VIDYALANKAR INSTITUTE OF TECHNOLOGY

Department of Information Technology

Lab Manual

Subject: **Computer Networks and Network Design**

SEM-IV

Prof. Vinita Bhandiwad and Prof Yash Shah

2020-21

Vision

To be recognized as a center of excellence in the field of Information Technology where learners are nurtured in a scholarly environment to evolve into competent professionals to benefit society.

Mission

- Evolve a curriculum which emphasizes on strong engineering fundamentals with the flexibility to choose advanced courses of interest and gain exposure to tools and techniques in Information Technology.
- Encourage a teaching-learning process in which highly competent faculty share a symbiotic association with the institutes of repute.
- Facilitate creation and dissemination of knowledge through a digitally enabled learning environment.
- Develop academic and infrastructural facilities with modern equipment and other learning resources and encourage reciprocal sharing with other institutes through networking.
- Establish a center of excellence to enhance academia – industry partnership and work on collaborative projects.

Program Educational Objectives:

PEO1	To prepare students for successful careers in industry that meet the needs of IT companies.
PEO2	To develop the ability among students to analyze technical concepts to software or product design
PEO3	To provide an environment for students to work on multidisciplinary projects as part of different teams to enhance their team building capabilities like leadership, motivation, teamwork etc
PEO4	To provide students with a sound foundation in the mathematical, scientific and engineering fundamentals necessary to formulate, solve and analyze engineering problems and to prepare them for graduate studies.
PEO5	To inculcate professional ethics and codes of professional practice.
PEO6	To provide students experience with the multifaceted aspects of using computers to solve problems and develop professional application software for some industry/society need by applying s/w engineering steps during this development.

Program Outcomes:

PO1	Graduates will demonstrate knowledge of applied mathematics, applied physics, applied chemistry, engineering drawing and mechanics, basic workshop practices, basic electronics and fundamentals of Computer and IT engineering.
PO2	Graduates will demonstrate an ability to identify, formulate and solve IT engineering problems.
PO3	Graduate will demonstrate an ability to design electronic circuits and conduct experiments, analyze and interpret data.
PO4	Graduates will demonstrate an ability to design software, system, component as per specification.
PO5	Graduates will demonstrate an ability to visualize and work on laboratory and multi-disciplinary tasks.
PO6	Graduate will demonstrate skills to use modern engineering tools, softwares and equipment to analyze, simulate, model and solve problems in OO Programming Languages using Object Oriented Concepts.

PO7	Graduate will demonstrate their skills in developing and managing internet based, client – server, different database(s), networking applications/projects and latest computing & communication technologies.
PO8	Graduate will be able to communicate effectively in both verbal and written form and demonstrate knowledge of professional and ethical responsibilities
PO9	Graduate will be broadly educated and have an understanding of the impact of engineering solutions on society and demonstrate awareness of contemporary issues.
PO10	Graduate will develop entrepreneurial approach and ability for life-long learning.

Vidyalankar Institute of Technology
Accredited A+ by NAAC

Lab Manual

Subject	Computer Networks and Network Design Lab
Semester	IV
Academic Year	2020-21
Software Requirements	Java/ Python/Wireshark/Cisco Packet Ttracer
Hardware Requirements	PC with Internet Connection
Theory Faculty In-charge	Prof. Vinita Bhandiwad/ Prof Yash Shah
Practical Faculty In-charge	Prof. Vinita Bhandiwad/ Prof Yash Shah
Laboratory	Lab 07C/D MSTEAMS
Lab Assistant	-----
Revised On	15th March 2021
Prepared By	Prof. Vinita Bhandiwad/ Prof Yash Shah
Sign	
Endorsed By HOD	

Lab Outcome

Subject: Network Lab	
LO1	To get familiar with the basic network administration commands
LO 2	To get familiar with the basic network administration command
LO 3	To understand the network simulator environment and visualize a network topology and observe its performance
LO 4	To implement client-server socket programs
LO5	To observe and study the traffic flow and the contents of protocol frames.
LO6	To design and configure a network for an organization

LIST OF EXPERIMENT

Sr. No.	Title of Experiments
1	Study and Perform various networking commands
2	Design a LAN Network and show the data communication between 4 nodes use Class C Addressing
3	Design an internetworking LAN Network and show the data communication between 6 nodes use Class B Addressing for configuring the end devices. Clearly show the router configuration steps
4	Design a college/Hospital Management network using cisco packet tracer
5	Demonstrate the TCP/IP protocol suite for the current packet running in the network using wireshark. Also demonstrate how to retrieve password using Wireshark
6	Implement Checksum error detection algorithm using Java/Python
7	Implement cyclic redundancy check algorithm using Java/Python
8	Implement Odd Parity check algorithm using Java/Python
9	Implement Even Parity check algorithm using Java/Python
10	Implement Routing algorithm using Dijkstra's method
11	Execute socket programming using Java/Python
12	Consider an IP address 192.168.20.1, identify its Network address, default subnet address, default broadcast address, number of host supported in the network ***(IP address can change)***
13	Consider an IP address 192.168.20.1, divide the network into two subnets, identify its Network address, default subnet address, default broadcast address, number of host supported in the network ***(IP address can change)***
14	Design various topologies of network using cisco packet tracer

Semester	IV
Subject	Computer Networks and Networking Design

Experiment Number	1	
Experiment Title	Understanding Basic networking Commands.	
Resources / Apparatus Required	Hardware: Compatible Computer System	Windows
Objectives	To understand basic networking commands.	

Theory	**1. Ipconfig/ifconfig:** • Stands for Internet protocol configuration. • It is a fast way of determining your computer's IP address and other information, such as the address of its default gateway. • To use the command, just type ipconfig at the Command Prompt. **2. Ping:** • The ping command is one of the most often used networking utilities for detecting devices on a network and for troubleshooting network problems. • When you ping a device, you send that device a short message, which it then sends back (the echo). • The general format is ping hostname or ping IP address. • ping www.google.com or ping 216.58.208.68 • If you receive a reply then the device is working OK, if you don’t then: ❖ The device is faulty, disconnected, switched off, incorrectly configured. ❖ Your network or the device you are working on is not working properly. **3. Hostname:** • The hostname is what a device is called on a network.

- In computer networking, a hostname (archaically node name) is a label that is assigned to a device connected to a computer network and that is used to identify the device in various forms of electronic communication.
- To use the command, just type hostname at the Command Prompt.

4. Trace route command:

- The traceroute command is used to determine the path between two connections. Often a connection to another device will have to go through multiple routers. The traceroute command will return the names or IP addresses of all the routers between two devices.
- 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.
- To use the command, just type tracert example.com at the Command Prompt.

5. ARP Command:

- The ARP commands to view, display, or modify the details/information in an ARP table/cache.
- Typically, a host uses ARP to determine the hardware address of another host.
- ARP stands for "Address Resolution Protocol" is a protocol for mapping an IP address to a physical MAC address on a local area network.
- When a computer wants to communicate with another computer on a different network, the IP address would be used. The IP address is like your mailing address while MAC address is like your name.
- To use this command, type
- arp first then arp -a (for displaying complete ARP cache) also we can also find the ARP cache entry for a specific IP address by specifying the IP address with arp command: arp -a 224.0.0.251

6. Netstat command:

- It comes from the word network statistics
- The Netstat command displays a variety of statistics about a computer's active TCP/IP connections.
- When dealing with excessive traffic and malicious software it's advantageous to be informed about the inbound and outbound connections to your computer

- Netstat is a cross-platform command, which means it's also available in other operating systems like macOS and Linux.
- To use the command type netstat on command prompt to see all active connection.
- Find out how to read only established/ LISTEN, CLOSE_WAIT, TIME_WAIT connection ports

7. Nslookup Command:

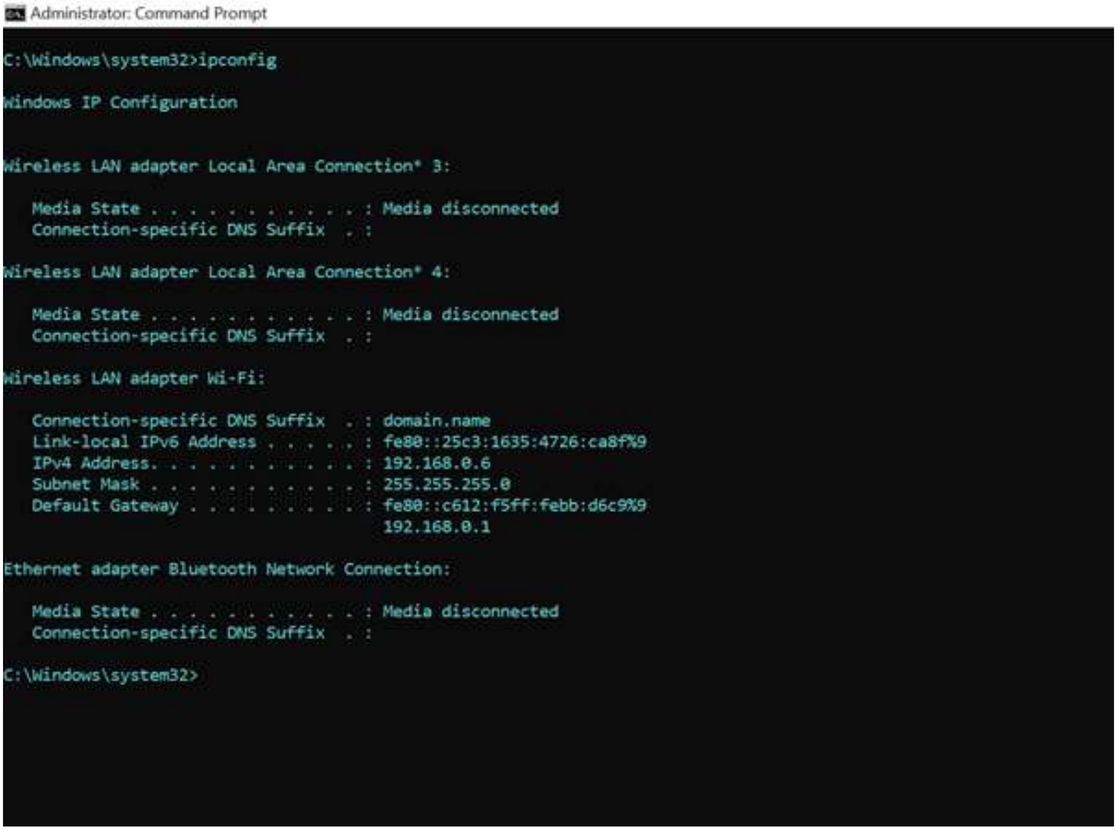
- The name nslookup stands for “name server look up.”
- It is used for querying the Domain Name System (DNS) to obtain domain name or IP address mapping information.
- The main use of nslookup is for troubleshooting DNS related problems.
- For example, if you can get to www.ebay.com by typing 66.135.192.87 in your browser's address bar but not by typing www.ebay.com, you have a DNS problem. The simplest use of nslookup is to look up the IP address for a given DNS name. For example, how did I know that 66.135.192.87 was the IP address for ebay.com? I used nslookup to find out .
- To use the command type nslookup google.com on command prompt to see all active connection.

8. Route command:

- In IP networks, routing tables are used to direct packets from one subnet to another.
- The Route command provides the device’s routing tables.
- To get this result, just type route print.

Output

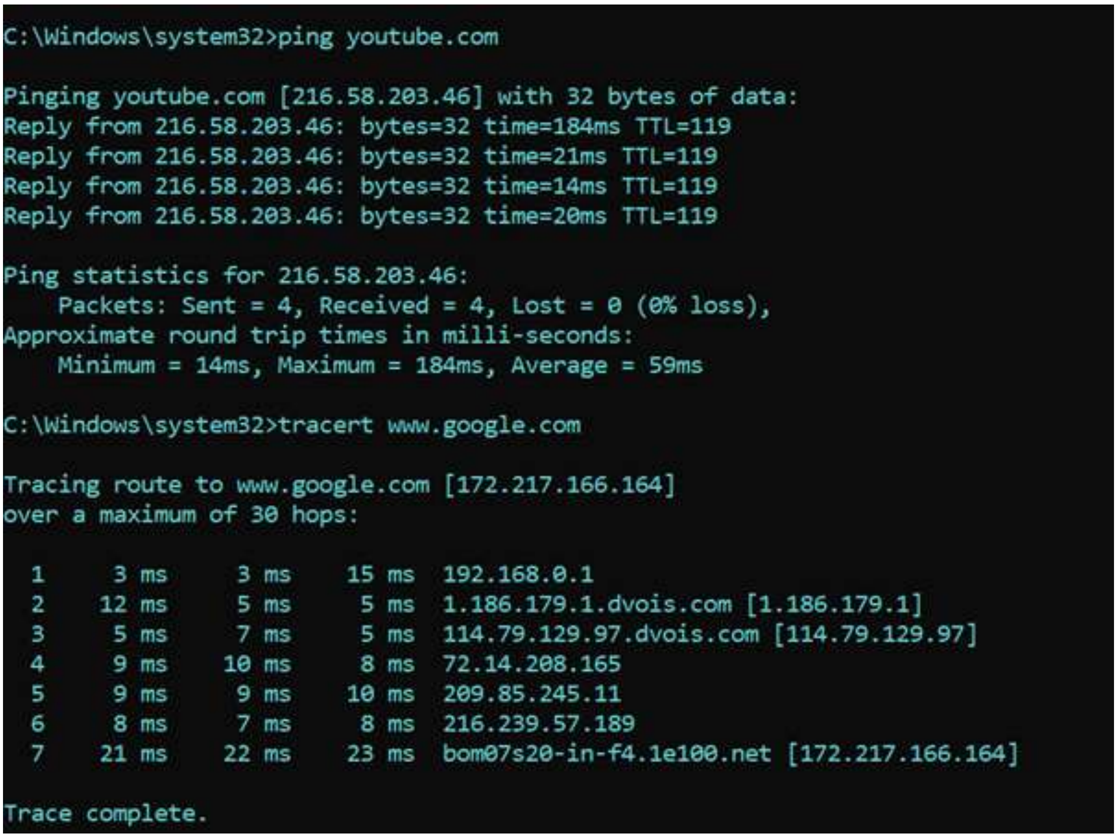
1.



