



# RAJ KUMAR GOEL INSTITUTE OF TECHNOLOGY

Approved by AICTE (Ministry of Education) & PCI (Ministry of Health & FW) GOI, Affiliated to Dr. APJ Abdul Kalam Technical University, Lucknow  
AKTU College Code-033

Accredited by NAAC ('A' Grade), NBA Accredited Programs (B.Tech-ECE, IT) & B.Pharma



## LABORATORY RECORD

Faculty Name	: Mr. Sanjay Srivastava	Department:	CSE
	: Computer Network Lab	Course Code:	BCS-653
Course Name	: 3rd/6th	NBA Code:	C-321
Year/Sem	: <a href="mailto:sansrfcs@rkgit.edu.in">sansrfcs@rkgit.edu.in</a>	Academic Year	2024-25
Email ID			

Department of Computer Science and Engineering



# **RAJ KUMAR GOEL INSTITUTE OF TECHNOLOGY**

Approved by AICTE (Ministry of Education) & PCI (Ministry of Health & FW) GOI, Affiliated to Dr. APJ Abdul Kalam Technical University, Lucknow

AKTU College Code-033

Accredited by NAAC ('A' Grade), NBA Accredited Programs (B.Tech-ECE, IT) & B.Pharma



## **VISION OF THE INSTITUTE**

To continually develop excellent professionals capable of providing sustainable solutions to challenging problems in their fields and prove responsible global citizens.

## **MISSION OF THE INSTITUTE**

We wish to serve the nation by becoming a reputed deemed university for providing value based professional education.

## **VISION OF THE DEPARTMENT**

To be recognized globally for delivering high quality education in the ever changing field of computer science & engineering, both of value & relevance to the communities we serve.

## **MISSION OF THE DEPARTMENT**

1. To provide quality education in both the theoretical and applied foundations of Computer Science and train students to effectively apply this education to solve real world problems.
2. To amplify their potential for lifelong high quality careers and give them a competitive advantage in the challenging global work environme



## PROGRAM EDUCATIONAL OUTCOMES (PEOs)

**PEO 1: Learning:** Our graduates to be competent with sound knowledge in field of Computer Science & Engineering.

**PEO 2: Employable:** To develop the ability among students to synthesize data and technical concepts for application to software product design for successful careers that meet the needs of Indian and multinational companies.

**PEO 3: Innovative:** To develop research oriented analytical ability among students to prepare them for making technical contribution to the society.

**PEO 4: Entrepreneur / Contribution:** To develop excellent leadership quality among students which they can use at different levels according to their experience and contribute for progress and development in the society.





## PROGRAM OUTCOMES (POs)

### Engineering Graduates will be able to:

**PO1: Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**PO3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO5: Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

**PO6: The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.



**PO7: Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**PO8: Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO9: Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**PO10: Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**PO11: Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**PO12: Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.



## PROGRAM SPECIFIC OUTCOMES (PSOs)

**PSO1:** The ability to use standard practices and suitable programming environment to develop software solutions.

**PSO2:** The ability to employ latest computer languages and platforms in creating innovative career opportunities.



### Outcome (Computer Networks Lab, BCS-653)

CO No.	Course Outcome (CO)	Relevant POs/ PSOs	Revised Bloom's Cognitive Process Level (BL)	Knowledge Category* (KC)
<b>CO1</b>	Examine the networking commands and configuring network hardware.	Apply	Conceptual & Procedural	Examine the networking commands and configuring Network hardware.
<b>CO2</b>	Demonstrate the working of Cisco packet tracer.	Apply	Conceptual & Procedural	Demonstrate the working of Cisco packet tracer.
<b>CO3</b>	Apply the concepts of stop and wait ARQ.	Apply	Conceptual & Procedural	Apply the concepts of stop and wait ARQ.
<b>CO4</b>	Construct the sockets for various applications.	Create	Conceptual & Procedural	Construct the sockets for various Applications.
<b>CO5</b>	Apply the knowledge to implement OSI layers	Apply	Conceptual & Procedural	Design and analyze functions of OSI session & presentation layer.
<b>CO6</b>	Analyze the working and performance of various protocols.	Analyze	Conceptual & Procedural	Analyze the working and performance of various protocols.

\*Knowledge Categories (BCS): **F**-Factual, **C**-Conceptual, **P**-Procedural, **M**-Metacognitive

### CO-PO Mapping (Computer Networks Lab, BCS-653)

Course Code: BCS653	Programme Outcome (PO)												PSO 1	PSO 2
	1	2	3	4	5	6	7	8	9	10	11	12	1	2
<b>CO1</b>	2	2	3	3	3	-	-	-	-	-	2	2	3	2





# RAJ KUMAR GOEL INSTITUTE OF TECHNOLOGY

Approved by AICTE (Ministry of Education) & PCI (Ministry of Health & FW) GOI, Affiliated to Dr. APJ Abdul Kalam Technical University, Lucknow

AKTU College Code-033

Accredited by NAAC ('A' Grade), NBA Accredited Programs (B.Tech-ECE, IT) & B.Pharma



<b>C02</b>	2	2	2	2	3	-	-	-	-	-	2	2	3	2
<b>C03</b>	3	2	3	3	3	-	-	-	-	-	2	2	3	2
<b>C04</b>	3	3	3	3	3	-	-	-	-	-	2	3	3	2
<b>C05</b>	3	2	3	2	3	-	-	-	-	-	2	2	3	2
<b>C06</b>	3	2	3	2	3	-	-	-	-	-	2	2	3	2

<b>PO Target</b>	<b>2.6</b>	<b>2.16</b>	<b>2.8</b>	<b>2.5</b>	<b>3.0</b>	<b>0.0</b>	<b>0.0</b>	<b>0.0</b>	<b>0.0</b>	<b>0.0</b>	<b>2.0</b>	<b>2.0</b>	<b>3.0</b>	<b>2.0</b>
------------------	------------	-------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	------------



## INDEX

S.NO	List of Program
1	To learn handling and configuration of networking hardware like RJ-45 connector, CAT- 6 cable, crimping tool, etc
2	Configuration of router, hub, switch etc. (using real devices or simulators)
3	Implementation of Stop and Wait Protocol and Sliding Window Protocol
4	Study of Socket Programming and Client – Server model
5	Write a code simulating ARP /RARP protocols.
6	Write a code simulating PING and TRACEROUTE commands
7	Create a socket for HTTP for web page upload and download
8	Write a program to implement RPC (Remote Procedure Call)
9	Implementation of Subnetting .
10	Applications using TCP Sockets like a. Chat b. File Transfer



11	Applications using TCP and UDP Sockets like c. DNS d. SNMP
12	Study of Network simulator (NS).and Simulation of Congestion Control Algorithms using NS

## INDEX

S.NO	List of Program
13	Perform a case study about the different routing algorithms to select the network path with its optimum and economical during data transfer. i. Link State routing ii. ii. Flooding iii. iii. Distance vector
14.	Running and using services/commands like ping, trace route, arp, telnet, etc.
15.	Network packet analysis using tools like Wireshark, tcpdump, etc.
16.	Network simulation using tools like Cisco Packet Tracer, NetSim, OMNeT++, NS2, NS3,



## INTRODUCTION

A computer network is a system that connects two or more computing devices for transmitting and sharing information. Computing devices include everything from a mobile phone to a server. These devices are connected using physical wires such as fiber optics, but they can also be wireless.

Computer Networks Lab/Web Programming Lab is a part of B.Tech 3RD Year (6th SEM) for CSE students. This lab provides information about working principle of various communication protocols, the network simulator environment and visualizes a network topology and observes its performance, to analyze the traffic flow and the contents of protocol frames.



## GUIDELINES FOR LABORTORY RECORD PREPARATION

While preparing the lab records, the student is required to adhere to the following guidelines:

### Contents to be included in Lab Records:

1. Cover page
2. Vision
3. Mission
4. PEOs
5. POs
6. PSOs
7. COs
8. CO-PO-PSO mapping
9. Index

### Experiments

Objective

Source code

Input-

Output

**A separate copy needs to be maintained for pre-lab written work** The student is required to make the Lab File as per the format given on the next two pages.



# RAJ KUMAR GOEL INSTITUTE OF TECHNOLOGY

Approved by AICTE (Ministry of Education) & PCI (Ministry of Health & FW) GOI, Affiliated to Dr. APJ Abdul Kalam Technical University, Lucknow

AKTU College Code-033

Accredited by NAAC ('A' Grade), NBA Accredited Programs (B.Tech-ECE, IT) & B.Pharma



## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



### COMPUTER NETWORK LAB (BCS-653)

<b>NAME</b>	
<b>ROLL NO.</b>	
<b>SECTION-BATCH</b>	



## INDEX

Serial no.	Experiment	Date of Execution	Date of Submission	Signature
1.	To learn handling and configuration of networking hardware like RJ-45 connector, CAT- 6 cable, crimping tool, etc			
2.	Configuration of router, hub, switch etc. (using real devices or simulators)			
3.	Implementation of Stop and Wait Protocol and Sliding Window Protocol			
4.	Study of Socket Programming and Client – Server model			
5.	Write a code simulating ARP /RARP protocols.			
6.	Write a code simulating PING and TRACEROUTE commands			
7.	Create a socket for HTTP for web page upload and download			
8.	Write a program to implement RPC (Remote Procedure Call)			
9.	Implementation of Subnetting .			
10.	Applications using TCP Sockets like a. Chat      b. File Transfer			
11.	Applications using TCP and UDP Sockets like a. DNS      b. SNMP			
12.	Study of Network simulator(NS).and Simulation of Congestion Control Algorithms using NS			
13.	Perform a case study about the different routing algorithms to select the network path with its optimum and economical during data transfer. i. Link State routing ii.Flooding iii. Distance vector			
14.	Running and using services / commands like ping, trace route,arp ,telnet, etc.			
15.	Network packet analysis using tools like Wireshark, tcpdump, etc.			
16.	Network simulation using tools like Cisco Packet Tracer, NetSim, OMNeT++, NS2, NS3,			



## GUIDELINES FOR ASSESSMENT

Students are provided with the details of the experiment (Aim, pre-experimental questions, procedure etc.) to be conducted in next lab and are expected to come prepared for each lab class.

Faculty ensures that students have completed the required pre-experiment questions and they complete the in-lab programming assignment(s) before the end of class. Given that the lab programs are meant to be formative in nature, students can ask faculty for help before and during the lab class.

Students' performance will be assessed in each lab based on the following Lab Assessment Components:

AC1: Performance (Max. marks =

5) AC2: Viva (Max. marks = 5)

AC3: Record (Max. marks = 5)

In each lab class, students will be awarded marks out of 5 under each component head, making it total out of 15 marks.





## EXPERIMENT-1

**Objective:** To learn handling and configuration of networking hardware like RJ-45 connector, CAT-6 cable, crimping tool, etc.

**Theory:** Ethernet cable Color-coded wiring sequences exist as a cabling industry standard. It allows cabling technicians to reliably predict how Ethernet cable is terminated on both ends so they can follow other technicians' work without having to guess or spend time deciphering the function and connections of each wire pair. Ethernet cable jack wiring follows the T568A and T568B standards.

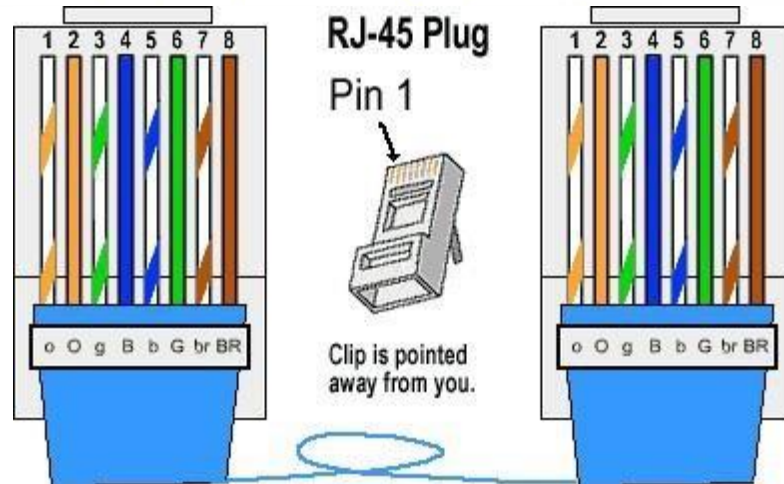
There are two kinds of Ethernet cables you can make, **Straight Through** and **Crossover**.

### Standard Cabling:

- 10BaseT and 100BaseT are most common mode of LAN. You can use UTP category-5 cable for both modes.
- A straight cable is used to connect a computer to a hub

**Diagram shows you how to prepare straight through wired connection**

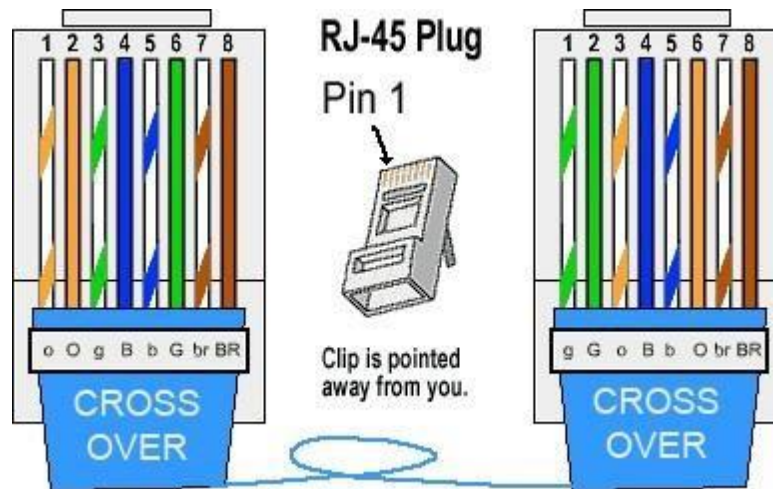
RJ45 Pin # (END 1)	Wire Color	Diagram End #1	RJ45 Pin # (END 2)	Wire Color	Diagram End #2
1	White/Orange		1	White/Green	
2	Orange		2	Green	
3	White/Green		3	White/Orange	
4	Blue		4	White/Brown	
5	White/Blue		5	Brown	
6	Green		6	Orange	
7	White/Brown		7	Blue	
8	Brown		8	White/Blue	



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

Diagram shows you how to prepare Cross wired connection

RJ45 Pin # (END 1)	Wire Color	Diagram End #1	RJ45 Pin # (END 2)	Wire Color	Diagram End #2
1	White/Orange		1	White/Green	
2	Orange		2	Green	
3	White/Green		3	White/Orange	
4	Blue		4	White/Brown	
5	White/Blue		5	Brown	
6	Green		6	Orange	
7	White/Brown		7	Blue	
8	Brown		8	White/Blue	



Bulk RJ45 Crimpable Connectors for CAT-6



RJ-45 Crimping tool

A crimping tool is a device used to conjoin two pieces of metal by deforming one or both of them in a way that causes them to hold each other. The result of the tool's work is called a crimp. A good example of crimping is the process of affixing a connector to the end of a cable. For instance, network cables and phone cables are created using a crimping tool (shown below) to join the RJ-45 and RJ-11 connectors to the both ends of either phone

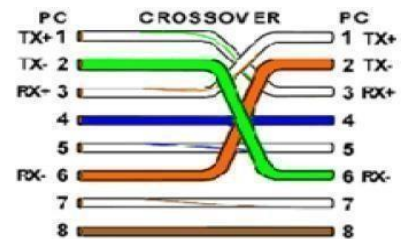
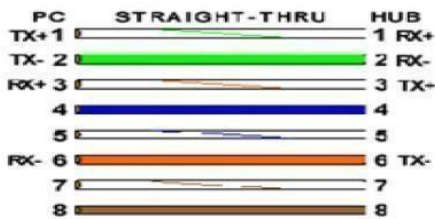




## Cable

### crimping steps:

1. Remove the outmost vinyl shield for 12mm at one end of the cable (we call this side A-side).
2. Arrange the metal wires in parallel
3. Insert the metal wires into RJ45 connector on keeping the metal wire arrangement.
4. Set the RJ45 connector (with the cable) on the pliers, and squeeze it tightly.
5. Make the other side of the cable (we call this side B-side) in the same way.
6. After you made it, you don't need to take care of the direction of the cable.



IO connector crimping: Run the full length of Ethernet cable in place, from endpoint to endpoint, making sure to leave excess.

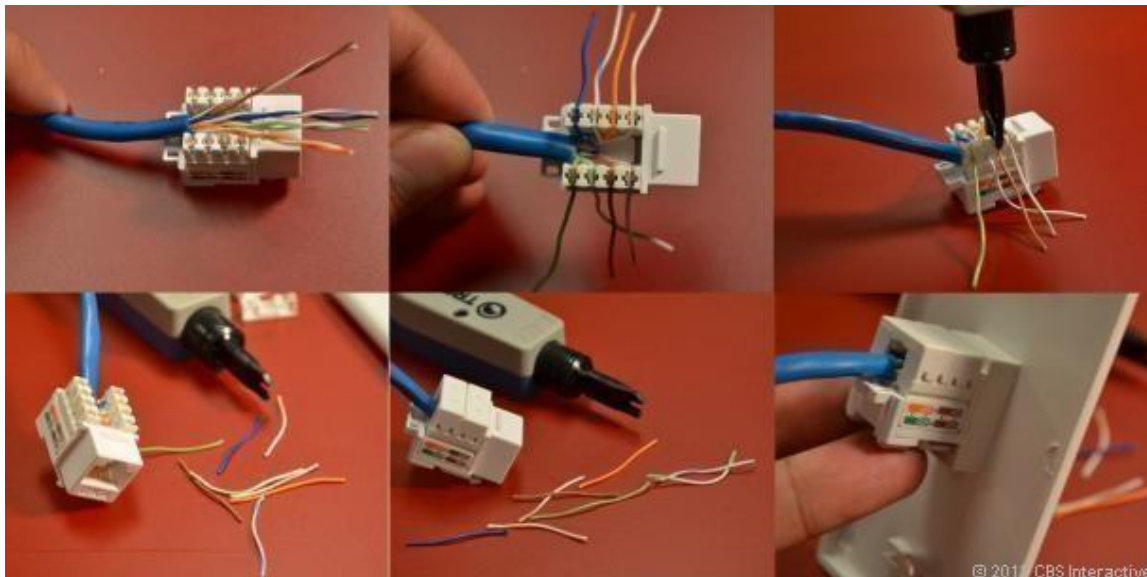
At one end, cut the wire to length leaving enough length to work, but not too much excess.

Strip off about 2 inches of the Ethernet cable sheath.

Align each of the colored wires according to the layout of the jack.

Use the punch down tool to insert each wire into the jack. Repeat the above steps for the second RJ45 jack.





Testing the crimped cable using a cable tester:

Step 1 : Skin off the cable jacket 3.0 cm long cable stripper up to cable

Step 2: Untwist each pair and straighten each wire 190 0 1.5 cm long.

Step 3 : Cut all the wires

Step 4 : Insert the wires into the RJ45 connector right white orange left brown the pins facing up

Step 5 : Place the connector into a crimping tool, and squeeze hard so that the handle reaches its full swing.

