL
http://pvg.edu.in
Go
Stop
Internet Technology
By GCES
Top


FTP Client
Physical
Config
Desktop
Attributes
Software/Services
Command Prompt
Packet Tracer PC Command Line 1.0
C:\>ftp pvg.edu.in
Trying to connect...pvg.edu.in
Connected to pvg.edu.in
220- Welcome to PT Ftp server
Username:user
331- Username ok, need password
Password:
230- Logged in
(passive mode On)
ftp>ls
Invalid or non supported command.
ftp>dir
Listing /ftp directory from pvg.edu.in:
0 : asa842-k8.bin 5571584
1 : asa923-k8.bin 30468096
2 : c1841-advipservicesk9-mz.124-15.T1.bin 33591768
3 : c1841-ipbase-mz.123-14.T7.bin 13832032
4 : c1841-ipbasek9-mz.124-12.bin 16599160
5 : c2600-advipservicesk9-mz.124-15.T1.bin 33591768
6 : c2600-i-mz.122-28.bin 5571584
7 : c2600-ipbasek9-mz.124-8.bin 13169700
8 : c2800nm-advipservicesk9-mz.124-15.T1.bin 50938004
9 : c2800nm-advipservicesk9-mz.151-4.M4.bin 33591768
10 : c2800nm-ipbase-mz.123-14.T7.bin 5571584
11 : c2800nm-ipbasek9-mz.124-8.bin 15522644
Top





Result

Thus the TCP/UDP performance has been simulated successfully

Ex No:8 B	TCP PROGRAM: MESSAGE REVERSAL

Aim

To write a java program for message reversal using TCP.

Algorithm

Tcp server

1. Create a server socket and wait for client connection.
2. Accept the client connection.
3. Receive a message from client.
4. Reverse the message.
5. Send the reverse message back to client.

TCP Client

1. Create a socket and connect to the server.

2. Read the message from the user.

3. Send the message to the server.

4. Receive the reversed message fom the server.

5. Display the server response.

Program

Server

```
import java.io.*;
import java.net.*;
public class TCPMessageServer {
public static void main(String[] args) {
try {
ServerSocket serverSocket = new ServerSocket(9876);
System.out.println("TCP server is running...");
```

```
Socket socket = serverSocket.accept();
BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
String message = in.readLine();
String reversedMessage = new StringBuilder(message).reverse().toString();
System.out.println("TCP Reversed Message: " + reversedMessage);
out.println(reversedMessage);
socket.close();
serverSocket.close();
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

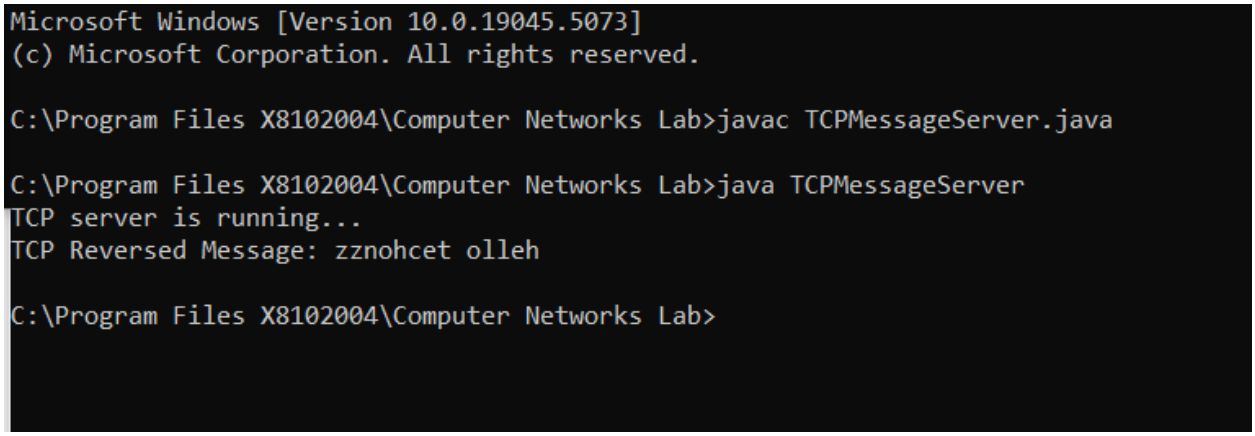
Client

```
import java.net.*;
public class TCPMessageClient {
public static void main(String[] args) {
try {
Socket socket = new Socket("localhost", 9876);
BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
 BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
System.out.print("Enter a message for TCP: ");
String message = userInput.readLine();
out.println(message);
String response = in.readLine();
```

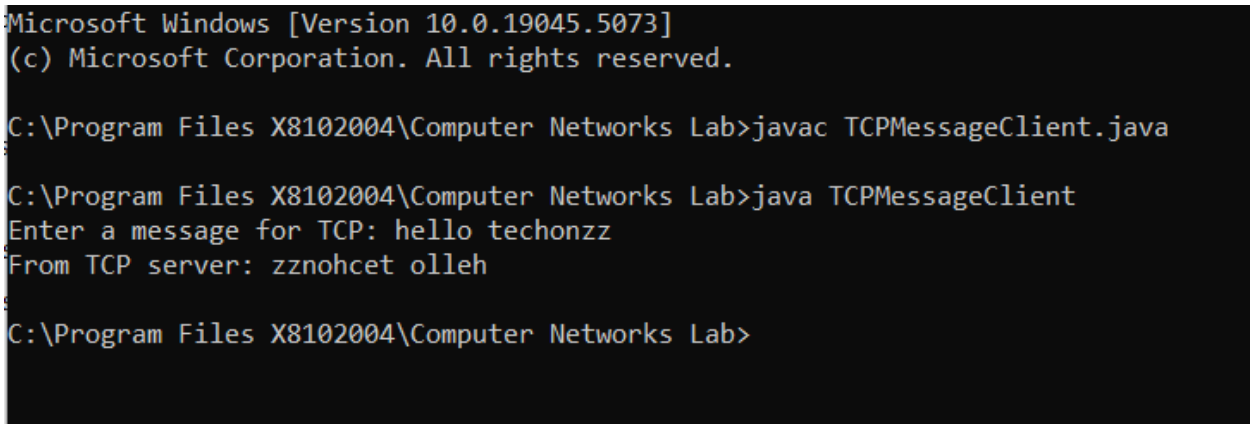
```
System.out.println("From TCP server: " + response);

socket.close();

}

catch (Exception e) {

e.printStackTrace();

 }}}
```

Output

Server



Client



Result

Thus the java program for message reversal using TCP was written executed and output was verified successfully.

Ex No:8 C	UDP PROGRAM: MESSAGE REVERSAL

Aim

To write a java program to message Reversal using UDP

Algorithm

UDP server

1. Create a data gram socket.
2. Receive a message from the client.
3. Optionally simulate packet loss.
4. Reverse the message.
5. Send the reverse message back to client.

UDP Client

1. Create a datagram socket.

2. Read the message from the user.

3. Send the message to the server.

4. Receive the reversed message from the server.

5. Display the server response.

Program

Server

```
import java.net.*;
public class UDPMessageServer {
public static void main(String[] args) {
DatagramSocket serverSocket = null;
try {
serverSocket = new DatagramSocket(9877);
byte[] receiveData = new byte[1024];
```

```
byte[] sendData;
        System.out.println("UDP server is running...");
while (true) {
DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
serverSocket.receive(receivePacket);
String message = new String(receivePacket.getData(), 0, receivePacket.getLength());
// Simulate random packet loss
if (Math.random() > 0.9) {
System.out.println("UDP packet lost: " + message);
continue; // Skip to the next iteration
}
// Reverse the message
String reversedMessage = new StringBuilder(message).reverse().toString();
sendData = ("UDP Reversed Message: " + reversedMessage).getBytes();
// Get the client's address and port
InetAddress clientAddress = receivePacket.getAddress();
int clientPort = receivePacket.getPort();
// Create and send the response packet
DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, clientAddress, clientPort);
serverSocket.send(sendPacket);
System.out.println("Processed message from client: " + message);
}
} catch (Exception e) {
e.printStackTrace();
} finally {
if (serverSocket != null && !serverSocket.isClosed()) {
serverSocket.close();
}
```