Administrator: Command Prompt

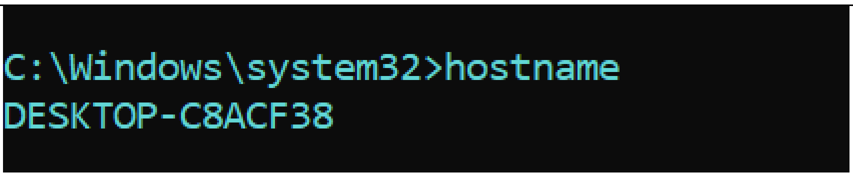
```
C:\Windows\system32>ipconfig

Windows IP Configuration


Wireless LAN adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 4:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . : domain.name
   Link-local IPv6 Address . . . . . : fe80::25c3:1635:4726:ca8f%9
   IPv4 Address. . . . . . . . . . . : 192.168.0.6
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::c612:f5ff:febb:d6c9%9
                                       192.168.0.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Windows\system32>
```

2.



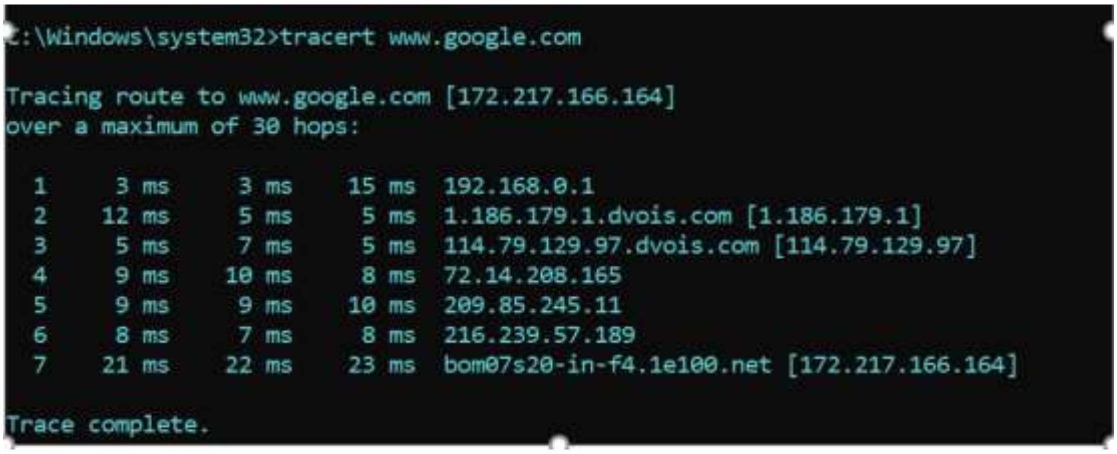
```
C:\Windows\system32>ping youtube.com

Pinging youtube.com [216.58.203.46] with 32 bytes of data:
Reply from 216.58.203.46: bytes=32 time=184ms TTL=119
Reply from 216.58.203.46: bytes=32 time=21ms TTL=119
Reply from 216.58.203.46: bytes=32 time=14ms TTL=119
Reply from 216.58.203.46: bytes=32 time=20ms TTL=119

Ping statistics for 216.58.203.46:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 14ms, Maximum = 184ms, Average = 59ms

C:\Windows\system32>tracert www.google.com

Tracing route to www.google.com [172.217.166.164]
over a maximum of 30 hops:

  1     3 ms     3 ms    15 ms  192.168.0.1
  2    12 ms     5 ms     5 ms  1.186.179.1.dvois.com [1.186.179.1]
  3     5 ms     7 ms     5 ms  114.79.129.97.dvois.com [114.79.129.97]
  4     9 ms    10 ms     8 ms  72.14.208.165
  5     9 ms     9 ms    10 ms  209.85.245.11
  6     8 ms     7 ms     8 ms  216.239.57.189
  7    21 ms    22 ms    23 ms  bom07s20-in-f4.1e100.net [172.217.166.164]

Trace complete.
```

3.



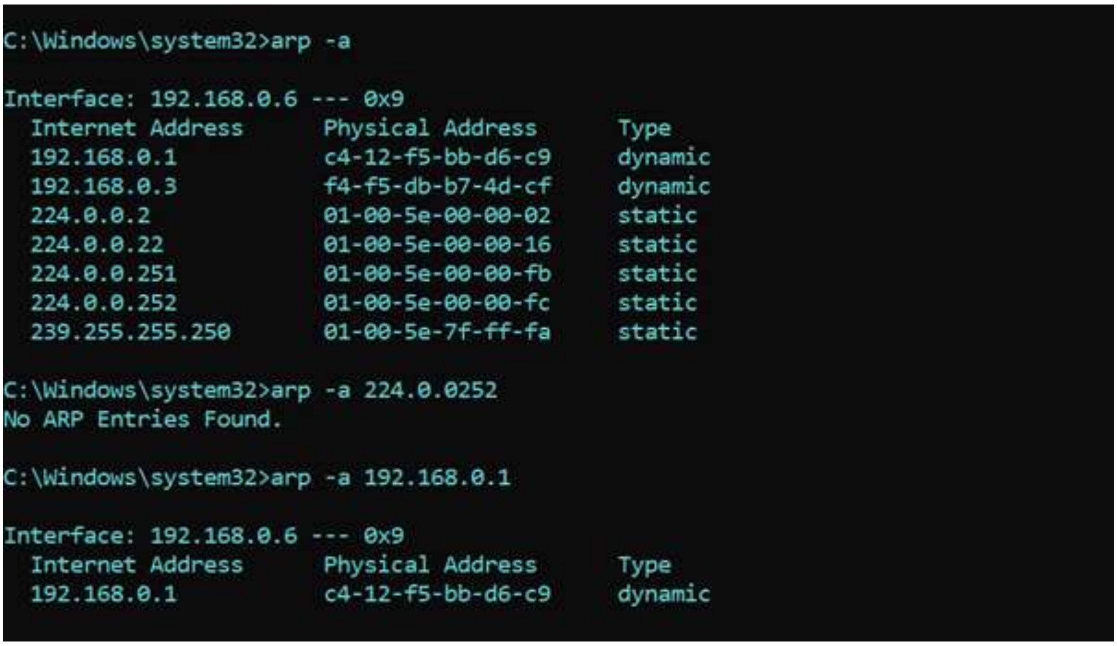
```
C:\Windows\system32>hostname
DESKTOP-C8ACF38
```

4.



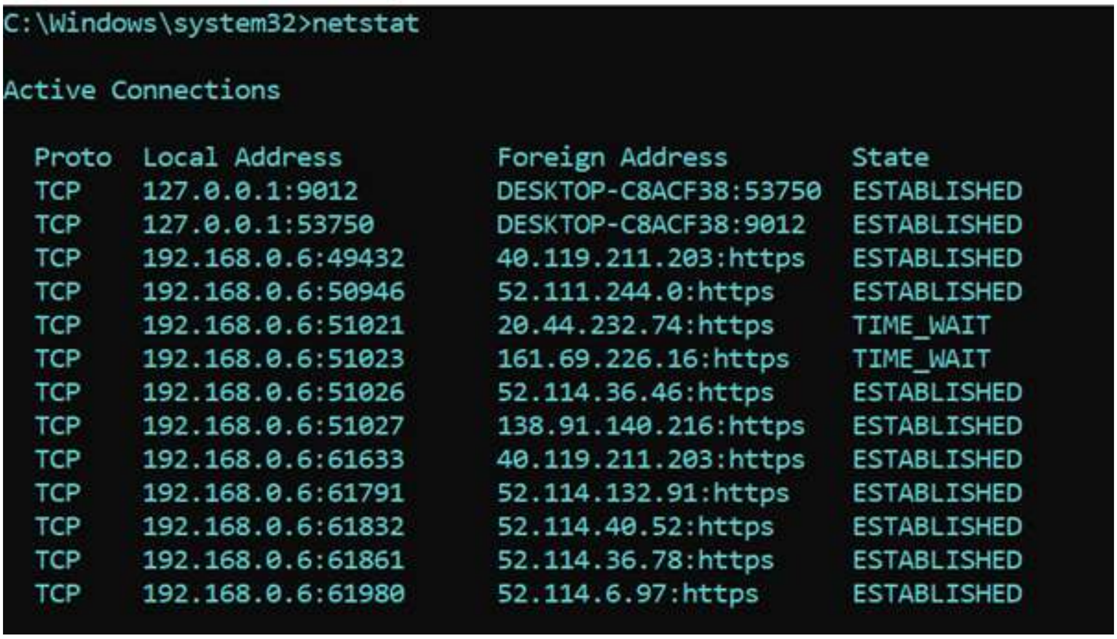
```
C:\Windows\system32>tracert www.google.com

Tracing route to www.google.com [172.217.166.164]
over a maximum of 30 hops:

  1     3 ms     3 ms    15 ms  192.168.0.1
  2    12 ms     5 ms     5 ms  1.186.179.1.dvois.com [1.186.179.1]
  3     5 ms     7 ms     5 ms  114.79.129.97.dvois.com [114.79.129.97]
  4     9 ms    10 ms     8 ms  72.14.208.165
  5     9 ms     9 ms    10 ms  209.85.245.11
  6     8 ms     7 ms     8 ms  216.239.57.189
  7    21 ms    22 ms    23 ms  bom07s20-in-f4.1e100.net [172.217.166.164]

Trace complete.
```

5.



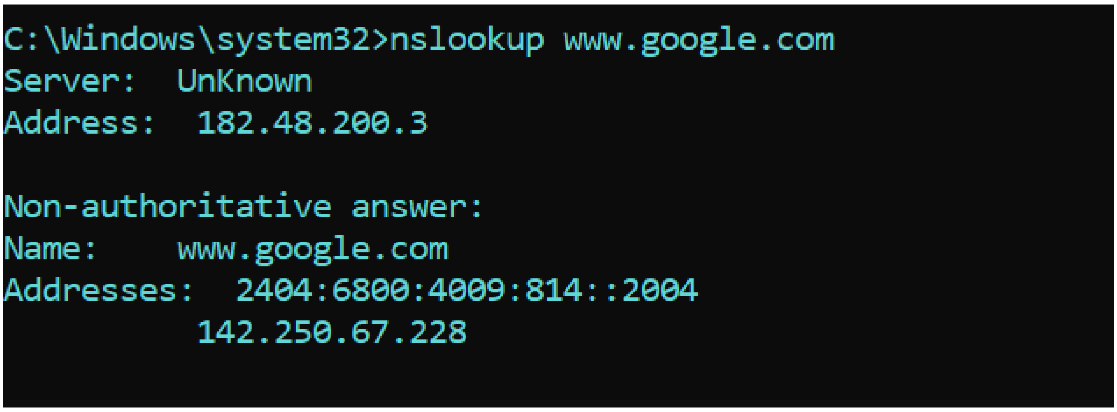
```
C:\Windows\system32>arp -a

Interface: 192.168.0.6 --- 0x9
  Internet Address      Physical Address      Type
  192.168.0.1           c4-12-f5-bb-d6-c9     dynamic
  192.168.0.3           f4-f5-db-b7-4d-cf     dynamic
  224.0.0.2             01-00-5e-00-00-02     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static

C:\Windows\system32>arp -a 224.0.0252
No ARP Entries Found.

C:\Windows\system32>arp -a 192.168.0.1

Interface: 192.168.0.6 --- 0x9
  Internet Address      Physical Address      Type
  192.168.0.1           c4-12-f5-bb-d6-c9     dynamic
```

6.



