DELHI TECHNOLOGICAL UNIVERSITY

Computer Networks Lab File

Submitted To:

Ms. Gull Kaur

Assistant Professor

Submitted By:

Sadaf A. Khan

2K19/CO/330

INDEX

S. No	EXPERIMENT	Dates
1	Program to convert Decimal IP Address to equivalent and vice-versa : : (a) Binary Notation (b) Octal Notation	07/01/2022
2	Implement bit, byte and character stuffing	21/01/2022
3	Implement cyclic redundancy check	28/01/2022
4	Implement stop and wait protocol	25/02/2022
5	Implement sliding window protocol (a)Go back N (b)Selective Repeat	04/03/2022 11/03/2022
6	Find class, network and host ID from IPv4 address	25/03/2022
7	Implement distance vector routing algorithm	01/04/2022
8	Implement link state routing algorithm	08/04/2022

PROGRAM - 1

- **AIM**

Program to convert Decimal IP Address to equivalent and vice-versa : :
(a) Binary Notation
(b) Octal Notation

- **THEORY**

An IP address is the identifier that enables your device to send or receive data packets across the internet. It holds information related to your location and therefore making devices available for two-way communication. The internet requires a process to distinguish between different networks, routers, and websites. Therefore, IP addresses provide the mechanism of doing so, and it forms an indispensable part in the working of the internet. You will notice that most of the IP addresses are essentially numerical. Still, as the world is witnessing a colossal growth of network users, the network developers had to add letters and some

addresses as internet usage grows.
An IP address is represented by a series of numbers segregated by periods(.). They are expressed in the form of four pairs - an example address might be 255.255.255.255 wherein each set can range from 0 to 255.

IP addresses are not produced randomly. They are generated mathematically and are further assigned by the IANA (Internet Assigned

Numbers Authority), a department of the ICANN.
ICANN stands for Internet Corporation for Assigned Names and Numbers. It is a non-profit corporation founded in the US back in 1998 with an aim to manage Internet security and enable it to be available by all.

- **CODE**

(a) Binary Notation

```
#include<iostream>
#include<sstream>
#include<bitset>
using namespace std;
int main()
{
int choice;
cout << "Which operation you want to perform : \n";
cout << "1.Decimal to Binary\n";
cout << "2.Binary to Decimal\n";
cin >> choice;
string ip;
cin >> ip;
stringstream object(ip);
string temp_number;
int i = 3;

while (getline(object, temp_number, '.'))
{
if (choice == 1)
{
cout << bitset<8> (stoi(temp_number)).to_string() << ((i != 0) ? "." : "");
```

```
4
}
else
{
cout << stoi(temp_number, 0, 2) << ((i != 0) ? "." : "");
}--i;
}
return 0;
}
```

(a) Octal Notation

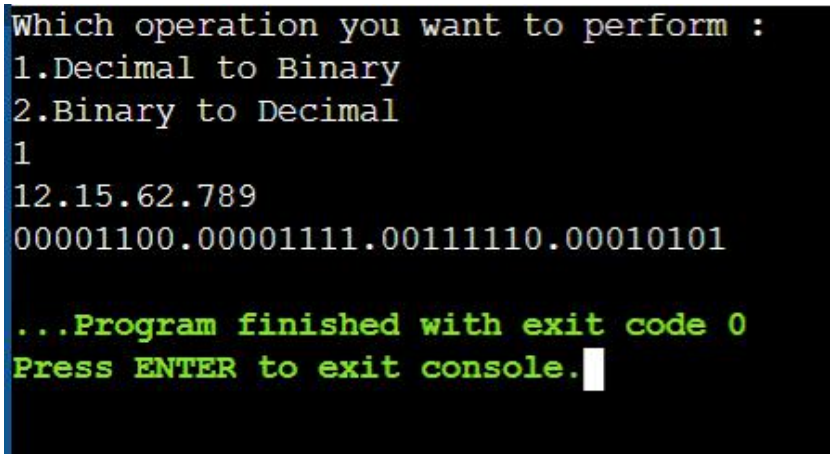
```
#include<iostream>
#include<sstream>
using namespace std;
int main()
{
int choice;
cout << "Which operation you want to perform : \n";
cout << "1.Decimal to Octal\n";
cout << "2.Octal to Decimal\n";
cin >> choice;
string ip;
cin >> ip;
stringstream object(ip);
string temp_number;
```

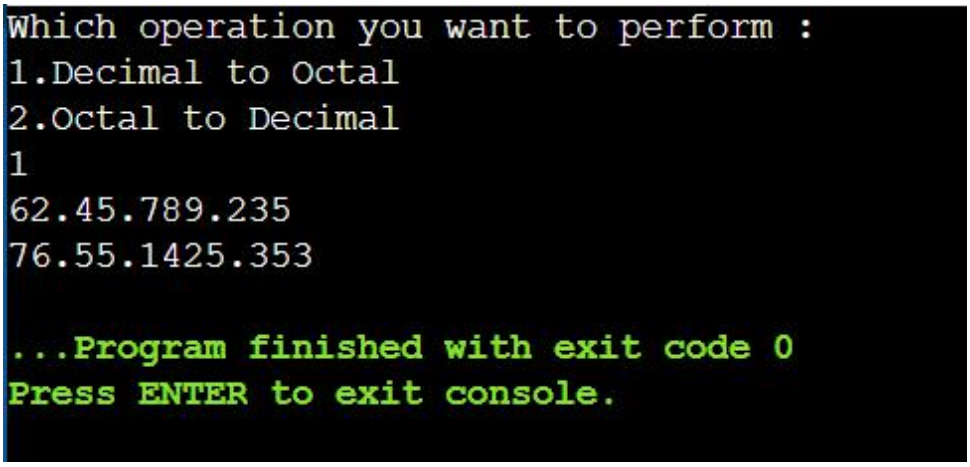
```
int i = 3;
while (getline(object, temp_number, '.'))
{
if (choice == 1)
{
stringstream octal_numer;
octal_numer << oct << stoi(temp_number);

4

cout << octal_numer.str() << ((i != 0) ? "." : "");
}
else
{
cout << stoi(temp_number, 0, 8) << ((i != 0) ? "." : "");
}--i;
}
return 0;
}
```

- **OUTPUT**





- **FINDINGS AND LEARNINGS**

We learnt about IP addresses and there Notaand its implementation andsignificance in computer networks.

PROGRAM - 2

- **AIM**

To implement different framing techniques using C/C++:

1. Bit stuffing
2. Byte stuffing
3. Character stuffing

- **THEORY**

Data link layer is responsible for something called Framing, which is the division of stream of bits from network layer into manageable units (calledframes). Each frame consists of sender's address and a destination address.The destination address defines where the packet is to go and the sender's address helps the recipient acknowledge the receipt.

Frames could be of fixed size or variable size. In fixed-size framing, there is noneed for defining the boundaries of the frames as the size itself can be used to define the end of the frame and the beginning of the next frame. But, in variable-size framing, we need a way to define the end of the frame and the beginning of the next frame. To separate one frame from the next, an 8-bit (or 1-byte) flag is added at the beginning and the end of a frame. But the problem with that is, any pattern used for the flag could also be part of the information. So, we use stuffing.

Security and Error detection are the most prominent features that are to be provided by any application which transfers data from one end to the other end.One of such a mechanism in tracking errors which may add up to the original data during transfer is known as Stuffing.

