**DELHI TECHNOLOGICAL UNIVERSITY**



**COMPUTER NETWORKS (CO306) LAB**

**VI Semester**

**Submitted by: Submitted to:**

Piyush Khanna Mr. Abebaw Alem

B. Tech. Computer Engineering (COE)

2K17/CO/224

|  |  |  |  |
| --- | --- | --- | --- |
| **INDEX** | | | |
| **S.NO.** | **PROGRAM TITLE** | **DATE** | **SIGNATURE** |
| 1. | Implement framing techniques – Bit Stuffing and Bit Unstuffing. |  |  |
| 2. | Write a program to implement error detection using CRC. |  |  |
| 3. | Write a program to implement stop wait protocol. |  |  |
| 4. | Write a program to implement sliding window protocol. |  |  |
| 5. | Write a program to show net ID, Host ID, and network class of the address. |  |  |
| 6. | Write a program to implement distance vector routing algorithm. |  |  |
| 7. | Write a program to implement link state routing algorithm. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

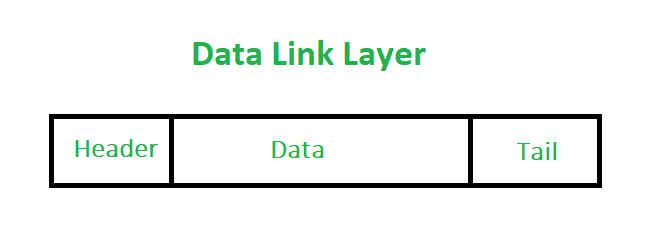
**Experiment 1**

**Aim**: - Implement framing techniques – Bit Stuffing and Bit Unstuffing.

**Theory**

Frames are the units of digital transmission particularly in computer networks and telecommunications. Frames are comparable to the packets of energy called photons in case of light energy. Frame is continuously used in Time Division Multiplexing process.

Framing is a point-to-point connection between two computers or devices consists of a wire in which data is transmitted as a stream of bits. However, these bits must be framed into discernible blocks of information. Framing is a function of the data link layer. It provides a way for a sender to transmit a set of bits that are meaningful to the receiver.



At data link layer, it extracts message from sender and provide it to receiver by providing sender’s and receiver’s address. The advantage of using frames is that data is broken up into recoverable chunks that can easily be checked for corruption.

**Problems in Framing** –

* Detecting start of the frame: When a frame is transmitted, every station must be able to detect it. Station detect frames by looking out for special sequence of bits that marks the beginning of the frame i.e. SFD (Starting Frame Delimeter).
* How do station detect a frame: Every station listen to link for SFD pattern through a sequential circuit. If SFD is detected, sequential circuit alerts station. Station checks destination address to accept or reject frame.
* Detecting end of frame: When to stop reading the frame.

**Bit Stuffing**: Let ED = 0111111 and if data = 0111111

* Sender stuffs a bit to break the pattern i.e. here appends a 0 in data = 01111101.
* Receiver receives the frame.
* If data contains 011101, receiver removes the 0 and reads the data.

**Code**

#include<iostream>

using namespace std;

string staffing(string str)

{

string ans;

int count=0;

for(int i=0;i<str.size();i++)

{

ans+=str[i];

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

count++;

else

count=0;

if(count==5)

{

count=0;

ans+='0';

}

}

return ans;

}

string unstaffing(string str)

{

int j=0,count=0;

for(int i=0;i<str.size();i++,j++)

{

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

count++;

else

count=0;

str[j]=str[i];

if(count==5)

{

count=0;

i++;

}

}

for(;j<str.size();j++)

str[j]='\0';

return str;

}

int main()

{

string str;

cout<<"Enter the data : ";

cin>>str;

string ans=staffing(str);

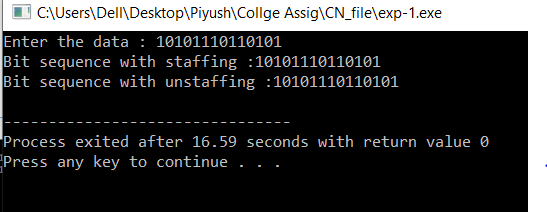
cout<<"Bit sequence with staffing :"<<ans<<endl;

cout<<"Bit sequence with unstaffing :"<<unstaffing(ans)<<endl;

return 0;

}

**Output**



**Learning & Finding**

In this experiment, I learnt about data link layer and framing techniques – staffing and unstaffing.