Step 6: Use a cable tester to test for proper continuity



**Result:**



# RAJ KUMAR GOEL INSTITUTE OF TECHNOLOGY

Approved by AICTE (Ministry of Education) & PCI (Ministry of Health & FW) GOI, Affiliated to Dr. APJ Abdul Kalam Technical University, Lucknow

AKTU College Code-033

Accredited by NAAC ('A' Grade), NBA Accredited Programs (B.Tech-ECE, IT) & B.Pharm



Cable Crimping, Standard Cabling and Cross Cabling, IO connector crimping and testing the crimped cable using a cable tester are done successfully



## EXPERIMENT-2

**Objective:** Configuration of router, hub, switch etc. (using real devices or simulators)

**Procedure:** Following should be done to understand this practical.

1. **Repeater:** Functioning at Physical Layer. A **Repeater** is an electronic device that receives a signal and retransmits it at a higher level and/or higher power, or onto the other side of an obstruction, so that the signal can cover longer distances. Repeater have two ports ,so cannot be use to connect for more than two devices

2. **Hub:** An **Ethernet hub**, **active hub**, **network hub**, **repeater hub**, **hub** or **concentrator** is a device for connecting multiple twisted pair or fiber optic Ethernet devices together and making them act as a single network segment. Hubs work at the physical layer (layer 1) of the OSI model. The device is a form of multiport repeater. Repeater hubs also participate in collision detection, forwarding a jam signal to all ports if it detects a collision.

3. **Switch:** A **network switch** or **switching hub** is a computer networking device that connects network segments. The term commonly refers to a network bridge that processes and routes data at the data link layer (layer 2) of the OSI model. Switches that additionally process data at the network layer (layer 3 and above) are often referred to as Layer 3 switches or multilayer switches.

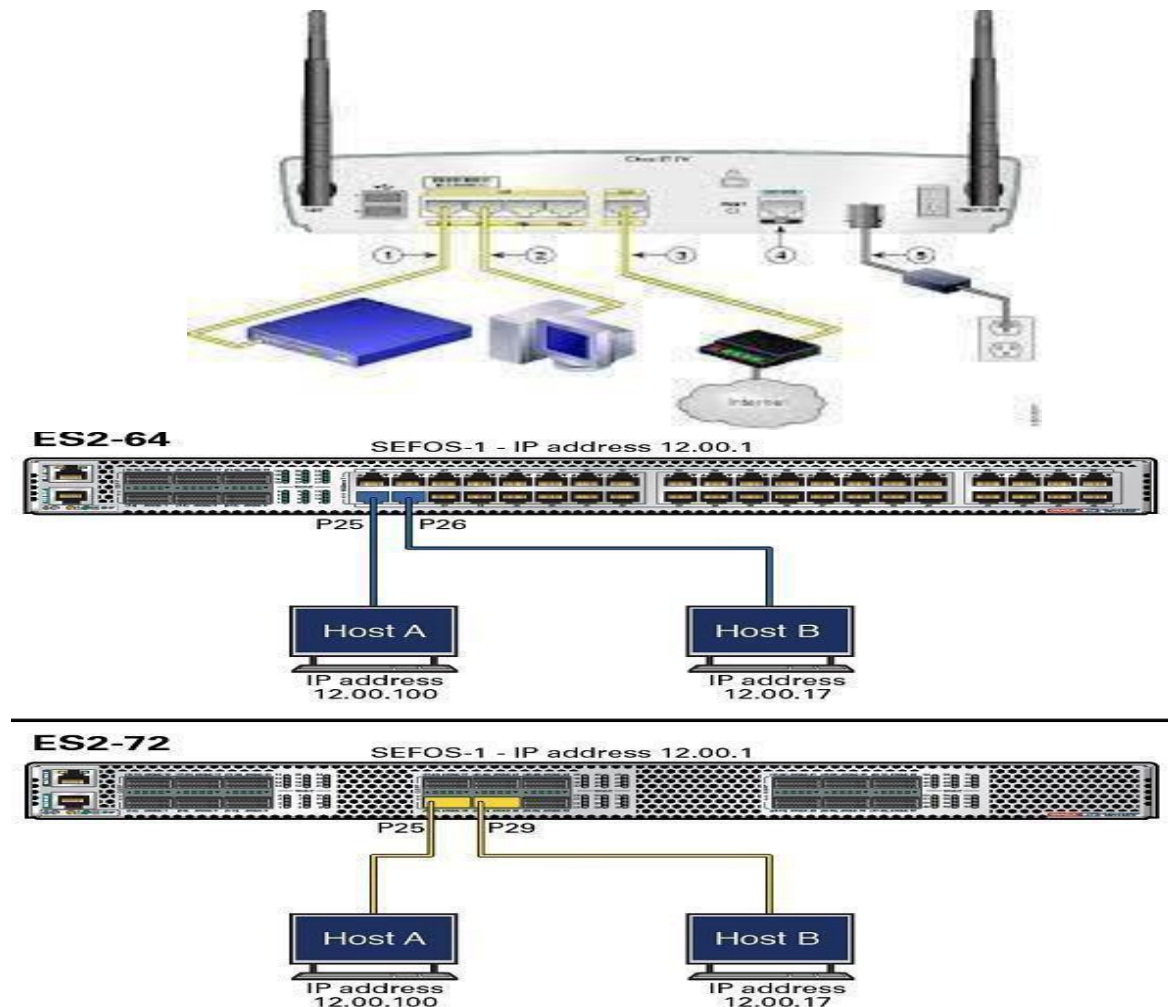
4. **Bridge:** A **network bridge** connects multiple network segments at the data link layer (Layer 2) of the OSI model. In Ethernet networks, the term *bridge* formally means a device that behaves according to the IEEE 802.1 D standards. A bridge and switch are very much alike; a switch being a bridge with numerous ports. *Switch* or *Layer 2 switch* is often used interchangeably with *bridge* .Bridges can analyze incoming data packets to determine if the bridge is able to send the given packet to another segment of the network.

5. **Router:** A **router** is an electronic device that interconnects two or more computer networks, and selectively interchanges packets of data between them. Each data packet contains address information that a router can use to determine if the source and destination are on the same network, or if the data packet must be transferred from one network to another. Where multiple routers are used in a large collection of interconnected networks, the routers exchange information about target system addresses, so that each router can build up a table showing the preferred paths between any two systems on the interconnected networks.

6. **Gate Way:** In a communications network, a network node equipped for interfacing with another network that uses different protocols.



- A gateway may contain devices such as protocol translators, impedance matching devices, rate converters, fault isolators, or signal translators as necessary to provide system interoperability. It also requires the establishment of mutually acceptable administrative procedures between both networks.
- A protocol translation/mapping gateway interconnects networks with different network protocol technologies by performing the required protocol conversions.





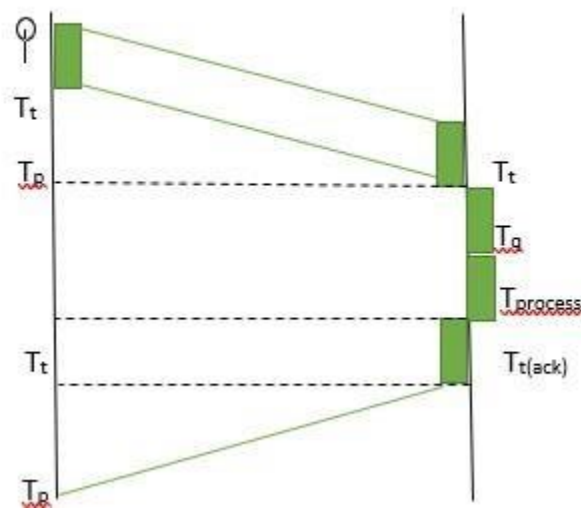


## EXPERIMENT-3

**Objective: Implementation of Stop and Wait Protocol and Sliding Window Protocol.**

**Theory:** It is the simplest flow control method in which the sender will send the packet and then wait for the acknowledgement by the receiver that it has received the packet then it will send the next packet. Stop and wait protocol is very easy to implement.

Time line diagram- Sender receiver



Total time taken to send is,

$$T_{total} = T_t(\text{data}) + T_p + T_q + T_{process} + T_t(\text{ack}) + T_p$$

( since,  $T_q$  and  $T_{process} = 0$ )

$$T_{total} = T_t(\text{data}) + 2T_p + T_t(\text{ack})$$

$$T_{total} = T_t(\text{data}) + 2T_p$$

(when  $T_t(\text{ack})$  is negligible)

### Efficiency

= useful time / total cycle time

$$= T_t / (T_t + 2T_p)$$

$$= 1 / (1 + 2a) \quad [a = T_p / T_t]$$

### Design

- **Sender Site:** The data link layer in the sender site waits for the network layer for a data packet. It then checks whether it can send the frame. If it receives a positive notification from the physical



layer, it makes frames out of the data and sends it. It then waits for an acknowledgement before sending the next frame.

- **Receiver Site:** The data link layer in the receiver site waits for a frame to arrive. When it arrives, the receiver processes it and delivers it to the network layer. It then sends an acknowledgement back to the sender.

**Algorithm: Sender Site Algorithm of Simplex Stop – and – Wait Protocol for Noiseless Channel**  
begin canSend = True;      //Allow the first frame to be

sent while (true)      //check repeatedly do

Wait\_For\_Event(); //wait for availability of packet if

( Event(Request\_For\_Transfer) AND canSend)

thenGet\_Data\_From\_Network\_Layer();

Make\_Frame();

Send\_Frame\_To\_Physical\_Layer(); canSend

= False; else if (

Event(Acknowledgement\_Arrival))

thenReceive\_ACK(); canSend =

True;

end if end while end

**Algorithm: Receiver Site Algorithm of Simplex Stop – and – Wait Protocol for Noiseless Channel**  
begin while (true)      //check

repeatedly do

Wait\_For\_Event();      //wait for arrival of frame      if (

Event(Frame\_Arrival) then

Receive\_Frame\_From\_Physical\_Layer();

Extract\_Data();

Deliver\_Data\_To\_Network\_Layer();

Send\_ACK();      end if      end while end

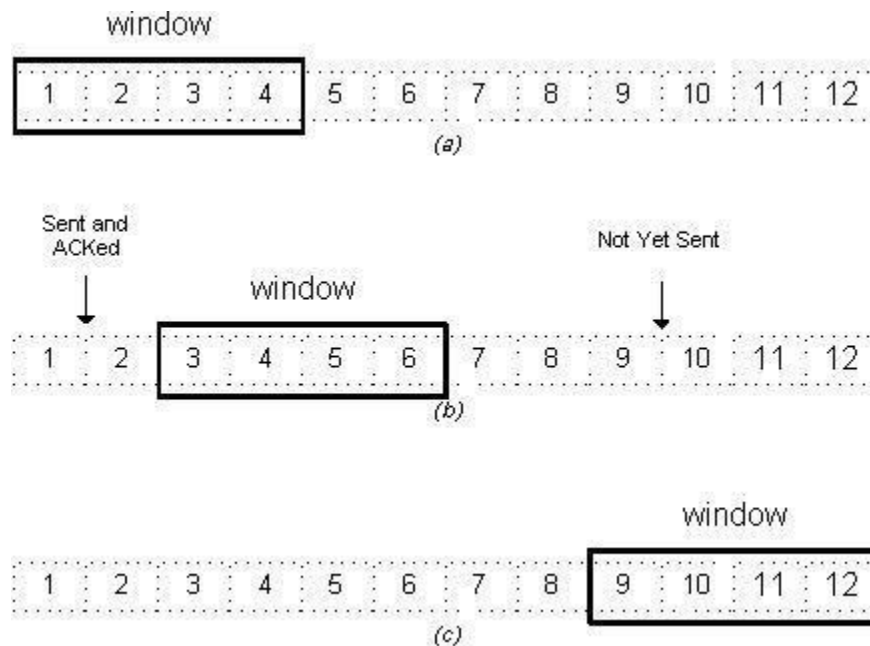


## Algorithm: Sliding Window Protocol

In computer networks sliding window protocol is a method to transmit data on a network. Sliding window protocol is applied on the Data Link Layer of OSI model. At data link layer data is in the form of frames. In Networking, Window simply means a buffer which has data frames that needs to be transmitted.

Both sender and receiver agrees on some window size. If window size= $w$  then after sending  $w$  frames sender waits for the acknowledgement (ack) of the first frame.

As soon as sender receives the acknowledgement of a frame it is replaced by the next frames to be transmitted by the sender. If receiver sends a collective or cumulative acknowledgement to sender then it understands that more than one frames are properly received, for eg:- if ack of frame 3 is received it understands that frame 1 and frame 2 are received properly.



In sliding window protocol the receiver has to have some memory to compensate any loss in transmission or if the frames are received unordered.

Sliding window works in full duplex mode

It is of two types:-

1. **Selective Repeat:** Sender transmits only that frame which is erroneous or is lost.
2. **Go back n:** Sender transmits all frames present in the window that occurs after the error bit including error bit also.



## Sliding Window Protocol Program in C

```
#include<stdio.h>
```

```
int main()
```

```
{ int w,i,f,frames[50];
```

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

```
printf("\nEnter number of frames to transmit:
```

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

```
",f); for(i=1;i<=f;i++)
```

```
scanf("%d",&frames[i]);
```

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

```
sent by the receiver\n\n",w); for(i=1;i<=f;i++)
```

```
{ if(i%w==0)
```

```
{ printf("%d\n",frames[i]); printf("Acknowledgement of above frames sent is received by sender\n\n");
```

```
} else printf("%d
```

```
",frames[i]);
```

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

```
return 0;
```

```
}
```

### **Output**

Enter window size: 3

Enter number of frames to transmit: 5

Enter 5 frames: 12 5 89 4 6

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

After sending 3 frames at each stage sender waits for acknowledgement sent by the receiver 12 5

89

Acknowledgement of above frames sent is received by sender

4 6



# RAJ KUMAR GOEL INSTITUTE OF TECHNOLOGY

Approved by AICTE (Ministry of Education) & PCI (Ministry of Health & FW) GOI, Affiliated to Dr. APJ Abdul Kalam Technical University, Lucknow

AKTU College Code-033

Accredited by NAAC ('A' Grade), NBA Accredited Programs (B.Tech-ECE, IT) & B.Pharma



Acknowledgement of above frames sent is received by sender



## Program 4

**Objective:** Study of Socket Programming and Client – Server model

### DESCRIPTION:

#### ❖ Socket

A socket is formally defined as an endpoint for communication between an application program, and the underlying network protocols. The two modes of services available are • Connection-oriented service

- Connection less service

#### Connectionless (UDP) vs Connection-Oriented (TCP) Servers

- Programmer can choose a connection-oriented server or a connectionless server based on their applications.
- In Internet Protocol terminology, the basic unit of data transfer is a datagram. This is basically a header followed by some data. The datagram socket is connectionless.
- **User Datagram Protocol (UDP):**
  1. Is a connectionless.
  2. A single socket can send and receive packets from many different computers.
  3. Best effort delivery.
  4. Some packets may be lost some packets may arrive out of order.
- **Transmission Control Protocol (TCP):**
  1. Is a connection-oriented.
  2. A client must connect a socket to a server.
  3. TCP socket provides bidirectional channel between client and server.
  4. Lost data is re-transmitted.
  5. Data is delivered in-order.
  6. Data is delivered as a stream of bytes.
  7. TCP uses flow control.
- It is simple for a single UDP server to accept data from multiple clients and reply.
- It is easier to cope with network problems using TCP.

#### CONNECTION-ORIENTED

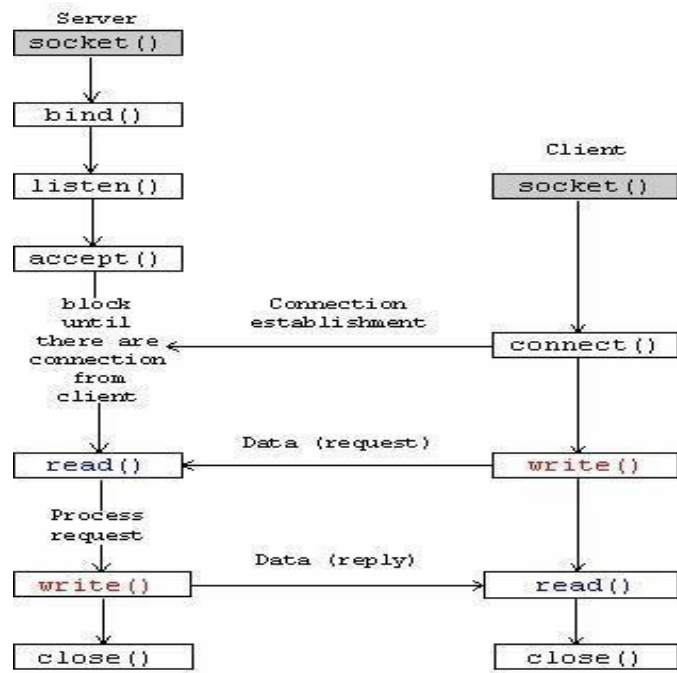
Connection oriented model defines a reliable delivery service. The figure shows a sequence of system calls for connection oriented communication. The server begins by carrying out a passive open as follows. The **socket** call creates a TCP socket. The **bind** call then binds the well-known port number of the server to the socket. The **listen** call turns the socket into a listening socket that can accept incoming connections from clients. Finally, the **accept** call puts the server process to sleep until the arrival of a client connection. The client does an active open. The socket call creates a socket on the client side, and the **connect** call establishes the TCP connection to the server with the specified destination socket address. When the TCP



connection is completed, the accept function at the server wakes up and returns the descriptor for the given connection, namely, the source IP address, the source port number, destination IP address and destination port number. The client and server are now ready to exchange information.

## Socket system calls for **connection-oriented service**

- Following figure illustrates the example of client/server relationship of the socket APIs for connection-oriented protocol (TCP).



## CONNECTION-LESS

In a connection-less mode an application program sends its data immediately without waiting for connection establishment. As a result the application program may waste its time by sending data when the other end is not ready to accept it. Moreover, data may not arrive at the other end if the network decides to discard it. If data arrives at the destination, it may not arrive in the same order as it was transmitted.

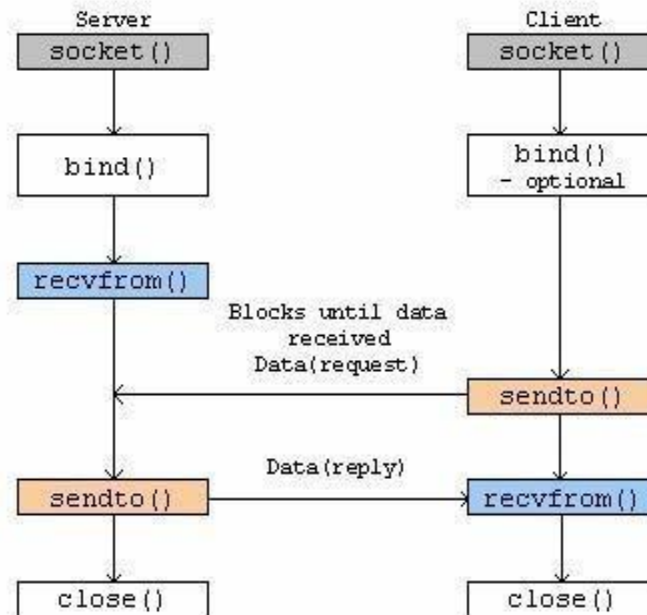
The connectionless mode is often said to provide best effort service, since the network would try its best to deliver the information but cannot guarantee the delivery.