1. Bit Stuffing: In bit stuffing, to limit the number of consecutive bits of the same value i.e., binary value in the data to be transmitted. A bit of the oppositevalue is inserted after the maximum allowed number of consecutive bits. HighLevel Data Link Control (HDLC) is based on bit stuffing.

e.g. After 5 consecutive 1-bits, a 0-bit is stuffed.

2. Byte Stuffing: A byte (usually escape character (ESC)), which has a predefined bit pattern is added to the data section of the frame when there is a character with the same

pattern as the flag. Whenever the receiver encountersthe ESC character, it removes from the data section and treats the next character as data, not a flag. Point to Point Protocol (PPP) is based on byte stuffing.

3. Character Stuffing: In character stuffing, DLESTX and DLEETX are used to denote the start and end of the character data with some constraints imposed onrepetition of characters as shown in the experiment.

- **ALGORITHM**

1. BIT-STUFFING

- Start
- Initialize the array for transmitted stream with the special bit pattern01111110 which indicates the beginning of the frame.
- Get the bit stream to be transmitted in the array.
- Check for five consecutive ones and if they occur, stuff a bit 0.
- Display the data transmitted as it appears on the data line after appending01111110 at the end.
- For de-stuffing, copy the transmitted data to another array after detectingthestuffed bits
- Display the received bit stream
- Stop

2. BYTE-STUFFING

- Start
- Append 01111110 at the beginning of the string
- Check the data if character is present; if character 01111110 is present in thestring (example 01011111101) insert another 01111110 in the string.
- Transmit 01111110 at the end of the string
- Display the string
- Stop

3. CHARACTER-STUFFING

- Start
- Append DLE STX at the beginning of the string
- Check the data if character is present; if character DLE is present in toe string(example DOODLE) insert another DLE in the string (ex: DOODLEDLE).
- Transmit DLE ETX at the end of the string
- Display the string

- Stop

- **CODE**

Bit stuffing :

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main()
{
    int a[50];
    int i, j, k, n, c, pos; c = 0;
    cout << "Enter no. of bits\n";
    cin >> n;
    cout << "Enter the bits\n";
    for (i = 0; i < n; i++)
        cin >> a[i];
    for (i = 0; i < n; i++)
    {
        if (a[i] == 1)
        {
            c++;
            if (c == 5)
            {
```

```
                    pos = i + 1; c = 0;
                    for (j = n - 1; j >= pos; j--)
                    {
                        k = j + 1; a[k] = a[j];
                    }
                    a[pos] = 0; n++;
                }
            }
            else c = 0;
        }
        cout << "data after stuffing\n";

        for (i = 0; i < n; i++)
            cout << a[i];
        return 0;
}
```

Byte stuffing :

```
#include<stdio.h>
using namespace std;
int main()
{
    int i, j, k, data[100], n;
    printf("Enter no. of bits:\n");
```

```
scanf("%d", &n);
printf("enter data to be sent:\n");
for (i = 0; i < n; i++) scanf("%d", &data[i]);
    printf("the array entered is \n");
for (i = 0; i < n; i++)
    printf("%d", data[i]);
printf("\n");
for (i = 0; i < n; i++)
{
    if (data[i] == 0 && data[i + 1] == 1 && data[i + 2] == 1 && data[i + 3] == 1 &&
data[i + 4] == 1 && data[i + 5] == 1 && data[i + 6] == 1 && data[i + 7] == 0)
    {
        i = i + 8; for (j = i + 8; j >= i; j--)
        {
            data[j + 8] = data[j]; n++;
        }
        for (k = 0; k < 8; k++)
        {
            if (k == 0 || k == 7) data[i] = 0;
            else data[i] = 1;

            i++;
        }
    }
```

```
        }
        printf("the byte stuffed array is:\n");
        for (i = 0; i < n; i++)
                printf("%d", data[i]);
        printf("\n");
        return 0;
 }
```

Character stuffing :

```
#include<iostream>
using namespace std;
int main()
{
        char s[50], data[100];
        int i, j, n, m;
        cout << "Enter Size" << endl;
        cin >> n;
        cout << "Enter Data" << endl;
        for (int i = 0; i < n; i++)
                cin >> s[i];
        data[0] = 'D';
        data[1] = 'L';
        data[2] = 'S';
        data[3] = 'T';
```

```
data[4] = 'E';
data[5] = 'X'; j = 6;
m = 6;
for (int i = 0; i < n; i++)
{
    if (s[i] == 'D' && s[i + 1] == 'L' && s[i + 2] == 'E')
    {
        data[j++] = 'D';
        data[j++] = 'L';

        data[j++] = 'E';
        data[j++] = 'D';
        data[j++] = 'L';
        data[j++] = 'E'; i += 2;
        m += 6;
    }
    else
    {
        data[j++] = s[i]; m++;
    }
}
data[m++] = 'D';
data[m++] = 'L';
```

```
        data[m++] = 'E';
        data[m++] = 'E';
        data[m++] = 'T';
        data[m++] = 'X';
        cout << endl;
        for (int i = 0; i < m; i++)
            cout << data[i] << " ";
        return 0;
}
```

- **OUTPUT**

Bit stuffing :

```
Enter no. of bits
12
Enter the bits
0 1 1 1 1 0 1 1 0 0 1 0
data after stuffing
011110110010

...Program finished with exit code 0
Press ENTER to exit console.
```

Byte stuffing :

```
Enter no. of bits:
11
enter data to be sent:
0 1 1 0 1 1 1 1 1 1 0
the array entered is
01101111110
the byte stuffed array is:
011011111100111111032535


...Program finished with exit code 0
Press ENTER to exit console.
```

Character stuffing :

```
Enter Size
5
Enter Data
R U T V S

D L S T E X R U T V S D L E E T X

...Program finished with exit code 0
Press ENTER to exit console.
```

- **DISCUSSION**

The Data Link Layer is the second layer in the OSI model, above the Physical Layer, which ensures that the error free data is transferred between the adjacentnodes in the network. It breaks the datagrams passed down by above layers andconvert them into frames ready for transfer. This is called Framing. It provides two main functionalities.

1. Reliable data transfer service between two peer network layers
2. Flow Control mechanism which regulates the flow of frames such that datacongestion is not there at slow receivers due to fast senders.

Bit stuffing is the process of inserting non-information bits into data to break up bit patterns to affect the synchronous transmission of information. Characterstuffing has the Same idea as bit-stuffing, but operates on bytes instead of bits. It Uses reserved characters to indicate the start and end of a frame. For instance, use the two-character sequence DLE STX (Data-Link Escape, Start ofTeXt) to signal the beginning of a frame, and the sequence DLE ETX (End of Text) to flag the frame's end.

- **FINDINGS AND LEARNINGS**

We learnt about stuffing and different stuffing techniques and theirimplementation and their uses in computer networks.

PROGRAM - 3

- **AIM**

Write a program to implement CRC (Cyclic Redundancy Check).

- **THEORY**

Error: A condition when the receiver's information does not match with the sender's information. During transmission, digital signals suffer from noise thatcan introduce errors in the binary bits travelling from sender to receiver. That means a 0 bit may change to 1 or a 1 bit may change to 0.

Error Detecting Codes (Implemented either at Data link layer or Transport Layer of OSI Model): Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted. To avoid this, we use error- detecting codes which are additional data added to a given digital message tohelp us detect if any error has occurred during transmission of the message. Basic approach used for error detection is the use of redundancy bits, where additional bits are added to facilitate detection of errors. Some popular techniques for error detection are:

1. Simple Parity check
2. Two-dimensional Parity check
3. Checksum
4. Cyclic redundancy check

Cyclic redundancy check (CRC): CRC is based on binary division. In CRC, asequence 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 atthis step there is no remainder, the data unit is assumed to be correct and is therefore accepted.

Aremainder indicates that the data unit has been damaged in transit andtherefore must be rejected.

- **ALGORITHM**

1. A string of n as is appended to the data unit. The length of predetermineddivisor is n+ 1.
2. The newly formed data unit i.e., original data + string of n as are divided by the divisor using binary division and remainder is obtained. This remainder iscalled CRC.
3. Now, string of n O's appended to data unit is replaced by the CRC remainder (which is also of n bit).
4. The data unit + CRC is then transmitted to receiver.
5. The receiver on receiving it divides data unit + CRC by the same divisor&checks the remainder.
6. If the remainder of division is zero, receiver assumes that there is no error indata and it accepts it.
7. If remainder is non-zero then there is an error in data and receiver rejects it.

- **CODE**

