EC3401 NETWORKS AND SECURITY LAB

Regulation : 2021

Branch : B.E. - ECE

Year & Semester : II Year / IV Semester

DEPARTMENT OF ECE

GOVERNMENT COLLEGE OF ENGINEERING

TIRUNELVELI

GOVERNMENT COLLEGE OF ENGINEERING
TIRUNELVELI - 627 007

2024 - 2025

Register Number:

CERTIFICATE

This is a bonafide record of work done by ……
…….……
Government College of Engineering, Tirunelveli during the year 2024-2025.

Place: Tirunelveli
Date:

Staff Incharge **Head of Department**

Submitted for the Anna University Practical Examination held at Government College of Engineering, Tirunelveli on …….

Internal Examiner **External Examiner**

S.NO	Date	Name of the experiment	Marks	Signature
1		Study Of C Programming		
2		Datalink layer framing methods Bit stuffing Character stuffing		
3		Error detection and correction technique Longitudinal Redundancy Check Cyclic Redundancy Check Hamming Technique		
4		Stop and Wait Protocol Sliding Window Protocol		
5		Go Back -N Protocol Selective Repeat Protocol		
6		Distance Vector Routing Algorithm Routing Information Protocol Bellman Ford Protocol		
7		Link State Routing Algorithm		
8		Data Encryption and Decryption Using Data Encryption Standard Algorithm		
9		Data Encryption and Decryption Using RSA Algorithm		
10		Client Server Model Using FTP Protocol		
11		Network Topology–Star, Bus And Ring Using NS2		
12		Operation Of CSMA/CD And CSMA/CA Using NS2		

Ex No.: 1

Date:

STUDY OF C PROGRAMMING

AIM:

To study the significant features of C programming.

INTRODUCTION:

C programming language is powerful and widely used procedural programming language initially developed by Dennis Ritchie early in the 1970's at Bell labs known for its efficiency, versatility and portability.C has been the foundation for numerous operating systems, System Software and applications.

HISTORY AND EVOLUTION:

C emerged from the need to develop the unix operating system, with its predecessor being the C programming language. Its design aimed to provide low-level access to memory, straight forward language constructs, and efficiency.Over the years, C has undergone various changes and becoming widely adopted standards.

KEY FEATURES:

1.Procedural Language: follows a procedural programming paradiagm, focusing on functions and procedures to accomplish tasks.

2.Efficiency:C provides direct access to hardware, efficiency memory management and minimal runtime overhead, making it suitable for system programming and embedded system

3.Portability: C programs can be compiled to run on different platforms with minimal modifications, that to its standardized syntax and libraries.

4.Extensibility: C allows for the creation of libraries and modules, facilitating code reuse and modularity.

5. Versatility: C is used in a wide range of applications, including system programming, device drivers, compilers, embedded system and high performance computing.

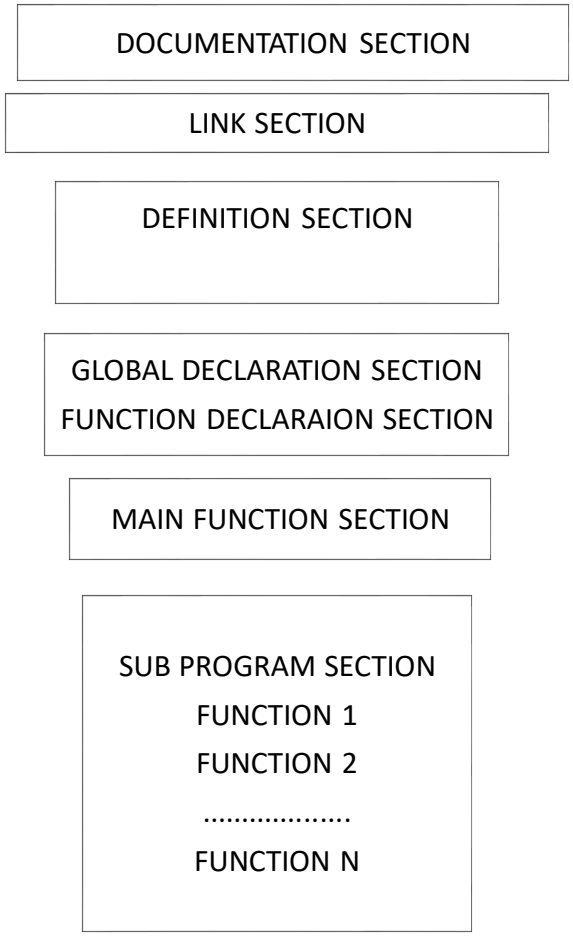
SYNTAX AND STRUCTURE:

C syntax is relatively simple and constant with a focus or readability efficiency. Programs are organised into function, each containing statements to perform specific tasks. Control Structures such as loops and conditionals allow for decision making and iteration. Data types, including integers, floating point numbers, characters and arrays provide flexibility in responding different types of data.

MEMORY MANAGEMENT:

C provides manual memory management through functions like malloc() and free() allowing developers precise control over memory allocation and deallocation.While offering flexibiling, this approach careful handling to avoid memory leaks and segmentation faults.

