VIGNAN'S
Foundation for Science, Technology & Research
UNIVERSITY
(Estd u/s 3 of UGC Act of 1956)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COMPUTER NETWORKS LAB MANUAL

Experiment No:1

1. **Study of Network devices in detail**

Aim: To implement Study of Network devices in detail

problem Description:

Introduction:

All the networks reqire devices to provide connectivity and functionality.
For a LAN to be able to access the backbone network, special devices are required.

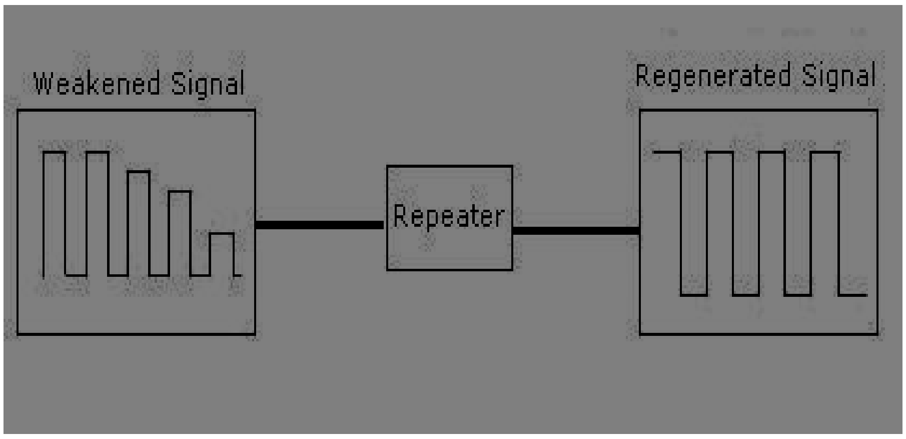
Using the OSI seven-layer reference model as the basis for comparison, networks can be interlinked at various layers. The layer at which networks are linked will depend on the type of networks that need to be connected. The various devices used to link lan segments operate at different layers of the OSI model. Each of the relay devices provides a different type of connectivity, performing different functions. Therefore, most networks will have more than one type of relay device in use. Relay devices support one of two general internal architectures, depending on the type of connectivity they are designed to provide. A local relay device is designed to connect two LANs directly. A remote relay device crosses at least one intermediate network.

In addition to this are given a detailed analysis of various networking topologies .

NETWORK DEVICES:

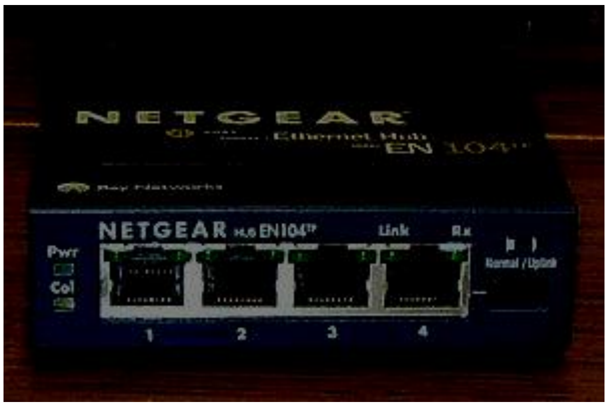
1.Repeaters:

The term repeater comes from the early days of visual communication,
There are many types of media, and each one has advantages and disadvantages. One of the disadvantages of the type of cable that we primarily use (CAT5 UTP) is cable length. The maximum length for UTP cable in a network is 100 meters if we need to extend our network beyond that limit, we must add a device to our network. This device is called a repeater.
The purpose of a repeater is regenerate and retime network signals at the bit level to allow them to travel a longer distance on the media. Repeaters are classified as Layer 1 devices in the OSI model, because they act only on the bit level and look at no other information.
To pass data through the repeater in a from one segment to the next, the packets and the Logical Link Control (LLC) protocols must be the same on the each segment. This means that a repeater will not enable communication, In other words, repeaters do not translate anything.



2.HUB:

hubs are used in networks that are used in twisted pair cabling to connect devices.hubs can also be joined together to create larger networks.hubs are of various sizes and shapes.Hubs are simple devices that direct data packets to all devices connected to the hub, regardless of whether the data package is destined for the device.hubs don’t perform any processing on the data it forwards nor it performs error checking.hubs enable communication between connected signals.the only disadvantage in hubs is that it can't divide a network without the help of switches,bridges,routers.Each switch port, bridge port, or router port forms a new collision domain. That is, devices connected to a single port share the network bandwidth, but they are protected from the interfering signals of devices on other ports.



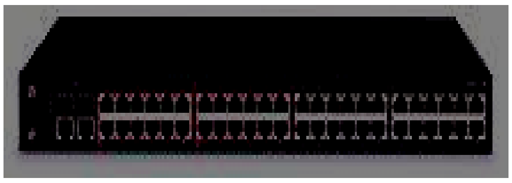
3.switches:

switch is a device that gathers the signals from devices that are connected to it, and then regenerates a new copy of each signal. Switches on the other hand are more advanced. Instead of broadcasting the frames everywhere, a switch actually checks for the destination MAC address and forward it to the relevant port to reach that computer only.If a destination MAC address is not in the table it forwards to all segments except the source segment. If the destination is same as the source, frame is discarded.Different speed levels are supported. They can be 10 Mb/s, 100 Mb/s, 1 Gb/s or more. Most common switching methods are:

a.Cut through: in this the packet begins to be forwarded as soon as it is received.it is very fast.no error checking.

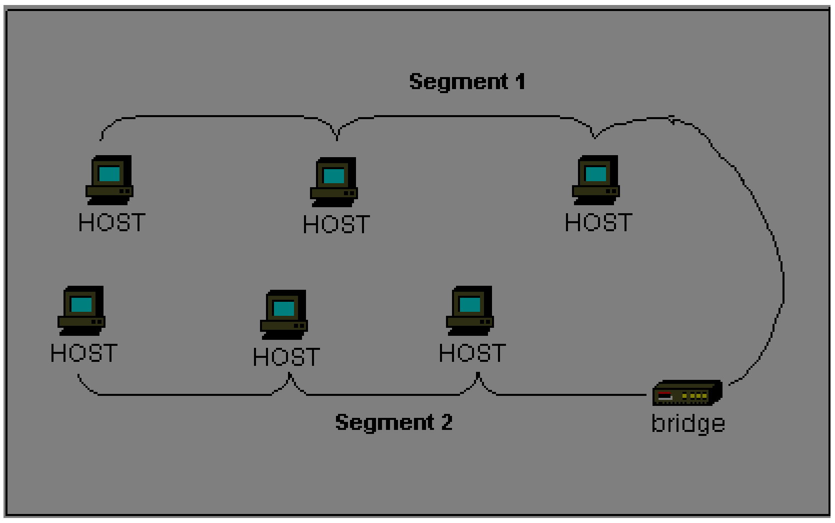
b.store and forward: here the entire packet is received and error is checked before its forwarding takes long time. Most common switches operate by learning the MAC addresses of all connected clients, servers, and peripherals, and associating each address with one of its ports. When a switch receives an incoming signal, it creates a temporary circuit between the sender and receiver. The temporary circuit provides two important benefits

C.FragmentFree:To take advantage of the error checking of store-andforward switching, but still offer performance levels nearing that of cutthrough switching, FragmentFree switching can be used.



4.Bridges:

A bridge is a device that connects two or more local area networks, or two or more segments of the same network. For example, suppose that your network includes both 10BaseT Ethernet and LocalTalk connections. You can use a bridge to connect these two networks so that they can share information with each other. In addition to connecting networks, bridges perform an additional, important function. possible for bridges to connect networks with different physical and data link level protocols. For example, you can use a bridge to connect a LocalTalk A Bridge connecting two networks network to a TokenRing network. Traditional bridges connect a single workgroup to another workgroup.