```
#include <iostream>

#include <conio.h>

using namespace std;

int main()

{

    int i, j, k, l; int fs;

    cout << "\n Enter Frame size: "; cin >> fs;

    int f[20];

    cout << "\n Enter Frame:"; for (i = 0; i < fs; i++) cin >> f[i];

    int gs;

    cout << "\n Enter Generator size: "; cin >> gs;

    int g[20];

    cout << "\n Enter Generator:"; for (i = 0; i < gs; i++)
```

```
        cin >> g[i];
cout << "\n Sender Side:"; cout << "\n Frame: ";

for (i = 0; i < fs; i++) cout << f[i];
cout << "\n Generator :"; for (i = 0; i < gs; i++) cout << g[i];
int rs = gs - 1;
cout << "\n Number of 0's to be appended: " << rs; for (i = fs; i < fs + rs; i++)
        f[i] = 0;
int temp[20]; for (i = 0; i < 20; i++) temp[i] = f[i];
cout << "\n Message after appending 0's :"; for (i = 0; i < fs + rs; i++)
        cout << temp[i]; for (i = 0; i < fs; i++)
{
        j = 0;
        k = i;
        if (temp[k] >= g[j])
        {
                for (j = 0, k = i; j < gs; j++, k++)
                {
                        if ((temp[k] == 1 && g[j] == 1) || (temp[k] == 0 && g[j] == 0)) temp[k] = 0;
                        else temp[k] = 1;
                }
        }
}
```

```
int crc[15]; for (i = 0, j = fs; i < rs; i++, j++) crc[i] = temp[j]; cout << "\n CRC bits: "; for (i = 0; i < rs; i++) cout << crc[i];

cout << "\n Transmitted Frame: ";



int tf[15]; for (i = 0; i < fs; i++) tf[i] = f[i];

for (i = fs, j = 0; i < fs + rs; i++, j++) tf[i] = crc[j]; for (i = 0; i < fs + rs; i++) cout << tf[i];

cout << "\n Receiver side : "; cout << "\n Received Frame: "; for (i = 0; i < fs + rs; i++) cout << tf[i]; for (i = 0; i < fs + rs; i++) temp[i] = tf[i]; for (i = 0; i < fs + rs; i++)

{

    j = 0;

    k = i;

    if (temp[k] >= g[j])

    {

        for (j = 0, k = i; j < gs; j++, k++)

        {

            if ((temp[k] == 1 && g[j] == 1) || (temp[k] == 0 && g[j] == 0)) temp[k] = 0;

            else temp[k] = 1;

        }

    }

}

cout << "\n Reminder: "; int rrem[15];

for (i = fs, j = 0; i < fs + rs; i++, j++) rrem[j] = temp[i]; for (i = 0; i < rs; i++) cout << rrem[i];

int flag = 0; for (i = 0; i < rs; i++)

{
```

```
        if (rrem[i] != 0) flag = 1;
    }
    if (flag == 0)
        cout << "\n Since Remainder Is 0 Hence Message Transmitted From Sender To Receiver Is Correct";
    else
        cout << "\n Since Remainder Is Not 0 Hence Message Transmitted From Sender To Receiver Contains Error";
    getch();
    return 0;
}
```

- **OUTPUT**

```
Enter Frame size: 5

Enter Frame:1 0 1 0 1

Enter Generator size: 4

Enter Generator:1 0 0 1

Sender Side:
Frame: 10101
Generator :1001
Number of 0's to be appended: 3
Message after appending 0's :10101000
CRC bits: 111
Transmitted Frame: 10101111
Receiver side :
Received Frame: 10101111
Reminder: 000
Since Remainder Is 0 Hence Message Transmitted From Sender To Receiver Is Correct