**Experiment 2**

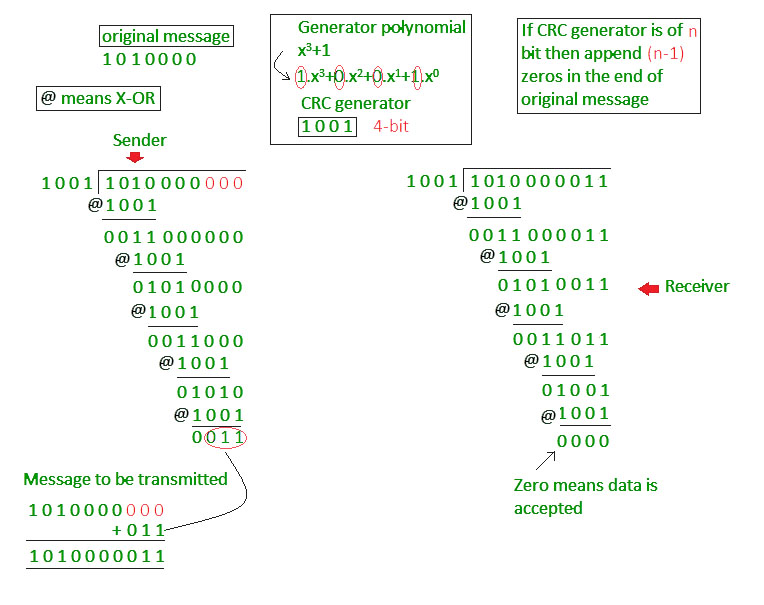
**Aim**: - Write a program to implement error detection using CRC.

**Theory**

In CRC (Cyclic Redundant Check), a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.

At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.

A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.



**Code**

#include<iostream>

#include<bits/stdc++.h>

using namespace std;

int returnAlphabet(char c)

{

if(c=='x'||c=='X')

return 0;

else if(c=='+')

return 1;

else if(c-'0'>=0&&c-'0'<=9)

return 2;

else return 3;

}

int main()

{

int arr[3][4]={{1,3,3,3},

{3,3,2,3},

{3,0,2,3}};

string str;

cout<<"Enter polynomial : ";

cin>>str;//input binary polynomail

int cur=0;//current state

vector<int> deg;//store degree of polynomial

int a=0;//store current number;

int maxdeg=-1;// Store maximum degree

for(int i=0;i<str.size();i++)

{

cur=arr[cur][returnAlphabet(str[i])];

if(cur==3){

cout<<"Error : write polynomial in appropriate form (e.g. x3+x1+x0)"<<endl;

return 0;

}

if(cur==2)

{

a=a\*10+(str[i]-'0');

}

else if(cur==0)

{

if(maxdeg<a)

maxdeg=a;

deg.push\_back(a);

a=0;

}

}

if(cur!=2)

{

cout<<"Error : write polynomial in appropriate form (e.g. x3+x1+x0)"<<endl;

return 0;

}

else{

deg.push\_back(a);

}

bool divisor[maxdeg+1]={0};

for(int i=0;i<deg.size();i++)

{

divisor[maxdeg-deg[i]]=1;

}

cout<<"Enter data word : ";

cin>>str;//Now str store data word

for(int i=0;i<maxdeg;i++)

str+='0';

cout<<"Divisor : ";

for(int i=0;i<=maxdeg;i++)

cout<<divisor[i];

cout<<endl;

bool dataword[str.size()];

bool tempdata[str.size()];

for(int i=0;i<str.size();i++)

{

if(str[i]=='0')

dataword[i]=0;

else if(str[i]=='1')

dataword[i]=1;

else {

cout<<"Error : Enter correct data word"<<endl;

return 0;