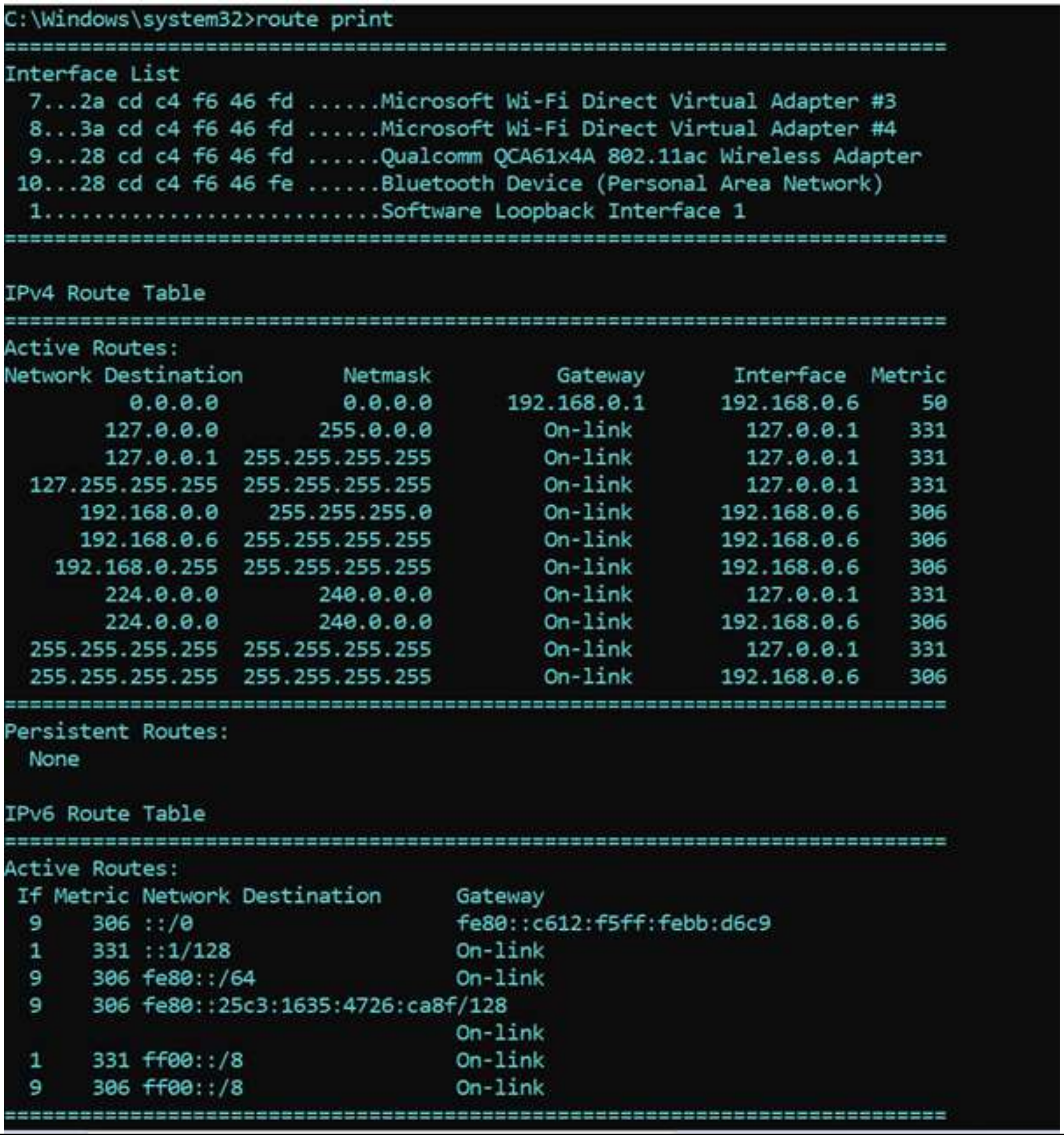
```
C:\Windows\system32>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:9012         DESKTOP-C8ACF38:53750  ESTABLISHED
  TCP    127.0.0.1:53750        DESKTOP-C8ACF38:9012   ESTABLISHED
  TCP    192.168.0.6:49432      40.119.211.203:https   ESTABLISHED
  TCP    192.168.0.6:50946      52.111.244.0:https     ESTABLISHED
  TCP    192.168.0.6:51021      20.44.232.74:https     TIME_WAIT
  TCP    192.168.0.6:51023      161.69.226.16:https    TIME_WAIT
  TCP    192.168.0.6:51026      52.114.36.46:https     ESTABLISHED
  TCP    192.168.0.6:51027      138.91.140.216:https   ESTABLISHED
  TCP    192.168.0.6:61633      40.119.211.203:https   ESTABLISHED
  TCP    192.168.0.6:61791      52.114.132.91:https    ESTABLISHED
  TCP    192.168.0.6:61832      52.114.40.52:https     ESTABLISHED
  TCP    192.168.0.6:61861      52.114.36.78:https     ESTABLISHED
  TCP    192.168.0.6:61980      52.114.6.97:https      ESTABLISHED
```

7.



```
C:\Windows\system32>nslookup www.google.com
Server:  UnKnown
Address:  182.48.200.3

Non-authoritative answer:
Name:    www.google.com
Addresses:  2404:6800:4009:814::2004
          142.250.67.228
```

8.



```
C:\Windows\system32>route print
===========================================================================
Interface List
  7...2a cd c4 f6 46 fd ......Microsoft Wi-Fi Direct Virtual Adapter #3
  8...3a cd c4 f6 46 fd ......Microsoft Wi-Fi Direct Virtual Adapter #4
  9...28 cd c4 f6 46 fd ......Qualcomm QCA61x4A 802.11ac Wireless Adapter
 10...28 cd c4 f6 46 fe ......Bluetooth Device (Personal Area Network)
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      192.168.0.1      192.168.0.6     50
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
      192.168.0.0    255.255.255.0         On-link       192.168.0.6    306
      192.168.0.6  255.255.255.255         On-link       192.168.0.6    306
    192.168.0.255  255.255.255.255         On-link       192.168.0.6    306
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link       192.168.0.6    306
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link       192.168.0.6    306
===========================================================================
Persistent Routes:
  None

IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
  9    306 ::/0                     fe80::c612:f5ff:febb:d6c9
  1    331 ::1/128                  On-link
  9    306 fe80::/64                On-link
  9    306 fe80::25c3:1635:4726:ca8f/128
                                    On-link
  1    331 ff00::/8                 On-link
  9    306 ff00::/8                 On-link
===========================================================================
```

Conclusion	Successfully learnt and implemented basic networking command

Semester	IV
Subject	Computer Networks and Networking Design

Experiment Number	2	
Experiment Title	Understanding basic topologies like:- 1. Bus 2. Ring 3. Star 4. Mesh 5. Hybrid	
Objective	To learn and perform the basic topologies used for building an efficient computer network.	
Resources / Apparatus Required	Hardware: Laptop	Software: Cisco Packet Tracer
Theory	**1.Bus Topology:-** In bus topology there is a main cable and all the devices are connected to this main cable through drop lines. There is a device called tap that connects the drop line to the main cable. Since all the data is transmitted over the main cable, there is a limit of drop lines and the distance a main cable can have.	

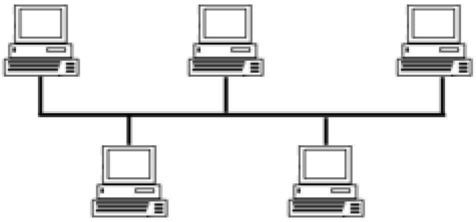
Advantages of bus topology

1. Easy installation, each cable needs to be connected with backbone cable.
2. Less cables required than Mesh and star topology

Disadvantages of bus topology

1. Difficultly in fault detection.
2. Not scalable as there is a limit of how many nodes you can connect with backbone cable

Bus Topology



http://www.computerhope.com

2.Ring Topology:-

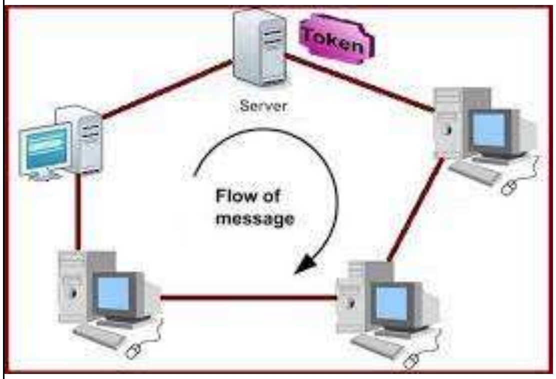
In ring topology each device is connected with the two devices on either side of it. There are two dedicated point to point links a device has with the devices on the either side of it. This structure forms a ring thus it is known as ring topology. If a device wants to send data to another device then it sends the data in one direction, each device in ring topology has a repeater, if the received data is intended for other device then repeater forwards this data until the intended device receives it.

Advantages of Ring Topology

1. Easy to install.
2. Managing is easier as to add or remove a device from the topology only two links are required to be changed.

Disadvantages of Ring Topology

1. A link failure can fail the entire network as the signal will not travel forward due to failure.
2. Data traffic issues, since all the data is circulating in a ring.



3. Star Topology:-

In star topology each device in the network is connected to a central device called hub. Unlike Mesh topology, star topology doesn’t allow direct communication between devices, a device must have to communicate through hub. If one device wants to send data to other device, it has to first send the data to hub and then the hub transmit that data to the designated device.

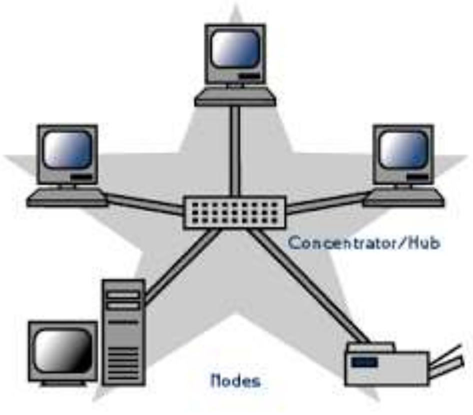
Advantages of Star topology

1. Less expensive because each device only need one I/O port and needs to be connected with hub with one link.
2. Easier to install
3. Less amount of cables required because each device needs to be connected with the hub only.
4. Robust, if one link fails, other links will work just fine.
5. Easy fault detection because the link can be easily identified.

Disadvantages of Star topology

1. If hub goes down everything goes down, none of the devices can work without hub.

2. Hub requires more resources and regular maintenance because it is the central system of star topology.



4.Mesh Topology:-

In mesh topology each device is connected to every other device on the network through a dedicated point-to-point link. When we say dedicated it means that the link only carries data for the two connected devices only. Lets say we have n devices in the network then each device must be connected with (n-1) devices of the network. Number of links in a mesh topology of n devices would be n(n-1)/2.

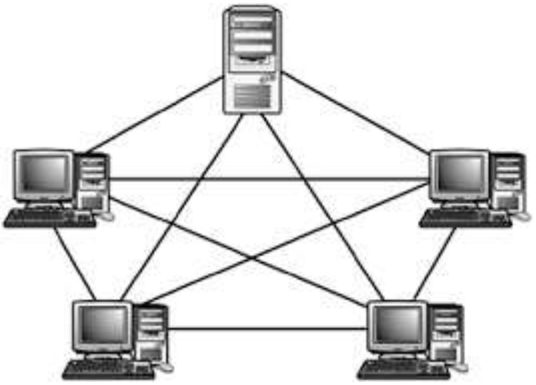
Advantages of Mesh topology

1. No data traffic issues as there is a dedicated link between two devices which means the link is only available for those two devices.
2. Mesh topology is reliable and robust as failure of one link doesn’t affect other links and the communication between other devices on the network.
3. Mesh topology is secure because there is a point to point link thus unauthorized access is not possible.
4. Fault detection is easy.

Disadvantages of Mesh topology

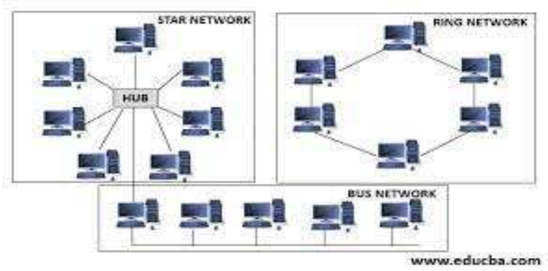
1. Amount of wires required to connected each system is tedious and headache.

2. Since each device needs to be connected with other devices, number of I/O ports required must be huge.
3. Scalability issues because a device cannot be connected with large number of devices with a dedicated point to point link.



5.Hybrid Topology:-

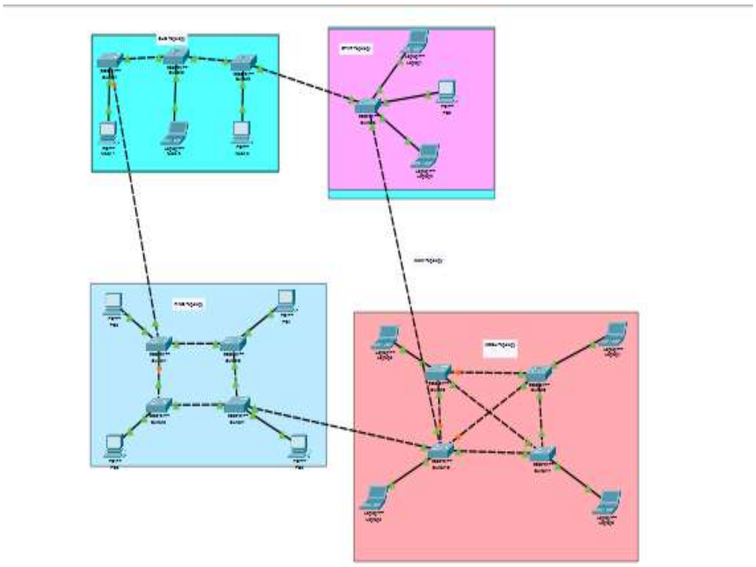
A combination of two or more topology is known as hybrid topology. For example a combination of star and mesh topology is known as hybrid topology.



Advantages of Hybrid topology

1. We can choose the topology based on the requirement for example, scalability is our concern then we can use star topology instead of bus technology.
2. Scalable as we can further connect other computer networks with the existing networks with different topologies.

Disadvantages of Hybrid topology

	1. Fault detection is difficult. 2. Installation is difficult. 3. Design is complex so maintenance is high thus expensive. Output:- 

Semester	IV
Subject	Computer Networks and Networking Design

Experiment Number	3	
Experiment Title	Design Interconnecting Network using Router in CPT.	
Objective	To learn and perform the interconnection between routers, switches and nodes among two buildings	
Resources / Apparatus Required	Hardware: Desktop	Software: Cisco packet Tracer
Theory	**1.Router** → A router is a device like a switch that routes data packets based on their IP addresses. → Router is mainly a Network Layer device. Routers normally connect LANs and WANs together and have a dynamically updating routing table based on which they make decisions on routing the data packets. → Router divide broadcast domains of hosts connected through it. 	