...Program finished with exit code 0
Press ENTER to exit console.
```

- **DISCUSSION**

During transmission, digital signals suffer from noise that can introduce errorsin the binary bits travelling from sender to receiver. That means a 0 bit may

- **FINDINGS AND LEARNINGS**

We learnt about CRC (Cyclic Redundancy Check) and its implementation andsignificance in computer networks.

PROGRAM - 4

- **AIM**

Write a program to implement STOP and WAIT protocol.

- **THEORY**

Stop and Wait Protocol is

- Used in Connection-oriented communication.
- It offers error and flow control.
- It is used in Data Link and Transport Layers.
- Stop and Wait ARQ mainly implements Sliding Window Protocol conceptwith Window Size 1.

Sender:

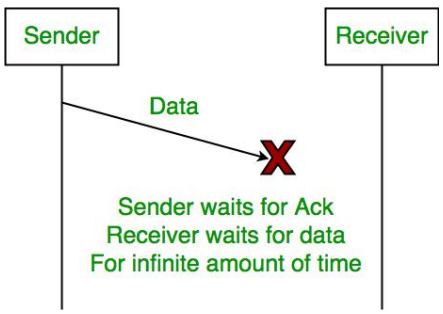
1. Send one data packet at a time.
2. Send next packet only after receiving acknowledgement for previous.

Receiver:

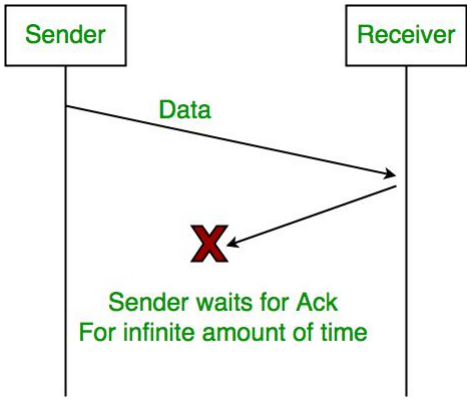
1. Send acknowledgement after receiving and consuming of data packet.
2. After consuming packet acknowledgement need to be sent (Flow Control).

Problems:

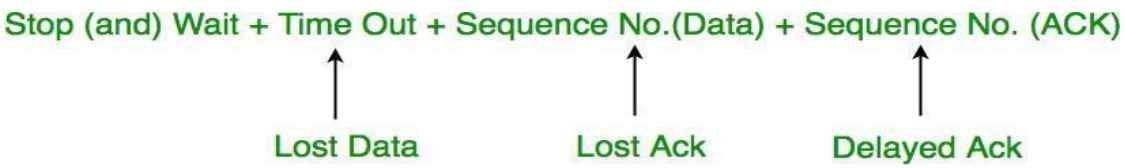
1. Lost Data



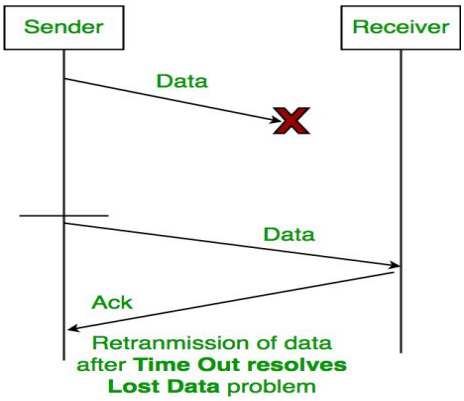
2. Lost Acknowledgement



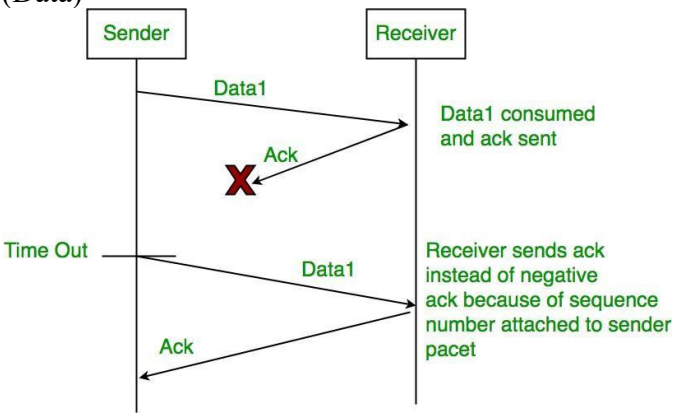
3. Delayed Acknowledgement/Data: After timeout on sender side, a long- delayed acknowledgement might be wrongly considered as acknowledgementof some other recent packet.



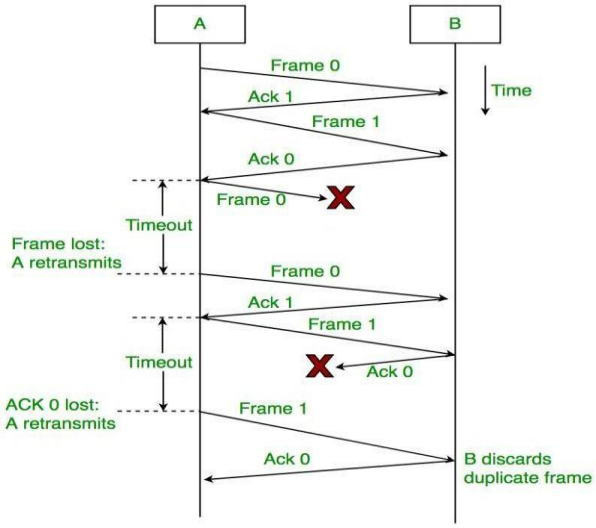
4. Time Out



5. Sequence Number (Data)



6. Delayed Acknowledgement This is resolved by introducing sequencenumber for acknowledgement also.



- **ALGORITHM**

At sender (Node A)

1. Accept packet from higher layer when available.
2. Assign number SN to it.
3. Transmit packet SN in frame with sequence #SN.
4. Wait for an error free frame from B:

a. if received and it contains RN greater than SN in the request #field, set SN 2RN and go to 1

b. if not received within given time go to 2 (with initial condition SN equal tozero)

At receiver (Node B)

1. Whenever an error free frame is received from a with a sequence # equal toRN release the receive packet to higher layers and increment RN.
2. At arbitrary times, but within bounded delay after receiving an error free frame from A, transmit a frame to A containing RN in the request # field(withinitial condition RN equal to zero).

- **CODE**

```
#include <conio.h>

#include <stdio.h>

#include <stdlib.h>

#define TIMEOUT 5

#define MAX_SEQ 1

#define TOT_PACKETS 5


#define inc(k) if(k<MAX_SEQ) k++; else k=0;


typedef struct

{

      int data;

} packet;


typedef struct
```

```
{
    int kind, seq, ack, err; packet info;
} frame; frame DATA;

typedef enum {frame_arrival, err, timeout, no_event} event_type;
void from_network_layer(packet *);
void to_network_layer(packet *);
void to_physical_layer(frame *);
void from_physical_layer(frame *);
void wait_for_event_sender(event_type *);
void wait_for_event_reciever(event_type *);
void receiver();
void sender();
int i = 1; //Data to be sent by sender char turn; //r , s
int DISCONNECT = 0;

int main()
{
    rand(); while (!DISCONNECT)
    {
        sender(); receiver();
    }
    getch(); return 0;
}
```

```
void sender()
{
    static int frame_to_send = 0; static frame s;
    packet buffer; event_type event;

    static int flag = 0; if (flag == 0)
    {
        from_network_layer(&buffer); s.info = buffer;
        s.seq = frame_to_send;
        printf("SENDER : Info = %d Seq No = %d ", s.info, s.seq); turn = 'r';
        to_physical_layer(&s); flag = 1;
    }
    wait_for_event_sender(&event); if (turn == 's')
    {
        if (event == frame_arrival)
        {
            from_network_layer(&buffer); inc(frame_to_send);
            s.info = buffer;
            s.seq = frame_to_send;
            printf("SENDER : Info = %d Seq No = %d ", s.info, s.seq); turn = 'r';
            to_physical_layer(&s);
        }
        if (event == timeout)
        {
            printf("SENDER : Resending Frame "); turn = 'r';
```

```
                to_physical_layer(&s);
            }
        }
}

void reciever()
{
    static int frame_expected = 0; frame r, s;
    event_type event;

    wait_for_event_reciever(&event); if (turn == 'r')
    {
        if (event == frame_arrival)
        {
            from_physical_layer(&r); if (r.seq == frame_expected)
            {
                to_network_layer(&r.info); inc(frame_expected);
            }
            else
                printf("RECIEVER : Acknowledgement Resent\n"); turn = 's';
            to_physical_layer(&s);
        }
        if (event == err)
        {
            printf("RECIEVER : Garbled Frame\n"); turn = 's'; //if frame not recieved
```

```
            } //sender shold send it again
        }
}

void from_network_layer(packet *buffer)
{
        (*buffer).data = i++;
}

void to_physical_layer(frame *s)
{       // 0 means error
        s->err = rand() % 4; //non zero means no error DATA = *s; //probability of error = 1/4
}

void to_network_layer(packet *buffer)
{
        printf("RECIEVER :Packet %d recieved , Ack Sent\n", (*buffer).data); if (i >
TOT_PACKETS) //if all packets recieved then disconnect
        {

            DISCONNECT = 1;
            printf("\nDISCONNECTED");
        }
}
```

```
void from_physical_layer(frame *buffer)
{
    *buffer = DATA;
}

void wait_for_event_sender(event_type * e)
{
    static int timer = 0; if (turn == 's')
    {
        timer++; if (timer == TIMEOUT)
        {
            *e = timeout;
            printf("SENDER : Ack not recieved=> TIMEOUT\n"); timer = 0;
            return;
        }
        if (DATA.err == 0)
            *e = err; else
        {
            timer = 0;
            *e = frame_arrival;
        }
    }
}

void wait_for_event_reciever(event_type * e)
```

```
{
    if (turn == 'r')
    {
        if (DATA.err == 0)
            *e = err; else