```
}
}
}
```

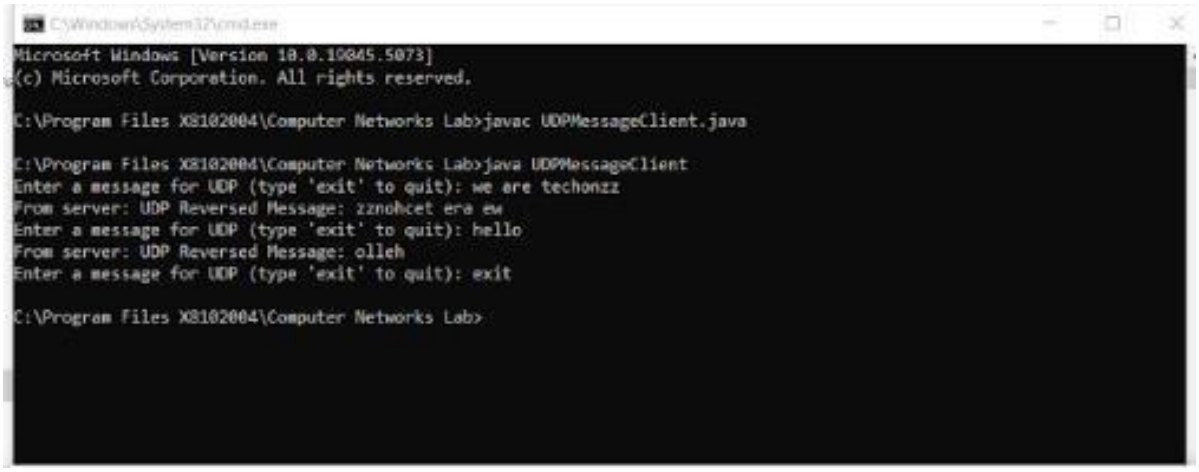
Client

```
import java.net.*;
import java.io.*;
public class UDPMessageClient {
public static void main(String[] args) {
DatagramSocket clientSocket = null;
try {
clientSocket = new DatagramSocket();
InetAddress serverAddress = InetAddress.getByName("localhost");
BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));
byte[] sendData;
byte[] receiveData = new byte[1024];
System.out.print("Enter a message for UDP (type 'exit' to quit): ");
String message;
while (!(message = userInput.readLine()).equalsIgnoreCase("exit")) {
sendData = message.getBytes();
// Create and send the packet
DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, serverAddress, 9877);
clientSocket.send(sendPacket);
// Receive the response
DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
clientSocket.receive(receivePacket);
String serverResponse = new String(receivePacket.getData(), 0, receivePacket.getLength());
```

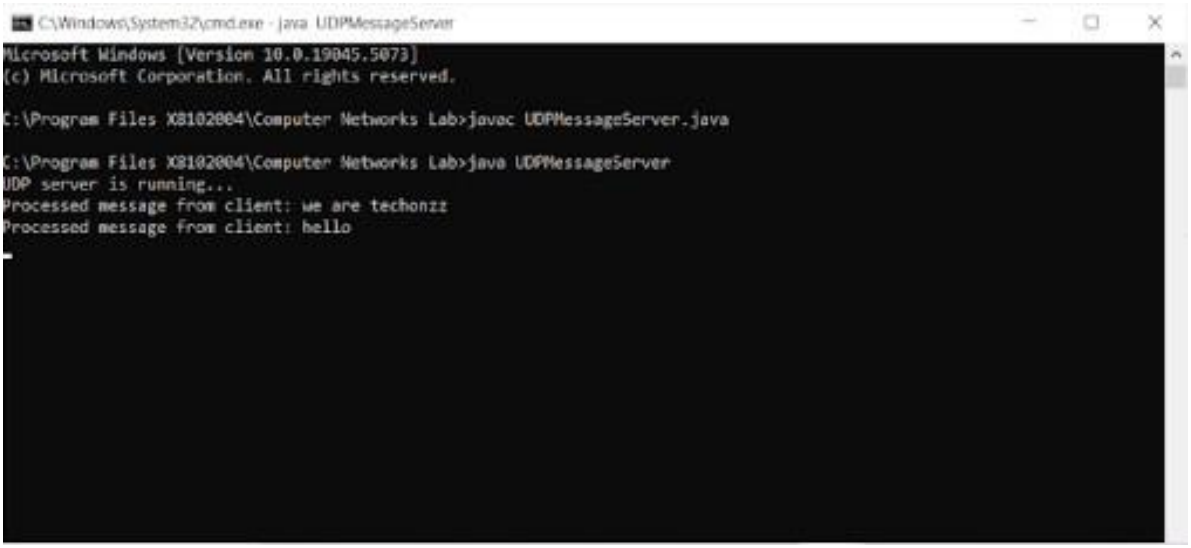
```
System.out.println("From server: " + serverResponse);

System.out.print("Enter a message for UDP (type 'exit' to quit): ");

}
} catch (Exception e) {

e.printStackTrace();

} finally {

if (clientSocket != null && !clientSocket.isClosed()) {

clientSocket.close();