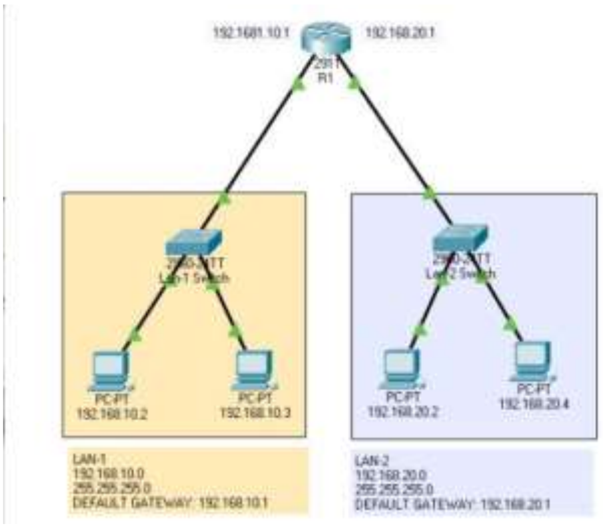
2.Switch

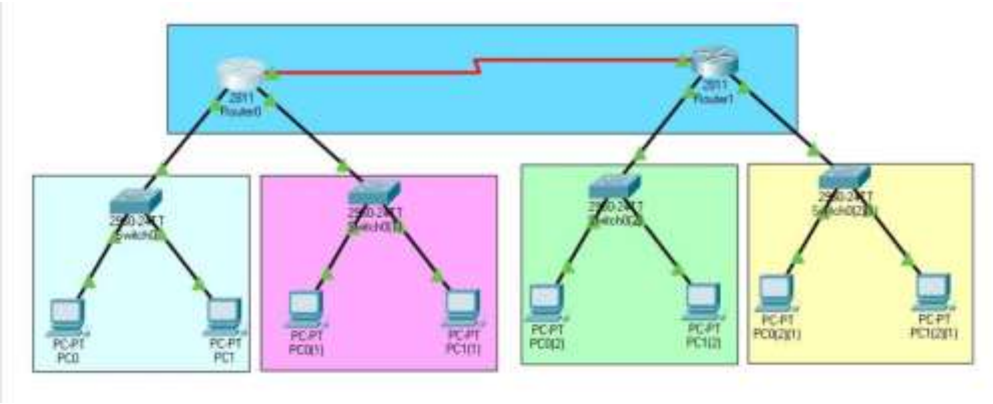
→ A switch is a multiport bridge with a buffer and a design that can boost its efficiency(a large number of ports imply less traffic) and performance.

→ A switch is a data link layer device. The switch can perform error checking before forwarding data, that makes it very efficient as it does not forward packets that have errors and forward good packets selectively to correct port only.

→ In other words, switch divides collision domain of hosts, but broadcast domain remains same.

Output



	 2811 Router0 2811 Router1 2960-24TT Switch0 2960-24TT Switch0(1) 2960-24TT Switch0(2) 2960-24TT Switch0(2)(1) PC-PT PC0 PC-PT PC1 PC-PT PC0(1) PC-PT PC1(1) PC-PT PC0(2) PC-PT PC1(2) PC-PT PC0(2)(1) PC-PT PC1(2)(1)

Semester	IV
Subject	Computer Networks and Networking Design

Experiment	4	
Problem Statement	Implement checksum algorithm using Java/Python programming language	
Resources / Apparatus Required	Hardware:Computer System	Software: Eclipse,CMD

Code

```
// Java code for Checksum_Sender

package checksum_sender;


import java.io.*;

import java.net.*;

import java.util.*;


public class Checksum_Sender

{

        // Setting maximum data length

        private int MAX = 100;


        // initialize socket and I/O streams

        private Socket socket = null;

        private ServerSocket servsock = null;
```

```
	private DataInputStream dis = null;
	private DataOutputStream dos = null;

	public Checksum_Sender(int port) throws IOException
	{
		servsock = new ServerSocket(port);

		// Used to block until a client connects to the server
		socket = servsock.accept();

		dis = new DataInputStream(socket.getInputStream());
		dos = new DataOutputStream(socket.getOutputStream());

		while (true)
		{
			int i, l, sum = 0, nob;
			Scanner sc = new Scanner(System.in);
			System.out.println("Enter data length");
			l = sc.nextInt();

			// Array to hold the data being entered
			int data[] = new int[MAX];
```

```
        // Array to hold the complement of each data
        int c_data[] = new int[MAX];

        System.out.println("Enter data to send");

        for (i = 0; i < l; i++)
        {
            data[i] = sc.nextInt();

            // Complementing the entered data
            // Here we find the number of bits required to represent
            // the data, like say 8 requires 1000, i.e 4 bits
            nob = (int)(Math.floor(Math.log(data[i]) / Math.log(2))) + 1;

            // Here we do a XOR of the data with the number 2^n -1,
            // where n is the nob calculated in previous step
            c_data[i] = ((1 << nob) - 1) ^ data[i];

            // Adding the complemented data and storing in sum
            sum += c_data[i];
```

```
                        }


                        // The sum(i.e checksum) is also sent along
with the data
                        data[i] = sum;
                        l += 1;


                        System.out.println("Checksum Calculated is : "
+ sum);
                        System.out.println("Data being sent along with
Checkum.....");


                        // Sends the data length to receiver
                        dos.writeInt(l);


                        // Sends the data one by one to receiver
                        for (int j = 0; j < l; j++)
                                dos.writeInt(data[j]);


                        // Displaying appropriate message depending
on feedback received
                        if (dis.readUTF().equals("success"))
                        {
                                System.out.println("Thanks for the
feedback!! Message received
                                                                Successfully!");
```

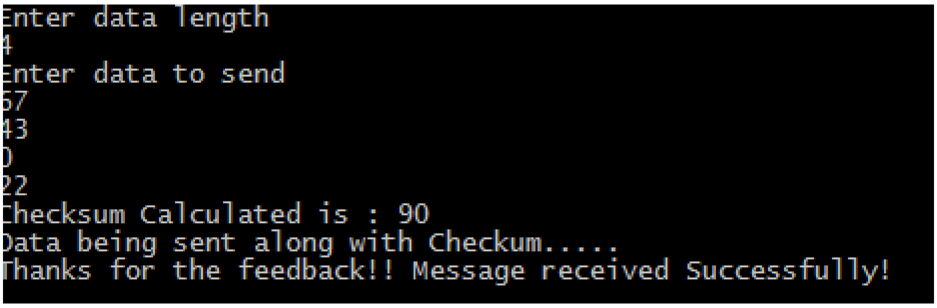
```
                break;
            }

            else if (dis.readUTF().equals("failure"))
            {
                System.out.println("Message was not received successfully!");
                break;
            }
        }

        // Closing all connections
        dis.close();
        dos.close();
        socket.close();
    }

    // Driver Method
    public static void main(String args[]) throws IOException
    {
        Checksum_Sender cs = new Checksum_Sender(45678);
    }
}
```

```
// Java code for Checksum_Receiver
package checksum_sender;

import java.net.*;
import java.io.*;
import java.util.*;

public class Checksum_Receiver {

        // Initialize socket and I/O streams
        private Socket s = null;
        private DataInputStream dis = null;
        private DataOutputStream dos = null;

        // Constructor to put ip address and port
        public Checksum_Receiver(InetAddress ip,int port)throws IOException
        {

                // Opens a socket for connection
                s = new Socket(ip,port);

                dis = new DataInputStream(s.getInputStream());
                dos = new DataOutputStream(s.getOutputStream());
        while (true)
        { Scanner sc = new Scanner(System.in);
            int i, l, nob, sum = 0, chk_sum;

            // Reads the data length sent by sender
            l = dis.readInt();

            // Initializes the arrays based on data length received
            int c_data[] = new int[l];
            int data[] = new int[l];

            System.out.println("Data received (alond with checksum) is");

            for(i = 0; i< data.length; i++)
            {
                // Reading the data being sent one by one
                data[i] = dis.readInt();
                System.out.println(data[i]);

                // Complementing the data being received
                nob = (int)(Math.floor(Math.log(data[i]) / Math.log(2))) + 1;
```

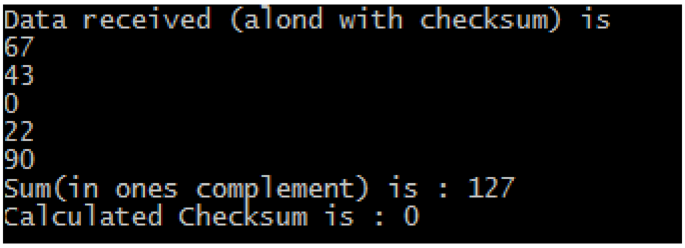
```
                        c_data[i] = ((1 << nob) - 1) ^ data[i];


                        // Adding the complemented data
                        sum += c_data[i];
                }
                System.out.println("Sum(in ones complement)
is : "+sum);


                // Complementing the sum
                nob = (int)(Math.floor(Math.log(sum) /
Math.log(2))) + 1;
                sum = ((1 << nob) - 1) ^ sum;
                System.out.println("Calculated Checksum is :
"+sum);


                // Checking whether final result is 0 or
something else
                // and sending feedback accordingly
                if(sum == 0)
                {
                        dos.writeUTF("success");
                        break;
                }
                else
                {
```

```
                                dos.writeUTF("failure");
                                break;
                        }
                }

                // Closing all connections
                dis.close();
                dos.close();
                s.close();
        }

        // Driver Method
        public static void main(String args[])throws IOException
        {
                // Getting ip address on which the receiver is running
                // Here, it is "localhost"
                InetAddress ip = InetAddress.getLocalHost();
                Checksum_Receiver cr = new Checksum_Receiver(ip,5000);
        }
}
```

Output



```
Enter data length
4
Enter data to send
67
43
0
22
Checksum Calculated is : 90
Data being sent along with Checkum.....
Thanks for the feedback!! Message received Successfully!
```

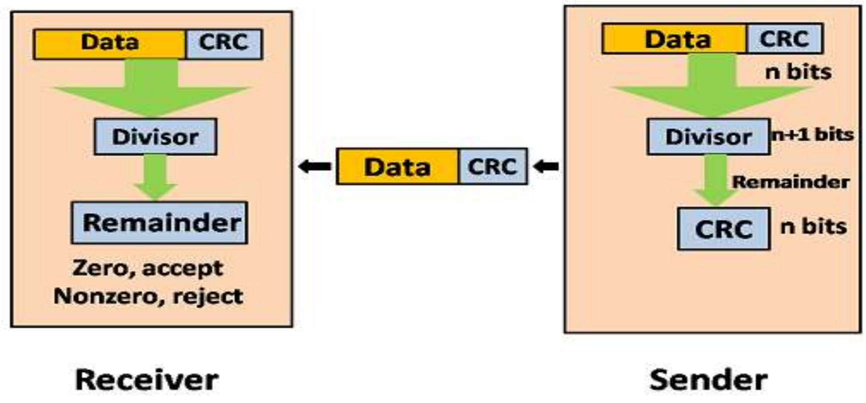


```
Data received (alond with checksum) is
67
43
0
22
90
Sum(in ones complement) is : 127
Calculated Checksum is : 0
```

Conclusion

Semester	IV
Subject	Computer Networks and Networking Design

Experiment Number	5	
Experiment Title	Implement CRC Error detection algorithm using Java/Python programming language	
Objective	To learn and implement CRC error detection in python.	
Resources / Apparatus Required	Hardware: Laptop with 8GB Ram	Software: Python 3.9/Pycharm
Theory	**CRC :** → CRC or Cyclic Redundancy Check is a method of detecting accidental changes/errors in the communication channel. → CRC stands for Cyclic Redundancy Check. → CRC uses Generator Polynomial which is available on both sender and receiver side. → **Sender side :** The binary data is first augmented by adding k-1 zeros in the end of the data. Use modulo-2 binary division to divide binary data by the key and store remainder of division. Append the remainder at the end of the data to form the encoded data and send the same → **Receiver side :** Perform modulo-2 division again and if the remainder is 0, then there are no errors. In this we will focus only on finding the remainder i.e. check word and the code word.	