The figure shows the sequence of system calls for a connectionless communication. No connection is established prior to data transfer. The `recvfrom` call returns when a Complete UDP data gram has been received.

## Socket system calls for **connection less service**

- The following figure illustrates the example of client/server relationship of the socket APIs for a connectionless protocol (UDP).









socket()	Create a new socket and return its descriptor.
bind()	Associate a socket with a port and address.
listen()	Establish queue for connection requests.
accept()	Accept a connection request.
connect()	Initiate a connection to a remote host.
recv()	Receive data from a socket descriptor.
send()	Send data to a socket descriptor.
read()	Reads from files, devices, sockets etc.
write()	Writes to files, devices, sockets etc.
close()	“One-way” close of a socket descriptor.
shutdown()	Allows you to cut off communication in a certain direction, or both ways just like close() does.

## Socket API Functions

### Socket API functions **socket()**

socket() creates an endpoint for communication and returns a file descriptor for the socket. socket() takes three arguments:

- *domain*, which specifies the protocol family of the created socket. For example:
  - PF\_INET for network protocol IPv4 or ◦ PF\_INET6 for IPv6.
  - PF\_UNIX for local socket (using a file).
- *type*, one of:
  - SOCK\_STREAM (reliable stream-oriented service or Stream Sockets)
  - SOCK\_DGRAM (datagram service or Datagram Sockets)
  - SOCK\_SEQPACKET (reliable sequenced packet service), or
  - SOCK\_RAW (raw protocols atop the network layer).
- *protocol* specifying the actual transport protocol to use. The most common are IPPROTO\_TCP, IPPROTO\_SCTP, IPPROTO\_UDP, IPPROTO\_DCCP. These protocols are specified in



<netinet/in.h>. The value “0” may be used to select a default protocol from the selected domain and type.

The function returns -1 if an error occurred. Otherwise, it returns an integer representing the newly-assigned descriptor. Prototype

## bind()

bind() assigns a socket an address. When a socket is created using socket(), it is only given a protocol family, but not assigned an address. This association with an address must be performed with the bind() system call before the socket can accept connections to other hosts. bind() takes three arguments:

- sockfd, a descriptor representing the socket to perform the bind on
- my\_addr, a pointer to a sockaddr structure representing the address to bind to.
- addrlen, a socklen\_t field specifying the size of the sockaddr structure.

Bind() returns 0 on success and -1 if an error occurs.

Prototype

## listen()

After a socket has been associated with an address, listen() prepares it for incoming connections. However, this is only necessary for the stream-oriented (connection-oriented) data modes, i.e., for socket types (SOCK\_STREAM, SOCK\_SEQPACKET). listen() requires two arguments:

- sockfd, a valid socket descriptor.
- backlog, an integer representing the number of pending connections that can be queued up at any one time. The operating system usually places a cap on this value. Once a connection is accepted, it is dequeued. On success, 0 is returned. If an error occurs, -1 is returned.

Prototype

## accept()

When an application is listening for stream-oriented connections from other hosts, it is notified of such events (cf. select() function) and must initialize the connection using the accept() function. Accept() creates a new socket for each connection and removes the connection from the listen queue. It takes the **int**

**bind(int sockfd, const struct sockaddr\* my\_addr, socklen\_t addrlen);** following arguments:



- `sockfd`, the descriptor of the listening socket that has the connection queued.
- `cliaddr`, a pointer to a `sockaddr` structure to receive the client's address information.
- `addrlen`, a pointer to a `socklen_t` location that specifies the size of the client address structure passed to `accept()`. When `accept()` returns, this location indicates how many bytes of the structure were actually used.

The `accept()` function returns the new socket descriptor for the accepted connection, or -1 if an error occurs. All further communication with the remote host now occurs via this new socket

## **`int listen(int sockfd , int backlog);`**

Datagram sockets do not require processing by `accept()` since the receiver may immediately respond to the request using the listening socket.

### Prototype **`connect()`**

The `connect()` system call *connects* a socket, identified by its file descriptor, to a remote host specified by that host's address in the argument list.

Certain types of sockets are *connectionless*, most commonly user datagram protocol sockets. For these sockets, `connect` takes on a special meaning: the default target for sending and receiving data gets set to the given address, allowing the use of functions such as `send()` and `recv()` on connectionless sockets.

`connect()` returns an integer representing the error code: 0 represents success, while -1 represents an error.

### Prototype

#### ❖ **Sending and receiving data over a socket**

After a connection is established, there are several ways to send information over the socket.

### **`read()`**



The most common way of reading data from a socket is using the read () system call, which is defined like this:

- socket - The socket from which we want to read.
- buffer - The buffer into which the system will write the data bytes.
- buflen - Size of the buffer, in bytes (actually, how much data we want to read).

The read system call returns one of the following values:

- 0 - The connection was closed by the remote host.
- -1 - The read system call was interrupted, or failed for some reason.
- n - The read system call put 'n' bytes into the buffer we supplied it with.

Note that read() might read less than the number of bytes we requested, due to unavailability of buffer space in the system.

## **write()**

The most common way of writing data to a socket is using the write() system call, which is defined like this:

- socket - The socket into which we want to write.
- buffer - The buffer from which the system will read the data bytes.
- buflen - Size of the buffer, in bytes (actually, how much data we want to write).

The write system call returns one of the following values:

- 0 - The connection was closed by the remote host.
- -1 - The write system call was interrupted, or failed for some reason.
- n - The write system call wrote 'n' bytes into the socket.

Note that the system keeps internal buffers, and the write system call write data to those buffers, not necessarily directly to the network. Thus, a successful write() doesn't mean the data arrived at the other end, or was even sent onto the network. Also, it could be that only some of the bytes were written, and not the actual number we requested. It is up to us to try to send the data again later on, when it's possible, and we'll show several methods for doing just that.

## **sendto() and recvfrom() for DATAGRAM (UDP)**



- Since datagram sockets aren't connected to a remote host, we need to give the destination address before we send a packet.
- The prototype is:
- This call is basically the same as the call to send() with the addition of two other pieces of information.
- to is a pointer to a structsockaddr (which you'll probably have as a structsockaddr\_in and cast it at the last minute) which contains the destination IP address and port.
- tolen can simply be set to sizeof(structsockaddr).
- Just like with send(), sendto() returns the number of bytes actually sent (which, again, might be less than the number of bytes you told it to send!), or -1 on error.
- Equally similar are recv() and recvfrom().

**The prototype of recvfrom() is:**

- Again, this is just like recv() with the addition of a couple fields.
- from is a pointer to a local structsockaddr that will be filled with the IP address and port of the originating machine.
- fromlen is a pointer to a local int that should be initialized to sizeof(structsockaddr). When the function returns, fromlen will contain the length of the address actually stored in from. recvfrom() returns the number of bytes received, or -1 on error (with errno set accordingly).
- Remember, if you connect() a datagram socket, you can then simply use send() and recv() for all your transactions.
- The socket itself is still a datagram socket and the packets still use UDP, but the socket interface will automatically add the destination and source information for you.

## **Socket Address structsockaddr\_in**

```
structsockaddr_in { u_charsin_len;  
u_shortsin_family;    // Address family
```



```
u_short sin_port; // Port number
struct in_addr sin_addr; // Internet or IP
address char
```

```
    sin_zero[8]; // Same size as struct sockaddr
```

```
};
```

- The `sin_family` field is the address family (always `AF_INET` for TCP and UDP).
- The `sin_port` field is the port number, and the `sin_addr` field is the Internet address. The `sin_zero` field is reserved, and you must set it to hexadecimal zeroes.
- Data type `struct in_addr` - this data type is used in certain contexts to contain an Internet host address. It has just one field, named `s_addr`, which records the host address number as an unsigned long int.
- `sockaddr_in` is a "specialized" `sockaddr`.
- `sin_addr` could be `u_long`.
- `sin_addr` is 4 bytes and 8 bytes are unused.
- `sockaddr_in` is used to specify an endpoint. • The `sin_port` and `sin_addr` must be in Network Byte Order.





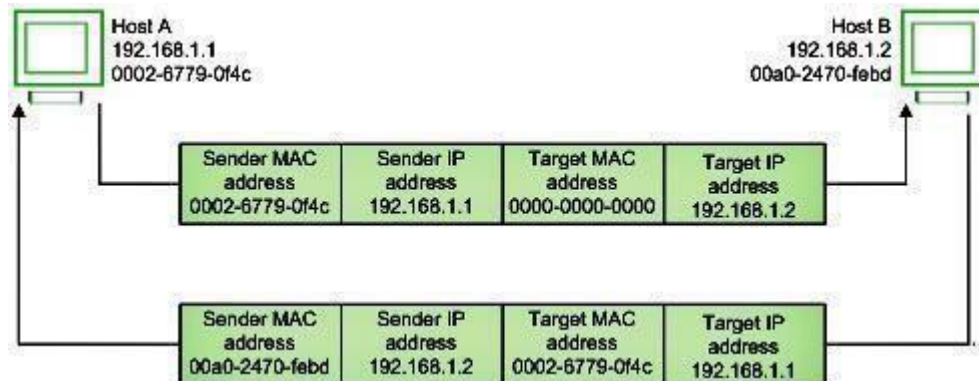
## EXPERIMENT-5

**Objective:** Write a code simulating ARP /RARP protocols

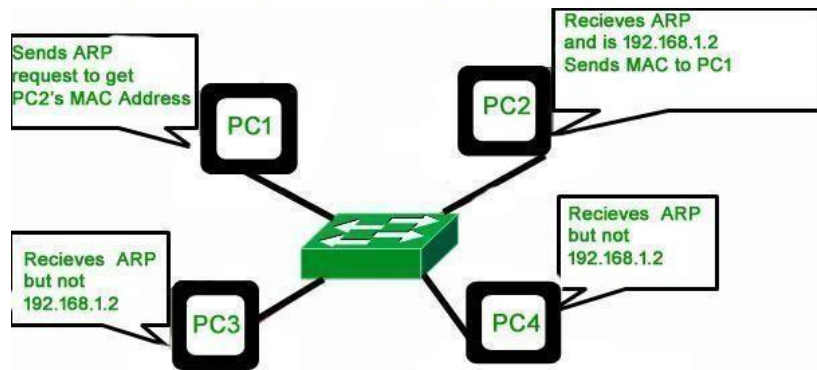
### Theory:

**Address Resolution Protocol (ARP)** – Address Resolution Protocol is a communication protocol used for discovering physical address associated with given network address. Typically, ARP is a network layer to data link layer mapping process, which is used to discover MAC address for given Internet Protocol Address.

In order to send the data to destination, having IP address is necessary but not sufficient; we also need the physical address of the destination machine. ARP is used to get the physical address (MAC address) of destination machine.



Before sending the IP packet, the MAC address of destination must be known. If not so, then sender broadcasts the ARP-discovery packet requesting the MAC address of intended destination. Since ARP-discovery is broadcast, every host inside that network will get this message but the packet will be discarded by everyone except that intended receiver host whose IP is associated. Now, this receiver will send a unicast packet with its MAC address (ARP-reply) to the sender of ARP-discovery packet. After the original sender receives the ARP-reply, it updates ARP-cache and start sending unicast message to the destination.



## Reverse Address Resolution Protocol (RARP) –

Reverse ARP is a networking protocol used by a client machine in a local area network to request its Internet Protocol address (IPv4) from the gateway-router's ARP table. The network administrator creates a table in gateway-router, which is used to map the MAC address to corresponding IP address.

When a new machine is setup or any machine which don't have memory to store IP address, needs an IP address for its own use. So the machine sends a RARP broadcast packet which contains its own MAC address in both sender and receiver hardware address field.



A special host configured inside the local area network, called as RARP-server is responsible to reply for these kind of broadcast packets. Now the RARP server attempt to find out the entry in IP to MAC address mapping table. If any entry matches in table, RARP server send the response packet to the requesting device along with IP address.

- LAN technologies like Ethernet, Ethernet II, Token Ring and Fiber Distributed Data Interface (FDDI) support the Address Resolution Protocol.
- RARP is not being used in today's networks. Because we have much great featured protocols like BOOTP (Bootstrap Protocol) and DHCP( Dynamic Host Configuration Protocol).

### C Program To Simulate ARP/RARP:

```
//ARP SERVER
```





```
#include<stdio.h>
#include<sys/types.h>
#include<sys/shm.h> #include<string.h>

main() { int shmid, a, i; char *ptr, *shmptr;
shmid=shmget(3000,10,IPC_CREAT | 0666);
shmptr=shmat(shmid,NULL,0); ptr=shmptr; for(i=0;i<3;i++)
{ puts("enter the mac");
scanf("%s",ptr);
a=strlen(ptr); printf("string
length:%d",a); ptr[a]= ' ' ;
puts("enter ip"); ptr=ptr+a+1;
scanf("%s",ptr); ptr[a]='\n' ;
ptr= ptr+a+1;
} ptr[strlen(ptr)]=
'\0';
printf("\n ARP table at serverside is=\n%s", shmptr); shmdt(shmptr);
}
```

ARP table at serverside is a.b.c.d

1.2.3.4

e.f.g.h 5.6.7.8

i.j.k.l 9.1.2.3

## //ARP CLIENT

```
#include<stdio.h>
#include<string.h>
#include<sys/types.h>
#include<sys/shm.h> main()
{
int shmid,a; char *ptr, *shmptr; char
ptr2[51], ip[12], mac[26];
shmid=shmget(3000,10,0666);
shmptr=shmat(shmid,NULL,0);
puts("the arp table is");
printf("%s",shmptr); printf("\n1.ARP\n
2.RARP\n 3.EXIT\n"); scanf("%d",&a);
switch(a) { case 1: puts("enter ip
address"); scanf("%s",ip);
```



```
ptr=strstr(shmptr, ip); ptr-=8;
sscanf(ptr,"%s%s",ptr2); printf("mac addr
is %s",ptr2); break; case 2:
puts("enter mac addr"); scanf("%s",mac);
ptr=strstr(shmptr, mac); sscanf(ptr,"%*s%s",ptr2);
printf("%s",ptr2); break; case 3: exit(1);
} }
```

## SAMPLE INPUT OUTPUT:

the arp table is a.b.c.d

1.2.3.4

e.f.g.h 5.6.7.8

i.j.k.l 9.1.2.3

1.ARP

2.RARP 3. EXIT enter your choice: 1 enter ip address: 1.2.3.4 mac addr is a.b.c.d

enter your choice:2 enter

mac address: e.f.g.h ip addr

is 5.6.7.8



## EXPERIMENT-6

**Objective:** Write a code simulating PING and TRACEROUTE commands

### Theory:

In computer networks, data is sent in small blocks known as packets. Each packet is transmitted individually and may also follow a different route to reach the destination. Once all these packets of the original message reach the destination, they are re-assembled to form the original message. But, sometimes, it may happen that the webserver is down, network congestion, or some other technical glitch is there, that may prevent the message from reaching the destination. To diagnose such congestions and network failures, we use two common programs namely Ping and Traceroute.

**Ping** – It is a utility that helps one to check if a particular IP address is accessible or not. Ping works by sending a packet to the specified address and waits for the reply. It also measures round trip time and reports errors.

Ping is also used in checking if the computers on a local network are active. For this, the user has to go in command prompt and type: ping 127.0.0.1, and if the address is active, the ping would return a message like this :

Pinging 127.0.0.1 with 32 bytes of data

Reply from 127.0.0.1: bytes=32 time<10ms TTL=32

Reply from 127.0.0.1: bytes=32 time<10ms TTL=32

Reply from 127.0.0.1: bytes=32 time<10ms TTL=32

Reply from 127.0.0.1: bytes=32 time<10ms TTL=32

The IP address 127.0.0.1 is the address of the local host and would receive a ping reply even if the sender is not connected to the internet.

**Traceroute** – It is a utility that traces a packet from your computer to the host, and will also show the number of steps (hops) required to reach there, along with the time by each step. Traceroute works by sending the packets of data with low survival time (Time to Live – TTL) which specifies how many steps (hops) can the packet survive before it is returned. When a packet can't reach the final destination and expires at an intermediate step, that node returns the packet and identifies itself. So, by increasing the TTL gradually, Traceroute is able to identify the intermediate hosts. If any of the hops come back with "Request timed out", it denotes network congestion and a reason for slow loading Web pages and dropped connections.

The main difference between Ping and Traceroute is that Ping is a quick and easy utility to tell if the specified server is reachable and how long will it take to send and receive data from the server whereas Traceroute finds the exact route taken to reach the server and time taken by each step (hop).

### **Algorithm:**



Step1: Start the program.

Step2: Import necessary packages.

Step3: Initialize the ping server with both sockets as null value.

Step4: Start the server socket.

Step5: At the client give the IP address of the server(by using ifconfig command in command prompt).

Step6: The client program is then started by starting socket.

Step7: At the receiver end, the client is pinged and traced. Step8: Stop the program.

## *Client Step1:*

Start the program.

Step2: Import necessary packages.

Step3: Initialize the ping client with both sockets as null value.

Step4: Start the socket.

Step5: Get the IP address of the server.

Step6: Ping the server.

Step7: At the receiver end, the server is pinged and traced. Step8:

Stop the program.

pingclient.java

```

/*...localhostport name and 5555-port number...*/ Socket
s=new Socket("127.0.0.1",5555);

/*... Get an input file handle from the socket and read the input...*/
DataInputStream dis=new DataInputStream(s.getInputStream()); PrintStream out=new
PrintStream(s.getOutputStream()); while(c<4){
    ...returns the current time in milliseconds... */
t1=System.currentTimeMillis();
str="Welcome to network programming
world"; out.println(str);
/*...readline() method read a line of text...*/
System.out.println(dis.readLine()); t2=System.currentTimeMillis();
System.out.println(";TTL="+t2-t1+"ms"); c++; pingserver.java

/*...ServerSocket object is used to establish the communication with
clients...*/ ServerSocketss=new ServerSocket(5555);

/*...accept(): Used to accept the client request...*/
Socket s=ss.accept();
int c=0; while(c<4) {

```



```
/*... Get an input file  
handle from the  
socket and read the  
input...*/
```

```
DataInputStream dis=new DataInputStream(s.getInputStream());
```

```
PrintStream out=new PrintStream(s.getOutputStream());
```

```
/*...readline() method read a line of text...*/
```

```
String str=dis.readLine(); out.println("Reply  
from"+InetAddress.getLocalHost()+";Length"+str.length()); c++;
```

**Sample**

**Output:**

```
user@administrator-ThinkCentre-E73: ~/Desktop  
user@administrator-ThinkCentre-E73:~/Desktop$ java pingclient  
Reply fromadministrator-ThinkCentre-E73/127.0.1.1;Length36  
;TTL=35ms  
Reply fromadministrator-ThinkCentre-E73/127.0.1.1;Length36  
;TTL=79ms  
Reply fromadministrator-ThinkCentre-E73/127.0.1.1;Length36  
;TTL=80ms  
Reply fromadministrator-ThinkCentre-E73/127.0.1.1;Length36  
;TTL=54ms  
user@administrator-ThinkCentre-E73:~/Desktop$
```

```
user@administrator-ThinkCentre-E73: ~/Desktop  
user@administrator-ThinkCentre-E73:~$ cd D*  
user@administrator-ThinkCentre-E73:~/Desktop$ javac pingserver.java  
Note: pingserver.java uses or overrides a deprecated API.  
Note: Recompile with -Xlint:deprecation for details.  
user@administrator-ThinkCentre-E73:~/Desktop$ java pingserver  
user@administrator-ThinkCentre-E73:~/Desktop$
```

## **Viva questions:**

1. Dose ping can measure round trip time?
2. What is ping sweep to trace the path from source to destination?
4. Difference between traceroute and ping.
- 5.What utility is used to find the number of routers between a source and destination?
6. Which protocol does ping use?
7. What is the 'ping' command useful for?