}

}

}

}
```

Output

Server



Client



Result

Thus the java program for message reversal using TCP was written executed and output was verified successfully.

Ex No: 9A	SIMULATION OF LINK STATE ROUTING ALGORITHM

Aim

To implement the Link State Routing Algorithm

Theory

Link State Vector Algorithm

❖ In Link state routing, each router share its information of its neighbors with every other router in the inter-network.

Knowledge about the neighborhood

❖ Instead of sending its entire routing table, a router sends information about its neighborhood only.

To all router

❖ Each router send this information to every other router on the internetworking, not just to its neighbors.

❖ If s does so by a process called “flooding” it means that a router sends its information.

Information sharing when there is a Change

❖ Each router sends out information about the neighbors when there is a change.

Information sharing

❖ Link state routing process use the same internet work as distance vector algorithm.

❖ Here each other sends its knowledge about is neighbors to every other router in the internet work.

❖ Cost is applied only by routers and not by any other station on a network, if cost was added by every station, instead of by routers alone, it would accumulate unpredictably.

❖ Cost is applied as a packet leaves the router rather then as if enters. Most networks are broadcast networks. When a packet is in network every station, including the router, can pick it up, we cannot assign any cost to a packet.

Advertiser Network Cost Neighbor Getting information about neighbors

❖ A router gets its information about its neighbors by periodically sending them a short greeting packet.

❖ If the neighbor responds to the greeting as expected, it is assumed to be alive and functioning.

Initialization

❖ Imagine that all routers in our sample internet work come up at the same time.

❖ Each router sends a greeting packet to its neighbors to find out the state of each link.

Link – State Database

❖ Every router every LSP and puts the information into a link-state database.

❖ Because every router receives the same LSPs every router builds the same database.

❖ It stores this database on its disk and uses it to calculate its routing table. If a router is added to be deleted from the system, the whole database must be shared for fast updating.

Procedure

1. Open Cisco Packet Tracer

Launch Cisco Packet Tracer.

2. Add and Connect Routers

Use the editor to add routers and connect them. Ensure proper physical connections between routers.

3. Assign IP Addresses

Configure IP addresses on router interfaces:

Router1: Interface Fa0/0: ip address 192.168.1.1 255.255.255.0

Router2: Interface Fa0/0: ip address 192.168.1.2 255.255.255.0

Interface Fa0/1: ip address 192.168.2.1 255.255.255.0

Enable OSPF:

Enable OSPF on the routers:

Router1: router ospf 1

network 192.168.1.0 0.0.0.255 area 0

Router2: router ospf 1

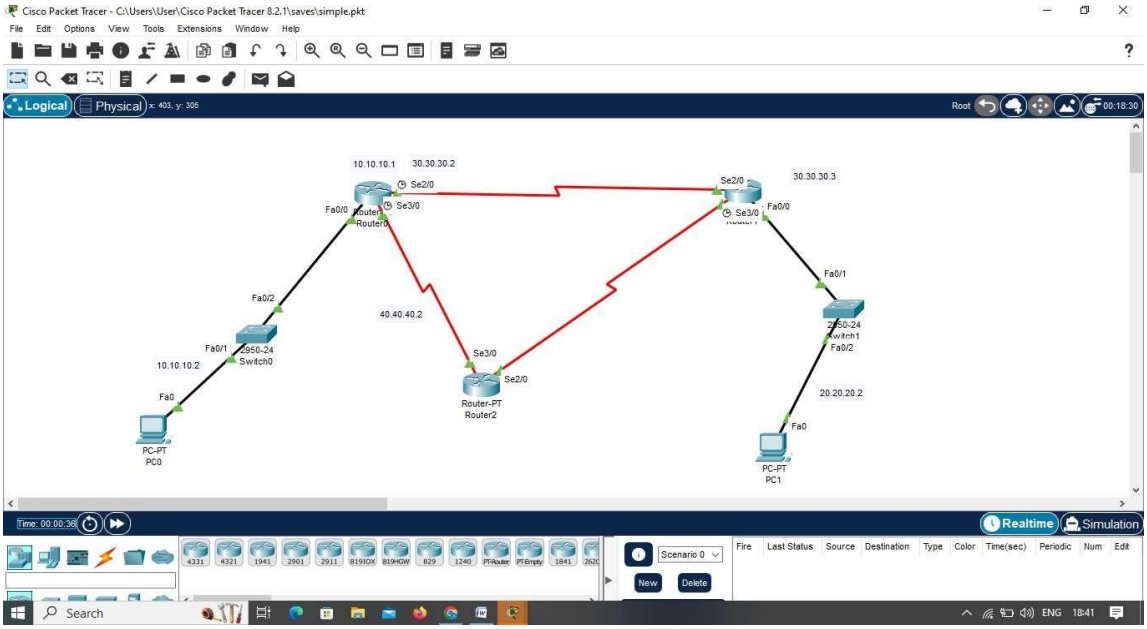
network 192.168.1.0 0.0.0.255 area 0

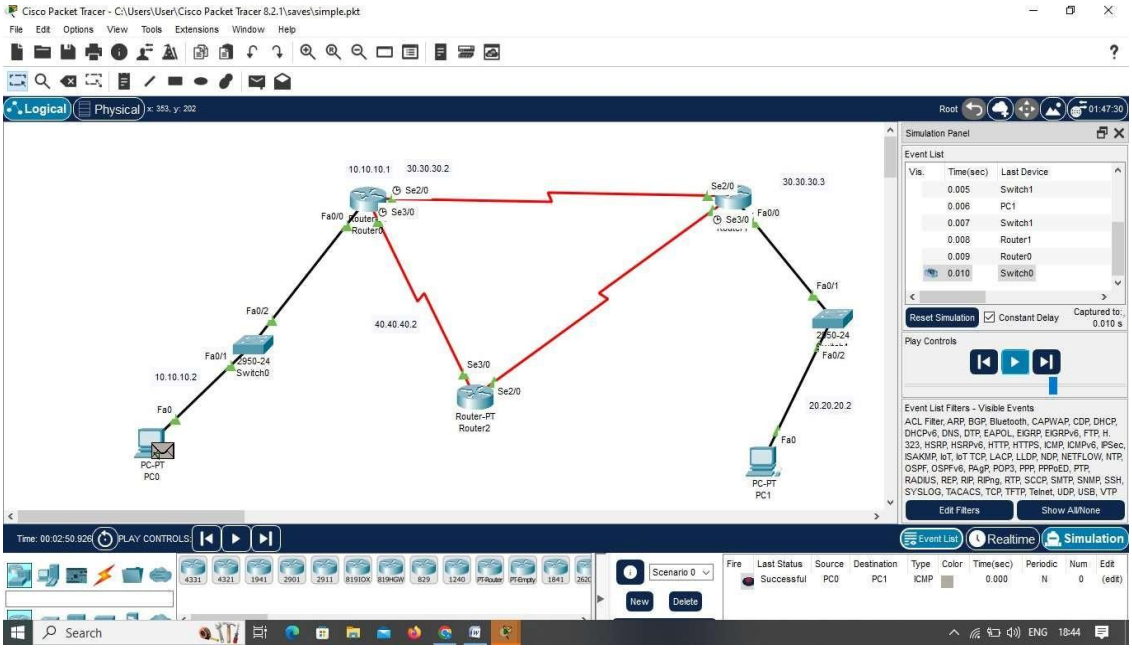
network 192.168.2.0 0.0.0.255 area 0

Connect Nearby Routers

OSPF automatically calculates the shortest path within an area. By enabling OSPF on both routers and defining network statements for their interfaces, you establish a connection between them.

Output





Result

Thus the Simulation of Link State Routing Algorithm using cisco packet tracer is studied Successfully.

EX NO: 9b	LINK STATE ROUTING ALGORITHM

Aim

To write a program to simulate Link State Routing algorithm.

Algorithm

Step 1: Create a network.

Step 2: Assign cost to the edges in the network**.**

Step 3: Initialize the distance of the node itself as zero.

Step 4: The node is taken and chosen as a root node of the tree, this creates the tree with a single node, and now set the total cost of each node to some value based on the information in Link State Database.

Step 5: Now the node selects one node, among all the nodes not in the tree like structure, which is nearest to the root, and adds this to the tree.The shape of the tree gets changed

Step 6: After this node is added to the tree, the cost of all the nodes not in the tree needs to be updated because the paths may have been changed.

Step 7: The node repeats the Step 5 and Step 6 until all the nodes are added in the tree.

Step 8: Print the shortest path from source to destination.

Program

```
import java.util.HashSet;

import java.util.Iterator;

import java.util.Scanner;

import java.util.Set;

public class Dijkstra {

   private int[] distances;

   private Set<Integer> settled;

   private Set<Integer> unsettled;
   private int number_of_nodes;

   private int[][] adjacencyMatrix;

   public Dijkstra(int number_of_nodes) {
```

```
    this.number_of_nodes = number_of_nodes;
    distances = new int[number_of_nodes + 1];
    settled = new HashSet<>();
    unsettled = new HashSet<>();
    adjacencyMatrix = new int[number_of_nodes + 1][number_of_nodes + 1];
}
public void dijkstra_algorithm(int[][] adjacency_matrix, int source) {
 int evaluationNode;
 for (int i = 1; i <= number_of_nodes; i++) {
        for (int j = 1; j <= number_of_nodes; j++) {
            adjacencyMatrix[i][j] = adjacency_matrix[i][j];
        }
    }
    for (int i = 1; i <= number_of_nodes; i++) {
        distances[i] = Integer.MAX_VALUE;
    }
    unsettled.add(source);
    distances[source] = 0;
    while (!unsettled.isEmpty()) {
        evaluationNode = getNodeWithMinimumDistanceFromUnsettled();
        unsettled.remove(evaluationNode);
        settled.add(evaluationNode);
```

```
            evaluateNeighbours(evaluationNode);
        }
    }
    private int getNodeWithMinimumDistanceFromUnsettled() {
        int min = Integer.MAX_VALUE;
        int node = 0;
        for (Integer vertex : unsettled) {
            if (distances[vertex] < min) {
                min = distances[vertex];
                node = vertex;
            }
        }
        return node;
    }
    private void evaluateNeighbours(int evaluationNode) {
        int edgeDistance;
        int newDistance;
        for (int destinationNode = 1; destinationNode <= number_of_nodes; destinationNode++) {
            if (!settled.contains(destinationNode)) {
                if (adjacencyMatrix[evaluationNode][destinationNode] != Integer.MAX_VALUE) {
                    edgeDistance = adjacencyMatrix[evaluationNode][destinationNode];
                    newDistance = distances[evaluationNode] + edgeDistance;
                    if (newDistance < distances[destinationNode]) {
                        distances[destinationNode] = newDistance;
                        unsettled.add(destinationNode);
```