}

tempdata[i]=dataword[i];

}

int ptr=0;

while(1)

{

if(ptr+maxdeg>=str.size())

{

break;

}

int tempptr=ptr;

for(int i=0;i<maxdeg+1;i++)

{

tempdata[tempptr+i]=tempdata[tempptr+i]^divisor[i];

if(tempptr==ptr&&tempdata[tempptr+i]==1)

{

ptr=tempptr+i;

}

}

}

bool r[maxdeg];

cout<<"Redundant bits : ";

for(int i=0;i<maxdeg;i++)

{

r[maxdeg-1-i]=tempdata[str.size()-1-i];

}

for(int i=0;i<maxdeg;i++)

cout<<r[i];

cout<<endl;

cout<<"Code word : ";

for(int i=0;i<str.size()-maxdeg;i++)

cout<<dataword[i];

for(int i=0;i<maxdeg;i++)

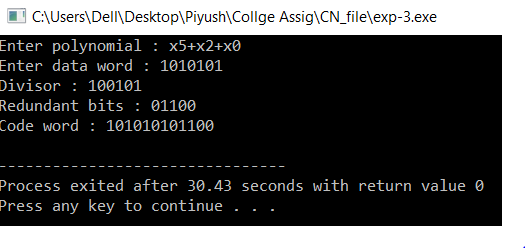
cout<<r[i];

cout<<endl;

return 0;

}

**Output**



**Learning & Finding**

In this experiment, I learnt about CRC and its implementation.

**Experiment 3**

**Aim**: - Write a program to implement stop wait protocol.

**Theory**

**Sender**:

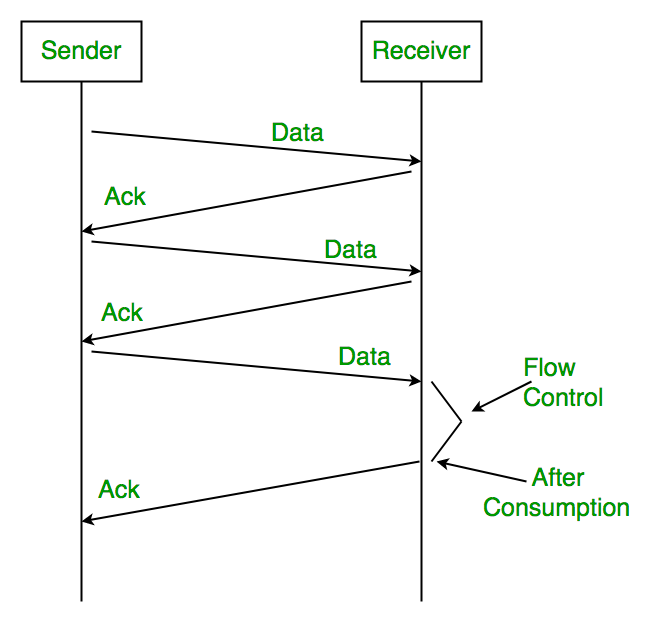
Rule 1. Send one data packet at a time.

Rule 2. Send next packet only after receiving acknowledgement for previous.

**Receiver**:

Rule 1. Send acknowledgement after receiving and consuming of data packet.

Rule 2. After consuming packet acknowledgement need to be sent (Flow Control)



**Code**

#include<bits/stdc++.h>

using namespace std;

void delay()

{

for(int i=0;i<10000;i++)

for(int j=0;j<10000+rand()%10000;j++){}

}

void delay2()

{

for(int i=0;i<10000;i++)

for(int j=0;j<100000+rand()%10000;j++){}

}

int main()

{

srand(time(NULL));

cout<<"Enter no of frames : ";

int n;

cin>>n;

int frame[n];

cout<<"Frame to be sent : ";

for(int i=0;i<n;i++)

{

frame[i] = rand()%100;

cout<<frame[i]<<" ";

}

cout<<endl;

cout<<"\tSender\t\t\t\tReceiver"<<endl;

for(int i=0;i<n;i++)