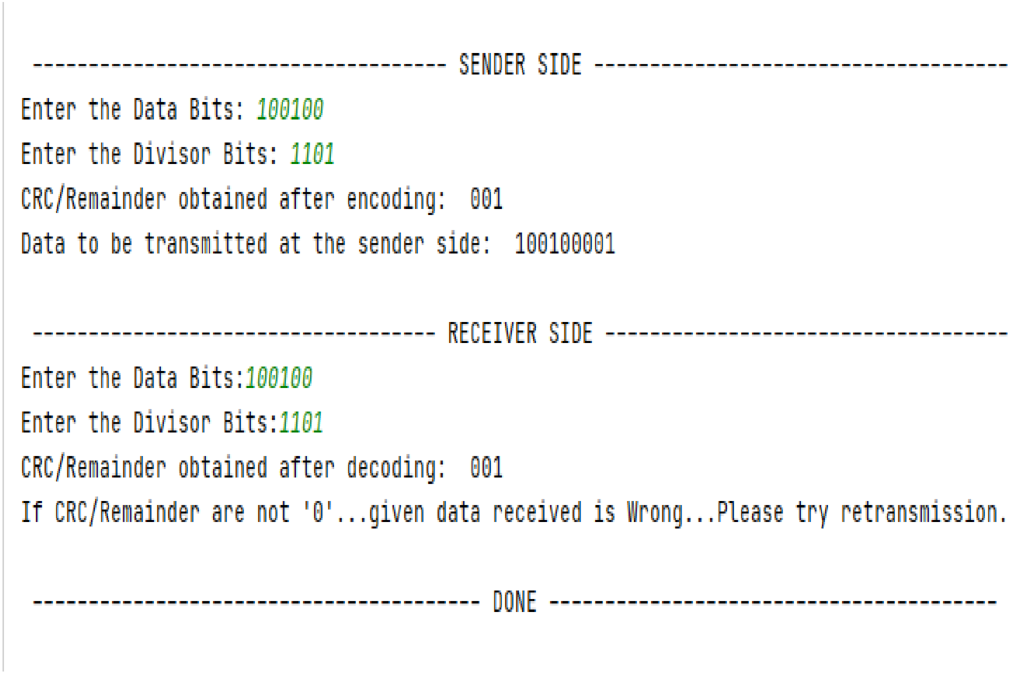


Code

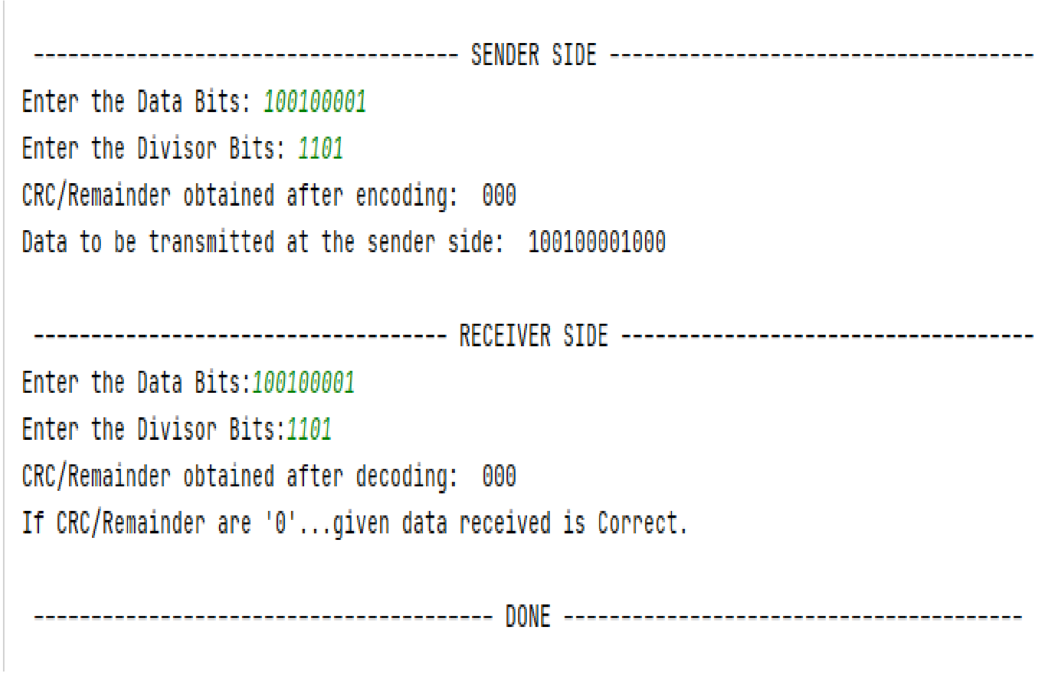
```
print("\n",37*"-","SENDER SIDE","-"*37)
def xor(a, b):
    result = []

    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')
    return ''.join(result)


def mod2div(divident, divisor):
    pick = len(divisor)
    tmp = divident[0: pick]

    while pick < len(divident):
        if tmp[0] == '1':
            tmp = xor(divisor, tmp) + divident[pick]
        else:
            tmp = xor('0' * pick, tmp) + divident[pick]

        pick += 1

    if tmp[0] == '1':
        tmp = xor(divisor, tmp)
    else:
        tmp = xor('0' * pick, tmp)

    checkword = tmp
    return checkword
```

```
def encodeData(data, key):
    l_key = len(key)
    appended_data = data + '0' * (l_key - 1)
    remainder = mod2div(appended_data, key)
    codeword = data + remainder
    print("CRC/Remainder obtained after encoding: ", remainder)
    print("Data to be transmitted at the sender side: ", codeword)

data= input("Enter the Data Bits: ")
key = input("Enter the Divisor Bits: ")
encodeData(data, key)



print("\n",36*"-","RECEIVER SIDE","-"*36)
def decodeData(data, key):
    l_key = len(key)
    appended_data = data + '0' * (l_key - 1)
    remainder = mod2div(appended_data, key)
    codeword = data + remainder
    print("CRC/Remainder obtained after decoding: ", remainder)
    temp = "0" * (len(key)-1)
    if remainder == temp:
        print("If CRC/Remainder are '0'...given data received is Correct.")
    else:
        print("If CRC/Remainder are not '0'...given data received is Wrong...Please try retransmission.")

data= input("Enter the Data Bits:")
key = input("Enter the Divisor Bits:")
decodeData(data, key)
print("\n",40*"-","DONE","-"*40,"\n")
```

Output



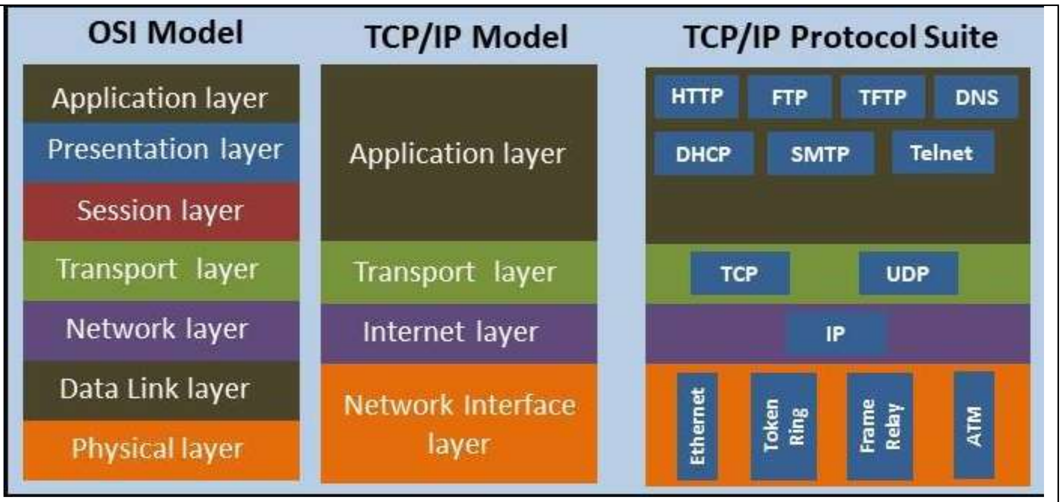
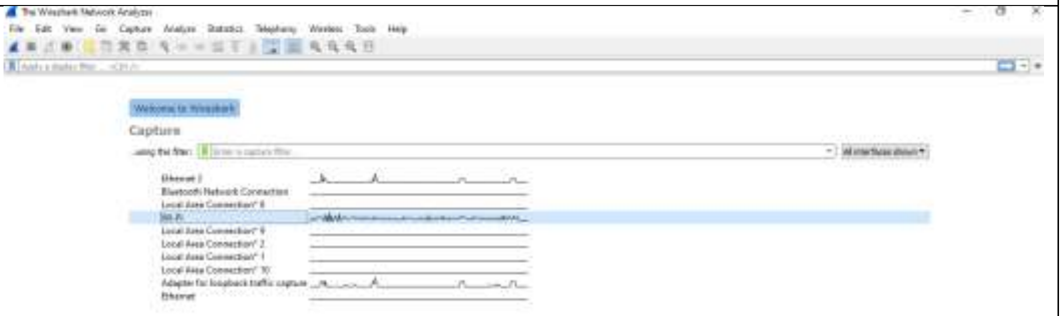
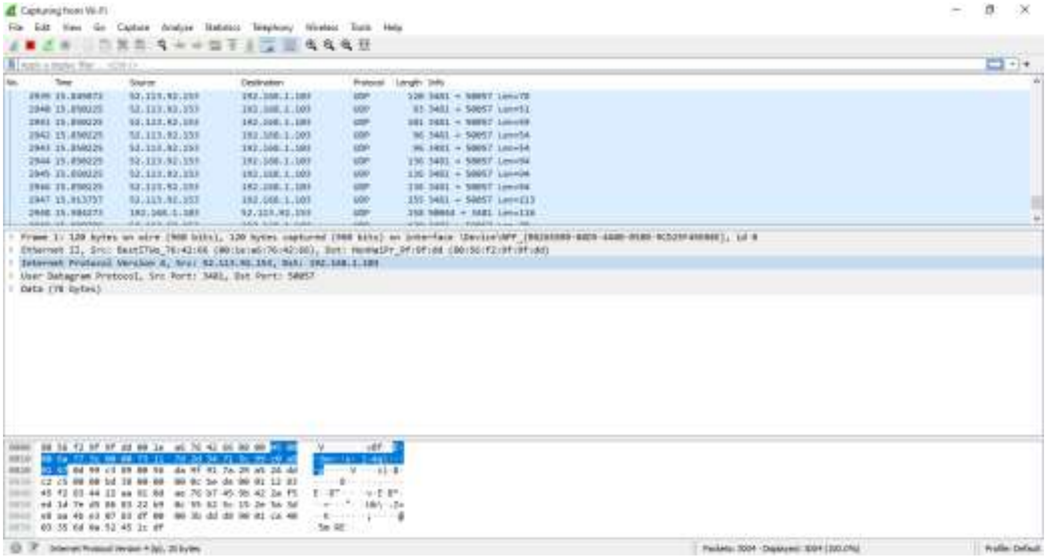
```
 ------------------------------------ SENDER SIDE ------------------------------------
Enter the Data Bits: 100100
Enter the Divisor Bits: 1101
CRC/Remainder obtained after encoding:  001
Data to be transmitted at the sender side:  100100001

 ----------------------------------- RECEIVER SIDE -----------------------------------
Enter the Data Bits:100100
Enter the Divisor Bits:1101
CRC/Remainder obtained after decoding:  001
If CRC/Remainder are not '0'...given data received is Wrong...Please try retransmission.

 --------------------------------------- DONE ---------------------------------------
```



```
 ------------------------------------- SENDER SIDE -------------------------------------
Enter the Data Bits: 100100001
Enter the Divisor Bits: 1101
CRC/Remainder obtained after encoding:  000
Data to be transmitted at the sender side:  100100001000

 ------------------------------------ RECEIVER SIDE ------------------------------------
Enter the Data Bits:100100001
Enter the Divisor Bits:1101
CRC/Remainder obtained after decoding:  000
If CRC/Remainder are '0'...given data received is Correct.

 ---------------------------------------- DONE ----------------------------------------
```

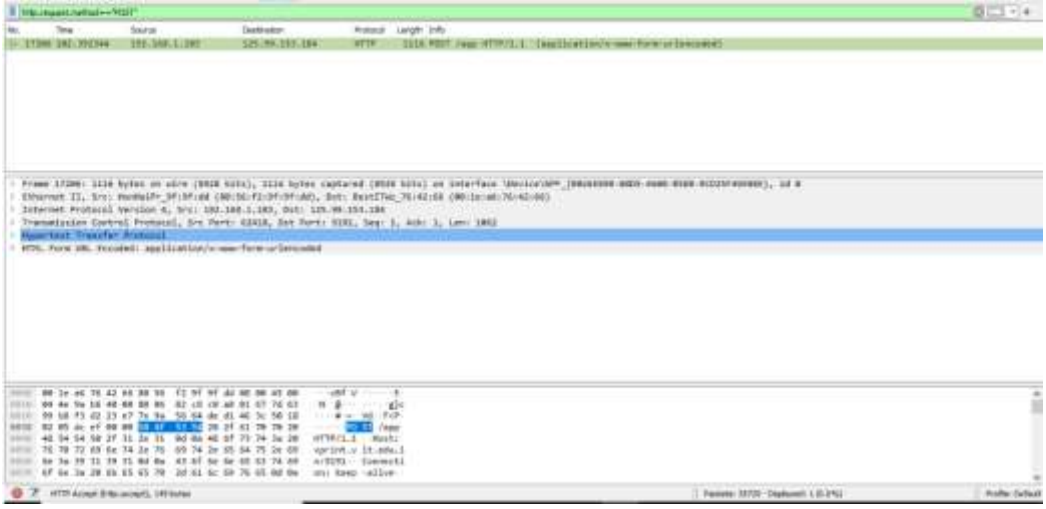
Semester	IV
Subject	Computer Networks and Networking Design

Experiment Number	6	
Experiment Title	Demonstrate the TCP/IP protocol suite for the current packet running.	
Objective	To study TCP/IP protocol suite.	
Resources / Apparatus Required	Hardware: PC with the Configuration of Intel Dual core Processor or higher, Minimum 2 GB RAM, Minimum 40 GB Hard disk, Network interface card.	Software: wireshark
Theory	**TCP/IP Protocol Suite :** → The Internet protocol suite is the conceptual model and set of communications protocols used in the Internet and similar computer networks. → It is commonly known as TCP/IP because the foundational protocols in the suite are the Transmission Control Protocol (TCP) and the Internet Protocol (IP). → During its development, versions of it were known as the Department of Defense (DOD) model because the development of the networking method was funded by the United States Department of Defense through DARPA. Its implementation is a protocol stack. → The Internet protocol suite provides end-to-end data communication specifying how data should be packetized, addressed, transmitted, routed, and received. → The TCP/IP protocol suite functions as an abstraction layer between internet applications and the routing/switching fabric.	