## EXPERIMENT-7

**Objective:** Create a socket for HTTP for web page upload and download

### Concept:

Concurrent Server: The server can be iterative, i.e. it iterates through each client and serves one request at a time. Alternatively, a server can handle multiple clients at the same time in parallel, and this type of a server is called a concurrent server.

### Algorithm: Server

**Step 1:** Create a socket and bind to the address. Leave socket unconnected.

**Step 2 :** Leave socket in passive mode, making it ready for use by a server.

**Step 3:** Repeatedly call accept to receive the next request from a client to handle the response with the through socket.

### Client

**Step 1:** Begin with a connection passed from the server (i.e., a socket for the connection).

**Step 2:** Use input streams; get the message from user to be given to the server.

**Step 3:** Use input streams read message given by server and print it.

**Step 4:** Use output streams to write message to the server.

**Step 5:** Close the connection and exit, i.e., slave terminates after handling all requests from one client.

### Sample Program:

#### ConServer.java

```
/*... Register service on port 8020...*/
ServerSocketss=new ServerSocket(8500);
System.out.println("Waiting for client..."); while(true)

    /*... ServerSocket in order to listen for and accept connections from clients...*/

    {Socket s=ss.accept();
    /*...getInputStream()-This method take the permission to write the data from client program
    toserver program and server program to client program...*/ BufferedReader
    br=new BufferedReader(new InputStreamReader(s.getInputStream())); cli_name=br.readLine();
    System.out.println("\nCLIENT NAME: "+cli_name); no=Integer.parseInt(br.readLine()); sq=no*no;
    PrintWriter pw=new PrintWriter(s.getOutputStream(),true); pw.println(sq);
```





```
System.out.println("OUTPUT - The square of "+no+" is "+sq);}}}
```

## ConClient1.java

```
{Socket s=new Socket("localhost",8500);
```

```
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
```

**/\*...Integer.parseInt() java method is used primarily in parsing a String method argument into an Integer object. The Integer object is a wrapper class for the int primitive data type ...\*/**

```
int num=Integer.parseInt(br.readLine());
```

**/\* ...getOutputStream()-This method is used to take the permission to read data from client system by the server or from the server system by the client...\*/**

```
PrintWriter pw=new PrintWriter(s.getOutputStream(),true); pw.println("Client  
1"); pw.println(num);
```

```
BufferedReader br1=new BufferedReader(new InputStreamReader(s.getInputStream()));  
intsqu=Integer.parseInt(br1.readLine());  
System.out.println("Square of "+num+" is "+squ+"\n");
```

## ConClient2.java

```
Socket s=new Socket("localhost",8500);
```

```
BufferedReader br=new BufferedReader(new InputStreamReader(System.in)); System.out.println("\nCLIENT  
2:\nEnter the number to find square: "); intnum=Integer.parseInt(br.readLine());
```

```
PrintWriter pw=new PrintWriter(s.getOutputStream(),true); pw.println("Client  
2"); pw.println(num);
```

```
BufferedReader br1=new BufferedReader(new InputStreamReader(s.getInputStream()));  
intsqu=Integer.parseInt(br1.readLine()); System.out.println("Square of  
"+num+" is "+squ+"\n");s.close();
```



```
skct@administrator-Lenovo-S510: ~/Desktop
skct@administrator-Lenovo-S510:~$
skct@administrator-Lenovo-S510:~$
skct@administrator-Lenovo-S510:~$ cd D*
skct@administrator-Lenovo-S510:~/Desktop$ javac ConClient1.java
skct@administrator-Lenovo-S510:~/Desktop$ java ConClient1

CLIENT 1:
Enter the number to find square:
7
Square of 7 is 49

skct@administrator-Lenovo-S510:~/Desktop$
```

```
skct@administrator-Lenovo-S510: ~/Desktop
skct@administrator-Lenovo-S510:~$ cd D*
skct@administrator-Lenovo-S510:~/Desktop$ javac ConServer.java
skct@administrator-Lenovo-S510:~/Desktop$ java ConServer
Waiting for client...

CLIENT NAME: Client 2
OUTPUT - The square of 5 is 25

CLIENT NAME: Client 1
OUTPUT - The square of 7 is 49
```

```
skct@administrator-Lenovo-S510: ~/Desktop
skct@administrator-Lenovo-S510:~$ cd D*
skct@administrator-Lenovo-S510:~/Desktop$ javac ConClient2.java
skct@administrator-Lenovo-S510:~/Desktop$ java ConClient2

CLIENT 2:
Enter the number to find square:
5
Square of 5 is 25

skct@administrator-Lenovo-S510:~/Desktop$
```

### Viva questions:

1. Define server and what are the types of server?
2. What are the three types of socket function?



3. What are concurrent servers?
4. Define Iterative server
5. Compare Iterative server and concurrent server
6. Explain socket address structure
7. List some character stream support classes 8.

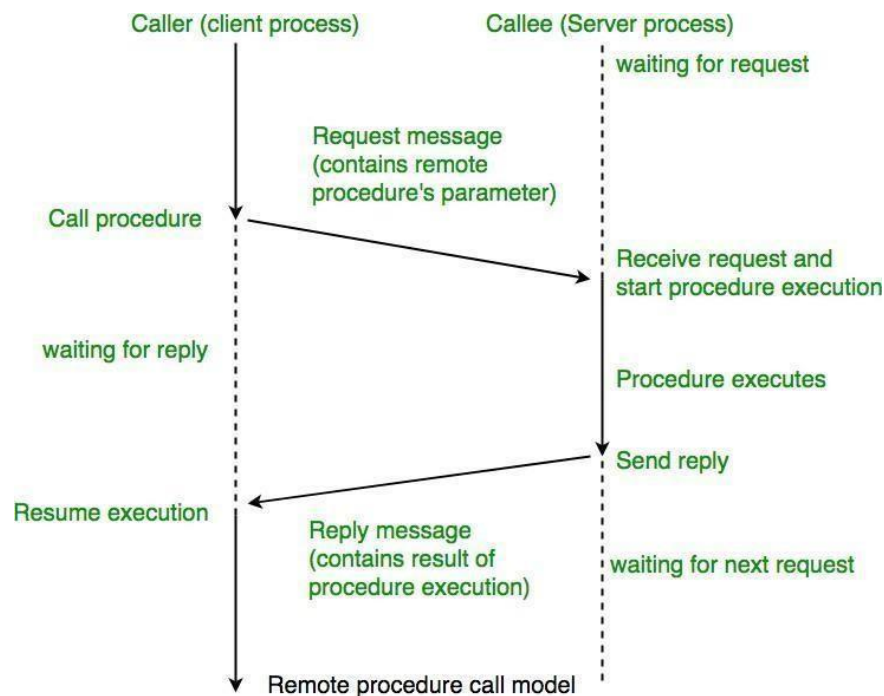
What do you mean by socket programming?



## EXPERIMENT-8

**Objective:** Write a program to implement RPC (Remote Procedure Call)

**Theory:** Remote Procedure Call (RPC) is a powerful technique for constructing **distributed, client-server based applications**. It is based on extending the conventional local procedure calling so that the **called procedure need not exist in the same address space as the calling procedure**. The two processes may be on the same system, or they may be on different systems with a network connecting them.



- 1.The calling environment is suspended, procedure parameters are transferred across the network to the environment where the procedure is to execute, and the procedure is executed there.
- 2.When the procedure finishes and produces its results, its results are transferred back to the calling environment, where execution resumes as if returning from a regular procedure call.

### The Programmer's Interface to RPC

This section addresses the C interface to RPC and describes how to write network applications using RPC. For a complete specification of the routines in the RPC library, see the rpc and related man pages.

### Simplified Interface



The simplified interface is the easiest level to use because it does not require the use of any other RPC routines. It also limits control of the underlying communications mechanisms. Program development at this level can be rapid, and is directly supported by the rpcgen compiler. For most applications, rpcgen and its facilities are sufficient. Some RPC services are not available as C functions, but they are available as RPC programs. The simplified interface library routines provide direct access to the RPC facilities for programs that do not require fine levels of control.

Routines such as rusers are in the RPC services library librpcsvc. rusers.c, below, is a program that displays the number of users on a remote host. It calls the RPC library routine, rusers.

## **Program:**

```
#include"rpc/rpc.h"

#include"square.h"

#include"stdio.h"

#include"stdlib.h" #include"math.h" square_out*squareproc_1_svc(square_in*inp,structsvc_req*rqstp)
{ staticsquare_out out;    out.resl=inp->arg1
*inp->arg1; return(&out);
}

// CLIENT FILENAME: client.c

#include"errno.h"

#include"rpc/rpc.h"

#include"square.h"

#include"stdio.h"

#include"stdlib.h" #include"math.h"

int main(intargc,char **argv)
{
    CLIENT *cl; square_in in;
square_out *outp; f(argc!=3)
    {
        printf("\n\n error:insufficient arguments!!!");
exit(-1);
    }
}
```





```
cl=clnt_create(argv[1],SQUARE_PROG,SQUARE_VERS,"tcp"); in.arg1=atol(argv[2]);
if(cl==NULL)
{
printf("\nerror:%s",strerror(errno));      exit(-1);
}
if((outp=squareproc_1(&in,cl))==NULL)
{
printf("\nerror   :%s",clnt_spperror(cl,argv[1]));      exit(-1);
}
printf("\n\n  result is      :      %ld",outp->res1);
exit(0);
}
// .h FILENAME: square.h
structsquare_in { /*input arg*/
long arg1;
};
structsquare_out {
/*op result*/ long
res1;
};
program SQUARE_PROG
{
version SQUARE_VERS
{
square_out SQUAREPROC(square_in)=1;/*proc no=1*/
}=1;/*version no*/
}=0x31230000;/*prog no*/ Output:
[root@localhost~]#rpcgen -C square.x
[root@localhost~]#cc -c client.c -o client.o
[root@localhost~]#cc -c square_clnt.c -o square_clnt.o
```





```
[root@localhost~]#cc -c square_xdr.c -o square.xdr.o
[root@localhost~]#cc -o client client.osquare_clnt.osquare_xdr.o
[root@localhost~]#cc -c client.cserver.csquare_xdr.c
[root@localhost~]#cc -c server.c -o server.o
[root@localhost~]#cc -c square_svc.c -o square_svc.o
[root@localhost~]#cc -o server server.osquare_svc.osquare_xdr.o
[root@localhost~]#./server &
[1] 2264
[root@localhost~]#./client localhost 4 result is:16
```



## EXPERIMENT-9

**Objective:** Implementation of Subnetting

### Theory:

If an organization was granted a large block in class A or B, it could divide the addresses into several contiguous groups and assign each group to smaller networks (called subnets) or, in rare cases, share part of the addresses with neighbors.

### Algorithm:

**Step1:** Get the input from the user by using scanner method.

**Step 2:** Read the input by using `nextLine()` and store it.

**Step 3:** Split the string based on string by using

`split("\\")`**Step4 :** Convert it into binary.

**Step 5:** calculating the network mask by using math and logarithmic **Step 6:**

get the first address by ANDding the last n bits with 0.

**Step7 :** get the last address by ANDding the last n bits with 1.

### Sample Coding:

```
//...Calculation of mask...// int bits =
(int)Math.ceil(Math.log(n)/Math.log(2)); /*eg if address = 120, log 120/log 2 gives log to the
base 2 => 6.9068, ceil gives us upper integer */
System.out.println("Number of bits required for address = "+bits); int
mask = 32-bits;

System.out.println("The subnet mask is = "+mask);

//...Calculation of first address and last address...//
intfbip[] = new int[32]; for(int i=0; i<32;i++) fbip[i]
= (int)bip.charAt(i)-48;

//convert cahraacter 0,1 to integer 0,1
for(int i=31;i>31-bits;i--)//Get first address by ANDing last n bits with 0 fbip[i] &=
```



```
0; String fip[] = {"", "", "", ""}; for(int i=0;i<32;i++)
fip[i/8] = new String(fip[i/8]+fbip[i]);

System.out.print("First address is = ");

for(int i=0;i<4;i++)
{
System.out.print(Integer.parseInt(fip[i],2)); if(i!=3)

System.out.print(".");

}
```

### Sample Output:

```
skct@administrator-Lenovo-S510: ~/Desktop
skct@administrator-Lenovo-S510:~$ cd D*
skct@administrator-Lenovo-S510:~/Desktop$ javac sub.java
skct@administrator-Lenovo-S510:~/Desktop$ java sub
ENTER IP:
172.17.2.64
MASK:
255.255.0.0
ADDRESS:
172.17.0.0.
skct@administrator-Lenovo-S510:~/Desktop$
```

### Viva questions:

1. What are the advantages of subnetting?
2. Your router has the following IP address on Ethernet0: 172.16.2.1/23. Which of the following can be valid host IDs on the LAN interface attached to the router?
  - a. 172.16.1.100
  - 2) 172.16.1.198
  - 3) 172.16.2.255
  - 4) 172.16.3.0

b. 1 only

c. 2 and 3 only

d. 3 and 4 only

e. None of the above



3. A network administrator is connecting hosts A and B directly through their Ethernet interfaces, as shown in the illustration. Ping attempts between the hosts are unsuccessful. What can be done to provide connectivity between the hosts?
4. What is the maximum number of IP addresses that can be assigned to hosts on a local subnet that uses the 255.255.255.224 subnet mask?
5. If an Ethernet port on a router were assigned an IP address of 172.16.112.1/25, what would be the valid subnet address of this host?
6. What are the network address, broadcast address, and the subnet mask for a host with the IP Address below? IP Address: 101. 39. 85. 201/ 23 Network Address: 101.39.84.0 Subnet Mask: 255.255.254.0 Broadcast Address: 101.39.85.255
7. On a VLSM network, which mask should you use on point-to-point WAN links in order to reduce the waste of IP addresses?
8. To test the IP stack on your local host, which IP address would you ping?



## EXPERIMENT-10(a)

**Objective:** Applications using TCP Sockets like Chat

A server program to establish the socket connection with the client for performing chat.

A client program which on establishing a connection with the server for performing chat.

### Concept:

It uses TCP socket communication .We have a server as well as a client.

Both can be run in the same machine or different machines. If both are running in the machine, the address to be given at the client side is local host address.

If both are running in different machines, then in the client side we need to specify the ip address of machine in which server application is running.

### Algorithm:

#### **Server**

**Step1:** Start the program and create server and client

sockets. Step2: Use input streams to get the message from user.

Step3: Use output streams to send message to the client.

Step4: Wait for client to display this message and write a new one to be displayed by the server.

Step5: Display message given at client using input streams read from socket.

Step6: Stop the program.

#### **Client**

Step1: Start the program and create a client socket that connects to the required host and port. Step2: Use input streams read message given by server and print it. Step3:

Use input streams; get the message from user to be given to the server. Step4: Use output streams to write message to the server.

Step5: Stop the program.



## Program: GossipServer.java

```
ServerSocket ssock = new ServerSocket(3000);
System.out.println("Server ready for chatting");
Socket sock = ssock.accept();

    /*...reading from keyboard (keyRead object)...*/
    BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));

    /*...sending to client (pwrite object)...*/ OutputStream ostream
    = sock.getOutputStream();

    PrintWriter pwrite = new PrintWriter(ostream, true);

    /*... receiving from server ( receiveRead object)...*/
    InputStream istream = sock.getInputStream();
    BufferedReader receiveRead = new BufferedReader(new
    InputStreamReader(istream)); String receiveMessage, sendMessage; while(true)
    {if((receiveMessage = receiveRead.readLine()) != null)
    {System.out.println(receiveMessage); }
```

## GossipClient.java

```
Socket sock = new Socket("127.0.0.1", 3000);

    /*...reading from keyboard (keyRead object)...*/
    BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));

    /*...sending to client (pwrite object)...*/ OutputStream ostream
    = sock.getOutputStream();

    PrintWriter pwrite = new PrintWriter(ostream, true);

    /*... receiving from server ( receiveRead object)...*/ InputStream istream =
    sock.getInputStream();

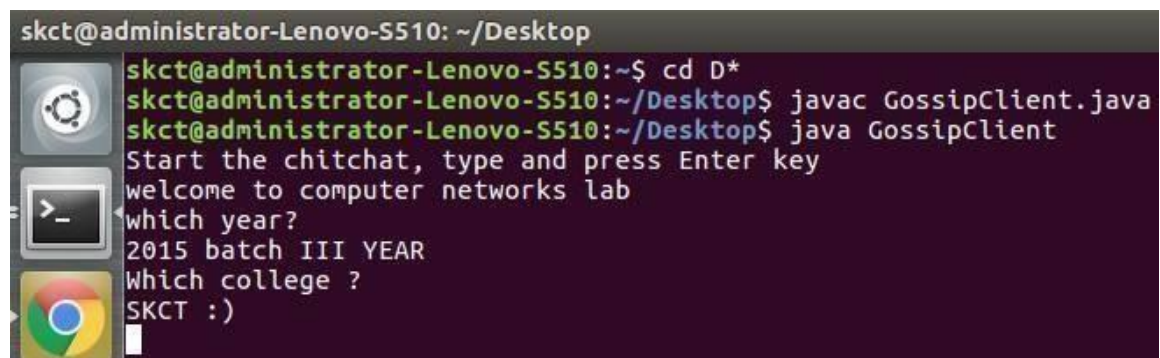
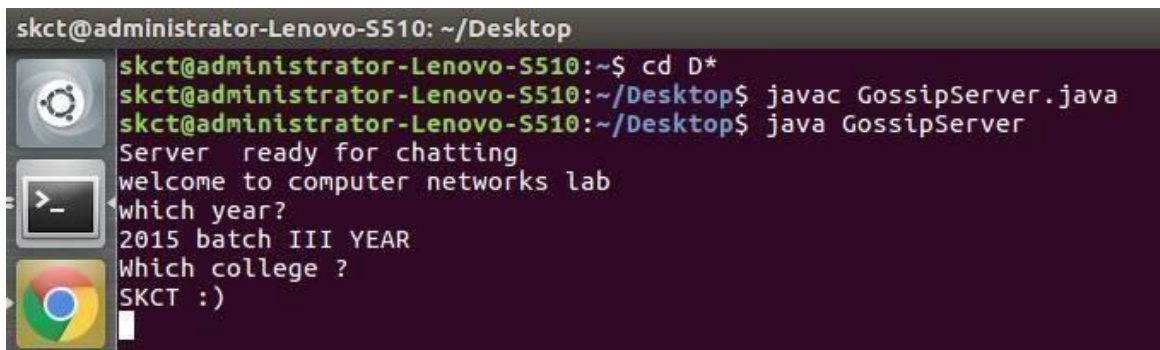
    BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));
    System.out.println("Start the chitchat, type and press Enter key"); String receiveMessage,
    sendMessage; while(true)
```





```
{sendMessage = keyRead.readLine(); /*... keyboard reading
...*/pwrite.println(sendMessage); /*... sending to server...*/ pwrite.flush();
/*... flush the data...*/ if((receiveMessage = receiveRead.readLine()) != null) /*...receive from
server...*/
{System.out.println(receiveMessage);} /*... displaying at DOS prompt...*/
```

## Sample Output:





## EXPERIMENT-10(b)

**Objective:** Applications using TCP Sockets like file Transfer

### Algorithm

#### Server

**Step1:** Import java packages and create class file server. **Step2:**

Create a new server socket and bind it to the port.