{

cout<<"\tSent frame : "<<frame[i]<<endl;

delay();

cout<<"\t\t\t\t\t"<<frame[i]<<" frame received"<<endl;

if(i>3)

{

cout<<"\t\t\t\t\tWait for processing..."<<endl;

delay();

cout<<"\t\t\t\t\tBuffer is available"<<endl;

}

cout<<"\t\t\t\t\tSent acknowledgment frame"<<endl;

delay();

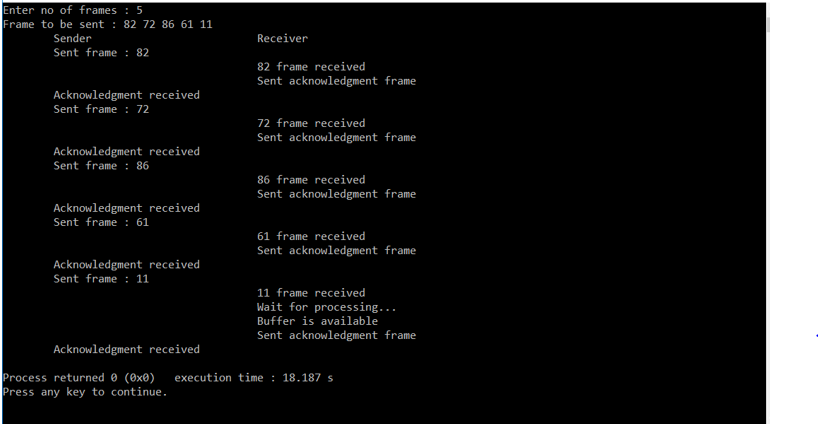
cout<<"\tAcknowledgment received"<<endl;

}

return 0;

}

**Output**



**Learning & Finding**

In this experiment, we learnt about stop and wait protocol and its implementation.

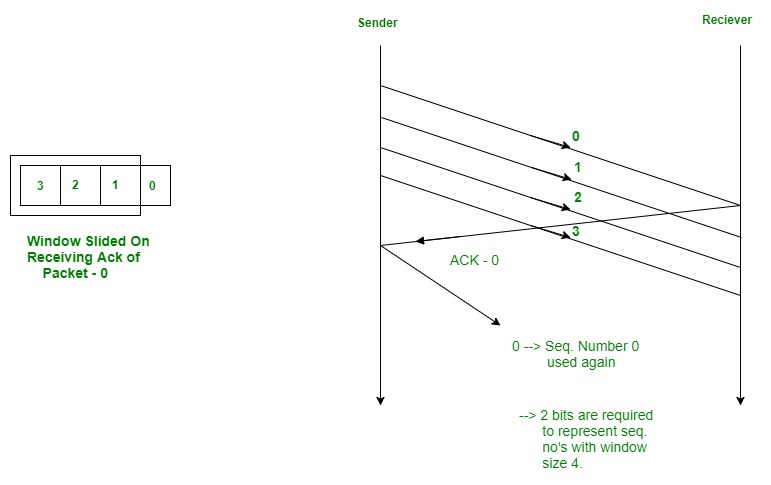
**Experiment 4**

**Aim**: - Write a program to implement sliding window protocol.

**Theory**

Sliding window protocols are data link layer protocols for reliable and sequential delivery of data frames. The sliding window is also used in Transmission Control Protocol.

In this protocol, multiple frames can be sent by a sender at a time before receiving an acknowledgment from the receiver. The term sliding window refers to the imaginary boxes to hold frames. Sliding window method is also known as windowing.