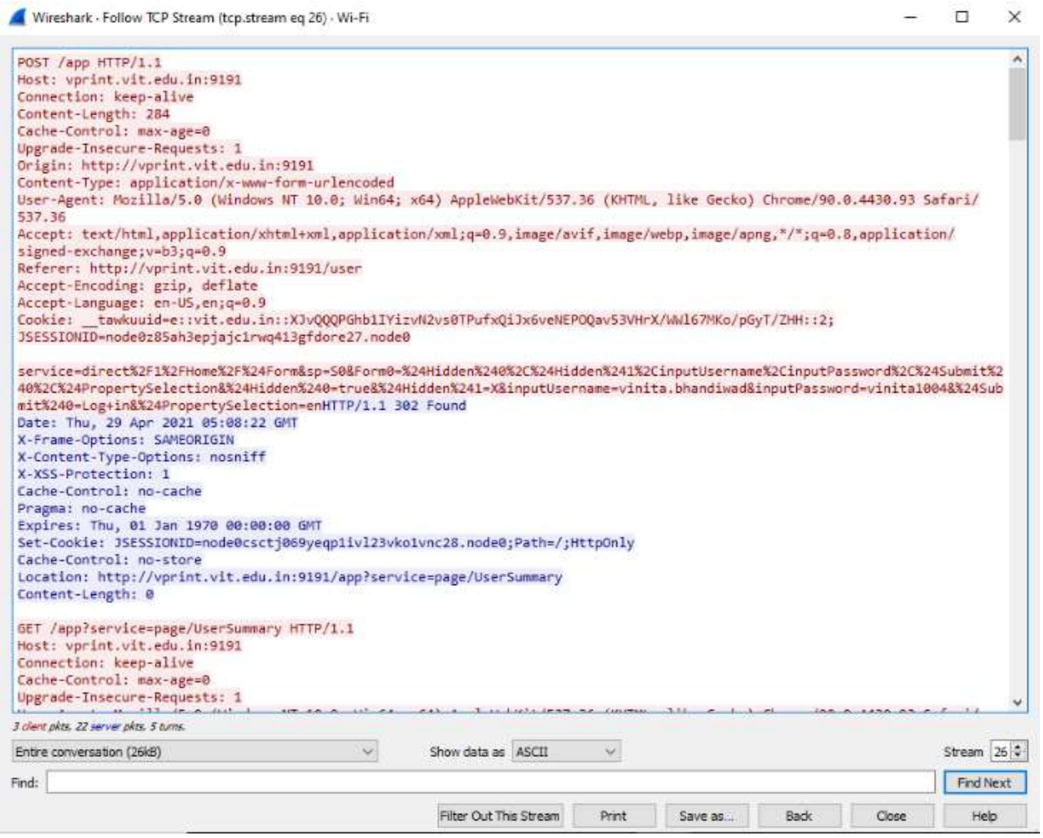
	
Output	 

```
Ethernet 2
Bluetooth Network Connection
Local Area Connection* 8
Wi-Fi
Local Area Connection* 9
Local Area Connection* 2
Local Area Connection* 1
Local Area Connection* 10
Adapter for loopback traffic capture
Ethernet
```

```
No.   Time        Source          Destination     Protocol  Length  Info
 2939 15.849873   52.113.92.153   192.168.1.103   UDP       120 3481 → 50057 Len=78
 2940 15.850225   52.113.92.153   192.168.1.103   UDP        93 3481 → 50057 Len=51
 2941 15.850225   52.113.92.153   192.168.1.103   UDP       101 3481 → 50057 Len=59
 2942 15.850225   52.113.92.153   192.168.1.103   UDP        96 3481 → 50057 Len=54
 2943 15.850225   52.113.92.153   192.168.1.103   UDP        96 3481 → 50057 Len=54
 2944 15.850225   52.113.92.153   192.168.1.103   UDP       136 3481 → 50057 Len=94
 2945 15.850225   52.113.92.153   192.168.1.103   UDP       136 3481 → 50057 Len=94
 2946 15.850225   52.113.92.153   192.168.1.103   UDP       136 3481 → 50057 Len=94
 2947 15.913757   52.113.92.153   192.168.1.103   UDP       155 3481 → 50057 Len=113
 2948 15.984273   192.168.1.103   52.113.92.153   UDP       158 50064 → 3481 Len=116

> Frame 1: 120 bytes on wire (960 bits), 120 bytes captured (960 bits) on interface \Device\NPF_{[illegible]}, id 0
> Ethernet II, Src: [illegible]_76:42:66 (00:[illegible]:76:42:66), Dst: [illegible]_9f:9f:dd (00:50:f2:9f:9f:dd)
> Internet Protocol Version 4, Src: 52.113.92.153, Dst: 192.168.1.103
> User Datagram Protocol, Src Port: 3481, Dst Port: 50057
> Data (78 bytes)

[illegible]

Internet Protocol Version 4 (ip), 20 bytes        Packets: 3004 · Displayed: 3004 (100.0%)        Profile: Default
```



```
http.request.method=="POST"
No.   Time         Source          Destination     Protocol Length Info
17206 182.392344   192.168.1.103   125.99.153.184  HTTP     1116   POST /app HTTP/1.1  (application/x-www-form-urlencoded)

Frame 17206: 1116 bytes on wire (8928 bits), 1116 bytes captured (8928 bits) on interface \Device\NPF_{...}, id 0
Ethernet II, Src: ..., Dst: ...
Internet Protocol Version 4, Src: 192.168.1.103, Dst: 125.99.153.184
Transmission Control Protocol, Src Port: 63418, Dst Port: 9191, Seq: 1, Ack: 1, Len: 1062
Hypertext Transfer Protocol
HTML Form URL Encoded: application/x-www-form-urlencoded
```



Wireshark · Follow TCP Stream (tcp.stream eq 26) · Wi-Fi

```
POST /app HTTP/1.1
Host: vprint.vit.edu.in:9191
Connection: keep-alive
Content-Length: 284
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://vprint.vit.edu.in:9191
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93 Safari/
537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/
signed-exchange;v=b3;q=0.9
Referer: http://vprint.vit.edu.in:9191/user
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: __tawkuuid=e::vit.edu.in::XJvQQQPGhb1IYizvN2vs0TPufxQiJx6veNEPOQav53VHrX/WWl67MKo/pGyT/ZHH::2;
JSESSIONID=node0z85ah3epjajc1rwq413gfdore27.node0

service=direct%2F1%2FHome%2F%24Form&sp=S0&Form0=%24Hidden%240%2C%24Hidden%241%2CinputUsername%2CinputPassword%2C%24Submit%2
40%2C%24PropertySelection&%24Hidden%240=true&%24Hidden%241=X&inputUsername=vinita.bhandiwad&inputPassword=vinita1004&%24Sub
mit%240=Log+in&%24PropertySelection=enHTTP/1.1 302 Found
Date: Thu, 29 Apr 2021 05:08:22 GMT
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-XSS-Protection: 1
Cache-Control: no-cache
Pragma: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: JSESSIONID=node0csctj069yeqp1ivl23vko1vnc28.node0;Path=/;HttpOnly
Cache-Control: no-store
Location: http://vprint.vit.edu.in:9191/app?service=page/UserSummary
Content-Length: 0

GET /app?service=page/UserSummary HTTP/1.1
Host: vprint.vit.edu.in:9191
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
```

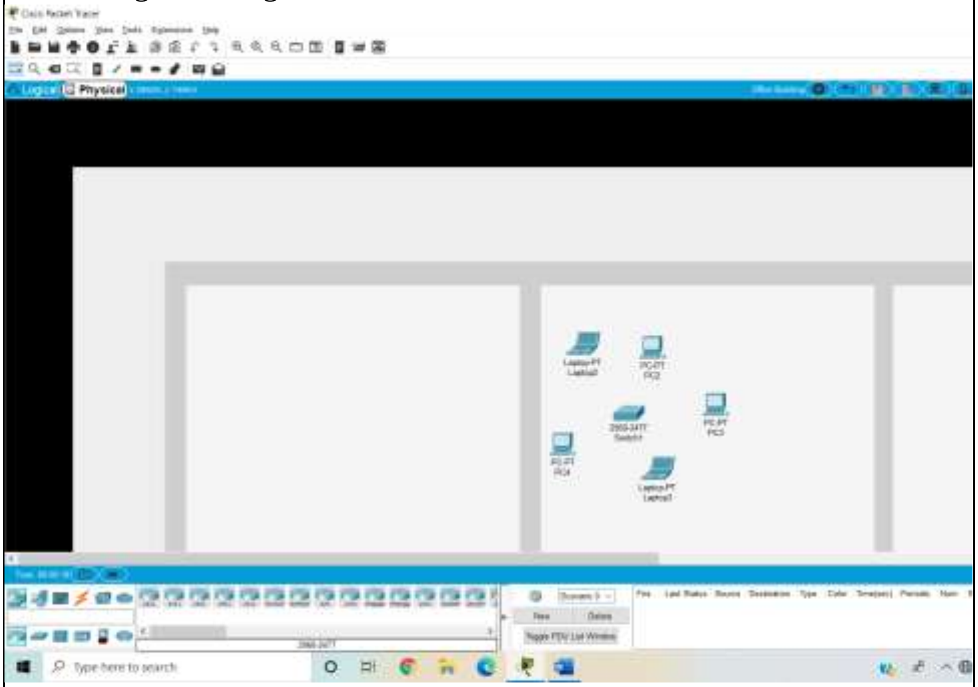
3 client pkts, 22 server pkts, 5 turns.

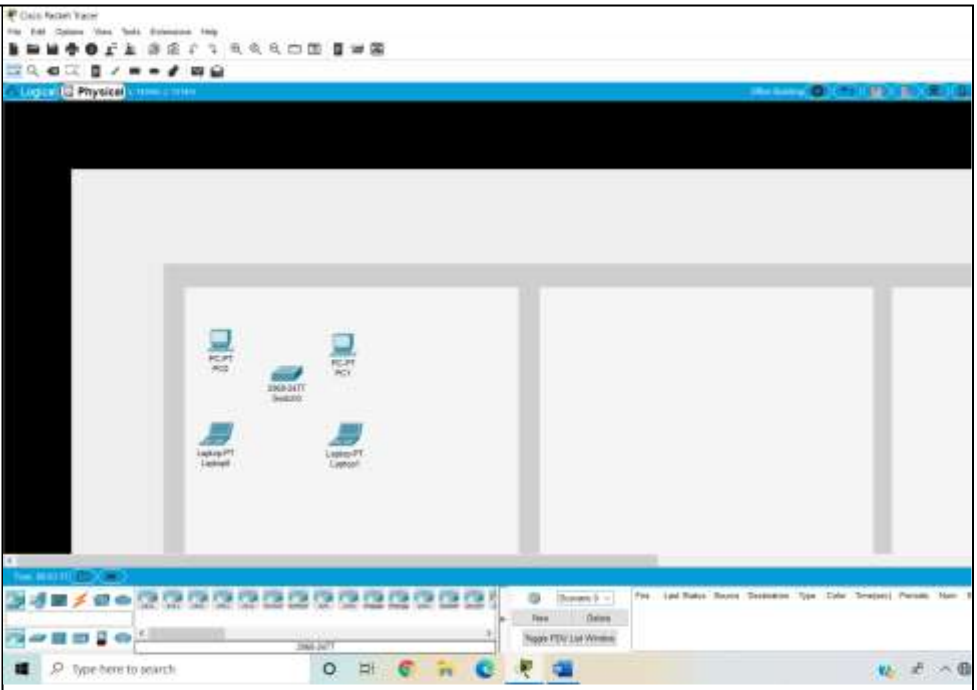
Entire conversation (26kB)