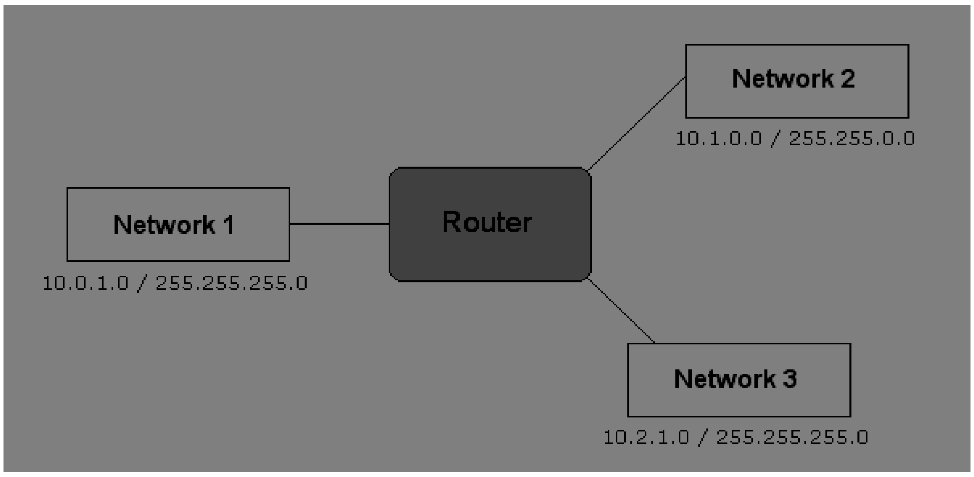
Placement Bridges should be positioned in the network using the 80/20 rule. This rule dictates that 80% of the data should be local and that the other 20% should be destined for devices on the other side of the bridge.
Bridging loops can occur when more than one bridge is implemented on the network. In this scenario, the bridges can confuse each other by leading one another to believe that a device is located on a certain segment when it is not.

Types of Bridges

Three types of bridges are used in networks:
Transparent bridge Derives its name from the fact that the devices on the network are unaware of its existence. A transparent bridge does nothing except block or forward data based on the MAC address. Source route bridge Used in Token Ring networks.
The source route bridge derives its name from the fact that the entire path that the packet is to take through the network is embedded within the packet.
Translational bridge Used to convert one networking data format to another; for example, from Token Ring to Ethernet and vice versa.

5.<u>Routers:</u>

routers are used to create larger networks by joining two network segments. Like bridges, routers are devices whose primary purpose is to connect two or more networks and to filter network signals so that only desired information travels between them. For example, routers are often used to regulate the flow of information between bridges, and they therefore can regulate network traffic more precisely. They also have another important capability:



Routers will only pass the information if the network address is known. This ability to control the data passing through the router reduces the amount of traffic between networks and allows routers to use these links more efficiently than bridge. Routers can filter traffic so that only authorized personnel can enter restricted areas. They can permit or deny network communications with a particular Web site. They can recommend the best route for information to travel. When a router receives a packet of data, it reads the header of the packet to determine the destination address. Once it has determined the address, it looks in its routing table to determine whether it knows how to reach the destination and, if it does, it forwards the packet to the next hop on the route.

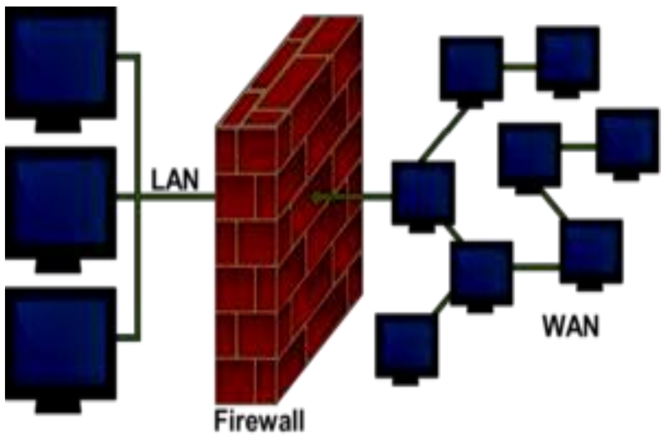
6.GATEWAYS:

Gateways make communication possible between different architectures and environments. They repackage and convert data going from one environment to another so that each environment can understand the other's environment data. A gateway repackages information to match the requirements of the destination system. Gateways can change the format of a message so that it will conform to the application program at the receiving end of the transfer. A gateway links two systems that do not use the same:

- Communication protocols
- Data formatting structures
- Languages
- Architecture

In many cases, the gateway functionality is incorporated into another device. Gateways operate at the network layer and above, but most of them at the application layer.

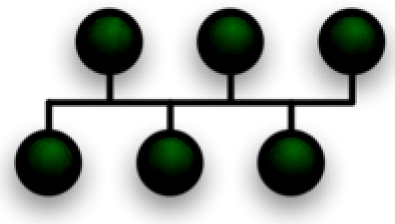
7.Fire walls:



A firewall is a device that prevents unauthorized electronic access to your network. The term firewall is generic, and includes many different kinds of protective hardware and software devices. Routers, discussed in the previous section, comprise one kind of firewall. A different kind of firewall might be created by installing software on a network server that is dedicated to the task of monitoring network activity. Yet another firewall consists of a standalone box. firewalls are used in networks of all sizes today. Hardware firewalls are often dedicated network devices that can be implemented with very little configuration and protect all systems behind the firewall from outside sources. Hardware firewalls are readily available and often combined with other devices today. For example, many broadband routers and wireless access points have firewall functionality built in. All firewalls have one thing in common: they guard your network, examining information inside every network packet. Based on a list of restrictions which you provide, the firewall allows or disables each packet from traveling any further. Firewalls can be divided into three major categories: packet- screening firewalls, proxy servers, and stateful inspection proxies. Packet-screening firewalls. Packet-screening firewalls operate by examining incoming or outgoing signals for information at OSI level 3.

III NETWORK TOPOLOGIES

TOPOLOGY – defines the structure of the network. There are two parts to the topology definition: the physical topology which is the actual layout of the wire (media) and the logical topology which defines how the media is accessed by the hosts.
It refers also to how computers are being connected with each other.

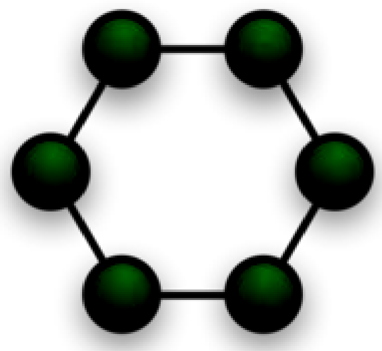


BUS TOPOLOGY: Uses a single backbone segment (length of cable) that all the hosts connect to directly. The idea is that is just like riding a bus. It has only one driver and many passengers who are riding. Bus networks (not to be confused with the system bus of a computer) use a common backbone to connect all devices. A single cable, the backbone functions as a shared communication medium that devices attach or tap into with an interface connector.

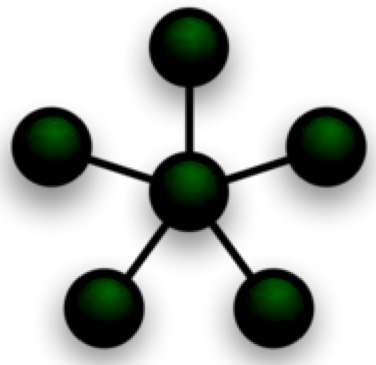
Linear bus: The type of network topology in which all of the nodes of the network are connected to a common transmission medium which has exactly two endpoints.

Distributed bus: The type of network topology in which all of the nodes of the network are connected to a common transmission medium which has more than two endpoints that are created by adding branches to the main section of the transmission medium – the physical distributed bus topology functions in exactly the same fashion as the physical linear bus topology.