**Step3:** Accept the client connection

**Step4:** Get the file name and stored into the BufferedReader.

**Step5:** Create a new object class file and realine.

**Step6:** If file is exists then FileReader read the content until EOF is reached. **Step7:** Stop the program.

#### Client

**Step1:** Import java packages and create class file server.

**Step2:** Create a new server socket and bind it to the port.

**Step3:** Now connection is established.

**Step4:** The object of a BufferedReader class is used for storing data content which has been retrieved from socket object.

**Step5:** The content of file is displayed in the client window and the connection is closed. **Step6:** Stop the program.

### Sample Code:

#### Se.java

```
{/*... Register service on port 15123...*/
```

```
ServerSocketserverSocket = new ServerSocket(15123);
```

```
/*... Wait and accept a connection...*/
```

```
Socket socket = serverSocket.accept();
```

```
System.out.println("Accepted connection : " + socket);
```

```
/*... enter the source file name which is to be transferred to client...*/
```

```
File transferFile = new File ("sunithanandhinifiletransfer.doc"); byte
```



```
[] bytearray = new byte [(int)transferFile.length()];
/*...FileInputStream is meant for reading streams of raw bytes such as image data...*/ FileInputStream
fin = new FileInputStream(transferFile);
BufferedInputStream bin = new
BufferedInputStream(fin);bin.read(bytearray,0,bytearray.length);

/*... Get a communication stream associated with the
socket...*/OutputStream os = socket.getOutputStream();
System.out.println("Sending Files...");
os.write(bytearray,0,bytearray.length); os.flush();socket.close();
```

## Cl.java

```
/*... Open your connection to a server, at port 15123...*/
Socket socket = new Socket("127.0.0.1",15123); byte [] bytearray = new byte [filesize];
InputStream is = socket.getInputStream();

/*... enter the destinationfile name which is to be transferred ...*/ FileOutputStream
fos = new
FileOutputStream("kalpanasonikafiletransfer.doc"); BufferedOutputStream bos
= new BufferedOutputStream(fos); bytesRead =
is.read(bytearray,0,bytearray.length); currentTot
= bytesRead;

do { bytesRead = is.read(bytearray, currentTot, (bytearray.length-currentTot));
if(bytesRead>= 0) currentTot += bytesRead;} while(bytesRead>
-1); bos.write(bytearray, 0 , currentTot); bos.flush(); bos.close(); socket.close();
} }
```

```
Terminal
skct@administrator-Lenovo-S510: ~/Desktop
skct@administrator-Lenovo-S510:~$ cd D*
skct@administrator-Lenovo-S510:~/Desktop$ javac Cl.java
skct@administrator-Lenovo-S510:~/Desktop$ java Cl
skct@administrator-Lenovo-S510:~/Desktop$
```



## **Viva questions:**

1. What are the types of protocol?
2. Define socket.
3. What information is needed to create a TCP Socket?
4. What are the two important TCP Socket classes?
5. What is the difference between the File and Random Access File classes?
6. What are some advantages and disadvantages of Java Sockets?
7. How to build File Input Stream object with byte array as a parameter
8. How to read file in byte array with File Input Stream



## EXPERIMENT-11

**Objective:** Applications using TCP and UDP Sockets like DNS & SNMP **Theory:**

1. The DNS client program sends a request to a DNS server to map the e-mail address to the corresponding IP address.
2. When the Internet was small, mapping was done by using a host file. The host file had only two columns: name and address.
3. The host that needs mapping can contact the closest computer holding the needed information. This method is used by the Domain Name System (DNS).

### **Algorithm:**

#### **Server**

**Step1:** Start the program.

**Step2:** Create the socket for the server.

**Step3:** Bind the socket to the port.

**Step4:** Listen for the incoming client connection.

**Step5:** Receive the IP address from the client to be resolved.

**Step6:** Get the domain name for the client.

**Step7:** Check the existence of the domain in the server.

**Step8:** If domain matches then send the corresponding address to the client.

**Step9:** Stop the program execution

#### **Client**

**Step1:** Start the Program.

**Step2:** Create the socket for the client.

**Step3:** Connect the socket to the Server.

**Step4:** Send the host name to the server to be resolved.

**Step5:** If the server corresponds then print the address and terminate the process

### **Sample Program:**



## Clientdns12.java

***/\*... datagram socket is the sending or receiving point for a packet delivery service.***

```
DatagramSocket client=new DatagramSocket();
```

***/\*...InetAddress class provides methods to get the IP of any host name...\*/***

```
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 DOMAIN NAME or IP address:"); String
```

```
str=in.readLine(); sendbyte=str.getBytes();
```

***/\*...send the data to the server(data,length,ip address and port number)...\*/*** 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("IP address or DOMAIN NAME: "+s.trim());
```

## **Serverdns12.java**

```
DatagramSocket server=new
```

```
DatagramSocket(1309); while(true)
```

```
{byte[] sendbyte=new byte[1024]; byte[]
```

```
receivebyte=new byte[1024];
```

***/\*..receiving the packet from client...\*/***

```
DatagramPacket receiver=new
```

```
DatagramPacket(receivebyte,receivebyte.length)
```

```
; server.receive(receiver);
```

```
String str=new String(receiver.getData());
```

```
String s=str.trim();
```

```
//System.out.println(s);
```

```
InetAddressaddr=receiver.getAddress(); int port=receiver.getPort();
```

***/\*... specify the IP address to map with its domain name...\*/*** String

```
ip[]={ "165.165.80.80", "165.165.79.1" };
```





```
/*domain name...*/  
String name[]={"www.skct.edu","www.sonika.com"};  
for(int i=0;i<ip.length;i++)  
{if(s.equals(ip[i]))  
  
{sendbyte=name[i].getBytes();  
DatagramPacket sender=new  
DatagramPacket(sendbyte,sendbyte.length,addr,port); server.send(sender); break;}  
else if(s.equals(name[i])){  
sendbyte=ip[i].getBytes();  
DatagramPacket sender=new  
DatagramPacket(sendbyte,sendbyte.length,addr,port); server.send(sender);
```

## Sample Output:

```
skct@administrator-Lenovo-S510: ~/Desktop  
skct@administrator-Lenovo-S510:~$ cd D*  
skct@administrator-Lenovo-S510:~/Desktop$ javac Serverdns12.java  
^[[skct@administrator-Lenovo-S510:~/Desktop$ java Serverdns12  
skct@administrator-Lenovo-S510:~/Desktop$ java Serverdns12  
skct@administrator-Lenovo-S510:~/Desktop$  
  
skct@administrator-Lenovo-S510:~/Desktop$ java Clientdns12  
Enter the DOMAIN NAME or IP address:  
www.sonika.com  
IP address or DOMAIN NAME: 165.165.79.1  
skct@administrator-Lenovo-S510:~/Desktop$
```

## APPLICATIONS (SNMP)

### Concept:

1. Simple Network Management Protocol (SNMP) is a framework for managing devices in an internet using TCP/IP.
2. It provides a set of fundamental operations for monitoring and maintaining an internet.
3. SNMP uses the concept of manager and agent  
➤ A manager is a host that runs the SNMP client program.



- A managed station called an agent, is a router that runs the SNMP server program

## Algorithm:

**Step 1:** Using start method the system is used to receive SNMP request.

**Step 2:** The assigned Port no 162 are used to send and receive trap.

**Step 3:** Set the address using the format ("udp:127.0.0.1/161")

**Step 4:** Create a variable binding and add the object identifier in OID format.

**Step 5:** Create the Protocol data unit object

**Step 6:** Listen enable listening for SNMP packet by using listen ().

**Step 8:** Get method asynchronous GET request PDU is send to the given target. **Step 7:**

Send a new get request for single OID and return response event for the request only, if notimeout has occurred.

**Step 8:** Send a new get request for multiple OIDS and return response event for the request only, if notimeout has occurred.

**Step 9:** Create target method which contains information about where the data should be fetched and how to return.

**Step 10:** If response is not NULL then no time out has occurred and the response was successfully delivered.

## Sample Code:

```
publicSNMPManager(String add)
{address = add; public static void main(String[] args)
throws IOException {
```

```
/*...Port 161 is used for Read and Other operations, Port 162 is
used for the trap generation ...*/
```

```
SNMPManager client = new
```

```
SNMPManager("udp:127.0.0.1/161"); client.start();
```

```
/*...OID - .1.3.6.1.2.1.1.1.0 =>SysDec, OID - .1.3.6.1.2.1.1.5.0
```



**=>SysName...\*/**

```
String sysDescr = client.getAsString(new OID(".1.3.6.1.2.1.1.1.0"));
System.out.println(sysDescr);}
```

**/\*... the listen() method listens for answers...\*/**

```
private void start() throws IOException {
    TransportMapping transport = new
    DefaultUdpTransportMapping(); snmp =
    new Snmp(transport); transport.listen();}
```

**/\*...Method which takes a single OID and returns the response from the agent as a String...\*/**

```
public String getAsString(OID oid) throws IOException { ResponseEvent
event = get(new OID[]{oid}); return event.getResponse().get(0).get
Variable().toString();}
```

**/\*...This method is capable of handling multiple OIDs paramoids Eturn throws IOException ...\*/**

```
public ResponseEvent get(OID oids[]) throws IOException { PDU pdu
= new PDU(); for (OID oid : oids) { pdu.add(new
VariableBinding(oid));} pdu.setType(PDU.GET);
ResponseEvent event = snmp.send(pdu, getTarget(), null); if(event != null) { return
event;} throw new RuntimeException("GET
timed out");}
```

**/\*... This method returns a Target, which contains information about where the data should be fetched and how to return ...\*/** private Target  
getTarget() {



```
Address targetAddress = GenericAddress.parse(address);  
CommunityTarget target = new CommunityTarget();  
target.setCommunity(new OctetString("public"));  
target.setAddress(targetAddress); target.setRetries(2);  
target.setTimeout(1500);  
target.setVersion(SnmpConstants.version2c); return target;}}
```

### **Sample Output:**

Hardware: x86 Family 6 Model 23 Stepping 10 AT/AT COMPATIBLE –

Software: Windows 2000 Version 5.1 (Build 2600 Multiprocessor Free)



## EXPERIMENT-12

**Objective:** Study of Network simulator (NS).and Simulation of Congestion Control

Algorithms using NS

### NET WORK SIMULATOR (NS2)

#### Ns overview

- ☐ Ns programming: A Quick start
- ☐ Case study I: A simple Wireless network
- ☐ Case study II: Create a new agent in Ns
- ☐ Ns Status
- ☐ Periodical release (ns-2.26, Feb 2003)
- ☐ Platform support
- ☐ FreeBSD, Linux, Solaris, Windows and Mac

#### Ns Functionalities

Routing, Transportation, Traffic sources, queuing disciplines, QoS

#### Wireless

Ad hoc routing, mobile IP, sensor-MAC Tracing, visualization and various utilities NS(Network Simulators) Most of the commercial simulators are GUI driven, while some network simulators are CLI driven. The network model / configuration describe the state of the network (nodes, routers, Switches and links) and the events (data transmissions, packet error etc.). The important outputs of simulations are the trace files. Trace files log every packet, every event that occurred in the simulation and are used for analysis. Network simulators can also provide other tools to facilitate visual analysis of trends and potential trouble spots.

Most network simulators use discrete event simulation, in which a list of pending "events" is stored, and those events are processed in order, with some events triggering future events such as the event of the arrival of a packet at one node triggering the event of the arrival of that packet at a downstream node. Simulation of networks is a very complex task. For example, if congestion is high, then estimation of the average occupancy is challenging because of high variance. To estimate the likelihood of a buffer overflow in a network, the time required for an accurate answer can be extremely large. Specialized techniques such as "control variants" and "importance sampling" have been developed to speed simulation.





## ***Examples of network simulators***

There are many both free/open-source and proprietary network simulators. Examples of notable network simulation software are, ordered after how often they are mentioned in research papers:

- ☐ ns (open source)
- ☐ OPNET (proprietary software)
- ☐ NetSim (proprietary software)

## ***Uses of network simulators***

Network simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple networked computers, routers and data links, network simulators are relatively fast and inexpensive. They allow engineers, researchers to test scenarios that might be particularly difficult or expensive to emulate using real hardware - for instance, simulating a scenario with several nodes or experimenting with a new protocol in the network. Network simulators are particularly useful in allowing researchers to test new networking protocols or changes to existing protocols in a controlled and reproducible environment. A typical network simulator encompasses a wide range of networking technologies and can help the users to build complex networks from basic building blocks such as a variety of nodes and links. With the help of simulators, one can design hierarchical networks using various types of nodes like computers, hubs, bridges, routers, switches, links, mobile units etc. Various types of Wide Area Network (WAN) technologies like TCP, ATM, IP etc. and Local Area Network (LAN) technologies like Ethernet, token rings etc., can all be simulated with a typical simulator and the user can test, analyse various standard results apart from devising some novel protocol or strategy for routing etc. Network simulators are also widely used to simulate battlefield networks in Network-centric warfare. There are a wide variety of network simulators, ranging from the very simple to the very complex.

Minimally, a network simulator must enable a user to represent a network topology, specifying the nodes on the network, the links between those nodes and the traffic between the nodes. More complicated systems may allow the user to specify everything about the protocols used to handle traffic in a network. Graphical applications allow users to easily visualize the workings of their simulated environment. Text-based applications may provide a less intuitive interface, but may permit more advanced forms of customization.

Packet loss occurs when one or more packets of data travelling across a computer network fail to reach their destination. Packet loss is distinguished as one of the three main error types encountered in digital communications; the other two being bit error and spurious packets caused due to noise. Packets can be lost in a network because they may be dropped when a queue in the network node overflows. The amount of packet loss during the steady state is another important property of a congestion control scheme. The larger the value of packet loss, the more difficult it is for transport layer protocols to maintain high bandwidths, the sensitivity to loss of individual packets, as well as to frequency and patterns of loss among longer packet sequences is strongly dependent on the application itself.





## ***Throughput***

This is the main performance measure characteristic, and most widely used. In communication networks, such as Ethernet or packet radio, throughput or network throughput is the average rate of successful message delivery over a communication channel. The throughput is usually measured in bit per second (bit/s or bps), and sometimes in data packet per second or data packets per time slot. This measure how soon the receiver is able to get a certain amount of data sent by the sender. It is determined as the ratio of the total data received to the end to end delay. Throughput is an important factor which directly impacts the network performance ***Delay***

Delay is the time elapsed while a packet travels from one point e.g., source premise or network ingress to destination premise or network egress. The larger the value of delay, the more difficult it is for transport layer protocols to maintain high bandwidths. We will calculate end to end delay

## ***Queue Length***

A queuing system in networks can be described as packets arriving for service, waiting for service if it is not immediate, and if having waited for service, leaving the system after being served. Thus queue length is very important characteristic to determine that how well the active queue management of the congestion control algorithm has been working.



## EXPERIMENT-13

**Objective:** Perform a case study about the different routing algorithms to select the network path with its optimum and economical during data transfer. i. Link State routing ii. Flooding iii. Distance vector

### a) LINK STATE ROUTING

Routing is the process of selecting best paths in a network. In the past, the term routing was also used to mean forwarding network traffic among networks. However this latter function is much better described as simply forwarding. Routing is performed for many kinds of networks, including the telephone network (circuit switching), electronic data networks (such as the Internet), and transportation networks. This article is concerned primarily with routing in electronic data networks using packet switching technology. In packet switching networks, routing directs packet forwarding (the transit of logically addressed network packets from their source toward their ultimate destination) through intermediate nodes. Intermediate nodes are typically network hardware devices such as routers, bridges, gateways, firewalls, or switches. General-purpose computers can also forward packets and perform routing, though they are not specialized hardware and may suffer from limited performance. The routing process usually directs forwarding on the basis of routing tables which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time. Multipath routing techniques enable the use of multiple alternative paths. In case of overlapping/equal routes, the following elements are considered in order to decide which routes get installed into the routing table (sorted by priority): Prefix-Length: where longer subnet masks are preferred (independent of whether it is within a routing protocol or over different routing protocol)

1. Metric: where a lower metric/cost is preferred (only valid within one and the same routing protocol)
2. Administrative distance: where a lower distance is preferred (only valid between different routing protocols) Routing, in a more narrow sense of the term, is often contrasted with bridging in its assumption that network addresses are structured and that similar addresses imply proximity within the network. Structured addresses allow a single routing table entry to represent the route to a group of devices. In large networks, structured addressing (routing, in the narrow sense) outperforms unstructured addressing (bridging). Routing has become the dominant form of addressing on the Internet. Bridging is still widely used within localized environments.



## b) FLOODING

Flooding is a simple routing algorithm in which every incoming packet is sent through every outgoing link except the one it arrived on. Flooding is used in bridging and in systems such as Usenet and peer-to-peer file sharing and as part of some routing protocols, including OSPF, DVMRP, and those used in ad-hoc wireless networks. There are generally two types of flooding available, Uncontrolled Flooding and Controlled Flooding. Uncontrolled Flooding is the fatal law of flooding. All nodes have neighbors and route packets indefinitely. More than two neighbors create a broadcast storm. Controlled Flooding has its own two algorithms to make it reliable, SNCF (Sequence Number Controlled Flooding) and RPF (Reverse Path Flooding). In SNCF, the node attaches its own address and sequence number to the packet, since every node has a memory of addresses and sequence numbers. If it receives a packet in memory, it drops it immediately while in RPF, the node will only send the packet forward. If it is received from the next node, it sends it back to the sender.

### *Algorithm*

There are several variants of flooding algorithm. Most work roughly as follows:

1. Each node acts as both a transmitter and a receiver.
2. Each node tries to forward every message to every one of its neighbors except the source node. This results in every message eventually being delivered to all reachable parts of the network. Algorithms may need to be more complex than this, since, in some case, precautions have to be taken to avoid wasted duplicate deliveries and infinite loops, and to allow messages to eventually expire from the system. A variant of flooding called selective flooding partially addresses these issues by only sending packets to routers in the same direction. In selective flooding the routers don't send every incoming packet on every line but only on those lines which are going approximately in the right direction.

### *Advantages*

- Packet can be delivered, it will (probably multiple times).
- Since flooding naturally utilizes every path through the network, it will also use the shortest path.
- This algorithm is very simple to implement.

### *Disadvantages*

Flooding can be costly in terms of wasted bandwidth. While a message may only have one destination it has to be sent to every host. In the case of a ping flood or a denial of service attack,



it can be harmful to the reliability of a computer network. Messages can become duplicated in the network further increasing the load on the networks bandwidth as well as requiring an increase in processing complexity to disregard duplicate messages. Duplicate packets may circulate forever, unless certain precautions are taken: Use a hop count or a time to live count and include it with each packet. This value should take into account the number of nodes that a packet may have to pass through on the way to its destination.

c)

## ***DISTANCE VECTOR ROUTING PROTOCOL USING NS2***

In computer communication theory relating to packet-switched networks, a **distance vector routing protocol** is one of the two major classes of routing protocols, the other major class being the link- state protocol. Distance-vector routing protocols use the Bellman–Ford algorithm, Ford–Fulkerson algorithm, or DUAL FSM (in the case of Cisco Systems protocols) to calculate paths.

A distance-vector routing protocol requires that a router informs its neighbours of topology changes periodically. Compared to link-state protocols, which require a router to inform all the nodes in a network of topology changes, distance-vector routing protocols have less computational complexity and message overhead. The term distance vector refers to the fact that the protocol manipulates vectors (arrays) of distances to other nodes in the network. The vector distance algorithm was the original ARPANET routing algorithm and was also used in the internet under the name of RIP (Routing Information Protocol). Examples of distancevector routing protocols include RIPv1 and RIPv2 and IGRP.

### ***Method:***

Routers using distance-vector protocol do not have knowledge of the entire path to a destination. Instead they use two methods:

1. Direction in which router or exit interface a packet should be forwarded.
2. Distance from its destination

Distance-vector protocols are based on calculating the direction and distance to any link in a network. "Direction" usually means the next hop address and the exit interface. "Distance" is a measure of the cost to reach a certain node. The least cost route between any two nodes is the route with minimum distance. Each node maintains a vector (table) of minimum distance to every node. The cost of reaching a destination is calculated using various route metrics. RIP uses the hop count of the destination whereas IGRP takes into account other information such as node delay and available bandwidth. Updates are performed periodically in a distancevector protocol where all or part of a router's routing table is sent to all its neighbors that are configured to use the same distance-vector routing protocol. RIP supports cross-platform distance vector routing whereas IGRP is a Cisco Systems proprietary distance vector routing protocol. Once a router has this information it is able to amend its own routing table to reflect the changes and then



inform its neighbors of the changes. This process has been described as routing by rumor<sup>4</sup> because routers are relying on the information they receive from other routers and cannot determine if the information is actually valid and true. There are a number of features which can be used to help with instability and inaccurate routing information.

EGP and BGP are not pure distance-vector routing protocols because a distance-vector protocol calculates routes based only on link costs whereas in BGP, for example, the local route preference value takes priority over the link cost.

### ***Count-to-infinity problem***

The Bellman–Ford algorithm does not prevent routing loops from happening and suffers from the countto infinity problem. The core of the count-to-infinity problem is that if A tells B that it has a path somewhere, there is no way for B to know if the path has B as a part of it. To see the problem clearly, imagine a subnet connected like A–B–C–D–E–F, and let the metric between the routers be "number of jumps". Now suppose that A is taken offline. In the vectorupdate-process B notices that the route to A, which was distance 1, is down – B does not receive the vector update from A. The problem is, B also gets an update from C, and C is still not aware of the fact that A is down – so it tells B that A is only two jumps from C (C to B to A), which is false. This slowly propagates through the network until it reaches infinity (in which case the algorithm corrects itself, due to the relaxation property of Bellman–Ford).





## EXPERIMENT-14

**Objective:** Running and using services/commands like ping, traceroute, arp, telnet, etc

### Tracert / traceroute

**Tracert:** Determines the path taken to a destination by sending Internet Control Message Protocol (ICMP) Echo Request messages to the destination with incrementally increasing Time to Live (TTL) field values. The path displayed is the list of near-side router interfaces of the routers in the path between a source host and a destination. The near-side interface is the interface of the router that is closest to the sending host in the path. Used without parameters, tracert displays help. This diagnostic tool determines the path taken to a destination by sending ICMP Echo Request messages with varying Time to Live (TTL) values to the destination. Each router along the path is required to decrement the TTL in an IP packet by at least 1 before forwarding it.

Effectively, the TTL is a maximum link counter. When the TTL on a packet reaches 0, the router is expected to return an ICMP Time Exceeded message to the source computer. Tracert determines the path by sending the first Echo Request message with a TTL of 1 and incrementing the TTL by 1 on each subsequent transmission until the target responds or the maximum number of hops is reached. The maximum number of hops is 30 by default and can be specified using the -h parameter.

The path is determined by examining the ICMP Time Exceeded messages returned by intermediate routers and the Echo Reply message returned by the destination. However, some routers do not return Time Exceeded messages for packets with expired TTL values and are invisible to the tracert command. In this case, a row of asterisks (\*) is displayed for that hop.

### Examples:

To trace the path to the host named [www.google.co.in](http://www.google.co.in) use following command

To trace the path to the host named [www.google.com](http://www.google.com) and prevent the resolution of each IP address to its name, type: **tracert -d [www.google.com](http://www.google.com)**

To trace the path to the host named [www.google.com](http://www.google.com) and use the loose source route 10.12.0.1-10.29.3.1-10.1.44.1, type: **tracert -j 10.12.0.1 10.29.3.1 10.1.44.1 [www.google.com](http://www.google.com)**