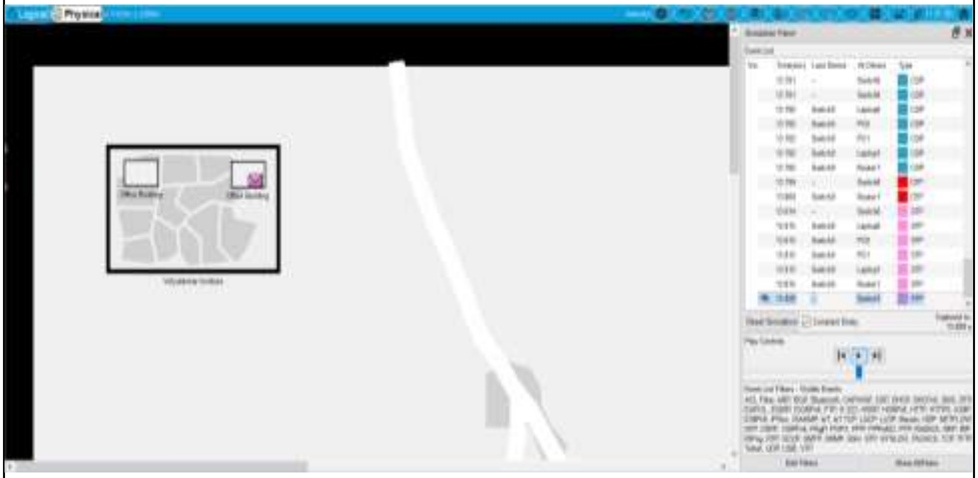
Show data as: ASCII

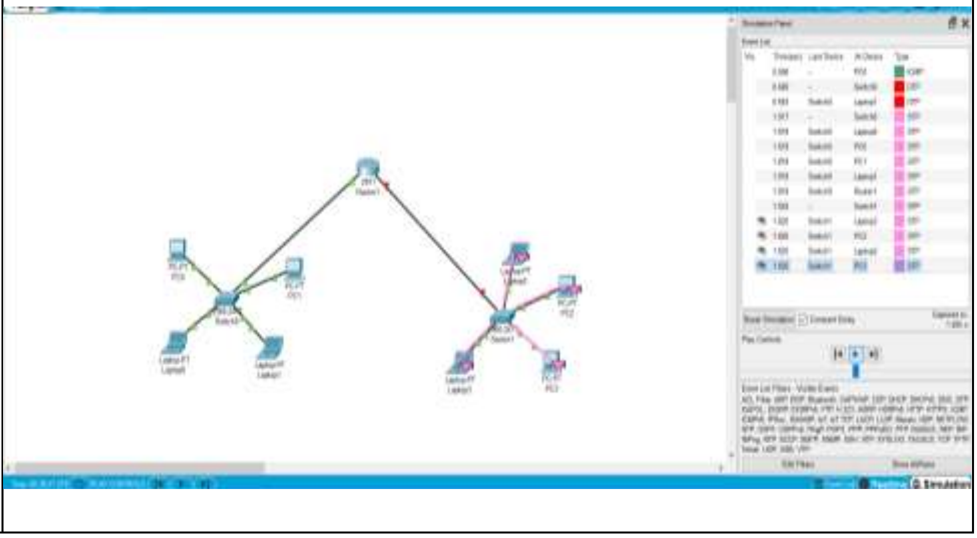
Stream: 26

Semester	IV
Subject	Computer Networks and Networking Design

Experiment Number	07	
Experiment Title	Design College LAN network using CPT.	
Resources / Apparatus Required	Hardware: Computer	Software: - Cisco Packet tracer
Objectives (Skill Set / Knowledge Tested / Imparted)	To Design College LAN network using CPT.	
Theory	Topology: Geometric representation of how the computers are connected to each other is known as topology. There are five types of topology – Mesh, Star, Bus, Ring and Hybrid.	
Output	First college building:  Cisco Packet Tracer Logical Physical Laptop-PT Laptop0 PC-PT PC2 2960-24TT Switch1 PC-PT PC3 PC-PT PC4 Laptop-PT Laptop1 New Delete Toggle PDU List Window Type here to search Second college building:	


Cisco Packet Tracer
Logical
Physical
PC-PT
PC2
PC-PT
PC1
2960-24TT
Switch0
Laptop-PT
Laptop0
Laptop-PT
Laptop1
Type here to search


Logical
Physical
Simulation Panel
Event List
Office Building
Office Building
Vidyalankar Institute


Simulation Panel
Event List
PC-PT
PC0
PC-PT
PC1
Switch0
Laptop-PT
Laptop0
Laptop-PT
Laptop1
2911
Router1
Switch1
Laptop-PT
PC-PT
PC2
Laptop-PT
Laptop3
PC-PT
PC3
Simulation

Semester	IV
Subject	Computer Networks and Networking Design

Experiment Number	8	
Experiment Title	Implement Dijkstra's algorithm	
Objective	To study Dijkstra's algorithm in python.	
Resources / Apparatus Required	Hardware: PC with the Configuration of Intel Dual core Processor or higher, Minimum 2 GB RAM, Minimum 40 GB Hard disk, Network interface card.	Software: Python 3.9/Pycharm

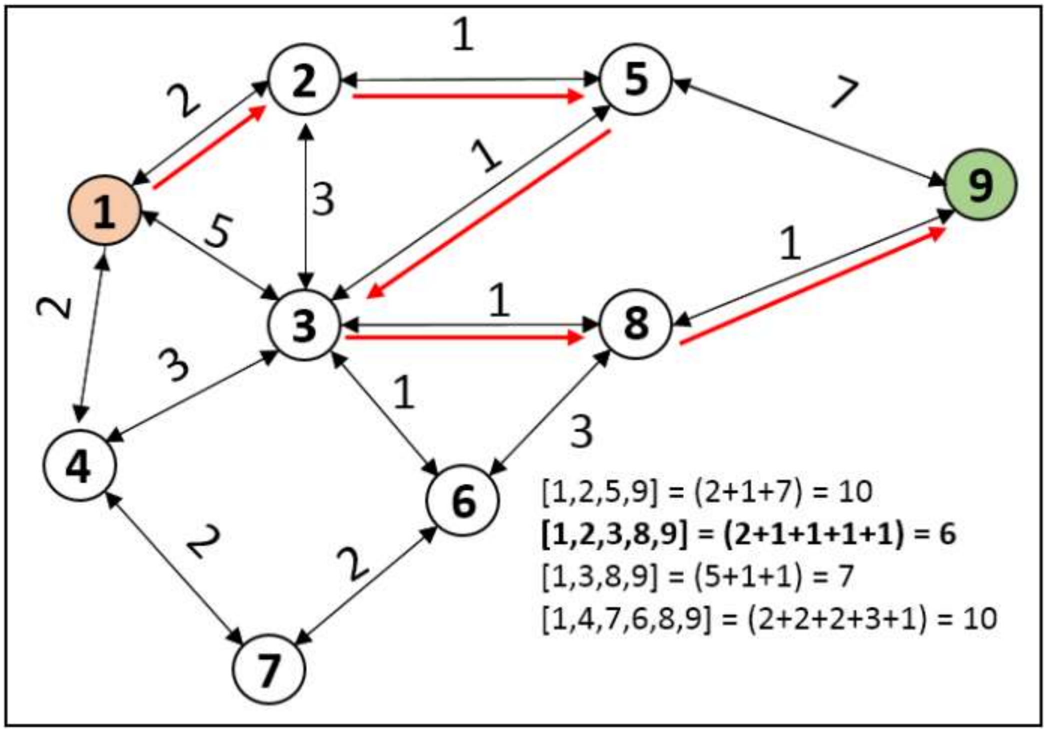
Theory

Dijkstra's Algorithm :

→ Dijkstra's algorithm allows us to find the shortest path between any two vertices of a graph.

→ It differs from the minimum spanning tree because the shortest distance between two vertices might not include all the vertices of the graph.

→ Eg :

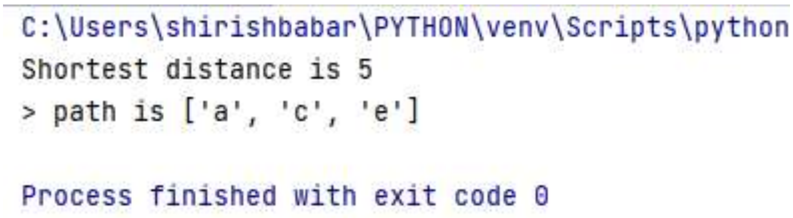


[1,2,5,9] = (2+1+7) = 10
[1,2,3,8,9] = (2+1+1+1+1) = 6
[1,3,8,9] = (5+1+1) = 7
[1,4,7,6,8,9] = (2+2+2+3+1) = 10

Code

```
graph = {'a': {'b': 10, 'c': 3}, 'b': {'c': 1, 'd': 2}, 'c': {'b': 4, 'd': 8, 'e': 2}, 'd': {'e': 7}, 'e': {'d': 9}}


def dijkstra(graph, start, goal):
    shortest_distance = {}
    predecessor = {}
    unseenNodes = graph
    infinity = 9999999
    path = []
    for node in unseenNodes:
        shortest_distance[node] = infinity
    shortest_distance[start] = 0

    while unseenNodes:
        minNode = None
        for node in unseenNodes:
            if minNode is None:
                minNode = node
            elif shortest_distance[node] < shortest_distance[minNode]:
                minNode = node

        for childNode, weight in graph[minNode].items():
            if weight + shortest_distance[minNode] <
shortest_distance[childNode]:
                shortest_distance[childNode] = weight +
shortest_distance[minNode]
                predecessor[childNode] = minNode
        unseenNodes.pop(minNode)

    currentNode = goal
    while currentNode != start:
        try:
            path.insert(0, currentNode)
            currentNode = predecessor[currentNode]
        except KeyError:
            print('The given path is not reachable.')
            break
    path.insert(0, start)
    if shortest_distance[goal] != infinity:
        print('Shortest distance is ' + str(shortest_distance[goal]))
        print('> path is ' + str(path))

dijkstra(graph, 'a', 'e')
```

Output



```
C:\Users\shirishbabar\PYTHON\venv\Scripts\python
Shortest distance is 5
> path is ['a', 'c', 'e']

Process finished with exit code 0
```

Semester	IV
Subject	Computer Networks and Networking Design

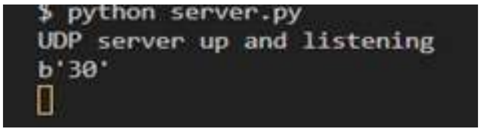
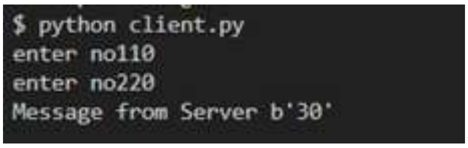
Experiment Number	9	
Experiment Title	Write a UDP-based socket program.	
Resources / Apparatus Required	Hardware: Computer	Software: Python IDLE
Objectives (Skill Set / Knowledge Tested / Imparted)		
Theory	Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.[Text Wrapping Break]They are the real backbones behind web browsing. In simpler terms there is a server and a client.	
Code	Following is the program to demonstrate UDP socket programing:- **Server.py** import socket localIP = "192.168.1.106" localPort = 20001 bufferSize = 1024 msgFromServer = "Hello UDP Client" bytesToSend = str.encode(msgFromServer)	

```
# Create a datagram socket

UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)

# Bind to address and ip

UDPServerSocket.bind((localIP, localPort))

print("UDP server up and listening")

# Listen for incoming datagrams

while(True):

   bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)

   message = bytesAddressPair[0]

   address = bytesAddressPair[1]

   no1 = UDPServerSocket.recvfrom(bufferSize)

   no2 = UDPServerSocket.recvfrom(bufferSize)

   clientMsg = "Message from Client:{}".format(message)

   clientIP  = "Client IP Address:{}".format(address)

   sum1 = str.encode(str(int(no1[0])+int(no2[0])))

   print(sum1)

   # Sending a reply to client

   UDPServerSocket.sendto(sum1,address)
```

Client.py

```
import socket

msgFromClient       = "Hello UDP Server"

bytesToSend         = str.encode(msgFromClient)

serverAddressPort   = ("192.168.1.106", 20001)

bufferSize          = 1024
```

	no1 = str.encode(input("enter no1"))
	no2 = str.encode(input("enter no2"))
	# Create a UDP socket at client side
	UDPClientSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
	# Send to server using created UDP socket
	UDPClientSocket.sendto(bytesToSend, serverAddressPort)
	UDPClientSocket.sendto(no1, serverAddressPort)
	UDPClientSocket.sendto(no2, serverAddressPort)
	msgFromServer = UDPClientSocket.recvfrom(bufferSize)
	msg = "Message from Server {}".format(msgFromServer[0])
	print(msg)

Output	**Server.py**
	
	$ python server.py
	UDP server up and listening
	b'30'
	Client.py
	
	$ python client.py
	enter no110
	enter no220
	Message from Server b'30'

Conclusion	Here we learnt to do Socket Programming.

Experiment Number	9	
Experiment Title	Write a TCP-based socket program.	
Resources / Apparatus Required	Hardware: Computer	Software: Python IDLE
Objectives (Skill Set / Knowledge Tested / Imparted)		
Theory	Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.[Text Wrapping Break]They are the real backbones behind web browsing. In simpler terms there is a server and a client.	

Code

Following is the program to demonstrate TCP socket programing:-

Server.py

```
#importing socket
import socket as s
#server socket
socket1 = s.socket()
print("Socket Created")
#address where the server should run
socket1.bind(('192.168.56.1',9999))
print("Bind Completed")
#specifing how many client's connection can be accepted/start lestning for ser
ver
socket1.listen(3)
print("Server Listening")
while True:
   #accept connection
   c,addr = socket1.accept()
   print("Connection established!client connected",addr)
   #receiving data from client
   n = c.recv(1024).decode()
   m = c.recv(1024).decode()
   inc = 0
   for i in n:
      if m == i:
         inc+=1
   print(inc)
   #transmitting data from server to client
```

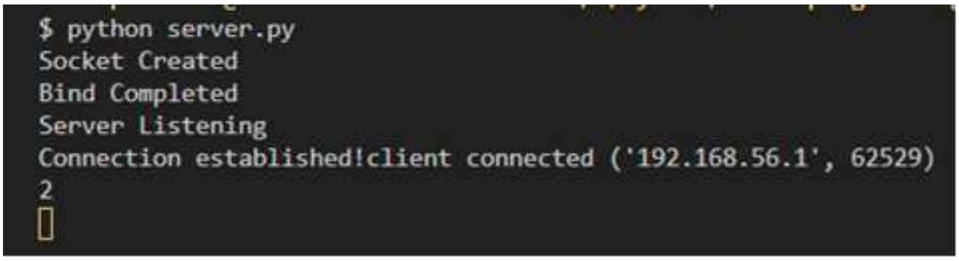
```
    c.send(bytes(str(inc),"utf-8"))
    c.close()
```

Client.py

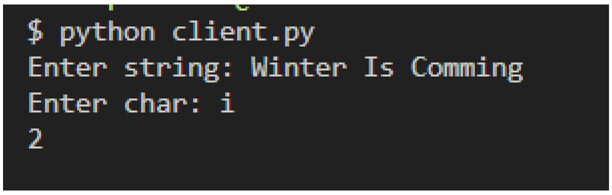
```
import socket as s

#create client socket

c = s.socket()

#connecting client socket with server

c.connect(('192.168.56.1',9999))

n = input("Enter string: ")

m = input("Enter char: ")

#sending data from client to server

c.send(bytes(n,"utf-8"))

#sending data from client to server

c.send(bytes(m,"utf-8"))

#receiving data from server

str = c.recv(1024)

#decoding the data

print(str.decode())
```

Output

Server.py



```
$ python server.py
Socket Created
Bind Completed
Server Listening
Connection established!client connected ('192.168.56.1', 62529)
2
▯
```

	Client.py  $ python client.py Enter string: Winter Is Comming Enter char: i 2
Conclusion	Here we learned to do Socket Programming.