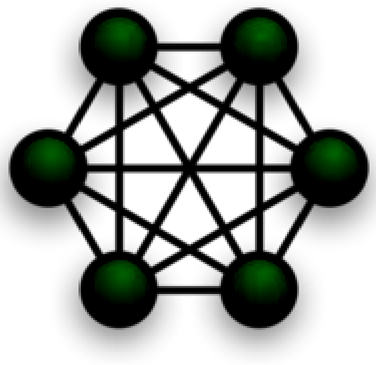
RING TOPOLOGY connects one host to the next and the last host to the first. This creates a physical ring of cable. In a ring network, every device has exactly two neighbors for communication purposes. All messages travel through a ring in the same direction (either "clockwise" or "counterclockwise"). A failure in any cable or device breaks the loop and can take down the entire network. To implement a ring network, one typically uses FDDI, SONET, or Token Ring technology. Ring topologies are found in some office buildings or school campuses.



STAR TOPOLGY: connects all cables to a central point of concentration. This point is usually a hub or switch. It has a focal point where all the resources are there. Many home networks use the star topology. A star network features a central connection point called a "hub node" that may be a network hub, switch or router. Devices typically connect to the hub with Unshielded Twisted Pair (UTP) Ethernet. Compared to the bus topology, a star network generally requires more cable, but a failure in any star network cable will only take down one computer's network access and not the entire LAN.



MESHTOPOLOGY: is used when there can be absolutely no break in communications. So as you can see in the graphic, each host has its connections to all other hosts. This also reflects the design of the internet which has multiple paths to any one location. Mesh topologies involve the concept of routes. Unlike each of the previous topologies, messages sent on a mesh network can take any of several possible paths from source to destination. Some WANs, most notably the Internet, employ mesh routing. A mesh network in which every device connects to every other is called a full mesh. As shown in the illustration below, partial mesh networks also exist in which some devices connect only indirectly to others.



TREE TOPOLOGY Tree topologies integrate multiple star topologies together onto a bus. In its simplest form, only hub devices connect directly to the tree bus, and each hub functions as the root of a tree of devices. This bus/star hybrid approach supports future expandability of the network much better than a bus (limited in the number of devices due to the broadcast traffic it generates) or a star (limited by the number of hub connection points) alone.

IV Conclusion

We have been introduced so far are the main traditional devices used to build networks, understanding how they work helps to understand the logic behind networks designing, however, now that technology advance quickly, it is possible to find new products in the market combining two or more of these devices into one. Thus this paper presents a precise survey of network devices and its functionalities along with different network topologies that form the base for ant network architecture.

Experiment No:2

2. Connect the computers in Local Area Network

Aim: A study on computers in LAN in detail

Problem Description:

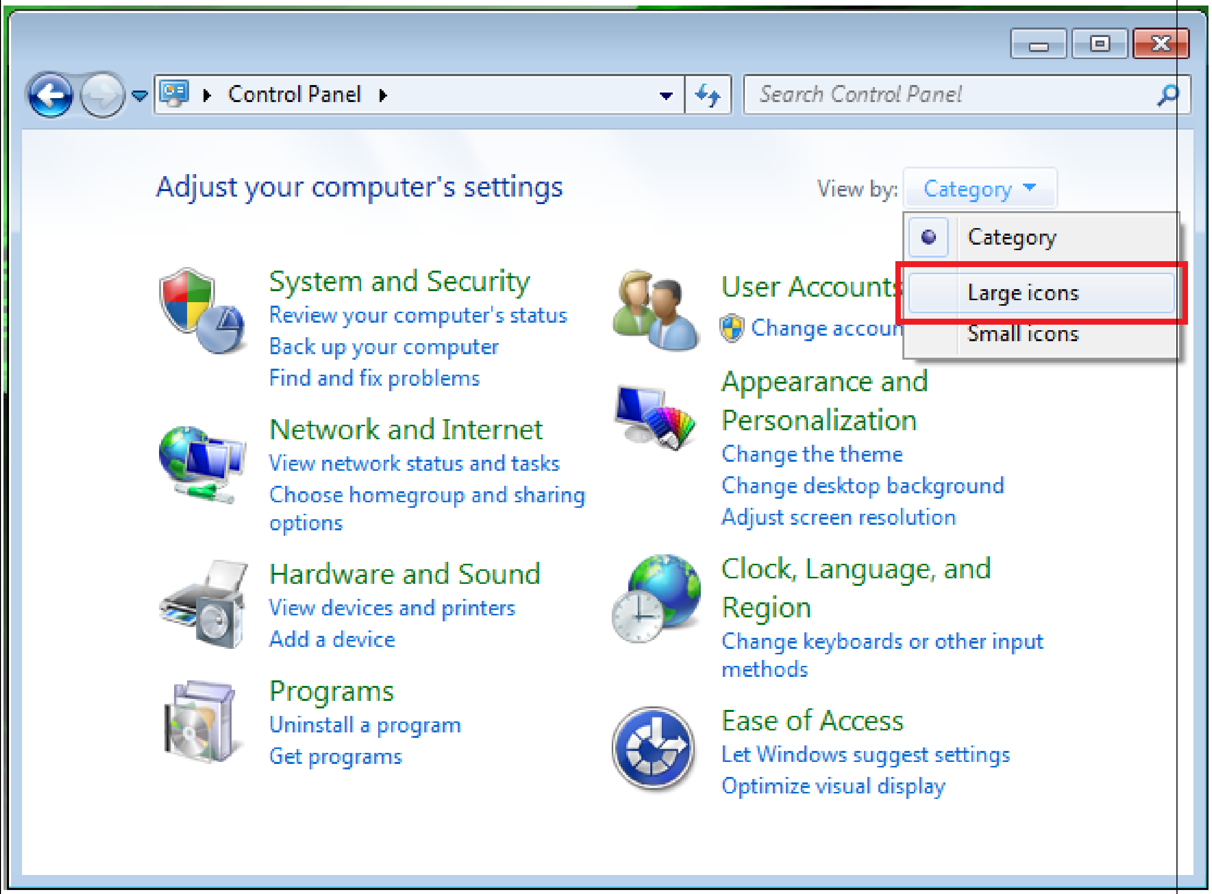
Study on Connecting the Computers into a Local Area Network.

Use the following steps to setup your LAN connection in Windows 7, when your computer is the device plugged in to your wireless radio or POE.

Note: - You will need your static IP settings and DNS settings before you continue.

First, go to the start menu and into Control Panel.

2. Once in Control Panel, if you are in Category view, click on the View by: drop down menu, and select Large icons.



3. Now that you are in Large icons view, find and double click on Network and Sharing Center.

4. Now that you are in Network and Sharing Center, click on the "Change adapter settings" link on the left.

5. In the Change adapters settings screen, find and right click on your Local Area Connection, and left click on Properties.

6. In the Local Area Connection Properties window, highlight Internet Protocol Version 4 (TCP/IPv4), and click on the Properties button.

7. In the Internet Protocol Version 4 (TCP/IPv4) Properties window, you will need to select "Use the following IP address" and "Use the following DNS server addresses:"

Enter in the corresponding settings per their section.

8. Once the settings have been entered in, click ok on the Internet Protocol Version 4 (TCP/IPv4) Properties window, then click on close on the Local Area Connections properties window.

At this point the Local Area Connection should say connected. See if you can browse to a web site at this time. If you are able to, your setup is complete.

Experiment No :3

3.Implementation of Data Link Framing method - Character Count.