            *e = frame_arrival;
    }
}
```

- **OUTPUT**

```
SENDER : Info = 1 Seq No = 0 RECIEVER :Packet 1 recieved , Ack Sent
SENDER : Info = 2 Seq No = 1 RECIEVER :Packet 2 recieved , Ack Sent
SENDER : Info = 3 Seq No = 0 RECIEVER :Packet 3 recieved , Ack Sent
SENDER : Info = 4 Seq No = 1 RECIEVER : Garbled Frame
SENDER : Ack not recieved=> TIMEOUT
SENDER : Resending Frame RECIEVER :Packet 4 recieved , Ack Sent
SENDER : Info = 5 Seq No = 0 RECIEVER :Packet 5 recieved , Ack Sent

DISCONNECTED

...Program finished with exit code 0
Press ENTER to exit console.
```

- **DISCUSSION**

1. Stop and Wait uses link between sender and receiver as half duplex link
2. Throughput = 1 Data packet/frame per RTT.
3. If Bandwidth*Delay product is very high, then stop and wait protocol is notso useful. The sender has to keep waiting for acknowledgements before sending the processed next packet.
4. It is an example for “Closed Loop OR connection oriented” protocols.

5. It is a special category of SWP where its window size is 1.

6. Irrespective of number of packets sender is having stop and wait protocolrequires only 2 sequence numbers 0 and 1.

- **FINDINGS AND LEARNINGS**

The Stop and Wait ARQ solves main three problems, but may cause big performance issues as sender always waits for acknowledgement even if it has next packet ready to send. Consider a situation where you have a high bandwidth connection and propagation delay is also high (you are connected tosome server in some other country though a high-speed connection). To solve this problem, we can send more than one packet at a time with a larger sequence number.

PROGRAM - 5

- **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 bya 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. The Sliding Window ARQ (Automatic Repeat Request) protocols are of two categories:

1. Go – Back – N ARQ: Go – Back – N ARQ provides for sending multiple frames before receiving the acknowledgment for the first frame. It uses the concept of sliding window, and so is also called sliding window protocol. The frames are sequentially numbered and a finite number of frames are sent. If theacknowledgment of a frame is not received within the time period, all frames starting from that frame are retransmitted.
2. Selective Repeat ARQ: This protocol also provides for sending multiple frames before receiving the acknowledgment for the first frame. However, hereonly the erroneous or lost frames are retransmitted, while the good frames are received and buffered.

- **ALGORITHM**

Step 1: Start the program.

Step 2: Open the input file in read mode.
Step 3: Read the size of the window.

Step 4: Select randomly the number of packets is to be transferred.Step 5: Read the content of the input file.

Step 6: Transfer the packet until it reaches the maximum defined size.

Step 7: Resume the window size and repeat the above two steps until packetsin.

Step 8: Close the file.
Step 9: Stop the
program.

- **CODE**

```
#include <conio.h>

#include<stdio.h>

#define ll long long int

#define pb push_back

#define ff first

#define ss second

#define mod 1000000007

int main()

{

	ios_base::sync_with_stdio(false);

	cin.tie(NULL);

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

	printf("Enter window size: ");

	scanf("%d", &w);

	printf("\nEnter number of frames to transmit: ");

	scanf("%d", &f);

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

	for (i = 1; i <= f; i++)

		scanf("%d", &frames[i]); //256348915267

	printf("\nAfter sending %d frames at each stage sender waits for acknowledgement sent by the receiver:\n\n", w);
```

```
    for (i = 1; i <= f; i++)
    {
        if (i % w == 0)
        {
            printf("%d\n", frames[i]);
            printf("Acknowledgement of above frames sent is received by sender\n\n");
        }
        else
            printf("%d ", frames[i]);
    }
    if (f % w != 0)
        printf("\nAcknowledgement of above frames sent is received by sender\n");
    return 0;
}
```

- **OUTPUT**