```
            }
        }
    }
}
public static void main(String[] args) {
    int[][] adjacency_matrix;
    int number_of_vertices;
    int source = 0, destination = 0;
    Scanner scan = new Scanner(System.in);
    try {
        System.out.println("Enter the number of vertices");
        number_of_vertices = scan.nextInt();
        adjacency_matrix = new int[number_of_vertices + 1][number_of_vertices + 1];
        System.out.println("Enter the Weighted Matrix for the graph");
        for (int i = 1; i <= number_of_vertices; i++) {
            for (int j = 1; j <= number_of_vertices; j++) {
                adjacency_matrix[i][j] = scan.nextInt();
                if (i == j) {
                    adjacency_matrix[i][j] = 0;
                } else if (adjacency_matrix[i][j] == 0) {
                    adjacency_matrix[i][j] = Integer.MAX_VALUE;
                }
            }
        }
```

```
            System.out.println("Enter the source ");
            source = scan.nextInt();
            System.out.println("Enter the destination ");
            destination = scan.nextInt();
            Dijkstra dijkstraAlgorithm = new Dijkstra(number_of_vertices);
            dijkstraAlgorithm.dijkstra_algorithm(adjacency_matrix,  source);
           System.out.println("The Shortest Path from source " + source + " to " + destination + " is: "
          + dijkstraAlgorithm.distances[destination]);
        } catch (Exception e) {
            System.out.println("Wrong Input Format");
        } finally {
            scan.close();
        }
    }
}
```

Output

```
C:\Users\lipsy>javac Dijkstra.java

C:\Users\lipsy>java Dijkstra
Enter the number of vertices
4
Enter the Weighted Matrix for the graph
0 1 3 0
1 0 1 1
3 1 0 4
0 1 4 0
Enter the source
1
Enter the destination
4
The Shortest Path from source 1 to 4 is: 2
```

Result

Thus a program to simulate Link State Routing algorithm is executed successfully.

Ex No: 9 C	SIMULATION OF DISTANCE VECTOR ROUTING ALGORITHM

Aim

To implement the Distance Vector Routing Algorithm and understand how routers share routing information to calculate the shortest path in a network.

Theory

In the Distance Vector routing algorithm, each router maintains a table of the best (shortest) distance to all other routers in the network. This algorithm works by exchanging this distance information with neighboring routers. Each router updates its routing table based on the distance vector received from its neighbors.

1. **Routing Table:**
   - A table maintained by each router that holds the distance (or cost) to reach every destination in the network.
2. **Distance Vector:**
   - The vector (list) of distances from a router to all other routers. Each router sends this vector to its immediate neighbors.
3. **Cost:**
   - The cost is typically based on the number of hops (or other metrics like bandwidth or delay). It helps in determining the shortest path to a destination.
4. **Periodic Updates:**
   - Routers periodically exchange routing tables with their neighbors. If a better route is found, the router updates its routing table and propagates the information to its neighbors.
5. **Convergence:**
   - Convergence happens when all routers have the correct routing table, meaning all the paths have been updated with the shortest distances.

Procedure

1. **Initialization:**
   - Initially, each router knows only the cost to reach its direct neighbors. For non-neighboring routers, the cost is considered to be infinite.
2. **Exchange of Information:**
   - Routers periodically exchange routing tables with their neighbors, and each router recalculates its routing table based on the received information.

3. **Bellman-Ford Algorithm:**
    - This algorithm is used to calculate the shortest path to each destination in the network by considering the shortest paths to all neighbors.
4. **Routing Table Update:**
    - If a router receives a routing table from a neighbor that offers a better (lower-cost) path to any destination, it updates its own routing table and forwards the new information to its neighbors.

Steps to implement distance vector routing algorithm

1. Open Cisco Packet Tracer

- Launch Cisco Packet Tracer to create a network topology.

2. Add and Connect Routers

- Add two or more routers to the workspace and connect them using appropriate cables (serial or Ethernet).

3. Assign IP Addresses

- Assign IP addresses to the interfaces on the routers to enable communication between them. Example:
    - **Router1:**
        - Interface Fa0/0: `ip address 192.168.10.1 255.255.255.0`
    - **Router2:**
        - Interface Fa0/0: `ip address 192.168.20.1 255.255.255.0`
    - **Router3:**
        - Interface Fa0/0: `ip address 192.168.30.1 255.255.255.0`
    - **PC1:**
        - IP address: `192.168.10.2 255.255.255.0`
        - Default gateway: `192.168.10.1`
    - **PC2:**
        - IP address: `192.168.20.2 255.255.255.0`
        - Default gateway: `192.168.20.1`
    - **PC3:**
        - IP address: `192.168.30.2 255.255.255.0`
        - Default gateway: `192.168.30.`

4. Enable Routing Protocol (RIP)

- For the Distance Vector routing algorithm, enable **RIP** (Routing Information Protocol) on each router.

- **Router1** Configuration:

```
Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#
Router(config-if)#exit
 Router(config)# router rip
 Router(config-router)# network 192.168.10.0
Router(config-router)#  network 10.0.0.0
Router(config-router)#
```

- **Router2** Configuration:

```
Router(config-if)#exit
Router(config)# router  rip
 Router(config-router)# network 192.168.20.0
 Router(config-router)# network 10.0.0.0
Router(config-router)#  network 10.0.0.0
Router(config-router)#
```

- **Router3** Configuration:

```
Router(config-if)#
Router(config-if)#exit
Router(config)# router  rip
 Router(config-router)# network 192.168.30.0
Router(config-router)#  network 20.0.0.0
Router(config-router)#
```

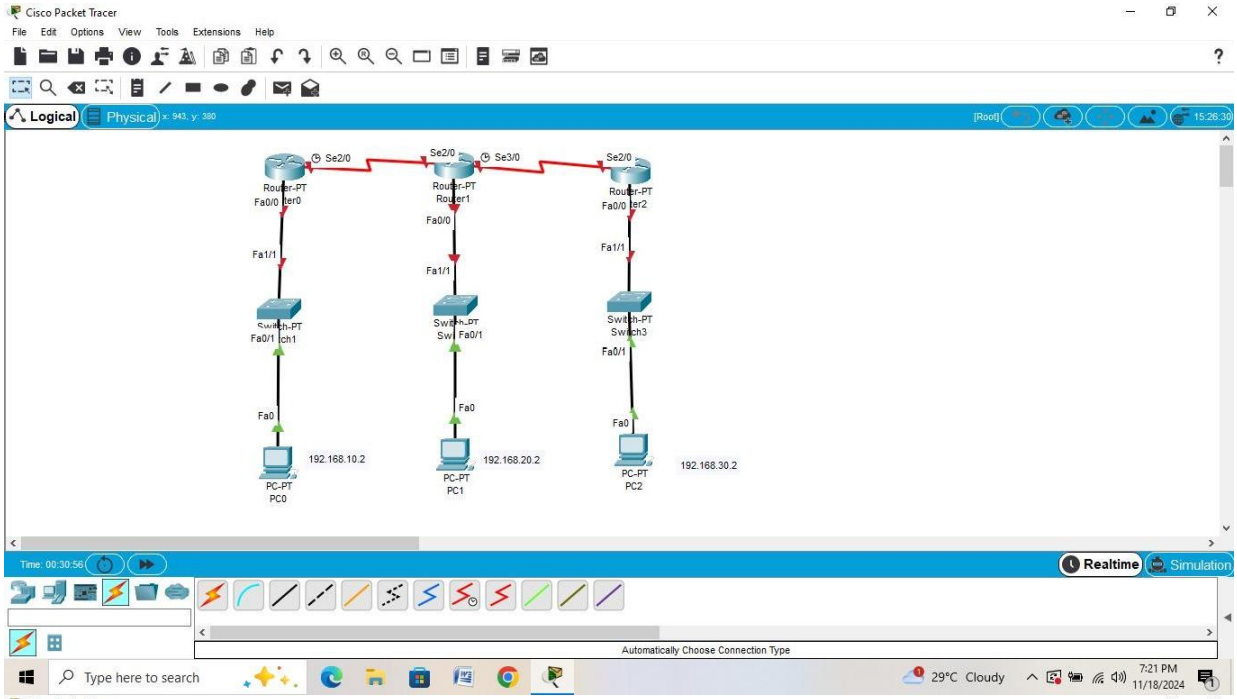
5. Establish Connection Between Routers

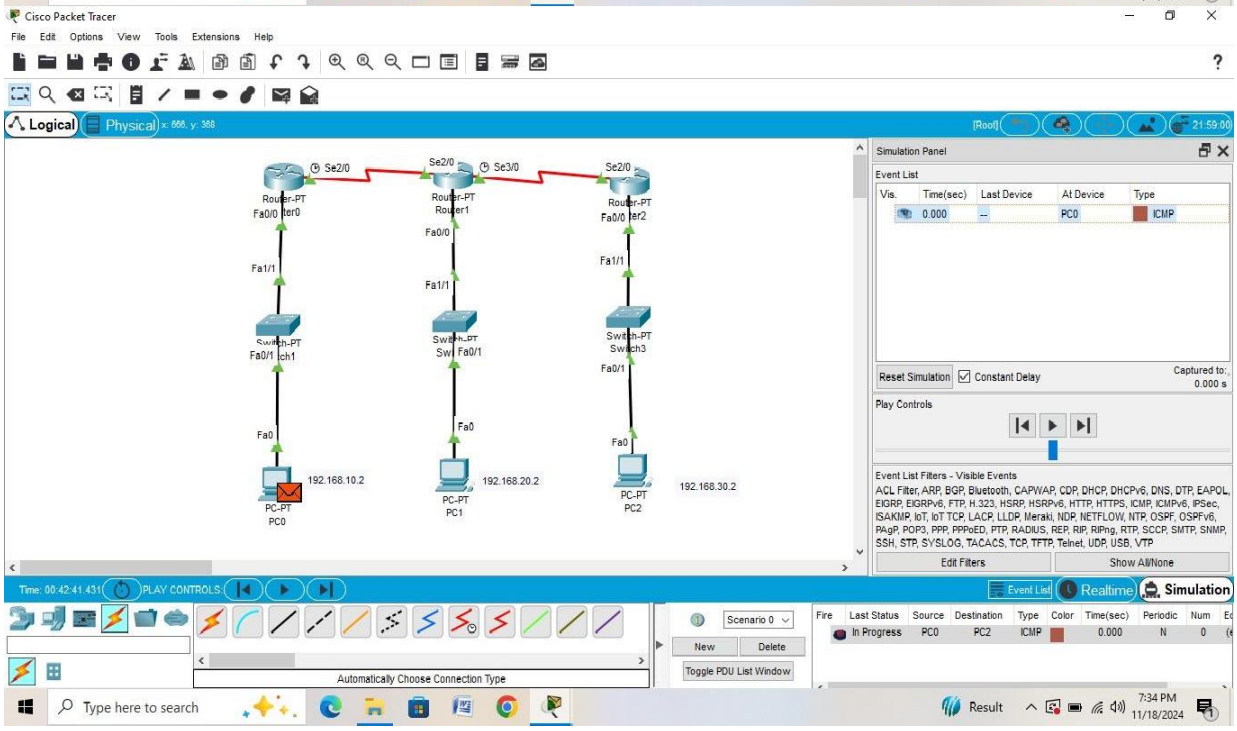
- When RIP is enabled, routers exchange their routing tables. This automatically builds the routing table for each router based on the shortest path to each destination.

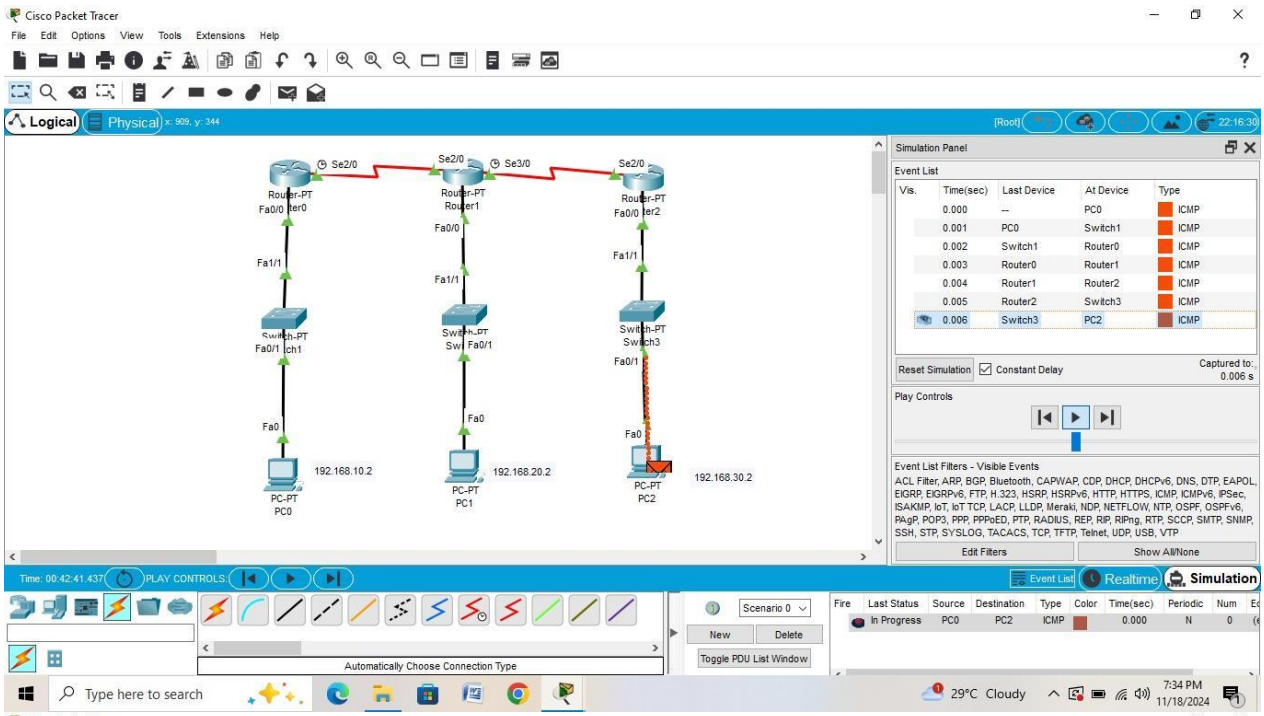
6. Monitor Updates

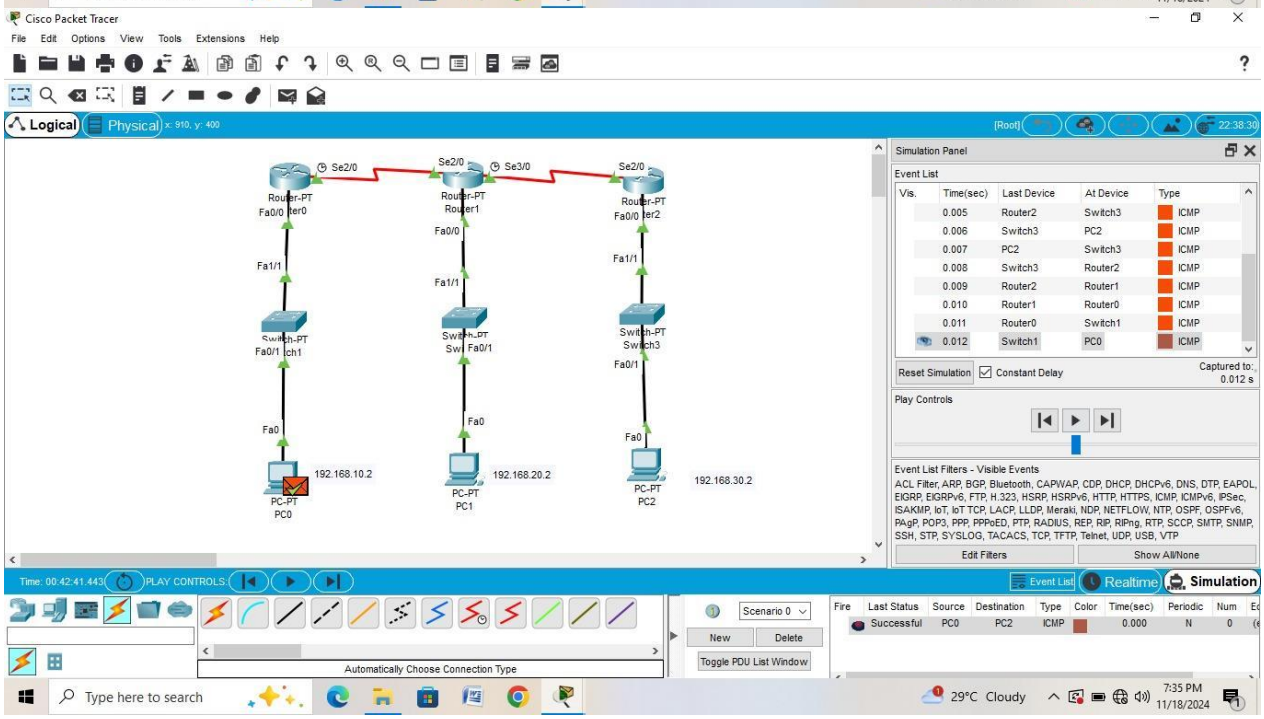
- Routers will periodically exchange updates. If a router finds a shorter path to a destination through its neighbor, it will update its routing table and send this updated information to its neighbors.

Output


Cisco Packet Tracer
File Edit Options View Tools Extensions Help
Logical Physical
Router-PT Router0
Router-PT Router1
Router-PT Router2
Se2/0 Se3/0
Fa0/0 Fa1/1
Switch-PT Switch1
Switch-PT Switch2
Switch-PT Switch3
Fa0/1 Fa0
PC-PT PC0
PC-PT PC1
PC-PT PC2
192.168.10.2
192.168.20.2
192.168.30.2
Realtime Simulation
Automatically Choose Connection Type
Type here to search
29°C Cloudy
7:21 PM
11/18/2024


Cisco Packet Tracer
File Edit Options View Tools Extensions Help
Logical Physical
Simulation Panel
Event List
Vis. Time(sec) Last Device At Device Type
0.000 -- PC0 ICMP
Reset Simulation
Constant Delay
Captured to: 0.000 s
Play Controls
Event List Filters - Visible Events
Edit Filters
Show All/None
Time: 00:42:41.431 PLAY CONTROLS:
Event List
Realtime
Simulation
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time(sec) Periodic Num
In Progress PC0 PC2 ICMP 0.000 N 0
Automatically Choose Connection Type
Type here to search
Result
7:34 PM
11/18/2024





Result

Thus the Simulation of Distance Vector Routing Algorithm using cisco packet tracer is studied successfully.

EX NO:9D	DISTANCE VECTOR ROUTING ALGORITHM USING JAVA

Aim

To write a program to simulate Distance Vector Routing algorithm.

Algorithm

Step 1: Create a network.

Step 2: Assign cost to the edges in the network.

Step 3: Initialize the distance of the node itself as zero.

Step 4: Assign the edge cost as the cost for the immediate neighbours and for other nodes assign the value as infinity.

Step 5: From time-to-time, each node sends its own distance vector estimate to neighbors. When a node x receives new DV estimate from any neighbor v, it saves v's distance vector and it updates its own DV using Bellman Ford equation:

$$Dx(y) = \min \{ C(x,v) + Dv(y)\} \text{ for each node } y \in N$$

Step 6: Print the routing table.

Program