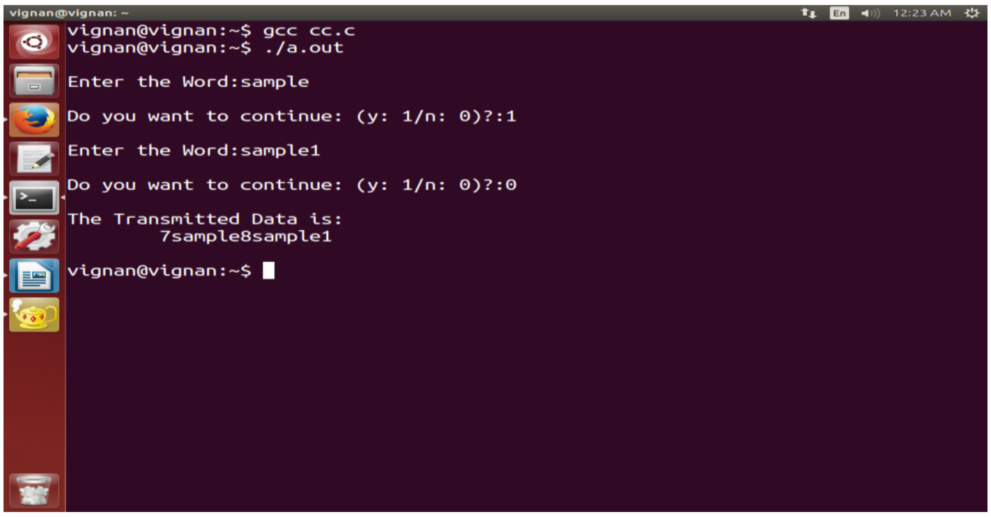
Aim: To implementation of Data Link Framing method by Character Count.

Problem Description: This method uses a field in the header to specify the number of characters in the frame. When the data link layer at the destination sees the character count,it knows how many characters follow, and hence where the end of the frame is. The disadvantage is that if the count is garbled by a transmission error, the destination will lose synchronization and will be unable to locate the start of the next frame. So, this method is rarely used.

Program Source Code:

```
#include<stdio.h>

#include<string.h>

char input[10][20];

intget_input();

voidmake_frames(int);

intcount_chars(int s);

int main()

{

intno_of_words=get_input();

make_frames(no_of_words);

return 0;

}

intget_input()

{

int answer;

inti=0;

do{

printf("\nEnter the Word:");
scanf("%s",input[i]);
fflush(stdin);
printf("\nDo you want to continue: (y: 1/n: 0)?:");
scanf("%d",&answer);
i++;
}while(answer!=0);
returni;
}
voidmake_frames(intno_of_words){
inti=0;
printf("\nThe Transmitted Data is:\n\t");
for(;i<no_of_words;i++)
printf("%d%s",(count_chars(i)+1),input[i]);
printf("\n\n");
}
intcount_chars(int index)
{
inti=0;
while(input[index][i]!='\0')
i++;
returni;
}
```

OUTPUT:



```
vignan@vignan:~$ gcc cc.c
vignan@vignan:~$ ./a.out

Enter the Word:sample

Do you want to continue: (y: 1/n: 0)?:1

Enter the Word:sample1

Do you want to continue: (y: 1/n: 0)?:0

The Transmitted Data is:
        7sample8sample1

vignan@vignan:~$
```

Experiment No :4

4.Implementation of Data link framing method - Bit stuffing and Destuffing.

Aim: To implementation of Data link framing method by Bit stuffing and Destuffing.

Problem Description:

Bit stuffing is the process of inserting non information bits into data to break up bit pattern to affect the synchronous transmission of information.

Bit stuffing is commonly used to bring bit stream upto a common transmission rate or to fill frames. Bit stuffing is also used for run-length limited coding.

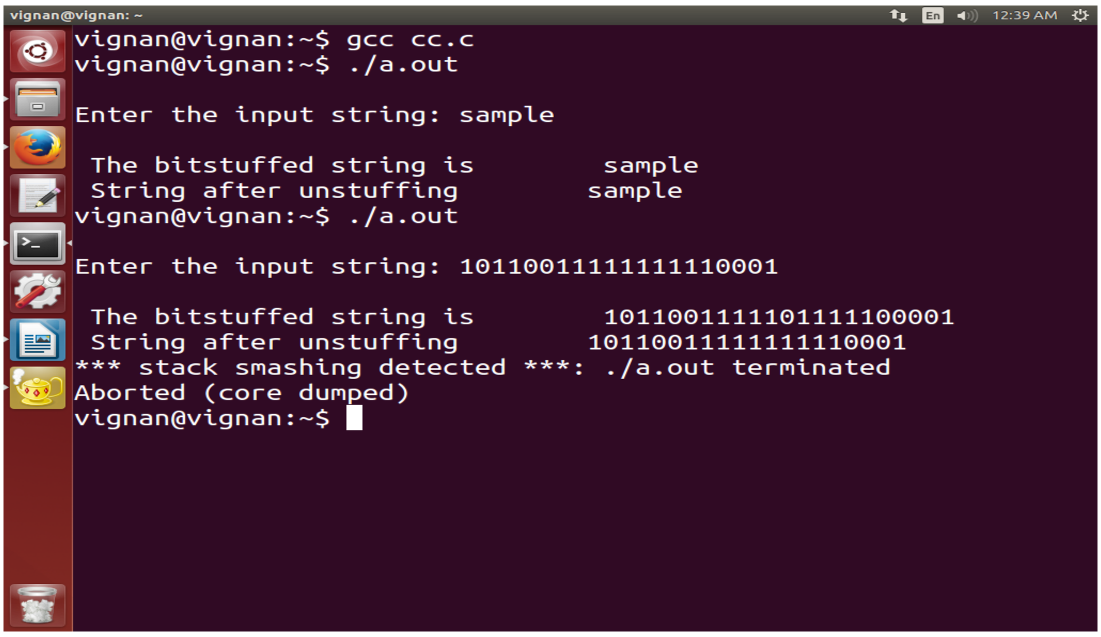
Program Source Code:

```
#include<stdio.h>

#include<string.h>

int main()

{

char a[10];

int count=0,i=0,j,l,k=0;

printf("\nEnter the input string:\t");

scanf("%s",a);

        l=strlen(a);

for(i=0;i<l;i++)

if(a[i]=='1')

        {

                k=i;

count=0;

while(a[k]=='1')

                {
```

```
count+=1;
k++;
if(count==5)
                        {
for(j=l+1;j>k;j--)
                                {
a[j]=a[j-1];
                                }
a[k]='0';
l++;
break;
                        }
i=k;
                }
        }
printf("\n The bitstuffed string is \t %s",a);
        //getch();
        k=0;
for(i=0;i<=l;i++)
        {
if(a[i]=='1')
                {
                        k=i;
count=0;
while(a[k]=='1')
                        {
```

```
count+=1;
k++;
if(count==5)
                                {
for(j=k;j<l+1;j++)
                                        {
a[j]=a[j+1];
                                        }
l--;
a[l+1]='0';
break;
                                }
                        }
i=k-1;
                }
        }
printf("\n String after unstuffing \t%s\n",a);
return 0;
}
```

OUTPUT:



```
vignan@vignan:~$ gcc cc.c
vignan@vignan:~$ ./a.out

Enter the input string: sample

 The bitstuffed string is        sample
 String after unstuffing        sample
vignan@vignan:~$ ./a.out

Enter the input string: 10110011111111110001

 The bitstuffed string is        1011001111101111100001
 String after unstuffing        10110011111111110001
*** stack smashing detected ***: ./a.out terminated
Aborted (core dumped)
vignan@vignan:~$ 
```

Experiment No :5

5.Implementation of Error detection method - even and odd parity.

Aim: To implementation of Error detection method - even and odd parity.

Problem Description:

In the case of even parity, for a given set of bits, the occurrences of bits whose value is 1 is counted. If that count is odd, the parity bit value is set to 1, making the total count of occurrences of 1's in the whole set(including the parity bit) an even number. If the count of 1's in a given set of bits is already even, the parity bit's value remains 0.