```
Enter window size: 5

Enter number of frames to transmit: 12

Enter 12 frames: 2 5 6 3 4 8 9 1 5 2 6 7

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

2 5 6 3 4
Acknowledgement of above frames sent is received by sender

8 9 1 5 2
Acknowledgement of above frames sent is received by sender

6 7
Acknowledgement of above frames sent is received by sender


...Program finished with exit code 0
Press ENTER to exit console.
```

- **DISCUSSION**

Characteristics of Sliding Window ARQ: Sliding window refers to animaginary box that hold the frames on both sender and receiver side.

1. It provides the upper limit on the number of frames that can be transmittedbefore requiring an acknowledgment.
2. Frames may be acknowledged by receiver at any point even when window isnot full on receiver side.
3. Frames may be transmitted by source even when window is not yet full onsender side.
4. The windows have a specific size in which the frames are numbered modulo- n, which means they are numbered from 0 to n-l. For e.g., if n = 8, theframes are numbered 0, 1,2,3,4,5,6, 7, 0, 1,2,3,4,5,6, 7, 0, 1, ...

5. The size of window is n-1. For e.g. In this case it is 7. Therefore, a maximumof n-l frames may be sent before an acknowledgment.

- **FINDINGS AND LEARNINGS**

We learnt about sliding window protocol and its implementation andsignificance in computer networks.

PROGRAM - 5 (b)

- **AIM**

Write a program to implement sliding window protocol.
(b) Selective Repeat

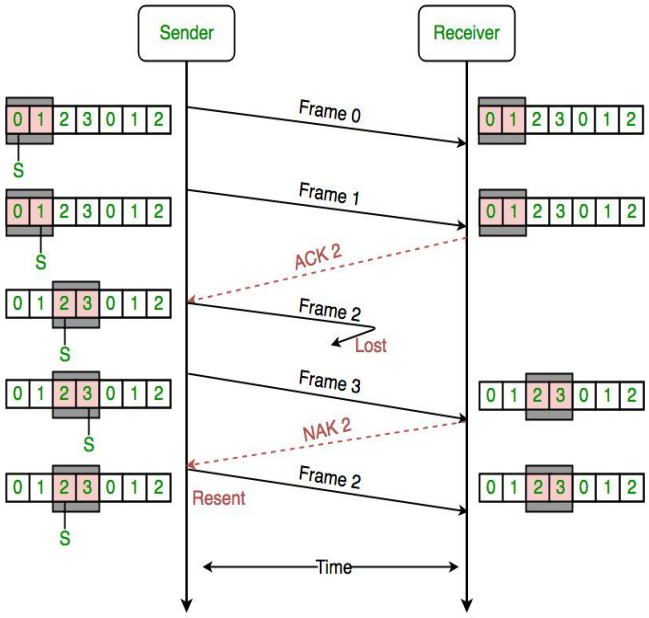
- **THEORY**

The Selective Repeat protocol, is to allow the receiver to accept and buffer the frames following a damaged or lost one.
Selective Repeat attempts to retransmit only those packets that are actually lost (due to errors) :
Receiver must be able to accept packets out of order.
Since receiver must release packets to higher layer in order, the receiver must be able to buffer some packets.

Retransmission requests :
Implicit – The receiver acknowledges every good packet, packets that are not ACKed before a time-out are assumed lost or in error.Notice that this approach must be used to be sure that every packet is eventually received.
Explicit – An explicit NAK (selective reject) can request retransmission of just one packet. This approach can expedite the retransmission but is not strictly needed.
One or both approaches are used in practice.

Selective Repeat Protocol (SRP) :
This protocol(SRP) is mostly identical to GBN protocol, except that buffers are used and the receiver, and the sender, each maintains a window of size. SRP works better when the link is very unreliable. Because in this case, re-transmission tends to happen more frequently, selectively retransmitting frames is more efficient than retransmitting all of them. SRP also requires full-duplex link. backward acknowledgments are also in progress.

- Sender's Windows (Ws) = Receiver's Windows (Wr).
- Window size should be less than or equal to half the sequence number in SR protocol. This is to avoid packets being recognized incorrectly. If the size of the window is greater than half the sequence number space, then if an ACK is lost, the sender may send new packets that the receiver believes are retransmissions.
- Sender can transmit new packets as long as their number is with W of all unACKed packets.
- Sender retransmit un-ACKed packets after a timeout – Or upon a NAK if NAK is employed.
- Receiver ACKs all correct packets.
- Receiver stores correct packets until they can be delivered in order to the higher layer.
- In Selective Repeat ARQ, the size of the sender and receiver window must be at most one-half of 2^m.



the sender only retransmits frames, for which a NAK is received

Efficiency of Selective Repeat Protocol (SRP) is same as GO-Back-N's efficiency :
Efficiency = N/(1+2a)
Where a = Propagation delay / Transmission delay
Buffers = N + N
Sequence number = N(sender side) + N (Receiver Side)

- **ALGORITHM**

Step 1: Start the program.

Step 2: Open the input file in read mode.
Step 3: Read the size of the window.

Step 4: Select randomly the number of packets is to be transferred.Step 5: Read the content of the input file.

Step 6: Transfer the packet until it reaches the maximum defined size.

Step 7: Resume the window size and repeat the above two steps until packetsin.

Step 8: Close the file.
Step 9: Stop the
program.

- **CODE**

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
using namespace std;
#define TOT_FRAMES 500
#define FRAMES_SEND 10
class sel_repeat
{
private:
    int fr_send_at_instance;
    int arr[TOT_FRAMES];
    int send[FRAMES_SEND];
    int rcvd[FRAMES_SEND];

    char rcvd_ack[FRAMES_SEND];

    int sw;

    int rw;          //tells expected frame

public:

    void input();

    void sender(int);

    void receiver(int);

};
```

```
void sel_repeat::input()
{
	int n;		//no. of bits for the frame
	int m;		//no. of frames from n bits
	int i
	cout << "Enter the no. of bits for the sequence no. : ";
	cin >> n;
	m = pow(2, n);
	int t = 0;
	fr_send_at_instance = (m / 2);
	for (i = 0; i < TOT_FRAMES; i++)
	{
		arr[i] = t;
		t = (t + 1) % m;
	}
	for (i = 0; i < fr_send_at_instance; i++)
	{
		send[i] = arr[i];
		rcvd[i] = arr[i];
		rcvd_ack[i] = 'n';
	}
	rw = sw = fr_send_at_instance;
	sender(m);
}
void sel_repeat::sender(int m)
{
	for (int i = 0; i < fr_send_at_instance; i++)
	{
		if (rcvd_ack[i] == 'n')
			cout << "SENDER : Frame " << send[i] << " is sent\n";
	}
	receiver(m);
}

void sel_repeat::receiver(int m)
{
	time_t t;
```

```
        int f;
        int j;
        int f1;
        int a1;
        char ch;
        srand((unsigned)time(&t));
        for (int i = 0; i < fr_send_at_instance; i++)
        {
            if (rcvd_ack[i] == 'n')
            {
                f = rand() % 10;
//if f=5 frame is discarded for some reason
//else frame is correctly recieved
                if (f != 5)
                {
                    for (int j = 0; j < fr_send_at_instance; j++)
                        if (rcvd[j] == send[i])
                        {
                            cout << "reciever:Frame" << rcvd[j] << "recieved correctly\n";
                            rcvd[j] = arr[rw];
                            rw = (rw + 1) % m;
                            break;
                        }
                    int j
                    if (j == fr_send_at_instance)
                        cout << "reciever:Duplicate frame" << send[i] << "discarded\n";
                    a1 = rand() % 5;
//if al==3 then ack is lost
//else recieved
                    if (a1 == 3)
                    {
                        cout << "(acknowledgement " << send[i] << " lost)\n";
                    cout << "(sender timeouts-->Resend the frame)\n";

                        rcvd_ack[i] = 'n';
```

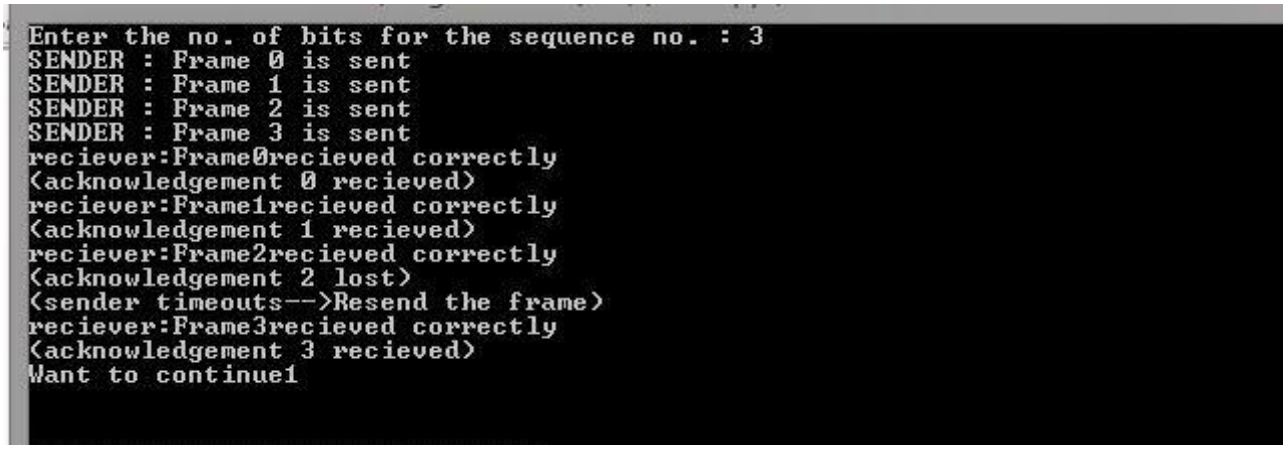
```
                    }
                    else
                    {
                        cout << "(acknowledgement " << send[i] << " recieved)\n";
                        rcvd_ack[i] = 'p';
                    }
                }
                else
                {   int ld = rand() % 2;
//if =0 then frame damaged
//else frame lost
                    if (ld == 0)
                    {
                        cout << "RECEIVER : Frame " << send[i] << " is damaged\n";
                        cout << "RECEIVER : Negative Acknowledgement " << send[i] << " sent\n";
                    }
                    else
                    {
                        cout << "RECEIVER : Frame " << send[i] << " is lost\n";
                        cout << "(SENDER TIMEOUTS-->RESEND THE FRAME)\n";
                    }
                    rcvd_ack[i] = 'n';
                }
            }
        }
        for (int j = 0; j < fr_send_at_instance; j++)
        {
            if (rcvd_ack[j] == 'n')
                break;
        }
        int i = 0;
        for (int k = j; k < fr_send_at_instance; k++)
        {
            send[i] = send[k];
            if (rcvd_ack[k] == 'n')
                rcvd_ack[i] = 'n';
```

```
            else
                rcvd_ack[i] = 'p';
            i++;
        }
        if (i != fr_send_at_instance)
        {
            for (int k = i; k < fr_send_at_instance; k++)
            {
                send[k] = arr[sw];
                sw = (sw + 1) % m;
                rcvd_ack[k] = 'n';
            }

        }
        cout << "Want to continue";
        cin >> ch;
        cout << "\n";
        if (ch == 'y')
            sender(m);
        else
            exit(0);
}

Int main()
{
        sel_repeat sr;
        sr.input();
}
```

- **OUTPUT**



- **DISCUSSION**

Characteristics of Sliding Window ARQ: Sliding window refers to animaginary box that hold the frames on both sender and receiver side.