**Code**

#include<bits/stdc++.h>

using namespace std;

int main()

{

int n,w;

int no\_tr=0;

srand(time(NULL));

cout<<"Enter the no. of frames : ";

cin>>n;

cout<<"Enter the window size : ";

cin>>w;

int i=1;

while(i<=n)

{

int x=0;

for(int j=i;j<i+w && j<=n;j++)

{

cout<<"Sent frame "<<j<<endl;

no\_tr++;

}

for(int j=i;j<i+w && j<=n;j++)

{

int flag = rand()%2;

if(!flag)

{

cout<<"Acknowledgment for frame "<<j<<endl; x++;

}

else

{

cout<<"Frame "<<j<<" Not Received"<<endl; cout<<"Retransmitting Window"<<endl;

break;

}

}

cout<<endl; i+=x;

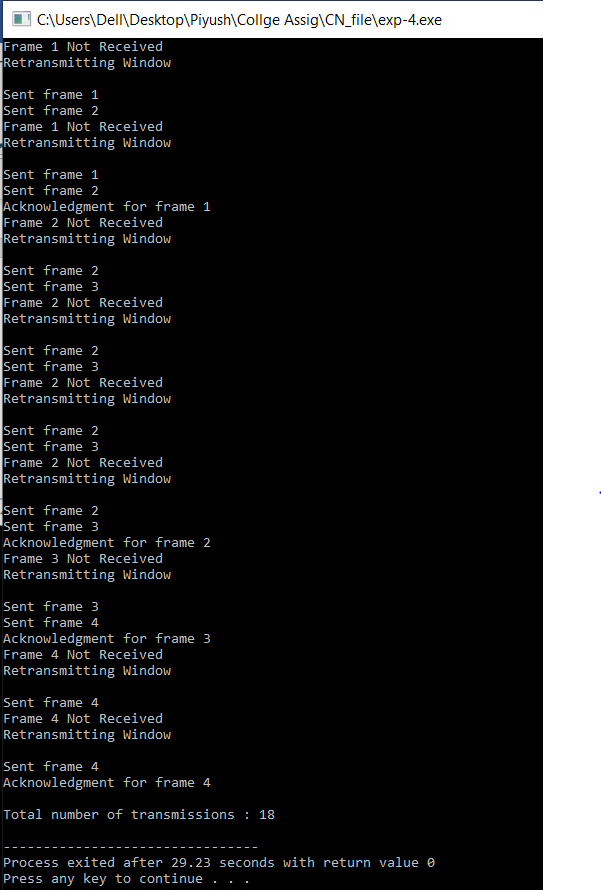
}

cout<<"Total number of transmissions : "<<no\_tr<<endl;

return 0;

}

**Output**



**Learning & Finding**

In this experiment, I learnt about sliding window protocol and its implementation.

**Experiment 5**

**Aim**: - Write a program to show net ID, Host ID, and network class of the address.

**Theory**

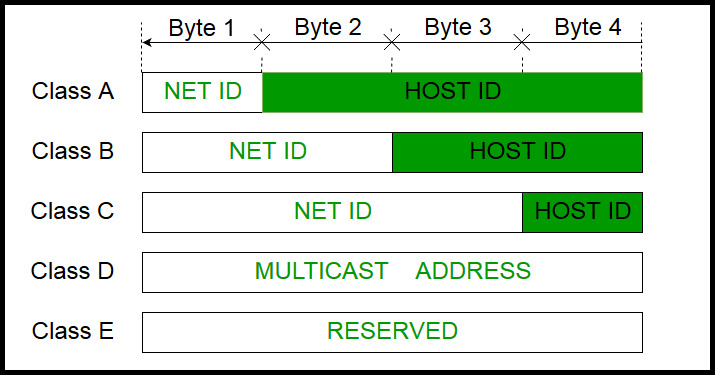
IP address is an address having information about how to reach a specific host, especially outside the LAN. An IP address is a 32 bit unique address having an address space of 232.

Generally, there are two notations in which IP address is written, dotted decimal notation and hexadecimal notation.