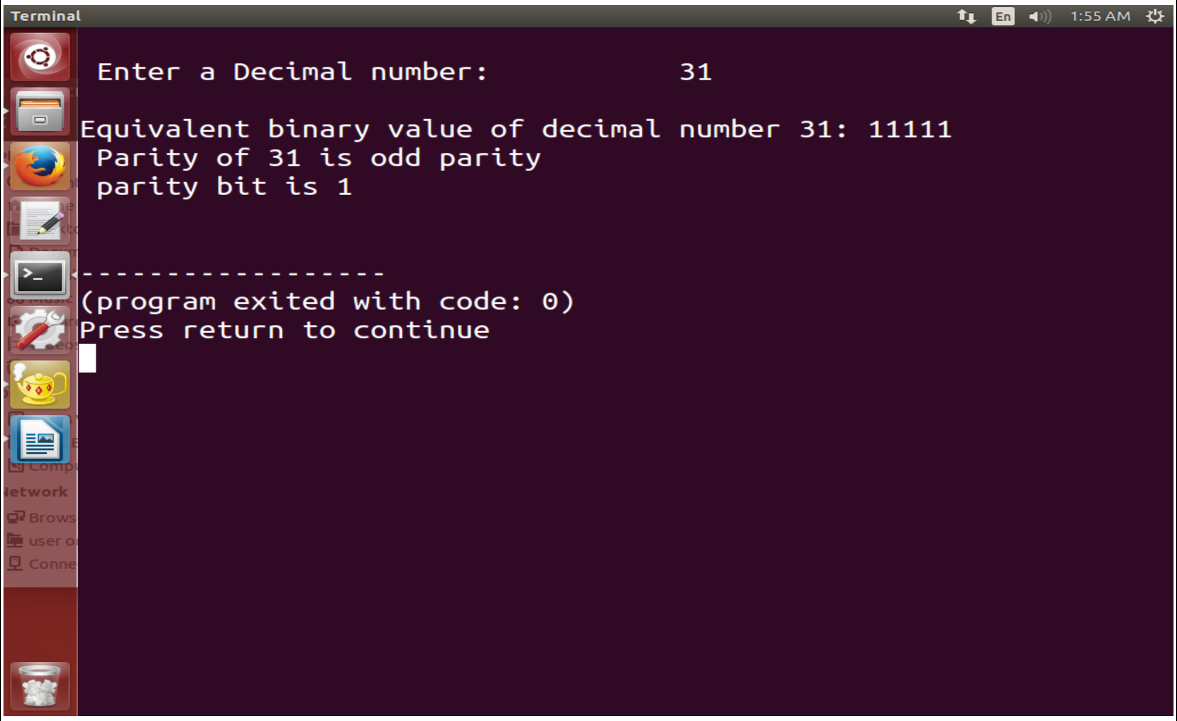
In the case of odd parity, the situation is reversed. For a given set of bits, if the count of bits with a value of 1 is even, the parity bit value is set to 1 making the total count of 1's in the whole set(including the parity bit) an odd number. If the count of bits with a value of 1 is odd, the count is already odd so the parity bit's value remains 0.

Program Source Code:

```
# include <stdio.h>

int main()

{

unsignedintn,count=0;

longint quotient;

intbinaryNumber[100],i=1,j;

printf ("\n Enter a Decimal number:\t");

scanf("%d",&n);

quotient = n;

while(quotient!=0)

    {

binaryNumber[i]= quotient % 2;

quotient = quotient / 2;

if(binaryNumber[i]==1)
```

```
              {
count++;
              }
i++;
    }
printf("\nEquivalent binary value of decimal number %d: ",n);
for(j = i -1 ;j> 0;j--)
printf("%d",binaryNumber[j]);
if(count%2==1)
printf("\n Parity of %d is odd parity \n parity bit is 1\n",n);
else
printf("\n Parity of %d is even parity\n parity bit is 0\n",n);
return 0;
}
```

OUTPUT:



```
 Enter a Decimal number:           31

Equivalent binary value of decimal number 31: 11111
 Parity of 31 is odd parity
 parity bit is 1


------------------
(program exited with code: 0)
Press return to continue
```

Experiment No :6

6.Implementation of Error detection method - CRC Polynomials.

Aim: To implementation of Error detection method - CRC Polynomials.

Problem Description:

A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short check value attached, based on the remainder of a polynomial division of their contents.

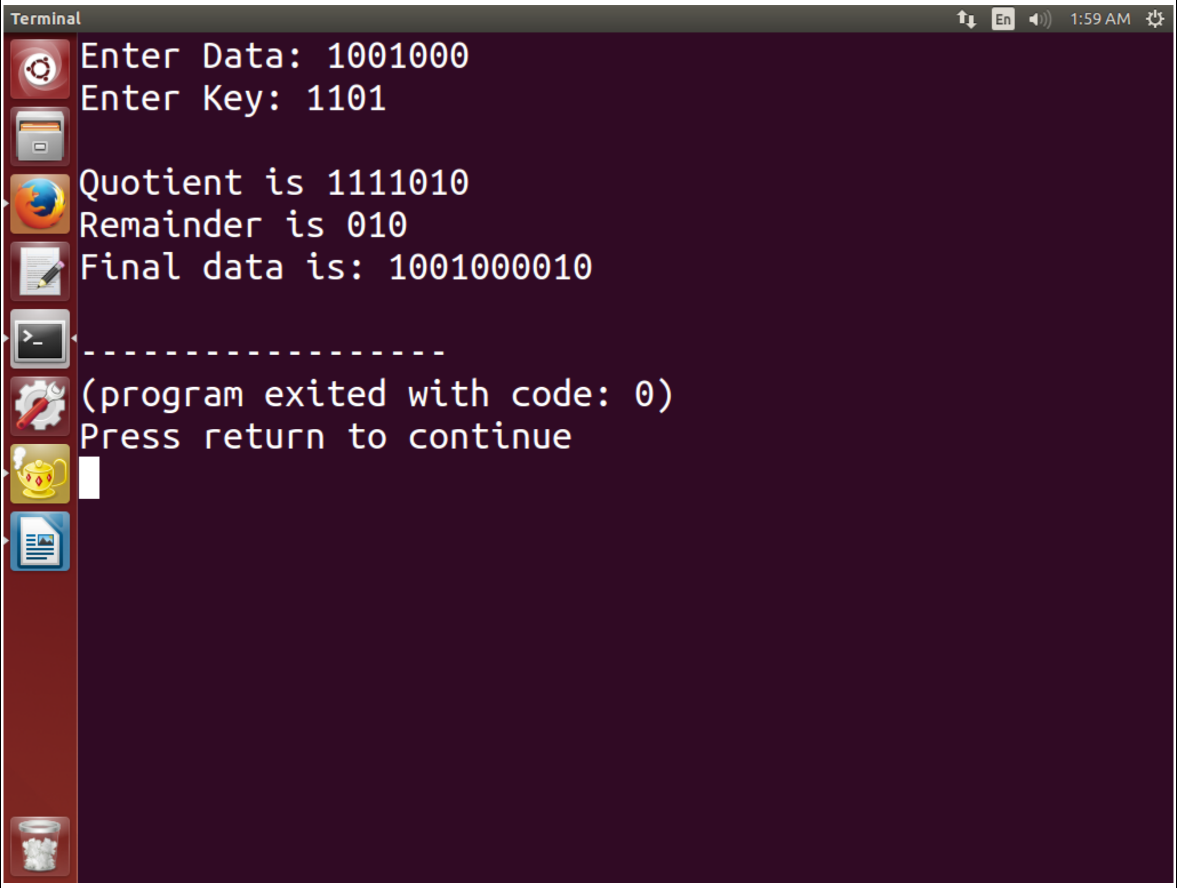
Program Source Code:

```
#include <stdio.h>

#include <string.h>

void main() {

        inti,j,keylen,msglen;

        char input[100], key[30],temp[30],quot[100],rem[30],key1[30];

        printf("Enter Data: ");

        gets(input);

        printf("Enter Key: ");

        gets(key);

        keylen=strlen(key);

        msglen=strlen(input);

        strcpy(key1,key);

        for (i=0;i<keylen-1;i++) {

                input[msglen+i]='0';

        }

        for (i=0;i<keylen;i++)

        temp[i]=input[i];

        for (i=0;i<msglen;i++) {
```

```
        quot[i]=temp[0];
        if(quot[i]=='0')
        for (j=0;j<keylen;j++)
        key[j]='0'; else
        for (j=0;j<keylen;j++)
        key[j]=key1[j];
        for (j=keylen-1;j>0;j--) {
            if(temp[j]==key[j])
            rem[j-1]='0'; else
            rem[j-1]='1';
        }
        rem[keylen-1]=input[i+keylen];
        strcpy(temp,rem);
    }
    strcpy(rem,temp);
    printf("\nQuotient is ");
    for (i=0;i<msglen;i++)
    printf("%c",quot[i]);
    printf("\nRemainder is ");
    for (i=0;i<keylen-1;i++)
    printf("%c",rem[i]);
    printf("\nFinal data is: ");
    for (i=0;i<msglen;i++)
    printf("%c",input[i]);
    for (i=0;i<keylen-1;i++)
    printf("%c",rem[i]);
```