```
import java.io.BufferedReader;
import java.io.*;
public class DVR {
        static int graph[][];
        static int via[][];
        static int rt[][];
        static int v;
        static int e;
        public static void main(String[] args) throws IOException{
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                System.out.println("Please enter the number of Vertices: ");
                v = Integer.parseInt(br.readLine());
                System.out.println("Please enter the number of Edges: ");
                e = Integer.parseInt(br.readLine());
                graph = new int[v][v];
                via = new int[v][v];
                rt = new int[v][v];
                for(int i = 0; i < v; i++)
                for(int j = 0; j < v; j++)
```

```
   {
if(i == j)
graph[i][j] = 0;
else
graph[i][j] = 9999;
   }

for(int i = 0; i < e; i++)
  {
System.out.println("Please enter data for Edge " + (i + 1) + ":");
System.out.print("Source: ");
int s = Integer.parseInt(br.readLine());
s--;
System.out.print("Destination:  ");
int d = Integer.parseInt(br.readLine());
d--;
System.out.print("Cost: ");
int c = Integer.parseInt(br.readLine());
graph[s][d] = c;
graph[d][s] = c;
  }

  dvr_calc_disp("The initial Routing Tables are: ");

System.out.print("Please enter the Source Node for the edge whose cost has changed: ");
int s = Integer.parseInt(br.readLine());
s--;
System.out.print("Please enter the Destination Node for the edge whose cost has changed: ");
int d = Integer.parseInt(br.readLine());
d--;
System.out.print("Please enter the new cost: ");
int c = Integer.parseInt(br.readLine());
graph[s][d] = c;
graph[d][s] = c;

  dvr_calc_disp("The new Routing Tables are: ");
 }

static void dvr_calc_disp(String message)
 {
System.out.println();
init_tables();
update_tables();
System.out.println(message);
print_tables();
System.out.println();
 }

static void update_table(int source)
 {
```

```
for(int i = 0; i < v; i++)
  {
if(graph[source][i] != 9999)
   {
int dist = graph[source][i];
for(int j = 0; j < v; j++)
    {
 int inter_dist = rt[i][j];
  if(via[i][j] == source)
       inter_dist = 9999;
if(dist + inter_dist < rt[source][j])
     {
rt[source][j] = dist + inter_dist;
via[source][j] = i;
     }
    }
   }
  }
 }

static void update_tables()
 {
int k = 0;
for(int i = 0; i < 4*v; i++)
  {
   update_table(k);
k++;
if(k == v)
    k = 0;
  }
 }

static void init_tables()
 {
for(int i = 0; i < v; i++)
  {
for(int j = 0; j < v; j++)
   {
if(i == j)
    {
rt[i][j] = 0;
via[i][j] = i;
     }
 else
     {
rt[i][j] = 9999;
via[i][j] = 100;
    }
   }
  }
```

```
 }

static void print_tables()
 {
for(int i = 0; i < v; i++)
  {
for(int j = 0; j < v; j++)
   {
System.out.print("Dist: " + rt[i][j] + "     ");
   }
System.out.println();
  }
 }

}
```

Output

```
Please enter the number of Vertices:
4
Please enter the number of Edges:
5
Please enter data for Edge 1:
Source: 1
Destination: 2
Cost: 1
Please enter data for Edge 2:
Source: 1
Destination: 3
Cost: 3
Please enter data for Edge 3:
Source: 2
Destination: 3
Cost: 1
Please enter data for Edge 4:
Source: 2
Destination: 4
Cost: 1
Please enter data for Edge 5:
Source: 3
Destination: 4
Cost: 4

The initial Routing Tables are:
Dist: 0    Dist: 1    Dist: 2    Dist: 2
Dist: 1    Dist: 0    Dist: 1    Dist: 1
Dist: 2    Dist: 1    Dist: 0    Dist: 2
Dist: 2    Dist: 1    Dist: 2    Dist: 0