Classful Addressing

The 32 bit IP address is divided into five sub-classes. These are:

* Class A
* Class B
* Class C
* Class D
* Class E



**Code**

#include<bits/stdc++.h>

using namespace std;

void division(string str, char ipClass)

{

char network[12], host[12];

for(int k = 0; k < 12; k++)

network[k] = host[k] = '\0';

if(ipClass == 'A')

{

int i = 0, j = 0;

while (str[j] != '.')

network[i++] = str[j++];

i = 0;

j++;

while(str[j] != '\0')

host[i++] = str[j++];

cout<<"Network ID : "<<network<<endl;

cout<<"Host ID : "<<host<<endl;

}

else if(ipClass == 'B')

{

int i = 0, j = 0, dotCount = 0;

while (dotCount < 2)

{

network[i++] = str[j++];

if (str[j] == '.')

dotCount++;

}

i = 0;

j++;

while(str[j] != '\0')

host[i++] = str[j++];

cout<<"Network ID : "<<network<<endl;

cout<<"Host ID : "<<host<<endl;

}

else if(ipClass == 'C')

{

int i = 0, j = 0, dotCount = 0;

while (dotCount < 3)

{

network[i++] = str[j++];

if(str[j] == '.')

dotCount++;

}

i = 0;

j++;

while(str[j] != '\0')

host[i++] = str[j++];

cout<<"Network ID : "<<network<<endl;

cout<<"Host ID : "<<host<<endl;

}

else

cout<<"No division in this class"<<endl;

}

char find\_class(string str)

{

int ip=0, i=0;

while(str[i] != '.')

{

ip = ip\*10 + (str[i] - '0');

i++;

}

if (ip >=1 && ip <= 126)

return 'A';

else if (ip >= 128 && ip <= 191)

return 'B';

else if (ip >= 192 && ip <= 223)

return 'C';

else if (ip >= 224 && ip <= 239)

return 'D';

return 'E';

}

int main()

{

string str;

cout<<"Enter ID Address : ";

cin>>str;

char ip\_class = find\_class(str);

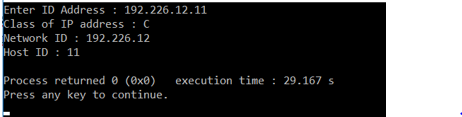
cout<<"Class of IP address : "<<ip\_class<<endl;

division(str, ip\_class);

return 0;

}

**Output**



**Learning & Finding**

In this experiment, I learnt about net ID, Host ID, and network class of the address and its implementation.