}

OUTPUT:



```
Enter Data: 1001000
Enter Key: 1101

Quotient is 1111010
Remainder is 010
Final data is: 1001000010

------------------
(program exited with code: 0)
Press return to continue
```

Experiment No :7

7.Implementation of Data Link protocols - Unrestricted simplex protocol

Aim: To implementation of Data Link protocols - Unrestricted simplex protocol

Problem Description:

In order to appreciate the step by step development of efficient and complex protocols such as SDLC, HDLC etc., we will begin with a simple but unrealistic protocol. In this protocol:

a. Data are transmitted in one direction only

b.The transmitting (Tx) and receiving (Rx) hosts are always ready

c.Processing time can be ignored

d.Infinite buffer space is available

e.No errors occur; i.e. no damaged frames and no lost frames (perfect channel)

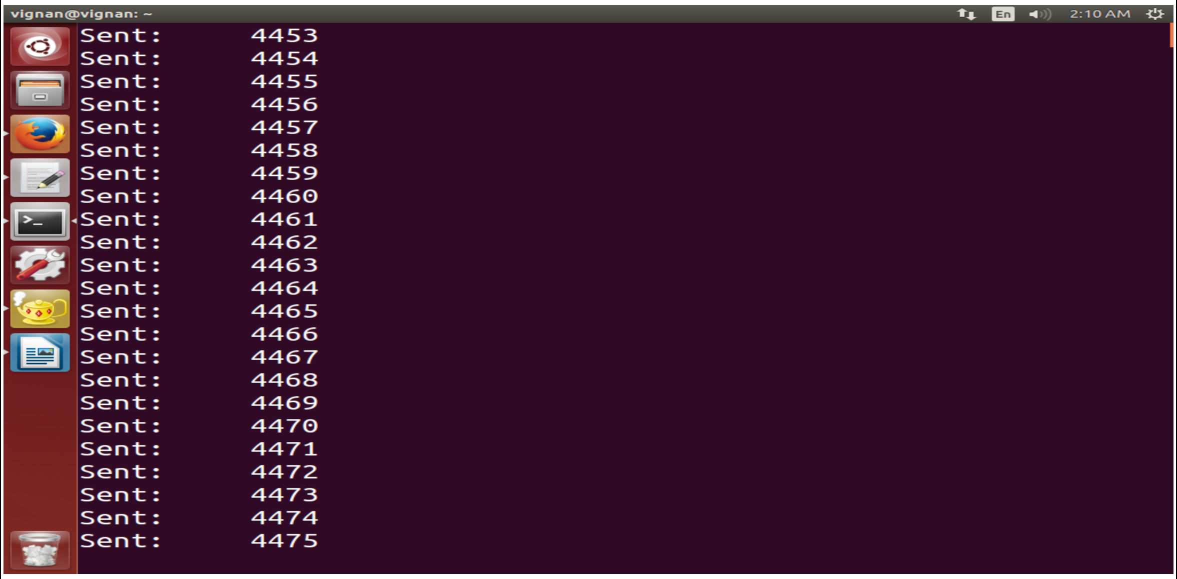
Program Source Code:

```
// An Unrestricted Simplex Protocol.
class Q
{
int n;
synchronizedint get()
{
System.out.println("Received: " + n);
return n;
}
synchronized void put(int n)
{
this.n = n;
System.out.println("Sent:     " + n);
```

```
}
}
class Sender implements Runnable
{
Q q;
Sender(Q q) ;
{
this.q = q;
new Thread(this, "Sender").start();
}
public void run()
{
inti = 0;
while(true)
{
q.put(i++);
}
}
}
class Receiver implements Runnable
{
Q q;
Receiver(Q q)
{
this.q = q;
new Thread(this, "Receiver").start();
```

```
}
public void run()
{
while(true)
{
q.get();
}
}
}
class Unrestricted
{
public static void main(String args[])
{
Q q = new Q();
new Sender(q);
new Receiver(q);
System.out.println("Press Control-C to stop.");
}
}
```

OUTPUT:



```
Sent:        4453
Sent:        4454
Sent:        4455
Sent:        4456
Sent:        4457
Sent:        4458
Sent:        4459
Sent:        4460
Sent:        4461
Sent:        4462
Sent:        4463
Sent:        4464
Sent:        4465
Sent:        4466
Sent:        4467
Sent:        4468
Sent:        4469
Sent:        4470
Sent:        4471
Sent:        4472
Sent:        4473
Sent:        4474
Sent:        4475
```

Experiment No :8

8.Implementation of data link protocols - Stop and Wait protocol.

Aim: To implementation of data link protocols - Stop and Wait protocol.

Problem Description:

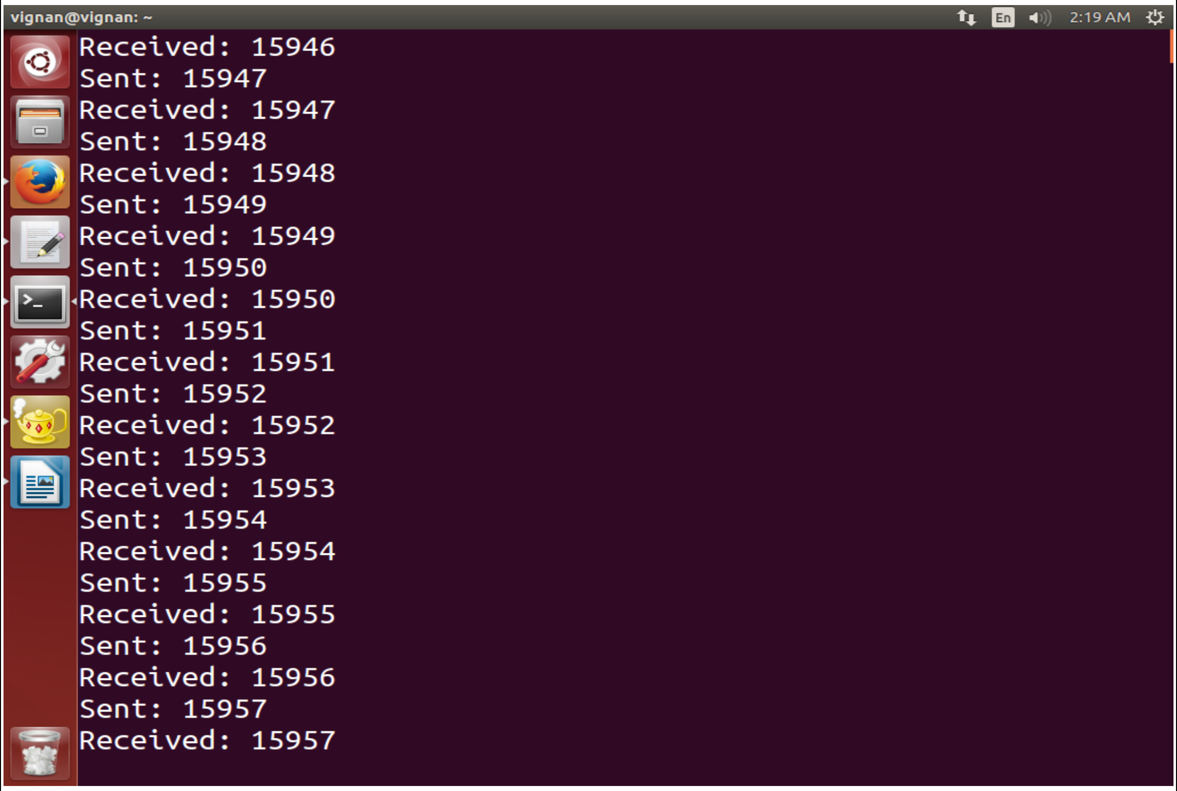
A stop-and-wait ARQ sender sends one frame at a time; it is a special case of the general sliding window protocol with both transmit and receive window sizes equal to 1 and more than one respectively . After sending each frame, the sender doesn't send any further frames until it receives an acknowledgement (ACK) signal. After receiving a good frame, the receiver sends an ACK. If the ACK does not reach the sender before a certain time, known as the timeout, the sender sends the same frame again. Timer is set after each frame transmission. The above behavior is the simplest Stop-and-Wait implementation.