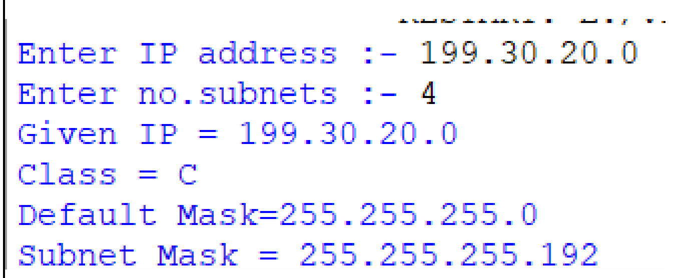
Semester	IV
Subject	Computer Networks and Networking Design

Experiment Number	10	
Experiment Title	Calculate IP addressing values.	
Objective	Considering IP address, calculate Network address, Broadcast address, Default Broadcast address,Number of Host.	
Resources / Apparatus Required	Hardware: Computer	Software: -Any Python IDLEs like Spyder or Pycharm

Code:

```
def A(ip,l,bit,n):
    start,end,first,last=[],[],["0","0","0","0"],["0","0","0","0"]
    print("Default Mask=255.0.0.0")
    l[0],l[1]="255",bit*'1'+(8-bit)*'0'
    l[1]=str(int("0b"+l[1],2))
    print("Subnet Mask =",".".join(l))
    b=int("0b"+(8-bit)*"1",2)
    c=0
    for i in range(n):
        ip[1]=str((b+1)*i)
        end.append(ip[1])
        ip[1]=str((b*(i+1)+c))
        start.append(ip[1])
        c+=1
    print("subnet startAdd endAdd")
    for i in range(n):
        print(i+1,end=" ")
        last[1]=end[i]
        last[0],last[2],last[3]=ip[0],"0","0"
        print(".".join(last),end=" ")
        first[1]=start[i]
    first[0],first[2],first[3]=ip[0],"255","255"
    print(".".join(first))
def B(ip,l,bit,n):
    start,end,first,last=[],[],["0","0","0","0"],["0","0","0","0"]
    print("Default Mask=255.255.0.0")
    l[0],l[1],l[2]="255","255",bit*'1'+(8-bit)*'0'
    l[2]=str(int("0b"+l[2],2))
    print("Subnet Mask =",".".join(l))
    b=int("0b"+(8-bit)*"1",2)
    c=0
    for i in range(n):
        ip[2]=(b+1)*i
        end.append(ip[2])
        ip[2]=(b*(i+1)+c)
        start.append(ip[2])
        c+=1
    print("subnet startAdd endAdd")
    for i in range(n):
        print(i+1,end=" ")
        last[2]=str(end[i])
        last[0],last[1],last[3]=ip[0],ip[1],"0"
        print(".".join(last),end=" ")
        first[2]=str(start[i])
        first[0],first[1],first[3]=ip[0],ip[1],"255"
        print(".".join(first))
def C(ip,l,bit,n):
    start,end,first,last=[],[],["0","0","0","0"],["0","0","0","0"]
    print("Default Mask=255.255.255.0")
    l[0],l[1],l[2],l[3]="255","255","255",bit*'1'+(8-bit)*'0'
    l[3]=str(int("0b"+l[3],2))
    print("Subnet Mask =",".".join(l))
    b=int("0b"+(8-bit)*"1",2)
    c=0
    for i in range(n):
        ip[3]=(b+1)*i
        end.append(ip[3])
        ip[3]=(b*(i+1)+c)
        start.append(ip[3])
        c+=1
    print("subnet startAdd endAdd")
    for i in range(n):
        print(i+1,end=" ")
        last[3]=str(end[i])
        last[0],last[1],last[2]=ip[0],ip[1],ip[2]
        print(".".join(last),end=" ")
        first[3]=str(start[i])
        first[0],first[1],first[2]=ip[0],ip[1],ip[2]
        print(".".join(first))
s,l=input("Enter IP address :- "),["0","0","0","0"]
n=int(input("Enter no.subnets :- "))
ip=s.split('.')
for i in range(1,9):
    if(2**i>=n):
        bit=i
        break
```

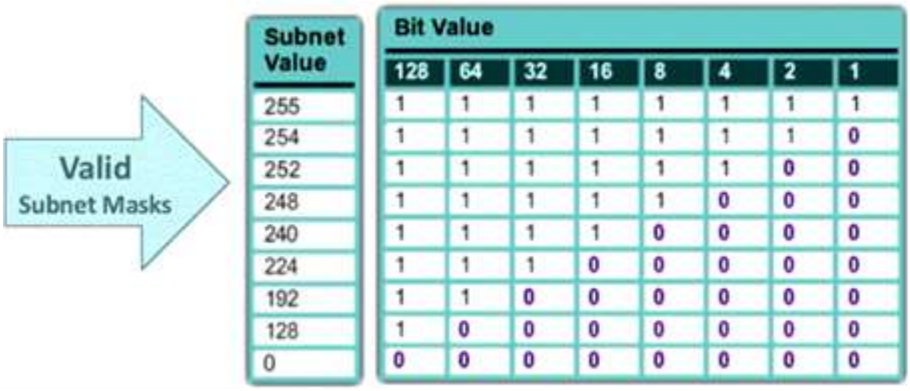
	print("Given IP =",s)
	if(int(ip[0]) in range(0,128)):
	    print("Class = A")
	    A(ip,l,bit,n)
	elif(int(ip[0]) in range(128,192)):
	    print("Class = B")
	    B(ip,l,bit,n)
	elif(int(ip[0]) in range(192,224)):
	    print("Class = C")
	    C(ip,l,bit,n)
	else:
	    print("IP is not valid")

Output:	 Enter IP address :- 199.30.20.0 Enter no.subnets :- 4 Given IP = 199.30.20.0 Class = C Default Mask=255.255.255.0 Subnet Mask = 255.255.255.192
Conclusion:	Thus the IP addressing code is calculated.

Semester	IV
Subject	Computer Networks and Networking Design

Experiment Number	Lab 11	
Experiment Title	Consider IP address and no of subnets required by the industry, display details of entire network and each network separately.	
Objective	Creation of subnets.	
Resources Required	Hardware: Desktop	Software: python, pycharm
Theory	**What is Subnetting?** Subnetting is the practice of dividing a network into two or smaller networks. It increases routing efficiency, which helps to enhance the security of the network and reduces the size of the broadcast domain. IP Subnetting designates high-order bits from the host as part of the network prefix. This method divides a network into smaller subnets. It also helps you to reduce the size of the routing tables, which is stored in routers. This method also helps you to extend the existing IP address base & restructures the IP address. **Why Use Subnetting?** • It helps you to maximise IP addressing efficiency. • Extend the life of IPV4. • Public IPV4 Addresses are scarce. • IPV4 Subnetting reduces network traffic by eliminating collision and broadcast traffic and thus improves overall performance. • This method allows you to apply network security policies at the interconnection between subnets. • Optimized IP network performance. • Facilitates spanning of large geographical distances. • Subnetting process helps to allocate IP addresses that prevent large numbers of IP network addresses from remaining unused. • Subnets are usually set up geographically for specific offices or particular teams within a business that allows their network traffic to stay within the location. • **What is Subnet Mask?** A subnet mask is a 32 bits address used to distinguish between a network address and a host address in IP address. A subnet mask identifies which part of an IP address is the network address an	

d the host address. They
are not shown inside the data packets traversing the Internet.
They carry the destination IP address, which a router will match with a subnet.


Valid
Subnet Masks

Subnet Value	Bit Value							
	128	64	32	16	8	4	2	1
255	1	1	1	1	1	1	1	1
254	1	1	1	1	1	1	1	0
252	1	1	1	1	1	1	0	0
248	1	1	1	1	1	0	0	0
240	1	1	1	1	0	0	0	0
224	1	1	1	0	0	0	0	0
192	1	1	0	0	0	0	0	0
128	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

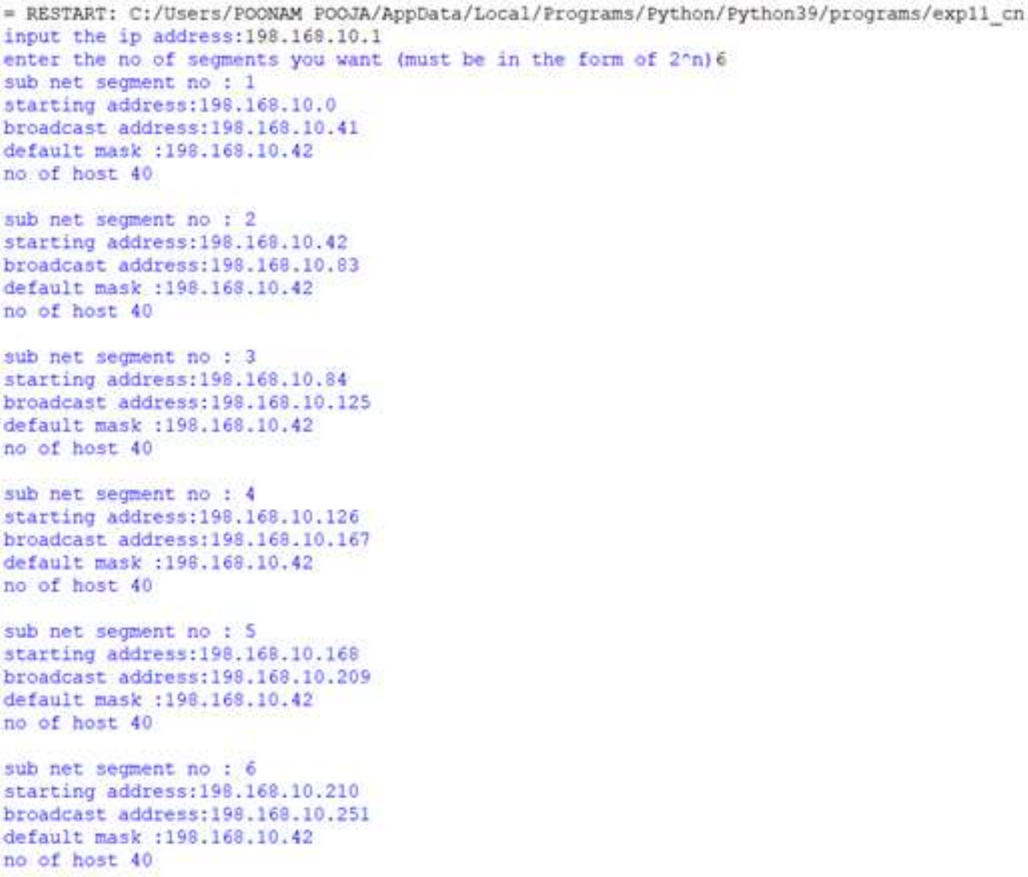
Two types of subnet masks are:

- The
  default Subnet Mask is the number of bits which is reserved by the address class. Using this default mask will accommodate a single network subnet in the relative class.
- A Custom Subnet Mask can be defined by an administrator to accommodate many Network

code

```
ipAdd = input('input the ip address:')
tempArray = ipAdd.split('.')
addrArray = []
str1 = "."

if len(tempArray) != 4:
    raise NameError('not a valid ip address')

for i in tempArray:
    num = int(i)
    if num < 0 or num > 255:
        raise NameError('not a valid ip address')
    addrArray.append(num)

noOfNetwork = int(input('enter the no of segments you want (must
 be in the form of 2^n)'))

networkRange = int(256 / noOfNetwork)
startadd = 0
for i in range(noOfNetwork):
    print('sub net segment no :', i + 1)
    print(f'starting address:{addrArray[0]}.{addrArray[1]}.{addr
Array[2]}.{startadd}')
    print(f'broadcast address:{addrArray[0]}.{addrArray[1]}.{add
rArray[2]}.{startadd + networkRange - 1}')
    print(f'default mask :{addrArray[0]}.{addrArray[1]}.{addrArr
ay[2]}.{networkRange}')
    print('no of host', networkRange - 2)
    print()
    startadd = startadd + networkRange
```

Output



```
= RESTART: C:/Users/POONAM POOJA/AppData/Local/Programs/Python/Python39/programs/exp11_cn
input the ip address:198.168.10.1
enter the no of segments you want (must be in the form of 2^n)6
sub net segment no : 1
starting address:198.168.10.0
broadcast address:198.168.10.41
default mask :198.168.10.42
no of host 40

sub net segment no : 2
starting address:198.168.10.42
broadcast address:198.168.10.83
default mask :198.168.10.42
no of host 40

sub net segment no : 3
starting address:198.168.10.84
broadcast address:198.168.10.125
default mask :198.168.10.42
no of host 40

sub net segment no : 4
starting address:198.168.10.126
broadcast address:198.168.10.167
default mask :198.168.10.42
no of host 40

sub net segment no : 5
starting address:198.168.10.168
broadcast address:198.168.10.209
default mask :198.168.10.42
no of host 40

sub net segment no : 6
starting address:198.168.10.210
broadcast address:198.168.10.251
default mask :198.168.10.42
no of host 40
```