**Experiment 6**

**Aim**: - Write a program to implement distance vector routing algorithm.

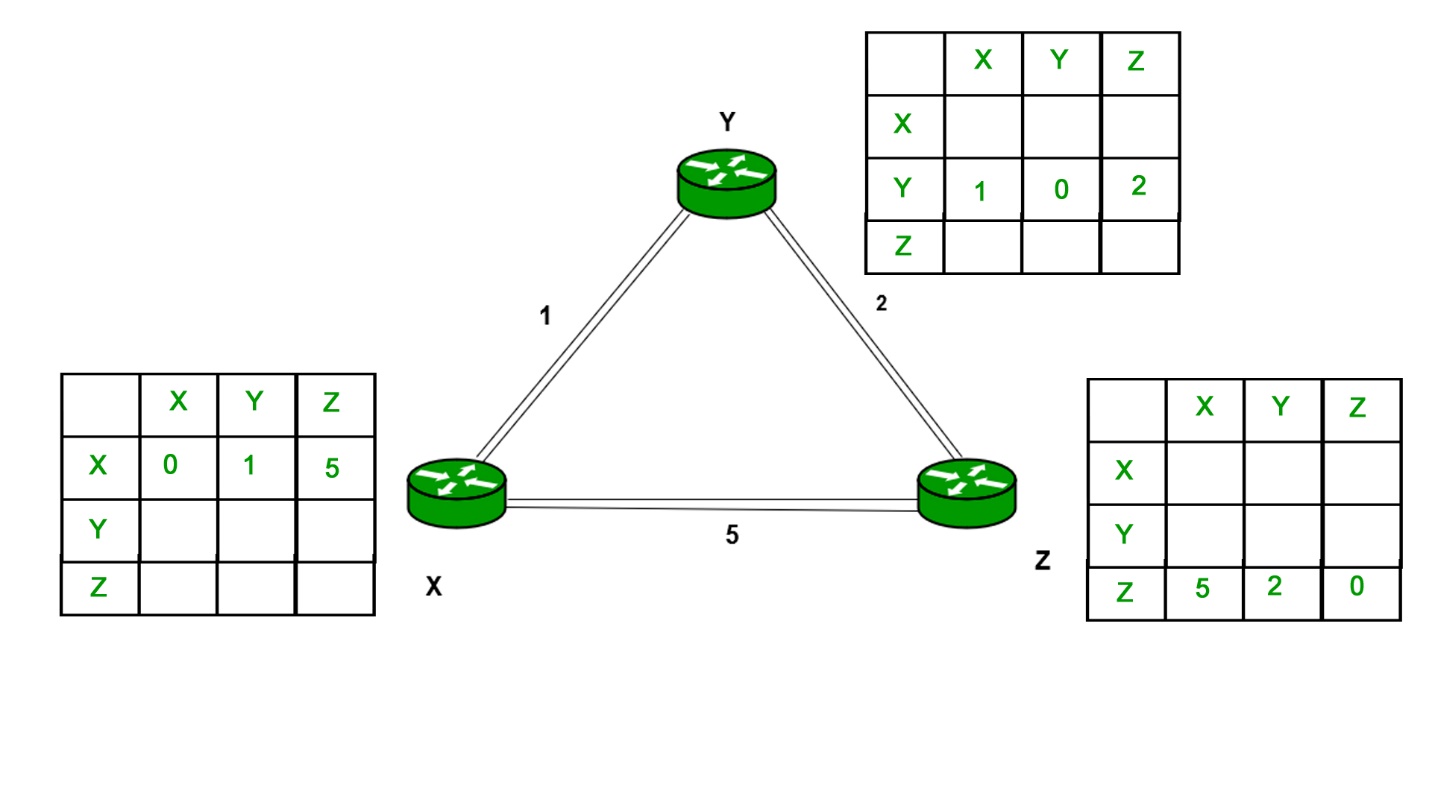
**Theory**

A distance-vector routing (DVR) protocol requires that a router inform its neighbors of topology changes periodically. Historically known as the old ARPANET routing algorithm (or known as Bellman-Ford algorithm).

Distance Vector Algorithm

1. A router transmits its distance vector to each of its neighbors in a routing packet.
2. Each router receives and saves the most recently received distance vector from each of its neighbors.
3. A router recalculates its distance vector when:

* It receives a distance vector from a neighbor containing different information than before.
* It discovers that a link to a neighbor has gone down.



**Code**

#include<bits/stdc++.h>

using namespace std;

struct node

{

unsigned dist[6];

unsigned from[6];

} DVR[10];

int main()

{

int costmat[6][6];

int nodes, i, j, k;

cout<<"Enter the no. of devices : ";

cin>>nodes;

cout<<"Enter the cost matrix : \n" ;

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

{

for(j = 0; j < nodes; j++)

{

cin>>costmat[i][j];

costmat[i][i] = 0;

DVR[i].dist[j] = costmat[i][j];

DVR[i].from[j] = j;

}

}

for(i = 0; i < nodes; i++) for(j = i+1; j < nodes; j++)

for(k = 0; k < nodes; k++)

if(DVR[i].dist[j] > costmat[i][k] + DVR[k].dist[j])

{

DVR[i].dist[j] = DVR[i].dist[k] + DVR[k].dist[j];

DVR[j].dist[i] = DVR[i].dist[j];

DVR[i].from[j] = k;

DVR[j].from[i] = k;

}

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

{

cout<<"\n\n For router: "<<i+1;

for(j = 0; j < nodes; j++)

cout<<"\t\n node "<<j+1<<" via "<<DVR[i].from[j]+1<<" Distance "<<DVR[i].dist[j];