Program Source Code:

```
// A Restricted Simplex Protocol.
class Q
{
int n;
booleanvalueSet = false;
synchronizedint get()
        {
if(!valueSet)
try
        {
wait();
        }
catch(InterruptedException e)
 {
System.out.println("InterruptedException caught");
}
System.out.println("Received: " + n);
valueSet = false;
notify();
return n;
        }
synchronized void put(int n)
        {
if(valueSet)
try
                {
wait();
                }
catch(InterruptedException e)
                {
System.out.println("InterruptedException caught");
                }
this.n = n;
valueSet = true;
System.out.println("Sent: " + n);
notify();
        }
}

class Sender implements Runnable
{
        Q q;
Sender(Q q)
        {
this.q = q;
new Thread(this, "Sender").start();
        }
public void run()
        {
inti = 0;
while(true)
                {
q.put(i++);
                }
        }
}

class Receiver implements Runnable
{
        Q q;
Receiver(Q q)
        {
this.q = q;
new Thread(this, "Receiver").start();
        }
public void run()
        {
```

```
while(true)
                {
q.get();
                }
        }
}

class Restricted
{
public static void main(String args[])
        {
                Q q = new Q();
new Sender(q);
new Receiver(q);
System.out.println("Press Control-C to stop.");
        }
}
```

OUTPUT:



```
Received: 15946
Sent: 15947
Received: 15947
Sent: 15948
Received: 15948
Sent: 15949
Received: 15949
Sent: 15950
Received: 15950
Sent: 15951
Received: 15951
Sent: 15952
Received: 15952
Sent: 15953
Received: 15953
Sent: 15954
Received: 15954
Sent: 15955
Received: 15955
Sent: 15956
Received: 15956
Sent: 15957
Received: 15957
```

Experiment No:10

10. Study of Network IP Addressing

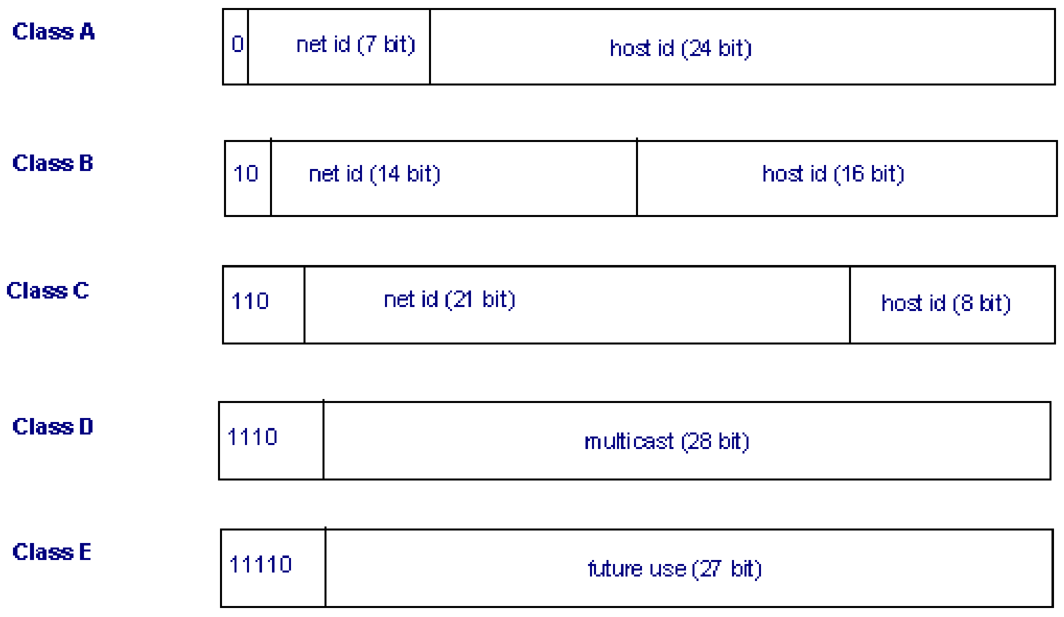
Aim: A detailed study on Network IP Addressing

problem Description:

IP Address Classification, IPv4

IP, Short for **Internet Protocol**, is an address of a computer or other network device on a network using IP or TCP/IP These addresses are similar to addresses used on houses and help data reach its appropriate destination on a network.

There are five classes of available IP ranges: Class A, Class B, Class C, Class D and Class E, while only A, B and C are commonly used. Each class allows for a range of valid IP addresses. Below is a listing of these addresses.



Class	Address Range	Supports
Class A	1.0.0.0 to 127.255.255.255	Supports 16 million hosts on each of 128 networks.
Class B	128.0.0.0 to 191.255.255.255	Supports 64K hosts on each of 16,384 networks.
Class C	192.0.0.0 to 223.255.255.255	Supports 256 hosts on each of 2 million networks.
Class D	224.0.0.0 to 239.255.255.255	Reserved for multicast groups.
Class E	240.0.0.0 to 254.255.255.255	Reserved for future use.

The Role of Subnet Masks

Subnet masks designate which bits of an IP address represent the network portion and which bits represent the host portion. Default subnet masks are used with Class A, Class B, and Class C IP addresses, as follows:

Class A: 255.0.0.0

Class B: 255.255.0.0

Class C: 255.255.255.0

The Class A subnet mask tells you the first 8 bits of the IP address represent the network portion of the address. The remaining 24 bits represent the host portion of the address. Let's say a host has the IP address 10.25.65.32. Using the default subnet mask, the network address would be 10.0.0.0. The host component of the address would be 25.65.32.

The Class B subnet mask tells you the first 16 bits of the IP address represent the network portion of the address. The remaining 16 bits represent the host address within the network. If a host has the IP address 172.20.33.33, the network portion of the address would be 172.20.0.0. The host component would be 33.33

The Class C subnet mask tells you the first 24 bits of the IP address represent the network portion of the address. The remaining 8 bits represent the host address within the network. If a host has the IP address 192.168.2.3, the network portion of the address would be 192.168.2.0. The host component would be 3.

The ANDing process

When a source host attempts to communicate with a destination host, the source host uses its subnet mask to determine whether the destination host is on the local network or a remote network. This is known as the *ANDing process*.

The AND function has the following properties:

- If the two compared values are both 1, the result is 1.
- If one of the values is 0 and the other is 1, the result is 0.
- If both of the compared values are 0, the result is 0.