```

C:\Users\LxsoftWin>tracert www.google.in

Tracing route to www.google.in [2404:6800:4002:804::2003]
over a maximum of 30 hops:

  1     1 ms    <1 ms    <1 ms    2405:205:1506:8af7::2a84:b8a0
  2     *      *        *        Request timed out.
  3    472 ms   1839 ms   *        2405:200:319:168::2
  4   1085 ms   829 ms   790 ms   2405:200:801:1600::91
  5    391 ms   1084 ms  1572 ms  2405:200:801:300::75
  6   2239 ms  1030 ms  1681 ms  2001:4860:1:1::1b6
  7     *      1022 ms  1179 ms  2001:4860:0:11de::1
  8   1009 ms  1253 ms  1623 ms  2001:4860:0:1::3d
  9   1170 ms   885 ms  1437 ms  del03s09-in-x03.1e100.net [2404:6800:4002:804::2003]

Trace complete.

C:\Users\LxsoftWin>_

```

## Syntax

tracert [-d] [-h MaximumHops] [-j HostList] [-w Timeout] [TargetName]

## Parameters

-d	Prevents tracert from attempting to resolve the IP addresses of intermediate routers to their names. This can speed up the display of tracert results.
-h	Maximum Hops Specifies the maximum number of hops in the path to search for the target (destination). The default is 30 hops.
-j	Host List Specifies that Echo Request messages use the Loose Source Route option in the IP header with the set of intermediate destinations specified in Host List. With loose source routing, successive intermediate destinations can be separated by one or multiple routers. The maximum number of addresses or names in the host list is 9. The Host List is a series of IP addresses (in dotted decimal notation) separated by spaces.
-w	Timeout Specifies the amount of time in milliseconds to wait for the ICMP Time Exceeded or Echo Reply message corresponding to a given Echo Request message to be received. If not received within the time-out, an asterisk (*) is displayed. The default time-out is 4000 (4 seconds).



## Ping

Verifies IP-level connectivity to another TCP/IP computer by sending Internet Control Message Protocol (ICMP) Echo Request messages. The receipt of corresponding Echo Reply messages are displayed, along with round-trip times. Ping is the primary TCP/IP command used to troubleshoot connectivity, reach ability, and name resolution.

```
Command Prompt

C:\Users\LxsoftWin>ping google.com

Pinging google.com [172.217.24.238] with 32 bytes of data:
Reply from 172.217.24.238: bytes=32 time=1451ms TTL=53
Reply from 172.217.24.238: bytes=32 time=599ms TTL=53
Reply from 172.217.24.238: bytes=32 time=1438ms TTL=53
Reply from 172.217.24.238: bytes=32 time=1656ms TTL=53

Ping statistics for 172.217.24.238:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 599ms, Maximum = 1656ms, Average = 1286ms

C:\Users\LxsoftWin>
```

You can use ping to test both the computer name and the IP address of the computer. If pinging the IP address is successful, but pinging the computer name is not, you might have a name resolution problem. In this case, ensure that the computer name you are specifying can be resolved through the local Hosts file, by using Domain Name System (DNS) queries, or through NetBIOS name resolution techniques.

### To test a TCP/IP configuration by using the ping command:

- To quickly obtain the TCP/IP configuration of a computer, open Command Prompt, and then type **ipconfig**. From the display of the ipconfig command, ensure that the network adapter for the TCP/IP configuration you are testing is not in a Media disconnected state.
- At the command prompt, ping the loopback address by typing **ping 127.0.0.1**
- Ping the IP address of the computer.
- Ping the IP address of the default gateway. If the ping command fails, verify that the default gateway IP address is correct and that the gateway (router) is operational.
- Ping the IP address of a remote host (a host that is on a different subnet). If the ping command fails, verify that the remote host IP address is correct, that the remote host is operational, and that all of the gateways (routers) between this computer and the remote host are operational.



- Ping the IP address of the DNS server. If the ping command fails, verify that the DNS server IP address is correct, that the DNS server is operational, and that all of the gateways (routers) between this computer and the DNS server are operational.

## ARP

Displays and modifies entries in the Address Resolution Protocol (ARP) cache, which contains one or more tables that are used to store IP addresses and their resolved Ethernet or Token Ring physical addresses. There is a separate table for each Ethernet or Token Ring network adapter installed on your computer.

### Syntax

Used without parameters	displays help
-a [InetAddr] [-N IfaceAddr]	Displays current ARP cache tables for all interfaces. To display the ARP cache entry for a specific IP address, use arp -a with the InetAddr parameter, where InetAddr is an IP address. To display the ARP cache table for a specific interface, use the -N IfaceAddr parameter where IfaceAddr is the IP address assigned to the interface. The -N parameter is case-sensitive.
-g [InetAddr] [-N IfaceAddr]	Identical to -a.
-d InetAddr [IfaceAddr]	Deletes an entry with a specific IP address, where InetAddr is the IP address. To delete an entry in a table for a specific interface, use the IfaceAddr parameter where IfaceAddr is the IP address assigned to the interface. To delete all entries, use the asterisk (*) wildcard character in place of InetAddr.
-s InetAddr EtherAddr [IfaceAddr]	Adds a static entry to the ARP cache that resolves the IP address a p [InetAddr to the physical address EtherAddr. To add a static ARP [IfaceAddr] entry to the table for a specific interface, use the IfaceAddr parameter [-s InetAddr EtherAddr [IfaceAddr]] where IfaceAddr is an IP address assigned to the interface.

### Parameters



## Examples:

To display the ARP cache tables for all interfaces use following command

arp -a

```
C:\Users\LxsoftWin>arp -a

Interface: 192.168.42.171 --- 0xd
  Internet Address      Physical Address      Type
  192.168.42.129        8e-df-54-4e-ac-fc    dynamic
  192.168.42.255        ff-ff-ff-ff-ff-ff    static
  224.0.0.22            01-00-5e-00-00-16    static
  224.0.0.252           01-00-5e-00-00-fc    static
  239.255.255.250       01-00-5e-7f-ff-fa    static
  255.255.255.255       ff-ff-ff-ff-ff-ff    static

Interface: 192.168.79.1 --- 0x14
  Internet Address      Physical Address      Type
  192.168.79.255        ff-ff-ff-ff-ff-ff    static
  224.0.0.22            01-00-5e-00-00-16    static
  224.0.0.252           01-00-5e-00-00-fc    static
  239.255.255.250       01-00-5e-7f-ff-fa    static

Interface: 192.168.23.1 --- 0x15
  Internet Address      Physical Address      Type
  192.168.23.255        ff-ff-ff-ff-ff-ff    static
  224.0.0.22            01-00-5e-00-00-16    static
  224.0.0.252           01-00-5e-00-00-fc    static
  239.255.255.250       01-00-5e-7f-ff-fa    static

C:\Users\LxsoftWin>_
```

To display the ARP cache table for the interface that is assigned the IP address 192.168.42.171

```
C:\Users\LxsoftWin>arp -a -N 192.168.42.171

Interface: 192.168.42.171 --- 0xd
  Internet Address      Physical Address      Type
  192.168.42.129        8e-df-54-4e-ac-fc    dynamic
  192.168.42.255        ff-ff-ff-ff-ff-ff    static
  224.0.0.22            01-00-5e-00-00-16    static
  224.0.0.252           01-00-5e-00-00-fc    static
  239.255.255.250       01-00-5e-7f-ff-fa    static
  255.255.255.255       ff-ff-ff-ff-ff-ff    static