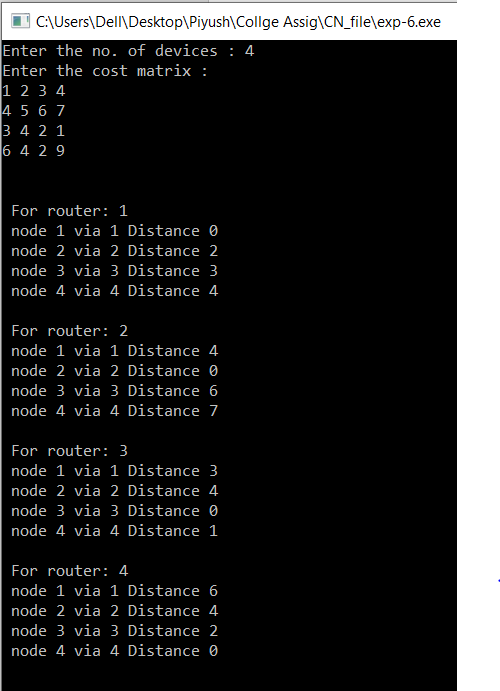
}

cout<<" \n\n ";

return 0;

}

**Output**



**Learning & Finding**

In this experiment, I learnt about distance vector routing algorithm and its implementation.

**Experiment 7**

**Aim**: - Write a program to implement link state routing algorithm.

**Theory**

Link state routing is the second family of routing protocols. While distance vector routers use a distributed algorithm to compute their routing tables, link-state routing uses link-state routers to exchange messages that allow each router to learn the entire network topology. Based on this learned topology, each router is then able to compute its routing table by using a shortest path computation.

Features of link state routing protocols

* Link state packet – A small packet that contains routing information.
* Link state database – A collection information gathered from link state packet.
* Shortest path first algorithm (Dijkstra algorithm) – A calculation performed on the database results into shortest path
* Routing table – A list of known paths and interfaces.

**Code**

#include<bits/stdc++.h>

using namespace std;

int main()

{

int count,src\_router,i,j,k,w,v,min;

int cost\_matrix[100][100],dist[100],last[100];

int flag[100];

cout<<"Enter the no. of routers : ";

cin>>count;

cout<<"Enter the cost matrix : "<<endl;

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

{

for(j=0;j<count;j++)

{

cin>>cost\_matrix[i][j];

if(cost\_matrix[i][j]<0)

cost\_matrix[i][j]=1000;

}

}

cout<<"Enter the source router : ";

cin>>src\_router;

for(v=0;v<count;v++)

{

flag[v]=0;

last[v]=src\_router;

dist[v]=cost\_matrix[src\_router][v];

}

flag[src\_router]=1;

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

{

min=1000;

for(w=0;w<count;w++)

{

if(!flag[w])

if(dist[w]<min)

{

v=w;

min=dist[w];

}

}

flag[v]=1;

for(w=0;w<count;w++)

{

if(!flag[w])

if(min+cost\_matrix[v][w]<dist[w])

{

dist[w]=min+cost\_matrix[v][w]; last[w]=v;

}

}

}

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

{

cout<<"\n"<<src\_router<<" -> "<<i<<" : Path taken : "<<i;

w=i;

while(w!=src\_router)

{

cout<<" <- "<<last[w];

w=last[w];

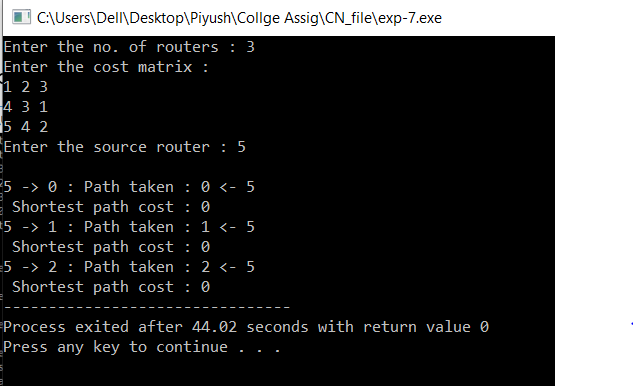
}

cout<<"\n Shortest path cost : "<<dist[i];

}

return 0;

}

**Output**

**Learning & Finding**

In this experiment, I learnt about sliding window protocol and its implementation.