The source and destination IP addresses are compared to the source's subnet mask using the ANDing process. An AND result is created for each of the addresses. If the result is the same, the hosts are on the same network. If the result is different, the destination host is on a remote network.

What Is a Default Gateway?

You will sometimes see the term **default gateway** on network configuration screens in Microsoft Windows. In computer networking, a default gateway is the device that passes traffic from the local subnet to devices on other subnets. The default gateway often connects a local network to the Internet, although internal gateways for local networks also exist.

Internet default gateways are typically one of two types:

- On home or small business networks with a broadband router to share the Internet connection, the home router serves as the default gateway.
- On home or small business networks without a router, such as for residences with dialup Internet access, a router at the Internet Service Provider location serves as the default gateway.
-

Default network gateways can also be configured using an ordinary computer instead of a router. These gateways use two network adapters, one connected to the local subnet and one to the outside network. Either routers or gateway computers can be used to network local subnets such as those in larger businesses.

What is Preferred DNS Server?

Preferred and Alternate DNS server - This is a server that translates domain names into IP addresses. Because domain names are alphabetic, they're easier to remember. The Internet however, is really based on IP addresses. Every time you use a domain name, therefore, a DNS service must translate the name into the corresponding IP address. For example, the domain name www.example.com might translate to 198.105.232.4.

Experiment No:11

11.Study of sockets in detail.

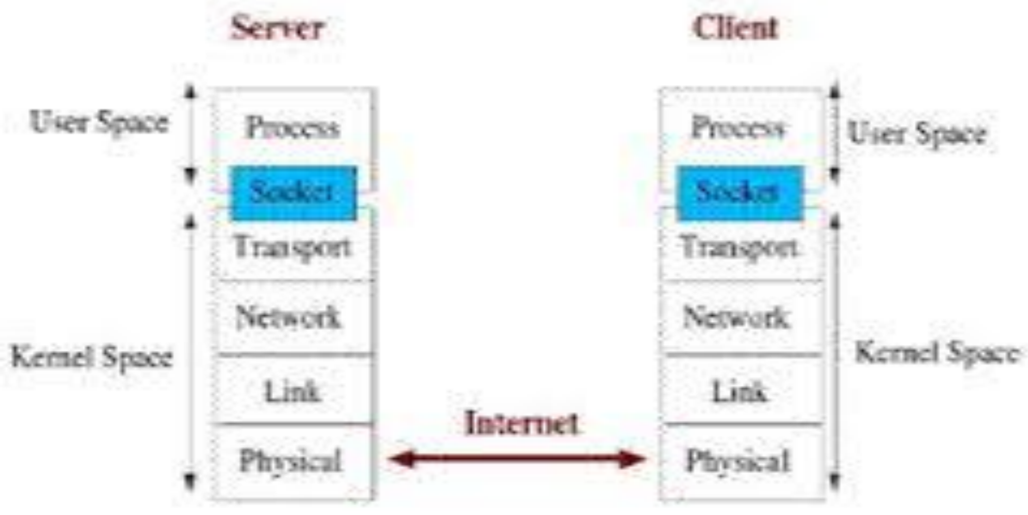
Aim: A detailed study on Sockets
problem Description:

Socket Programming

What is a socket?
Socket: An interface between an application process and transport layer The application process can send/receive messages to/from another application process (local or remote)via a socket In Unix jargon, a socket is a file descriptor – an integer associated with an open file

Types of Sockets:
- Internet Sockets,
- unix sockets,
- X.25 sockets etc

Internet sockets characterized by IP Address (4 bytes) and port number (2 bytes)



Types of Internet Sockets

Stream Sockets (SOCK_STREAM) :

Connection oriented Rely on TCP to provide reliable two-way connected communication

Datagram Sockets (SOCK_DGRAM) :

Rely on UDP Connection is unreliable

Two types of "Byte ordering"

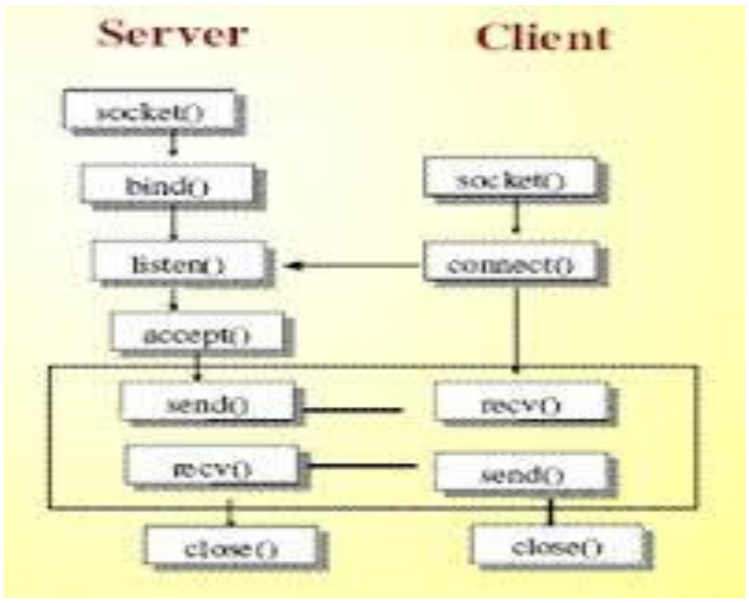
Network Byte Order: High-order byte of the number is stored in memory at the lowest address

Host Byte Order: Low-order byte of the number is stored in memory at the lowest address Network stack (TCP/IP) expects Network Byte Order

Conversions:

- htons() - Host to Network Short
- htonl() – Host to Network Long
- ntohs() Network to Host Short
- ntohl() - Network to Host Long Conne

Connection oriented protocols



socket() - int socket(int domain, int type, int protocol);

domain should be set to AF _ INET

type can be SOCK_STREAM or SOCK_DGRAM

set protocol to 0 to have socket choose the correct protocol based on type socket() returns a socket descriptor for use in later system calls or -1 on error

socket structures

structsockaddr: Holds socket address information for many types of sockets

structsockaddr _ in: A parallel structure that makes it easy to reference elements of the socket address

Dealing with IP Addresses

intinet _ aton(const char *cp, structin_addr *inp);

inet _ aton() gives non-zero on success and zero on failure

To convert binary IP to string: inet_noa()
printf("%s",inet_ntoa(my_addr.sin_addr));

bind() - what port am I on?

Used to associate a socket with a port on the local machine The port number is used by the kernel to match an incoming packet to a process

int bind(intsockfd, structsockaddr *my_addr, intaddrlen)

connect() - Hello!

Connects to a remote host

int connect(intsockfd, structsockaddr *serv_addr, intaddrlen)

listen() - Call me please!

Waits for incoming connections

int listen(intsockfd, int backlog);

accept() - Thank you for calling !

accept() gets the pending connection on the port you are listen()ing on

int accept(intsockfd, void *addr, int *addrlen);

send() and recv() - Let's talk!

The two functions are for communicating over stream sockets or connected datagram sockets.

- int send(intsockfd, const void *msg, intlen, int flags);
- intrecv(intsockfd, void *buf, intlen, int flags);

sendto() and recvfrom() –

intsendto(intsockfd, const void *msg, intlen, int flags, conststructsockaddr *to, inttolen);

intrecvfrom(intsockfd, void *buf, intlen, int flags, structsockaddr *from, int *fromlen);

close() - Bye Bye!

int close(intsockfd);

Miscellaneous Routines

intgetpeername(intsockfd, structsockaddr *addr, int *addrlen);

Will tell who is at the other end of a connected stream socket and store that info in address

intgethostname(char *hostname, size_t size);

Will get the name of the computer your program is running on and store that info in hostname

Miscellaneous structures

```
structhostent *gethostbyname(const char *name);
```

Summary

Sockets help application process to communicate with each other using standard Unix file descriptors   Many routines exist to help ease the process of communication