Please enter the Source Node for the edge whose cost has changed: 2
Please enter the Destination Node for the edge whose cost has changed: 4
Please enter the new cost: 10

The new Routing Tables are:
Dist: 0    Dist: 1    Dist: 2    Dist: 6
Dist: 1    Dist: 0    Dist: 1    Dist: 5
Dist: 2    Dist: 1    Dist: 0    Dist: 4
Dist: 6    Dist: 5    Dist: 4    Dist: 0
```

Result

Thus a program to simulate Distance Vector Routing algorithm is executed successfully.

EX NO: 10 A	SIMULATION OF ERROR CORRECTION CODE (CRC)

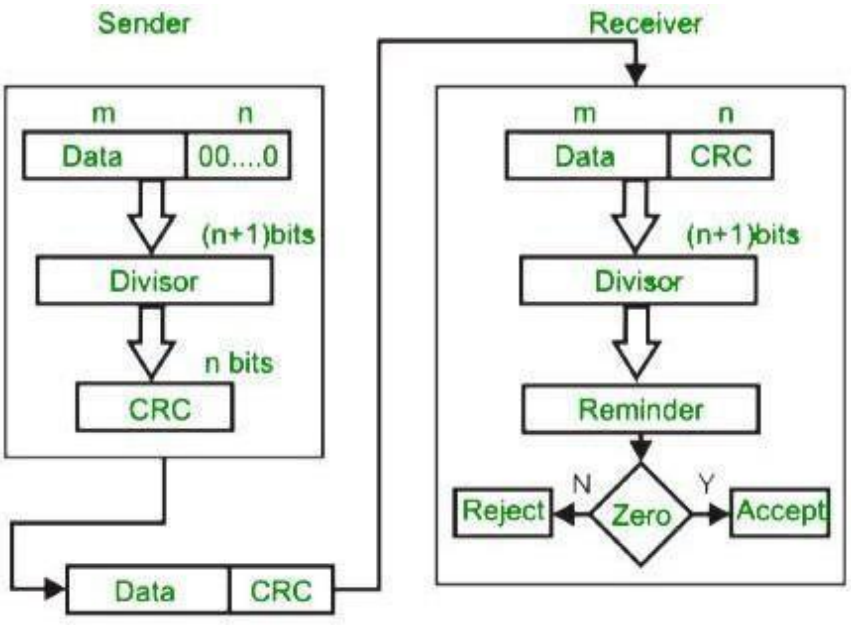
Aim

To simulate the Error Correction Code using Java.

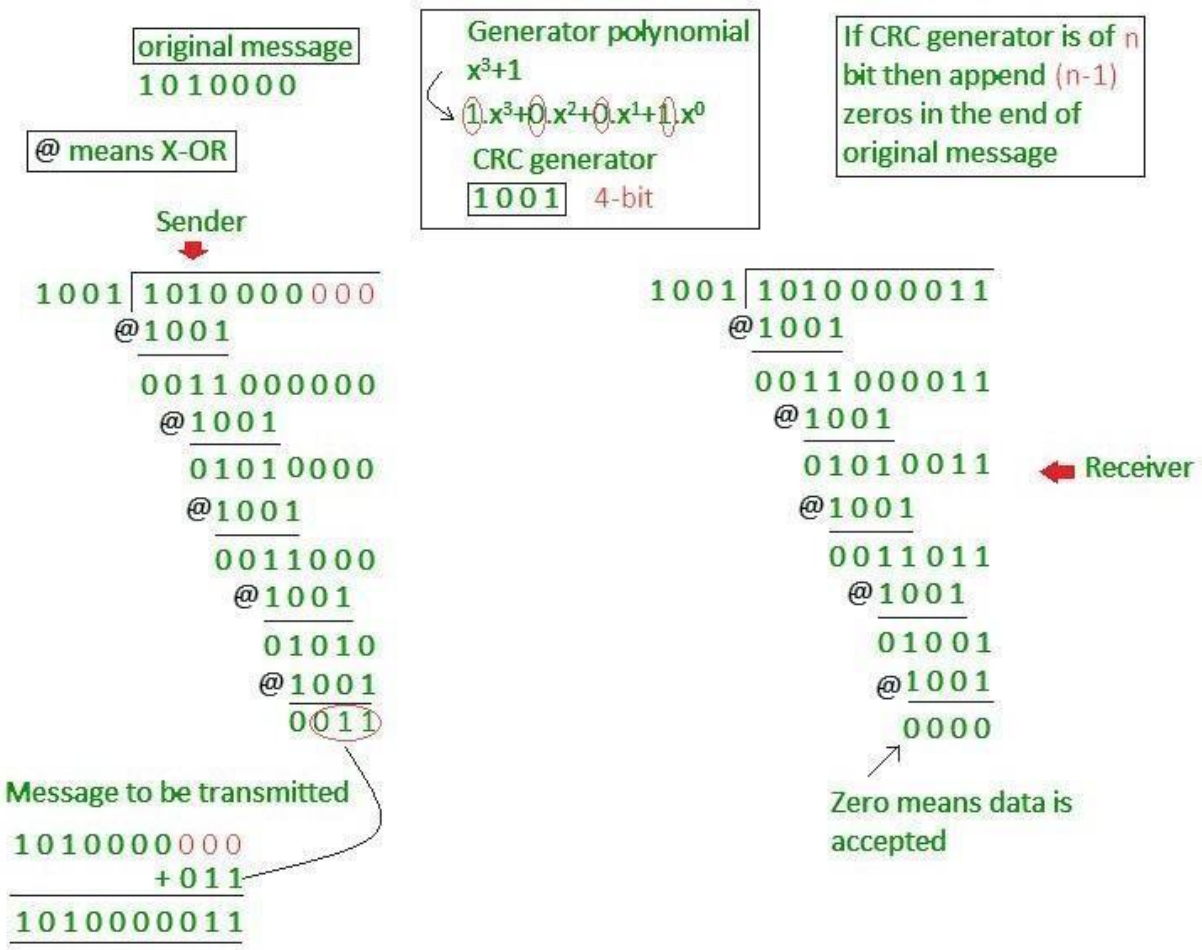
Theory

Cyclic redundancy check (CRC)

- Unlike checksum scheme, which is based on addition, CRC is based on binary division.
- In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.
- At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.
- A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.



Example



Algorithm

1. Open the editor and type the program for error detection.
2. Get the input in the form of bits.
3. Append the redundancy bits.
4. Divide the appended data using a divisor polynomial.
5. The resulting data should be transmitted to the receiver.
6. At the receiver the received data is entered.
7. The same process is repeated at the receiver.
8. If the remainder is zero there is no error otherwise there is some error in the received bits.
9. Run the program.

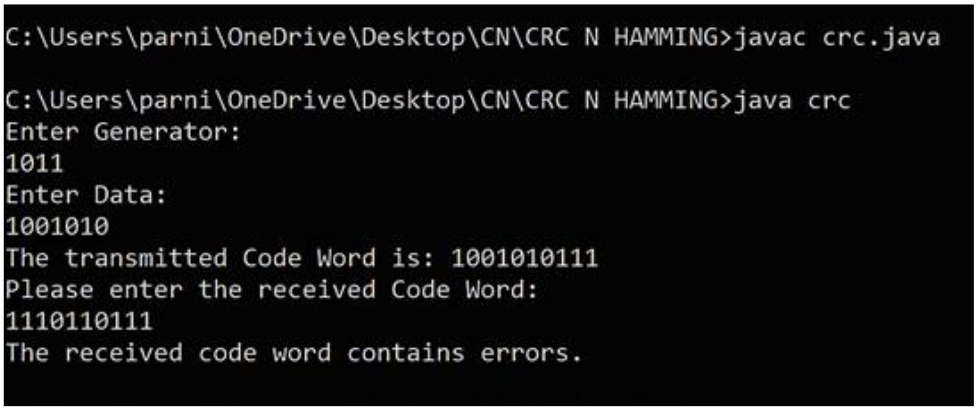
Program