STRUCTURE OF PROGRAM:



STANDARED LIBRARIES:

C standard library provides a set of functions for common tasks like input /output operations / string manipulation mathematical computations and memory management. Additionally platform specific libraries extend is capabilities for interacting with the underlying system.

CONCLUSION:

C programming language remains a cover stone in software development appreciated for its efficiency, versatility and low-level control. Despite the emergence of higher-level languages,C continues to be indispensable in various domains, laying the groundwork as a fundamental skill for programmers worldwide.

RESULT:

Thus the study of C programming was written and understood successfully.

Ex No.:2A

Date:

DATA LINK LAYER FRAMING METHODS-BIT STUFFING

AIM:

To implement the data link layer framing using bit stuffing using C programming.

SOFTWARE REQUIRED:

- Turbo C++ Software
- Dev C++ Software

THEORY:

In the OSI model, the data link layer divides the data received from the physical layer into data frames. Data frame is a sequence of bits. It has 4 parts namely, Header, payload, Trailer and Flag. Data frame is of 2 types. They are fixed length and variable length data frame. Bit stuffing is used for management purposes to add some extra bits in payload of frames. These bits don't carry any information and used to mark the end of bit stream. Bit Stuffing is a process of inserting an extra bit as 0, once the frame sequence encountered 5 consecutive 1'S.

ForExample:

Let an arr[]be {1,1,1,1,1,1}. Then the output will be 1111101.

Similarly,

Let an arr[]be {1,0,1,0,1,0}. Then the output will be 101010.

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

OUTPUT:

```
Enter frame size: 5
Enter the frame in the form of 0 and 1:
1
1
1
1
1
After Bit Stuffing: 111110
```

PROGRAM:

```
#include<stdio.h>
#include<string.h>
int main()
{
int a[20],b[30],i,j,k,count,n;
printf("Enter frame size :");
scanf("%d",&n);
printf("Enter the frame in the form of 0 and 1:");
for( i=0;i<n;i++)
        scanf("%d",&a[i]);
i=0;
count=1;
j=0;
while(i<n)
{
        if( a[i]==1)
        {
        b[j] = a[i]
        for(k=i+1; a[k]==1&&k<n&&count<5; k++)
        {
        j++;;
        b[j]=a[k];
        count++;
        if(count==5)
        {
        j++;
                b[j]=0;
        }
        i=k;
        }
        }
```

```
else
{
b[j]=a[ i];
}
i++;
j++;
}
printf("After Bit Stuffing :");
for(i=0; i<j; i++)
        printf("%d",b[i]);
return 0;
}
```

RESULT:

Thus the program to implement data link layer framing using bit stuffing was written,executed and output was verified successfully.

Ex No.: 2B

Date:

DATA LINK LAYER FRAMING METHODS – CHARACTER STUFFING

AIM:

To write a C program to implement data link layer framing method using Character Stuffing.

Software Required:

- Turbo C++ Software
- Dev C++ Software

Theory:

Character stuffing is also known as byte stuffing or character-oriented framing and is same as that of bit stuffing but byte Stuffing actually operates on bytes whereas bit stuffing operates on bits. In byte stuffing special byte that is basically known as ESC (Escape Character) that has predefined pattern is generally added to data section of the data stream or frame when there is message or character that has same pattern as that of flag byte.

But receiver removes this ESC and keeps data part that causes some problems. In simple words, we can say that character stuffing is addition of 1 additional byte, if there is presence of ESC or flag in text.

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

PROGRAM:

```
#include <stdio.h>
#include <string.h>
#include <conio.h> // getche() is in conio.h
int main() {
```

OUTPUT:

Enter string:MLRITM

Enter position: 2

Enter the character: dldl

Frame after stuffing:dTestXMdldlLRITMdTestX

```
int i = 0, j = 0, n, pos;
char a[20], b[50], ch;
printf("Enter string: ");
scanf("%s", a);
   n = strlen(a);
printf("Enter position: ");
scanf("%d", &pos);
   // Validate position
while (pos > n || pos < 1) {
printf("Invalid position, enter again: ");
scanf("%d", &pos);
   }

printf("Enter the character: ");
ch = getche();
printf("\n");
   // Initial stuffing
b[0] = 'd';
b[1] = 'T';
b[2] = 'e';
   j = 3; // Start filling from index 3
for (i = 0; i < n; i++) {
          if (i == pos - 1) { // Insert character at given position
b[j] = ch;
j++;
       }
b[j] = a[i]; // Copy original characters
j++;
   }
   // Add stuffing at the end
b[j] = 'd';
b[j + 1] = 'l';
```

```
b[j + 2] = 'e';
b[j + 3] = 'e';
b[j + 4] = 't';
b[j + 5] = 'x';
b[j + 6] = '\0'; // Null terminator for string
printf("\nFrame after stuffing:\n");
printf("%s\n", b);
return 0;
}
```

RESULT:

Thus the C program to implement data link layer framing using character stuffing was written,executed and the output was obtained.