6. It provides the upper limit on the number of frames that can be transmittedbefore requiring an acknowledgment.
7. Frames may be acknowledged by receiver at any point even when window isnot full on receiver side.
8. Frames may be transmitted by source even when window is not yet full onsender side.
9. The windows have a specific size in which the frames are numbered modulo- n, which means they are numbered from 0 to n-l. For e.g., if n = 8, theframes are numbered 0, 1,2,3,4,5,6, 7, 0, 1,2,3,4,5,6, 7, 0, 1, ...

10. The size of window is n-1. For e.g. In this case it is 7. Therefore, a maximumof n-l frames may be sent before an acknowledgment.

- **FINDINGS AND LEARNINGS**

We learnt about sliding window protocol (Selective Repeat) and its implementation and significance in computer networks.

PROGRAM - 6

- **AIM**

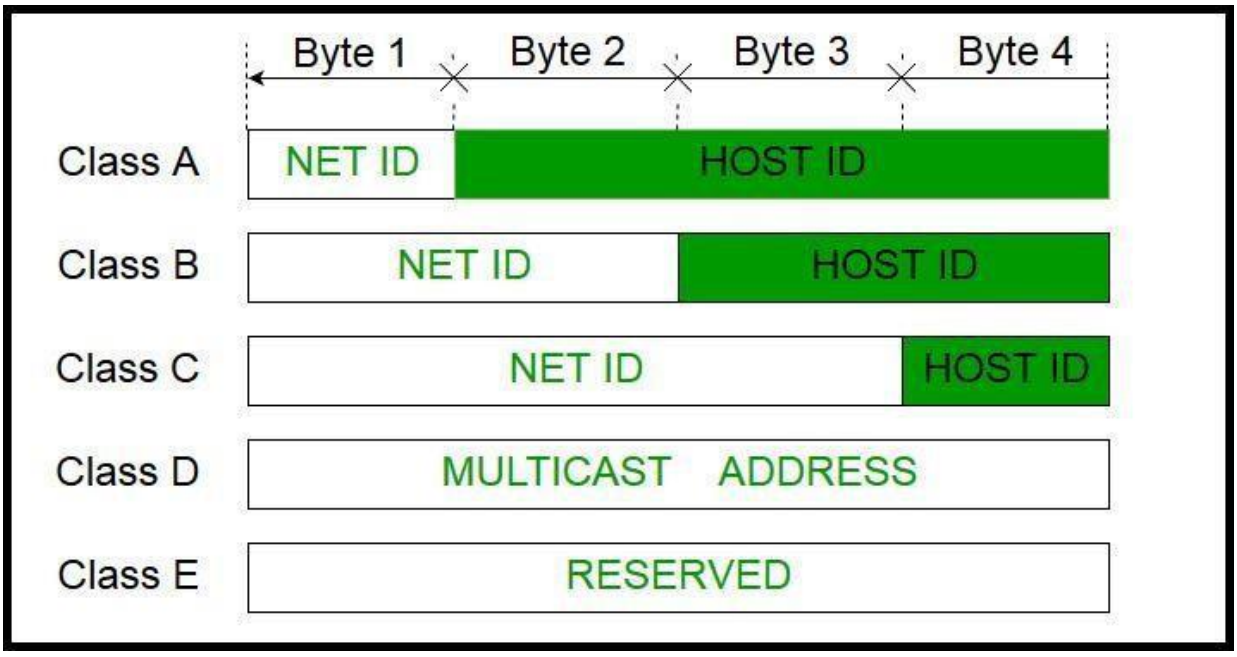
Write a program to implement and find class, network and host ID from givenIPv4 address.

- **THEORY**

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

- Class A
- Class B
- Class C
- Class D
- Class E

Each of these classes has a valid range of IP addresses. Classes D and E are reserved for multicast and experimental purposes respectively. The order of bitsin the first octet determines the classes of IP address. IPv4 address is divided into two parts Network ID and Host ID. The class of IP address is used to determine the bits used for network ID and host ID and the number of total networks and hosts possible in that particular class. Each ISP or network administrator assigns IP address to each device that is connected to its network.



- **ALGORITHM**

1. For determining the class: The idea is to check first octet of IP address. Aswe know, for class A first octet will range from 1 – 126, for class B first octetwill range from 128 – 191, for class C first octet will range from 192- 223, for class D first octet will range from 224 – 239, for class E first octet will range from 240 – 255.
2. For determining the Network and Host ID: We know that Subnet Mask for Class A is 8, for Class B is 16 and for Class C is 24 whereas Class D and Eis not divided into Network and Host ID.

An IP address has two components, the network address and the host address.A subnet mask separates the IP address into the network and host addresses (<network><host>). Subnetting further divides the host part of an IP address into a subnet and host address (<network><subnet><host>) if additional subnetwork is needed. It is called a subnet mask because it is used to identify network address of an IP address by performing a bitwise AND operation on the netmask.

- **CODE**

```
#include <bits/stdc++.h>

using namespace std;

int main()

{

    ios_base::sync_with_stdio(false); cin.tie(NULL);

    int i, c, k, n; c = k = 0; string s;

    cout << "Enter IPv4 address "; cin >> s;

    n = s.length(); for (i = 0; i < n; i++)

    {

        if (s[i] == '.')

        {

            k = stoi(s.substr(0, i));
```

```
                break;
            }
        }    //checking for a valid address if(k<1 || k>255)
        {
            cout << "Wrong IPv4 address entered"; return 0;
        } if (k <= 126)
        {
            cout << "Class: A\n";
            cout << "Netword ID: " << k << '\n'; cout << "Host ID: " << s.substr(i + 1);
        }
        else if (k <= 191)
        {
            cout << "Class: B\n"; for (i = 0; i < n; i++)
            {
                if (s[i] == '.')
                {
                    if (c == 1)
                    {
                        cout << "Netword ID: " << s.substr(0, i) << '\n'; cout << "Host ID: " <<
s.substr(i + 1);
                        break;
                    } c++;
                }
            }
```

```
    }
    else if (k <= 223)
    {
        cout << "Class: C\n"; for (i = 0; i < n; i++)
        {
            if (s[i] == '.')
            {
                if (c == 2)
                {
                    cout << "Netword ID: " << s.substr(0, i) << '\n'; cout << "Host ID: " <<
s.substr(i + 1);
                    break;
                } c++;
            }
        }
    }
    else if (k <= 239)
    {
        cout << "Class: D\n";
        cout << "Netword ID: Does not exist\n"; cout << "Host ID: Does not exist\n";
    }
    else
    {
        cout << "Class: E\n";
        cout << "Netword ID: Does not exist\n"; cout << "Host ID: Does not exist\n";
```

```
    }
    return 0;
}
```

- **OUTPUT**

```
Enter IPv4 address 192.226.12.11
Class: C
Netword ID: 192.226.12
Host ID: 11

...Program finished with exit code 0
Press ENTER to exit console.
```

```
Enter IPv4 address 130.45.151.154
Class: B
Netword ID: 130.45
Host ID: 151.154

...Program finished with exit code 0
Press ENTER to exit console.
```

```
Enter IPv4 address 1.4.5.5
Class: A
Netword ID: 1
Host ID: 4.5.5

...Program finished with exit code 0
Press ENTER to exit console.
```

- **FINDINGS AND LEARNINGS**

We learnt about host id, net id and network class and their uses andsignificance in computer network.