C:\Users\LxsoftWin>_
```





## TELNET

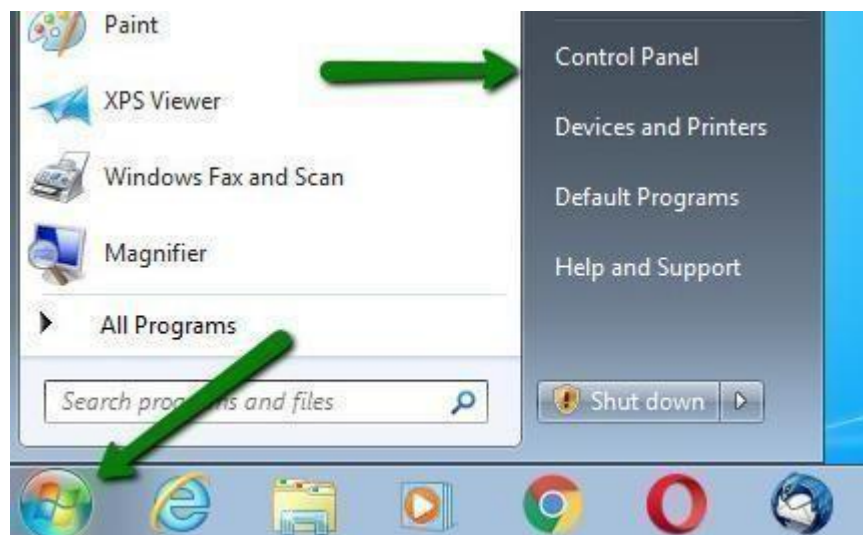
The telnet command is used for connection and communication with a remote or local host via the Telnet TCP/IP protocol.

You can enter a domain or IP address and try connecting to it via the chosen port. In case the port is not specified, telnet utility tries to connect via the default port 23.

The command is really useful in cases when you need to check whether the needed port is open on your computer and on the side of the remote host. **How to use Telnet For Windows**

Telnet is disabled on Windows by default. To enable it, perform these steps:

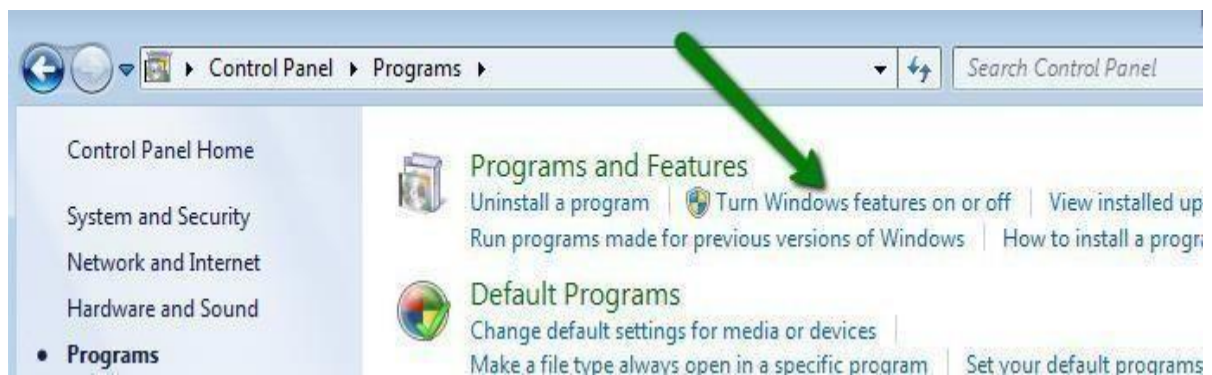
1. Press the **Start** button > **Control Panel**:



2. **Go to the Programs section:**

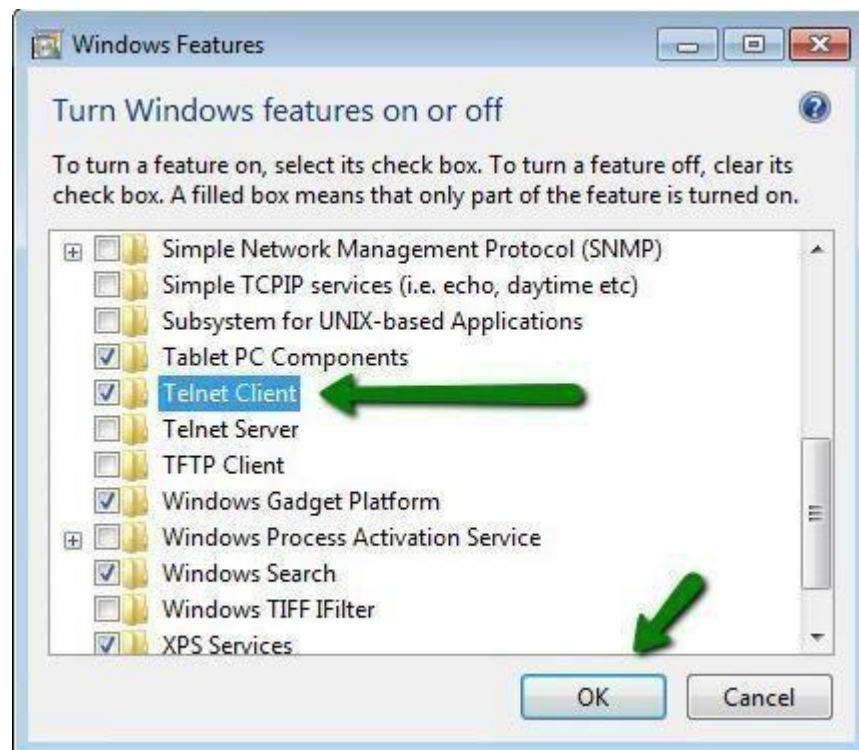


3. **Select Programs and Features > Turn Windows Features on or off:**



4. **Scroll down the list available in the Windows Features window > check Telnet Client option > press OK > wait a few moments for the changes to be applied**





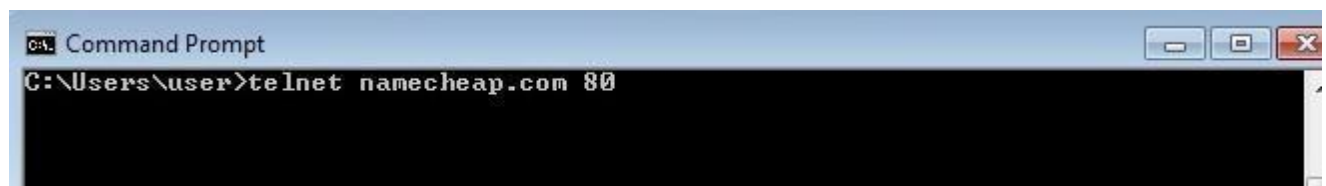
5. Telnet is enabled now, so we can run it in the same way as other commands:

- Select the **Start** button > click on the **Run** option.
- In the command line, type in **cmd** and press **Enter**.
- After that, type in the following and press **Enter**:

telnet [domain name or IP] [port number]

for example:

**telnet namecheap.com 80**



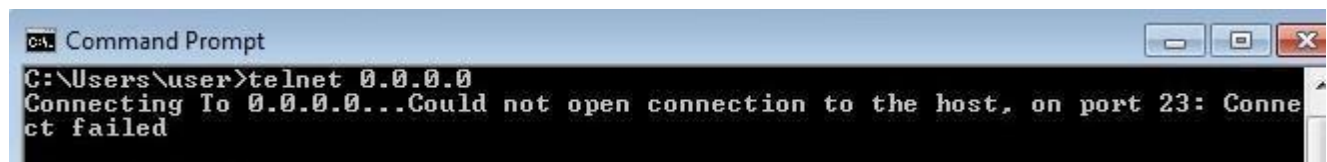


If you see the blank output after that, you have connected successfully. To quit you can press **CTRL + C** or any key:



As a result of successful telnet, we can conclude, that the entered domain or IP exists, and the chosen port is open on your computer and on the side of the target host.

If connection has not been established, the following error will appear:



Possible reasons for it are:

- domain or IP does not exist/not available/entered with a typo
- the chosen port is closed by a security software or via network configurations on your computer or on the side of the host you connect to
- connection/networking issue has occurred



## EXPERIMENT-15

**Objective:** Network packet analysis using tools like Wireshark, tcpdump, etc.

### Theory: tcpdump

The fundamental tool of almost all network traffic collection is tcpdump. It is an open-source application that comes installed on almost all Unix-like operating systems. Tcpdump is an excellent collection tool and comes complete with a very complex filtering language. It's essential to know how to filter the data at collection time to end up with a manageable chunk of data to analyze. Capturing all data from a network device on even a moderately busy network can create too much data to analyze efficiently.

In some rare cases, allowing tcpdump to output its capture directly to your screen may be enough to find what you're looking for. For example, in writing this article, captured some traffic and noticed that machine was sending traffic to an IP address. It turns out that machine was sending data to a Google IP address of 172.217.11.142.

It seems that even when Chrome is not running in the foreground it remains running as a service. it would not have necessarily noticed this without a packet analysis to tip me off. it re-captured some more tcpdump data but this time told tcpdump to write the data to a file that it opened in Wireshark (more on that later). Here's that entry:

No.	Time	Source	Destination	Protocol	Length	Info
49	1.17886551	192.168.2.146	172.217.11.142	TCP	74	48638 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=21637930 TSecr=0 WS=128
52	1.276719348	172.217.11.142	192.168.2.146	TCP	74	443 → 48638 [SYN, ACK] Seq=0 Ack=1 Win=42408 Len=0 MSS=1380 SACK_PERM=1 TSval=1726570383 TSecr=21637930 WS=256
53	1.276750884	192.168.2.146	172.217.11.142	TCP	66	48638 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=21637956 TSecr=1726570383
56	1.278370469	192.168.2.146	172.217.11.142	TLSv1.2	640	Client Hello
61	1.365636811	172.217.11.142	192.168.2.146	TCP	66	443 → 48638 [ACK] Seq=1 Ack=575 Win=43776 Len=0 TSval=1726570473 TSecr=21637956
62	1.365864393	172.217.11.142	192.168.2.146	TLSv1.2	222	Server Hello, Change Cipher Spec, Hello Request, Hello Request
63	1.365981311	192.168.2.146	172.217.11.142	TCP	66	48638 → 443 [ACK] Seq=575 Ack=157 Win=38336 Len=0 TSval=21637978 TSecr=1726570473
64	1.366352516	192.168.2.146	172.217.11.142	TLSv1.2	117	Change Cipher Spec, Hello Request, Hello Request
65	1.367913336	192.168.2.146	172.217.11.142	TLSv1.2	243	Application Data
66	1.367178953	192.168.2.146	172.217.11.142	TLSv1.2	785	Application Data
70	1.436799548	172.217.11.142	192.168.2.146	TCP	66	443 → 48638 [ACK] Seq=157 Ack=1522 Win=46336 Len=0 TSval=1726570555 TSecr=21637978
71	1.436831070	172.217.11.142	192.168.2.146	TLSv1.2	139	Application Data
72	1.436847607	172.217.11.142	192.168.2.146	TLSv1.2	184	Application Data
73	1.436994599	192.168.2.146	172.217.11.142	TLSv1.2	184	Application Data
78	1.465503263	172.217.11.142	192.168.2.146	TLSv1.2	431	Application Data
79	1.465808228	172.217.11.142	192.168.2.146	TLSv1.2	1484	Application Data
80	1.465822660	192.168.2.146	172.217.11.142	TCP	66	48638 → 443 [ACK] Seq=1560 Ack=2047 Win=34304 Len=0 TSval=21638003 TSecr=1726570589
81	1.465828891	172.217.11.142	192.168.2.146	TLSv1.2	1484	Application Data
82	1.466003849	172.217.11.142	192.168.2.146	TLSv1.2	1484	Application Data
83	1.466008560	192.168.2.146	172.217.11.142	TCP	66	48638 → 443 [ACK] Seq=1560 Ack=4883 Win=40064 Len=0 TSval=21638003 TSecr=1726570589
84	1.466015634	172.217.11.142	192.168.2.146	TLSv1.2	1484	Application Data
85	1.466336458	172.217.11.142	192.168.2.146	TLSv1.2	4329	Application Data, Application Data, Application Data
86	1.466359777	192.168.2.146	172.217.11.142	TCP	66	48638 → 443 [ACK] Seq=1560 Ack=10555 Win=51456 Len=0 TSval=21638003 TSecr=1726570589
87	1.466602992	172.217.11.142	192.168.2.146	TLSv1.2	1484	Application Data

Tcpdump is a favorite tool among sysadmins because it is a command-line tool. This means that it doesn't require a full-blown desktop to run. It is unusual for production servers to provide a desktop because of the resources that would take, so command-line tools are preferred. As with many advanced tools, tcpdump has a very rich and arcane language that takes some time to master. **Key Features:**

- Command line tool
- Packet capture capabilities
- Free to use



A few of the very basic commands involve selecting the network interface from which to collect data, and writing that data to a file so it can be exported for analysis elsewhere. The -i and -w switches are used for this.

This produces a capture file:

The standard TCP capture file is a pcap file. It is not text so it can only be read by an analysis program that knows how to read pcap files.

## Wireshark

Wireshark is probably the next best-known tool in any sysadmin's toolkit. It can not only capture data, but also provides some advanced analysis tools. Adding to its appeal, Wireshark is open source, and has been ported over to almost every server operating system that exists. Starting life named Ethereal,

Wireshark now runs everywhere, including as a standalone portable app. # tcpdump -i eth0 -w tcpdump\_packets

tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144

If you're analyzing traffic on a server with a desktop installed, Wireshark can do it all for you. The ^C51 packets captured collected packets can then be analyzed all in one spot. However, desktops are not common on servers, so in many cases, you'll want to capture the network data packets remotely and then pull the resulting pcap file into Wireshark.

At first launch, Wireshark allows you to either load an existing pcap file, or start capturing. If you elect file tcpdump\_packets

to capture network traffic, you can optionally specify filters to pare down the amount of data Wireshark captures. tcpdump\_packets: tcpdump capture file (little-endian) - version 2.4 (Ethernet, c collects. Since its analysis tools are so good, it's less important to ensure you surgically identify the data at collection time with Wireshark. If you don't specify a filter, Wireshark will simply collect all network data that your selected interface observes.



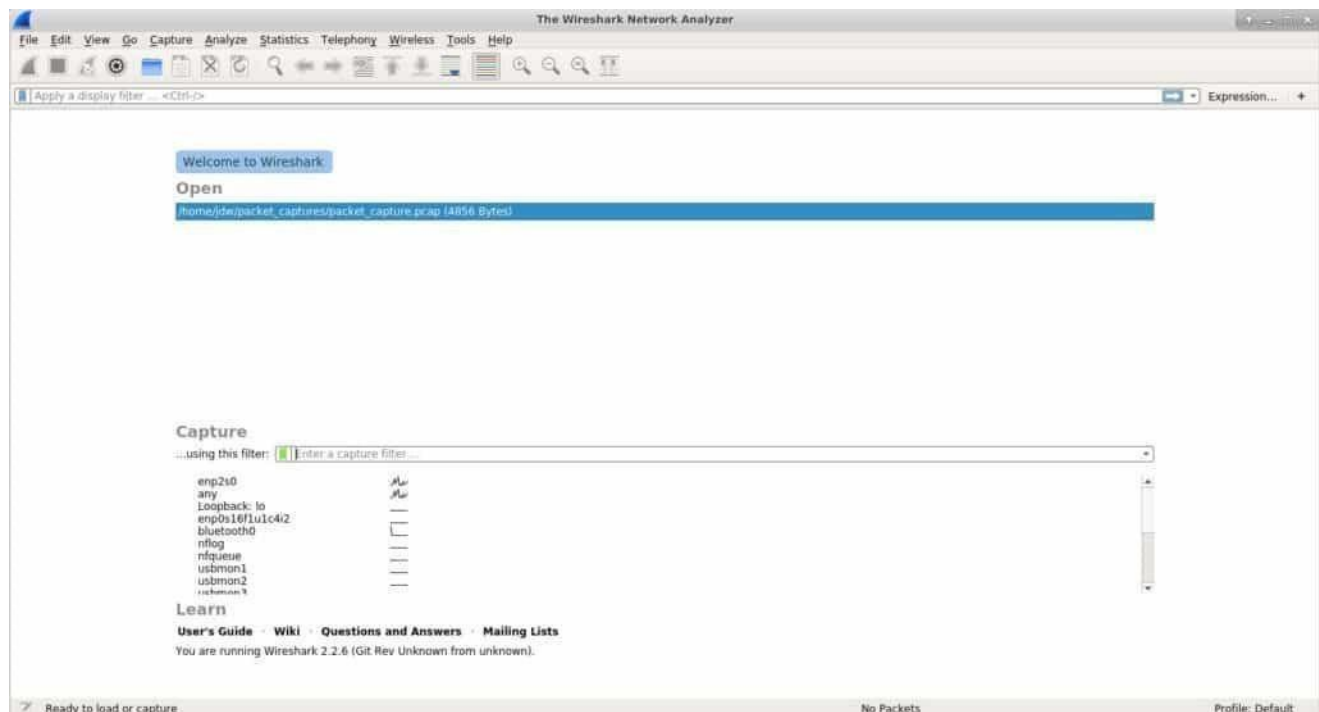


# RAJ KUMAR GOEL INSTITUTE OF TECHNOLOGY

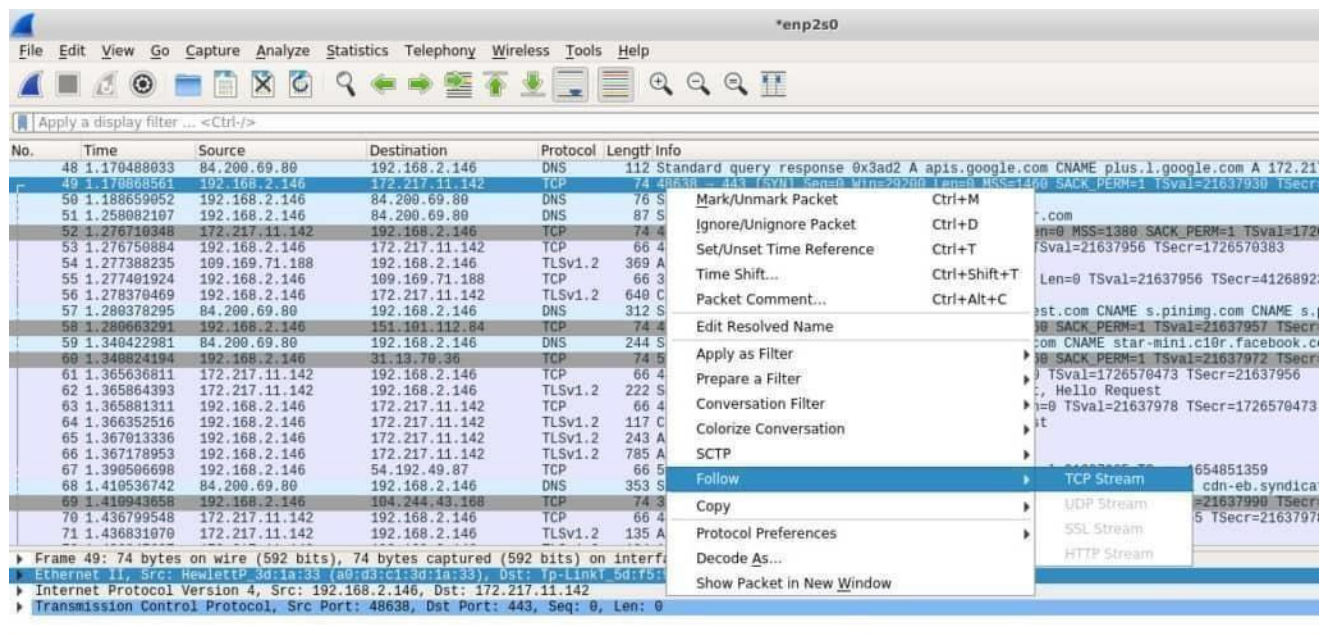
Approved by AICTE (Ministry of Education) & PCI (Ministry of Health & FW) GOI, Affiliated to Dr. APJ Abdul Kalam Technical University, Lucknow

AKTU College Code-033

Accredited by NAAC ('A' Grade), NBA Accredited Programs (B.Tech-ECE, IT) & B.Pharma



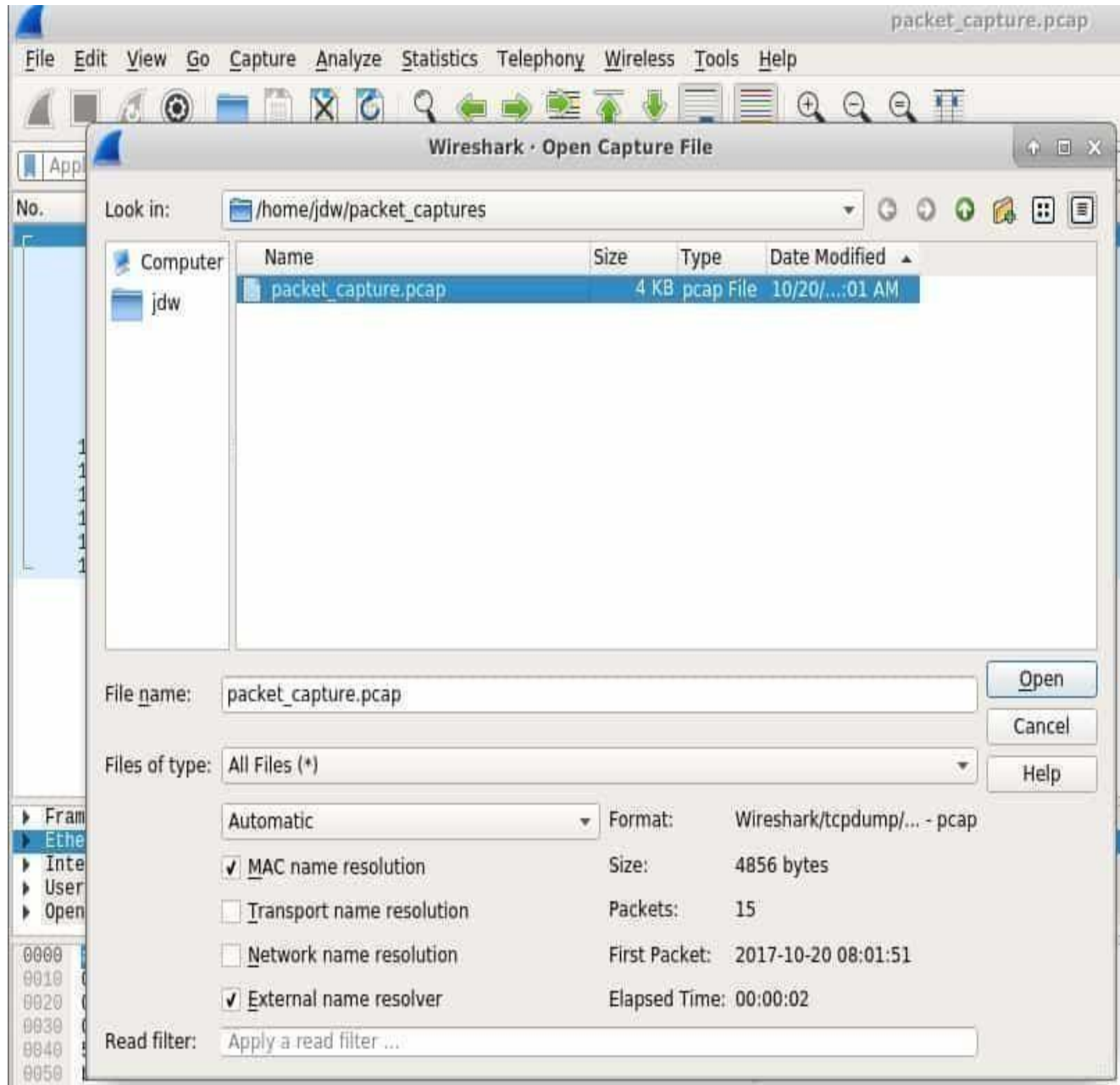
One of the most useful tools Wireshark provides is the ability to follow a *stream*. It's probably most useful to think of a stream as an entire conversation. In the screenshot below we can see a lot of data has been captured, but what is most interesting is that Google IP address. It can right-click it and *Follow* the TCP Stream to see the entire conversation.



If you've captured traffic elsewhere, you can import the pcap file using Wireshark's *File* -



> *Open* dialogue. The same filters and tools that can be used for natively captured network data are available for imported files.







## EXPERIMENT-16

**Objective:** Network simulation using tools like Cisco Packet Tracer, NetSim, OMNeT++, NS2, NS3, etc

### **Theory:**

Simulation is a very important technology in modern time. Computer assisted simulation can model hypothetical and real-life objects or activities on a computer to study the well-designed structure. A network simulator is a system of implementing the network on the computer through which the performance of the network is calculated. The computer assisted simulation technologies are applied in the simulation of networking algorithms. The functional network field is narrower than general simulation and it is natural that more specific requirements will be placed on network simulations.

Network simulator allows the researchers to test the scenarios that are difficult or expensive to simulate in real world. Design of various network topologies using nodes, hosts, hubs, bridges, routers and mobile units etc. is possible. The network simulators are of various types which can be compared on the basis of: range (simple to the complex), specification of nodes, links and traffic between the nodes. Specifying about the protocols used to handle traffic in a network, user friendly applications (allow users to easily visualize the simulated environment.), text-based applications (permit more advanced forms of customization) and programming-oriented tools (providing a programming framework that customizes to create an application that simulates the networking environment to be tested).

Network simulators are used by people from different areas such as academic researchers, industrialized sectors and Quality Assurance (QA) to design, simulate and analyze the performance of different network protocols. They can also be used to evaluate the outcome of the different parameters of the protocols being studied. Normally a network simulator comprises of wide range of networking technologies and protocols that help users to build complex networks from basic building blocks like clusters of nodes and links. With their help, different network topologies can be designed using various types of nodes such as endhosts, network bridges, routers, hubs, optical link-layer devices and mobile units.

### **Concepts in network simulation**

Generally, network simulators try to represent the real world networks and it is a useful technique, given that the activities of a network can be modeled by calculating the interaction between the different network components (they can be end-host or network entities such as routers, packets or physical links) using mathematical formulas. They can also be modeled by actually or virtually capturing and playing back experimental observations from real networks. Upon receipt of the observation data from simulation experiments, the behavior of the network and protocols supported are analyzed in a series of offline test experiments. All types of attributes can also be modified in a controlled manner to assess how the network can behave under different parameter combinations. Another feature of network simulation worth noticing is that the simulation program can be used and analyzed together with various strategy, links, applications etc. Typically, users can then adapt the simulator to fulfill their exact needs. Simulators support the most popular protocols and networks such as WLAN, TCP and WSN.



## Simulators

Most of the commercial simulators are Graphical User Interface (GUI) driven, while some network simulators are Command-Line Interface (CLI). The design of the network describes the state of the network (nodes, routers, switches and links) and the events (data transfer, transmission delay, packet error etc.). The major output of simulation is the trace files which log every packet and event that occurred during simulation and is used for analysis. Also provides other tools to facilitate visual analysis of trends and potential trouble spots. Most of the network simulators are discrete event, in which the list of pending "events" are stored and processed in order. Some events triggers the future events (i.e.) the event of the arrival of a packet at one node triggering the event of the arrival of that packet at a downstream node.

Simulation of networks is a very difficult task. For example, if blocking is high, then evaluation of the average occupancy is challenging because of high variance. To evaluate the probability of buffer overflow in a network, the time required for a precise answer can be enormously large. Techniques like "control variants" and "sampling" have been developed to speed simulation.

## List of Network Simulators

There are many both free/open-source and proprietary simulators. Examples of notable simulation software are ordered based on how frequently they are

NS2 (Network Simulator 2)

1. NS3 (Network Simulator 3)
2. OPNET.
3. OMNeT++.
4. NetSim.

## Uses of Network Simulators

Network simulators serve a variety of requirements. Simulators are relatively fast and economical when compared to the cost and time involved in setting up an entire bed containing multiple network computers, data links and routers. They authorize researchers to test scenarios that might be particularly difficult or expensive to emulate using a real hardware - for instance simulating a scenario with several nodes or experimenting with a new protocol in the network. Simulators are mainly useful in allowing researchers to test new networking protocols or changes to existing protocols in controlled and reproducible surroundings.