```
import java.io.*;
class CRC {
    public static void main(String args[]) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter Generator:");
        String gen = br.readLine();
        System.out.println("Enter Data:");
        String data = br.readLine();
        String code = data;
        while (code.length() < (data.length() + gen.length() - 1)) {
            code = code + "0";
        }
        code = data + div(code, gen);
        System.out.println("The transmitted Code Word is: " + code);
        System.out.println("Please enter the received Code Word:");
        String rec = br.readLine();
        if (Integer.parseInt(div(rec, gen)) == 0) {
            System.out.println("The received code word contains no errors.");
        } else {
            System.out.println("The received code word contains errors.");
        }
    }
    static String div(String num1, String num2) {
```

```
int pointer = num2.length();
String result = num1.substring(0, pointer);
String remainder = "";
for (int i = 0; i < num2.length(); i++) {
    if (result.charAt(i) == num2.charAt(i)) {
        remainder += "0";
    } else {
        remainder += "1";
    }
}
while (pointer < num1.length()) {
    if (remainder.charAt(0) == '0') {
        remainder = remainder.substring(1, remainder.length());
        remainder = remainder + String.valueOf(num1.charAt(pointer));
        pointer++;
    }
    result = remainder;
    remainder = "";
    for (int i = 0; i < num2.length(); i++) {
        if (result.charAt(i) == num2.charAt(i)) {
            remainder += "0";
        } else {
            remainder += "1";
        }
```

```
        }
      }
      return remainder.substring(1, remainder.length());
    }
}
```

Output



Result

Thus a program to simulate error correction code was executed successfully.

Ex No:10 B	SIMULATION OF ERROR CORRECTION - HAMMING CODE

Aim

To write a program for the simulation of Error Correction - Hamming code.

Algorithm

1. Write the bit positions in binary form.
2. The bit positions should be started from 1(1, 10, 11, 100, etc.).
3. Mark all those bits as parity bits which are a power of 2 (1, 2, 4, 8, etc.).
4. Mark all the other bit positions as data bits.
5. Each data bit is included in a unique set of parity bits, as determined by its bit position in binary form.
    i. All those bits positions whose binary representation have 1 in the 4th position from the least significant bit (8-15, 24-31, 40-47, etc.) are covered by the 8th parity bit.
    ii. All those bits positions whose binary representation have 1 in the 3rd position from the least significant bit(4-7, 12-15, 20-23, etc.) are covered by the 4th parity bit.
    iii. All those bits positions whose binary representation have 1 in the second position from the least significant bit(2, 3, 6, 7, 10, 11, etc.) are covered by the 2nd parity bit.
    iv. All those bits positions whose binary representation have 1 in the least significant position(1, 3, 5, 7, 9, 11, etc.) are covered by the second parity bit.
    v. All the bits, where the bitwise AND of the parity position
    vi. The bit position is non-zero, are covered by each parity bit.
6. Since we check for even parity, set a parity bit to 1 if the total number of ones in the positions it checks is odd.
7. Set a parity bit to 0 when the total number of ones in the bit positions it covers is an even.

Program

```
package javaTpoint.JavaExample;
import java.util.Scanner;
class HammingCodeExample {
    public static void main(String args[]) {
        int size, hammingCodeSize, errorPosition;
        int arr[];
        int hammingCode[];
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of data bits:");
        size = sc.nextInt();
        arr = new int[size];
        for (int i = 0; i < size; i++) {
            System.out.println("Enter " + (size - i) + "-bit of the data:");
            arr[size - i - 1] = sc.nextInt();
        }
        System.out.println("The data you entered is:");
        for (int i = 0; i < size; i++) {
            System.out.print(arr[size - i - 1]);
        }
        System.out.println();
        hammingCode = getHammingCode(arr);
        hammingCodeSize = hammingCode.length;
        System.out.println("The generated Hamming code is:");
        for (int i = 0; i < hammingCodeSize; i++) {
            System.out.print(hammingCode[hammingCodeSize - i - 1]);
        }
        System.out.println();
        System.out.println("Enter the position of a bit to alter (0 for no error):");
        errorPosition = sc.nextInt();
        sc.close();
        if (errorPosition != 0)
        {
```

```
        hammingCode[errorPosition - 1] = (hammingCode[errorPosition - 1] + 1) % 2;
    }
    System.out.println("Sent Data is:");
    for (int i = 0; i < hammingCodeSize; i++) {
        System.out.print(hammingCode[hammingCodeSize - i - 1]);
    }
    System.out.println();
    receiveData(hammingCode, hammingCodeSize - arr.length);
}
static int[] getHammingCode(int data[]) {
    int size = data.length;
    int i = 0, parityBits = 0, j = 0, k = 0;
    while (i < size) {
        if (Math.pow(2, parityBits) == (i + parityBits + 1)) {
            parityBits++;
        }
        else
        {
            i++;
        }
    }
    int returnData[] = new int[size + parityBits];
    for (i = 1; i <= returnData.length; i++) {
        if (Math.pow(2, j) == i) {
            returnData[i - 1] = 2;
            j++;
        }
        else
        {
            returnData[k + j] = data[k++];
        }
    }
```

```
    for (i = 0; i < parityBits; i++)
    {
        returnData[(int) Math.pow(2, i) - 1] = getParityBit(returnData, i);
    }
    return returnData;
}
static int getParityBit(int returnData[], int pow) {
    int parityBit = 0;
    int size = returnData.length;
    for (int i = 0; i < size; i++) {
        if (returnData[i] != 2) {
            int k = i + 1;
            String str = Integer.toBinaryString(k);
            int temp = (Integer.parseInt(str) / (int) Math.pow(10, pow)) % 10;
            if (temp == 1 && returnData[i] == 1) {
                parityBit = (parityBit + 1) % 2;
            }
        }
    }
    return parityBit;
}
static void receiveData(int data[], int parityBits) {
    int size = data.length;
    int parityArray[] = new int[parityBits];
    String errorLoc = "";
    for (int pow = 0; pow < parityBits; pow++) {
        for (int i = 0; i < size; i++) {
            int j = i + 1;
            String str = Integer.toBinaryString(j);
            int bit = (Integer.parseInt(str) / (int) Math.pow(10, pow)) % 10;
            if (bit == 1 && data[i] == 1) {
                parityArray[pow] = (parityArray[pow] + 1) % 2;
            }
```

```
            }
            errorLoc = parityArray[pow] + errorLoc;
        }
        int finalLoc = Integer.parseInt(errorLoc, 2);
        if (finalLoc != 0) {
            System.out.println("Error is found at location " + finalLoc + ".");
            data[finalLoc - 1] = (data[finalLoc - 1] + 1) % 2;
            System.out.println("After correcting the error, the code is:");
            for (int i = 0; i < size; i++) {
                System.out.print(data[size - i - 1]);
            }
            System.out.println();
        }
        else
        {
            System.out.println("There is no error in the received data.");
        }
        System.out.println("The data sent from the sender:");
        int pow = parityBits - 1;
        for (int k = size; k > 0; k--) {
            if (Math.pow(2, pow) != k) {
                System.out.print(data[k - 1]);
            }
            else
            {
                pow--;
            }
        }
        System.out.println();
    }
}
```

Output:

```
Enter the bits size for the data.
7
Enter 7-bit of the data:
1
Enter 6-bit of the data:
1
Enter 5-bit of the data:
0
Enter 4-bit of the data:
1
Enter 3-bit of the data:
0
Enter 2-bit of the data:
0
Enter 1-bit of the data:
1
The data which you enter is:
1101001
The hamming code generated for your data is:
11001001101
For detecting error at the reciever end, enter position of a bit to alter original
3
Sent Data is:
11001001001
Error is found at location 3.
After correcting the error, the code is:
11001001101
The data sent from the sender:
1101001
```

Result

Thus the program for Error Correction using Hamming Code was implemented and the output was verified successfully.