PROGRAM - 7

- **AIM**

Write a program to implement distance vector routing algorithm.

- **THEORY**

The Distance vector algorithm is iterative, asynchronous and distributed.

1. Distributed: It is distributed in that each node receives information from oneor more of its directly attached neighbours, performs calculation and then distributes the result back to its neighbours.
2. Iterative: It is iterative in that its process continues until no more informationis available to be exchanged between neighbours.
3. Asynchronous: It does not require that all of its nodes operate in the lockstep with each other.

The Distance vector algorithm is a dynamic algorithm. It is mainly used in ARPANET, and RIP. Each router maintains a distance table known as Vector.Three Keys to understand the working of Distance Vector Routing Algorithm:

1) Knowledge about the whole network: Each router shares its knowledge through the entire network. The router sends its collected knowledge about thenetwork to its neighbours.
2) Routing only to neighbours: The router sends its knowledge about the network to only those routers which have direct links. The router sends whatever it has about the network through the ports. The information is received by the router and uses the information to update its own routing table.
3) Information sharing at regular intervals: Within 30 seconds, the router sendsthe information to the neighboring routers.

With the Distance Vector Routing algorithm, the node x contains the followingrouting information:

1. For each neighbor v, the cost c(x,v) is the path cost from x to directlyattached neighbor, v.

2. The distance vector x, i.e., Dx = [Dx(y) : y in N], containing its cost to alldestinations, y, in N.

3. The distance vector of each of its neighbors, i.e., Dv = [Dv(y) : y in N] foreach neighbor v of x.

- **ALGORITHM**

At each node x,

Initialization

for all destinations y in N:

Dx(y) = c(x,y) // If y is not a neighbour then c(x,y) = ∞for each neighbour w

Dw(y) = ? for all destination y in N.
for each neighbour w

send distance vector Dx = [Dx(y) : y in N] to wloop

wait(until I receive any distance vector from some neighbour w)for each y in N:

Dx(y) = minv{c(x,v)+Dv(y)}

If Dx(y) is changed for any destination y

Send distance vector Dx = [Dx(y) : y in N] to all neighbours

- **CODE**

```
#include <bits/stdc++.h>

using namespace std;

struct node

{

    unsigned dist[20];
```

```
    unsigned from[20];
} rt[10];
int main()
{
    int costmat[20][20]; int nodes, i, j, k, count = 0;
    cout << "\nEnter the number of nodes : "; cin >> nodes;
    cout << "\nEnter the cost matrix : \n"; for (i = 0; i < nodes; i++)
    {
        for (j = 0; j < nodes; j++)

        {
            cin >> costmat[i][j]; costmat[i][i] = 0; rt[i].dist[j] = costmat[i][j]; rt[i].from[j] = j;
        }
    }
    do
    {
        count = 0; for (i = 0; i < nodes; i++)
        {
            for (j = 0; j < nodes; j++)
            {
                for (k = 0; k < nodes; k++)
                {
                    if (rt[i].dist[j] > costmat[i][k] + rt[k].dist[j])
                    {
                        rt[i].dist[j] = rt[i].dist[k] + rt[k].dist[j]; rt[i].from[j] = k;
```

```
                                count++;
                            }
                        }
                    }
                }
        }while (count != 0);

        for (i = 0; i < nodes; i++)
        {
            cout << "\n For router" << i + 1 << " "; for (j = 0; j < nodes; j++)
                cout << "\n Node " << j + 1 << " via " << rt[i].from[j] + 1 << " Distance " <<
rt[i].dist[j];
        }
        cout << ("\n\n");
        return 0;
}
```

- **OUTPUT**

```
Enter the number of nodes : 4

Enter the cost matrix :
0 3 5 8
4 0 6 2
9 1 0 3
6 7 8 0

 For router1
 Node 1 via 1 Distance 0
 Node 2 via 2 Distance 3
 Node 3 via 3 Distance 5
 Node 4 via 2 Distance 5
 For router2
 Node 1 via 1 Distance 4
 Node 2 via 2 Distance 0
 Node 3 via 3 Distance 6
 Node 4 via 4 Distance 2
 For router3
 Node 1 via 2 Distance 5
 Node 2 via 2 Distance 1
 Node 3 via 3 Distance 0
 Node 4 via 4 Distance 3
 For router4
 Node 1 via 1 Distance 6
 Node 2 via 2 Distance 7
 Node 3 via 3 Distance 8
 Node 4 via 4 Distance 0


...Program finished with exit code 0
Press ENTER to exit console.
```

- **DISCUSSION**

In Distance Vector Routing:

1. Only distance vectors are exchanged.
2. “Next hop”values are not exchanged. This is because it results in exchangingthe large amount of data which consumes more bandwidth.
3. While preparing a new routing tableA router takes into considerationonlythe distance vectors it has obtained from its neighboring routers.
4. t does not take into consideration its old routing table.

The algorithm keeps on repeating periodically and never stops. This is toupdate the shortest path in case any link goes down or topology changes.

Routing tables are prepared total (n-1) times if there are n routers in the given network. This is because shortest path between any 2 nodes contains at most n-1 edges if there are n nodes in the graph.

- **FINDINGS AND LEARNINGS**

We learnt about distance vector routing algorithm and its implementation andsignificance in computer networks.

PROGRAM - 8

- **AIM**

Write a program to implement link state routing algorithm..

- **THEORY**

The Link state routing is the second family of routing protocols. Whiledistancevector 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 –

1. Link state packet – Asmall packet that contains routing information.
2. Link state database – A collection information gathered from link statepacket.
3. Shortest path first algorithm (Dijkstra algorithm) – A calculation performedon the database results into shortest path.
4. Routing table – A list of known paths andinterfaces.

- **ALGORITHM**

To find shortest path, each node needs to run the famous Dijkstra algorithm.This famous algorithm uses the following steps:

1. The node is taken and chosen as a root node of the tree, this creates the tree with a single node, and now set the total cost of each node to some value basedon the information in Link State Database.
2. Now the node selects one node, among all the nodes not in the tree like structure, which is nearest to the root, and adds this to the tree. The shapeofthe tree gets changed.
3. After this node is added to the tree, the cost of all the nodes not in the treeneeds to be updated because the paths may have been changed.
4. The node repeats the Step 2. and Step 3. until all the nodes are added in thetree.

- **CODE**

```
#include <bits/stdc++.h>


#define ll long long int
```

```
#define pb push_back
#define ff first
#define ss second
#define mod 1000000007
using namespace std;
int main()
{
	ios_base::sync_with_stdio(false); cin.tie(NULL);
	int count, src_router, i, j, k, w, v, min;
	int cost_matrix[100][100], dist[100], last[100]; int flag[100];
	cout << "\nEnter the no of routers : "; cin >> count;
	cout << "\nEnter the cost matrix values :\n";
	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 << "\nEnter 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] && 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 << " :\nPath taken : " << i; w = i;
            while (w != src_router)
            {
                cout << "<--" << last[w]; w = last[w];
            }
            cout << "\nShortest path cost : " << dist[i];
        }
        return 0;
}
```

- **OUTPUT**