A typical simulator encompasses a wide range of networking technologies and can help the users to build complex networks from basic building blocks such as selection of nodes and links. Various types of nodes in Hierarchical networks resembling computers, hubs, bridges, routers, links, switches mobile units etc can be designed with the help of simulators.

Various types of Wide Area Network (WAN) technologies like TCP, ATM, IP etc. and Local Area Network (LAN) technologies like Ethernet, token rings etc., can be imitated with a simulator and the user can examine various standard results apart from devising some novel protocol or routing strategy. Network simulators are widely used to simulate battlefield networks in Network-centric warfare.



There are ample varieties of simulators, ranging from simple to complex. A simple simulator must enable a user to represent network topology, to specify nodes on the network, the links and the traffic between the nodes. More complex systems may permit the user to specify everything about the protocols used to handle traffic in a network. User friendly applications permit users to envision easily the working mechanism of their simulated situation. Text-based applications offer a less sensitive interface, but permits more advanced forms of customization.

## Overview of Network Simulators

Currently there are many network simulators that have different features in different aspects. Short lists of the current network simulators include NS-2, NS-3, OPNET, OMNeT++, NETSIM, QualNet, and JSim. These network simulators are selected for discussion regarding their features, advantages and restrictions.

### NS2

The Ns2 is a discrete event simulator targeted at packet level networking research and provides substantial support to simulate group of protocols like TCP, UDP, FTP and HTTP. It comprises of two simulation tools. Ns-2 is primarily UNIX based and fully simulates a layered wire or wireless network from the physical radio transmission channel to high-level applications. The simulator is written in C++ and a script language called OTcl.

**C++:** C++ is fast to run but slower to change, making it suitable for detailed protocol implementation.

**Otcl:** OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration. Ns provides glue to make objects and variables appear on both languages.

NS2 uses an OTcl interpreter by which the user writes an OTcl script that defines the network, (number of nodes and links) the transaction in the network (sources destinations, type of traffic) and the type of protocols used. The outcome of the simulation is a trace file that can be used for data processing (calculate delay, throughput etc). To visualize the simulation, a program called Network Animator (NAM) is used. It visualizes the packets as they propagate throughout the network. The ns-2 simulator has numerous features that make it suitable for our simulations.

- A network environment for ad-hoc networks,
- Wireless channel modules (e.g.802.11),
- Routing along multiple paths,
- Mobile hosts for wireless cellular networks.
- Download of ns-2 source code is possible and can be compiled for multiple platforms.

### **Advantages:**



1. NS2 has large number of available models, realistic mobility models, powerful and flexible scripting and simulation setup, large user community and ongoing development.
2. NS2 provides an easy traffic and movement pattern by including an efficient energy model.
3. It provides a set of randomized mobility model and there are several projects to bring advanced mobility models to the simulators.
4. Complex scenarios can be easily tested.
5. Popular for its modularity.

## Limitations:

1. NS2 needs to be recompilation every time if there is a change in the user code.
2. Real system is too complex to model i.e. complicated infrastructure.

## NS3

The *ns-3* simulator is a discrete-event network simulator for Internet systems, targeted primarily for research and learning purpose. The ns-3 project, started in 2006, is open-source free software, licensed under the GNU GPLv2 license. It will rely on the current contributions of the community to develop new models, debug or maintain the existing ones, and share the results. Ns3 is mainly used on LINUX systems and not limited to internet based systems alone.

**C++:** implementation of simulation and core model. Ns-3 is built as a library which may be statically or dynamically linked to a C++ main program. These libraries describe the beginning of simulation and their topology.

**Python:** C++ wrapped by Python. Python programs to import an “ns3” module. The features of NS3 simulator are given below.

1. Modular, documented core
2. C++ programs and Python scripting
3. Alignment with real systems
4. Software integrations
5. Virtualization and test bed integration
6. Attribute system
7. Updated models

## Advantages:

1. High modularity than its ancestor NS2.
2. Support simulation for TCP, UDP, ICMP, IPv4, multicast routing, P2P and CSMA protocols.
3. Support for ported code should make model validation easier and more credible.



4. Much more flexible than any other simulators.
5. Wide range of use in both optimization and expansion of the existing networks.

## **Limitations:**

1. NS3 still suffers from lack of credibility.
2. NS3 is intended to replicate the successful mode of NS2 in which various organizations contributed to the models and components based on the framework of NS2.
3. NS3 needs a lot of specialized maintainers in order to avail the merits of NS3 as the commercial OPNET network simulators.
4. Active maintainers are required to respond to the user questions, bug reports and help to Test & validate the system.

## **OMNET++**

It is a component-based, modular and open architecture discrete event simulator framework. The most common use of OMNeT++ is for simulation of networks, but it is also used for queuing network simulations and other areas as well. It is licensed under its own Academic Public License, which permits GNU Public License like freedom but only in noncommercial settings. It provides component architecture for models.

**C++:** The *C++ class library* comprises of simulation kernel and utility classes (for random number generation, statistics collection, topology discovery etc) -- this one is used to create simulation components (*simple modules* and channels); infrastructure to assemble simulations from these components and configure (*NED language, ini files*); runtime user interfaces or *environments* for simulations (*Tkenv, Cmdenv*); an Eclipse-based simulation IDE for designing, running and evaluating simulations; extension interfaces for real-time simulation, emulation, MRIP, parallel distributed simulation, database connectivity and so on.

## **The OMNeT++ components include:**

1. Simulation kernel library
2. Compiler for the NED topology description language (nedc)
3. Graphical network editor for NED files (GNED)
4. GUI for simulation execution, links into simulation executable (Tkenv)
5. Command-line user interface for simulation execution (Cmdenv)
6. Graphical output vector plotting tool (Plove)
7. Graphical output scalars visualization tool (Scalars)
8. Model documentation tool (opp\_neddoc)





9. Utilities (random number seed generation tool, make file creation tool, etc.)
10. Documentation, sample simulations, etc.

### **Advantages:**

- Provides a powerful GUI environment.
- Tracing and debugging are much easier than other simulators.
- Accurately models most hardware and include the modeling of physical phenomena.

### **Limitations:**

- It does not offer a great variety of protocols and very few protocols have been implemented, leaving users with significant background work.
- Poor analysis and management of typical performance. The mobility extension is relatively incomplete

## **NETSIM**

NetSim is a discrete event simulator developed by Tetcos in 1997, in association with Indian Institute of Science. It has also been featured with Computer Networks and Internets V edition by Dr. Douglas Comer, published by Prentice Hall. It has an object-oriented system simulating environment to support simulation and analysis of voice and data communication scenarios for High Frequency Global Communication Systems (HFGCS).

**Java:** It creating fast, platform independent software that could be used in simple, consumer electronic products. Java designed for simple, efficient, platform-independent program for creating WWW-based programs. Using Java one can create small programs called applets that are entrenched into an HTML document and viewable on any Java-compatible browser. Java applets are compiled into a set of byte-codes, or machine-independent processing instructions.

### **Features:**

- NetSim modeling and simulation are supported for Aloha, Slotted Aloha, Token Ring/Bus, Ethernet CSMA/CD, Fast Ethernet, WLAN - IEEE 802.11 a/b/g/n and e, X.25, Frame Relay, TCP, UDP, IPv4 and IPv6, Routing - RIP, OSPF, BGP, MPLS, MANET, GSM, CDMA, Wireless Sensor Network, Zigbee, Cognitive radio)[5].
- It simulates a wide variety of Cisco routers, including 2500 series, 2600 series, 2800 series, and 3600 series, as well as the Cisco Catalyst 1900 series, 2900 series, and 3500 series switches. Protocol libraries are available as open C code for user modification. This can help to avoid the time consuming process such as encoding, customization and configuring commercial simulators to meet customer specific needs. Along with the Boson Virtual Packet Technology engine NetSim utilizes Boson's proprietary Router & Simulator EROUTER software technologies, to produce individual packets. These packets are routed and switched through the simulated network, allowing the simulator to build an appropriate virtual routing table and





simulate proper networking. Other simulation products on the market do not support this level of functionality.

- It can be used to create a simulation of the topology of corporate network and help practice trouble-shooting without using devices on the production network.

## **Advantages:**

1. NetSim has a GUI which features drag and drop functionality for devices, links etc. i.e. 2.

Modeling in NetSim is simple and user friendly.

3. It has a built in analysis framework that provides intra and inter-protocol performance comparison with graphical options.
4. Data packet and control packet flow can be visual-ized through NetSim built-in packet animator.
5. It is easy to learn all about NetSim. **Limitations:**

1. NetSim is a single process discrete event simulator. A single event queue is used for the simulation which at any given time contains one entry for each station on the network.
2. Free version of NetSim is not available.

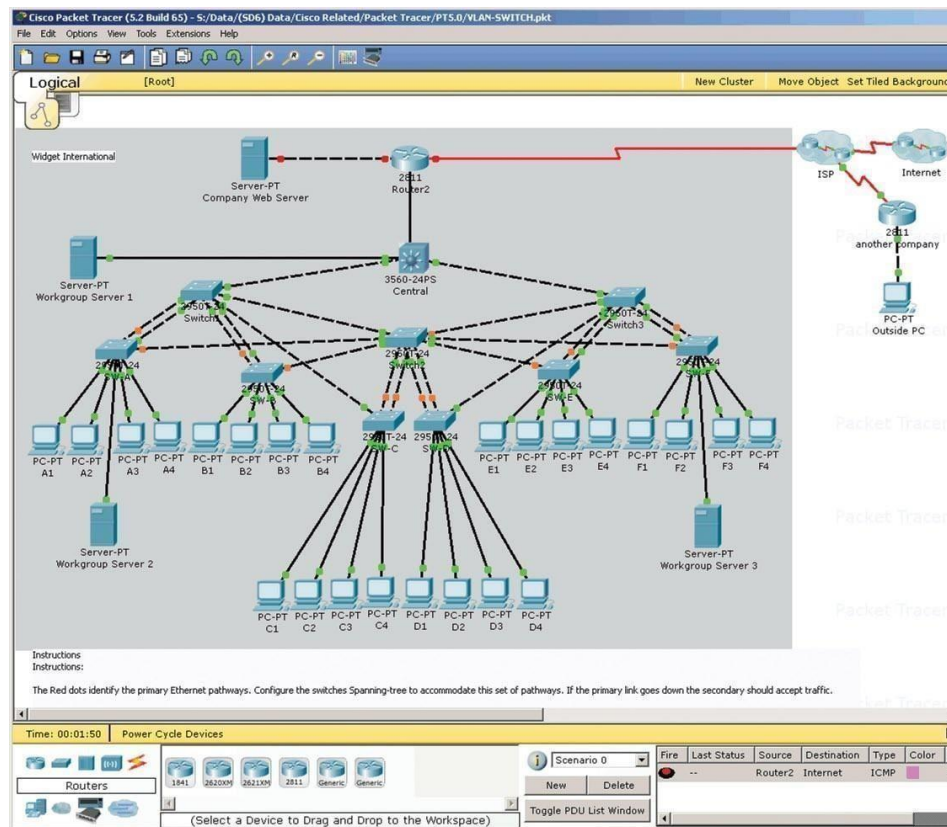
## **Cisco Packet Tracer**

Cisco Packet Tracer is a powerful network simulation program that allows students to experiment with network behavior and ask “what if” questions. As an integral part of the Networking Academy comprehensive learning experience, Packet Tracer provides simulation, visualization, authoring, assessment, and collaboration capabilities to facilitate the teaching and learning of complex technology concepts.

Packet Tracer supplements physical equipment in the classroom by allowing students to create a network with an almost unlimited number of devices, encouraging practice, discovery, and troubleshooting. The simulation-based learning environment helps students develop 21st century skills such as decision making, creative and critical thinking, and problem solving. Packet Tracer complements the Networking Academy curricula, allowing instructors to easily teach and demonstrate complex technical concepts and networking systems design. Instructors can customize individual or multiuser activities, providing hands-on lessons for students that offer value and relevance in their classrooms. Students can build, configure, and troubleshoot networks using virtual equipment and simulated connections, alone or in collaboration with other students. Packet Tracer offers an effective, interactive environment for learning networking concepts and protocols. Most importantly, Packet



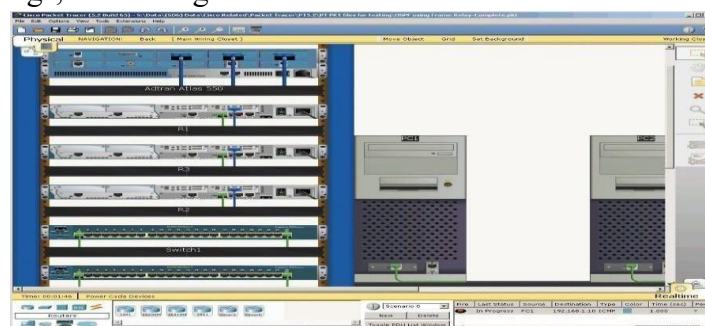
Tracer helps students and instructors create their own virtual “network worlds” for exploration, experimentation, and explanation of networking concepts and technologies.



Packet Tracer’s drag-and-drop interface allows students to configure and validate system architecture

## Key Features Packet Tracer Workspaces:

Cisco Packet Tracer has two workspaces—logical and physical. The logical workspace allows users to build logical network topologies by placing, connecting, and clustering virtual network devices. The physical workspace provides a graphical physical dimension of the logical network, giving a sense of scale and placement in how network devices such as routers, switches, and hosts would look in a real environment. The physical view also provides geographic representations of networks, including multiple cities, buildings, and wiring closets.





## Packet Tracer Modes:

Cisco Packet Tracer provides two operating modes to visualize the behavior of a network—real-time mode and simulation mode. In real-time mode the network behaves as real devices do, with immediate real-time response for all network activities. The real-time mode gives students a viable alternative to real equipment and allows them to gain configuration practice before working with real equipment.

In simulation mode the user can see and control time intervals, the inner workings of data transfer, and the propagation of data across a network. This helps students understand the fundamental concepts behind network operations.

Layer	Cisco Packet Tracer Supported Protocols
Application	<ul style="list-style-type: none"><li>• FTP , SMTP, POP3, HTTP, TFTP, Telnet, SSH, DNS, DHCP, NTP, SNMP, AAA, ISR VOIP, SCCP config and calls ISR command support, Call Manager Express</li></ul>
Transport	<ul style="list-style-type: none"><li>• TCP and UDP, TCP Nagle Algorithm &amp; IP Fragmentation, RTP</li></ul>
Network	<ul style="list-style-type: none"><li>• BGP, IPv4, ICMP, ARP, IPv6, ICMPv6, IPsec, RIPv1/ v2/ng, Multi-Area OSPF, EIGRP, Static Routing, Route Redistribution, Multilayer Switching, L3 QoS, NAT, CBAL, , Zone-based policy firewall and Intrusion Protection, System on the ISR, GRE VPN, IPsec VPN</li></ul>
Network Access/ Interface	<ul style="list-style-type: none"><li>• Ethernet (802.3), 802.11, HDLC, Frame Relay, PPP, PPPoE, STP, RSTP, VTP, DTP, CDP, 802.1q, PAgP, L2 QoS, SLARP, Simple WEP, WPA, EAP</li></ul>