EX.NO: 3A

DATE:

ERROR DETECTION /CORRECTION TECHNIQUES – LRC

AIM:

To write a C program to implement error detection and correction using longitudinal redundancy check.

SOFTWARE REQUIRED:

- Turbo C++ Software
- Dev C++ Software

THEORY:

Longitudinal Redundancy Check (LRC) is also known as 2D parity check. In this method, data which the user wants to send is organized into rows and columns. A block of bit is divided into table or matrix of rows and columns. In order to detect an error, a redundant bit is added to the whole block and this block is transmitted to receiver. The user uses this redundant row to detect error. After checking the data for errors, receiver accepts the data and discards the redundant rows of bits.

For example: If a block of 32 bits is to be transmitted, it is divided into 4 rows and 8 columns which is shown in the following figure. In this matrix of bits, a parity bit (odd or even) is calculated for each column. It means 32 bits data plus 8 redundant bits are transmitted to receiver. Whenever data reaches at the destination, receiver uses LRC to detect error in data.

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

PROGRAM:

```
#include <stdio.h>

#include <string.h>

int main()

{

int l1, bit[100], count = 0, choice;
```

OUTPUT:

Enter the length of data stream: 10

Enter the data stream: 1

1

0

1

0

1

1

1

1

0

Number of 1's are 7

Enter the choice to implement parity bit

1-Sender side

2-Receiver side

1

The data stream after adding parity bit is:

1 1 0 1 0 1 1 1 1 0 1

```
printf("Enter the length of data stream: ");
scanf("%d", &l1);
printf("\nEnter the data stream: ");
for(int i = 0; i < l1; i++) {
scanf("%d", &bit[i]);
if(bit[i] == 1)
count++;
   }
printf("Number of 1's are %d\n", count);
printf("\nEnter the choice to implement parity bit");
printf("\n1-Sender side\n2-Receiver side\n");
scanf("%d", &choice);
switch(choice)
   {
case 1:
if(count % 2 == 0)
bit[l1] = 0;
else
bit[l1] = 1;
printf("\nThe data stream after adding parity bit is:\n");
for(int i = 0; i <= l1; i++)
printf("%d ", bit[i]);
break;
case 2:
if(count % 2 == 0)
printf("There is no error in the received data stream\n");
else
printf("There is an error in the received data stream\n");
break;
default:
printf("Invalid choice\n");
break;
```

```
    }
getchar();
return 0;
}
```

RESULT:

Thus the C program to implement error detection and correction using longitudinal redundancy check was written , executed and the output was verified.

EX.NO:3B

DATE:

ERROR DETECTION /CORRECTION TECHNIQUES – CRC

AIM:

To write a C program to implement error detection and correction using cyclic redundancy check.

SOFTWARE REQUIRED:

- Turbo C++ Software
- Dev C++ Software

THEORY:

CRC or cyclic Redundancy check is a method of detecting accidental changes/ errors in the communication change. CRC users Generator Polynomial which is available on both sender and receiver side.

An example generator polynomial is of the form x^2+x+1. This generator polynomial represents key 1011.

SENDER SIDE (Generation of Encoded Data for Data and Generator Polynomial or key):

- The binary data is first augmented by adding k-1 zeroes in the end of the data.
- Use MODULO-2 BINARY DIVIDION to divide binary data by the key and store remainder of division.

3. Append the remainder at the end of the data to firm the encoded data and send the same.

RECEIVER SIDE (Check if there are errors introduced in transmission):

- Perform the modulo-2 division again and if the remainder is 0, then there are no errors.

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

OUTPUT:

Enter data: 1101

Enter Key : 10

Quotient is : 101

Remainder is: 10

Final Data is: 11010

PROGRAM:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main() {
        int i,j,keylen,msglen;
        char input[100], key[30],temp[30],quot[100],rem[30],key1[30];
        clrscr();
        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]);
        getch();
}
```

RESULT:

Thus the C program to implement error detection and correction using cyclic redundancy check was written, executed and the output is verified.

EX.NO.:3C

DATE:

ERROR DETECTION /CORRECTION TECHNIQUES – HAMMING CODE

AIM:

To write a c program to implement error detection and correction using hamming code.

SOFTWARE REQUIRED:

- Turbo C++ Software
- Dev C++ Software

THEORY:

Hamming code is an error correction system that can detect and correct errors when data is stored or transmitted. It requires adding additional PARITY BITS with the data. It is commonly used in error correction mode (ECC) RAM. It's named after its Inventor, Richard W. Hamming. Hamming code uses a block parity mechanism. The amount of parity data added to the Hamming code is given by the formula 2^p> = d+p+1, where is the number of parity bits and d is the number of data bits. For example, if you wanted to transmit 7 data bits, the formula would be 2^4> =7+4+1, in 4 parity bits required.

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

PROGRAM:

```
#include<stdio.h>
#include<math.h>
int input[32];
int code[32];
int ham_calc(int,int);
void main()
{
int n,i,p_n=0,c_l,j,k;
printf("Please enter the length of the Data Word: ");
```

OUTPUT:

```
Please enter the length of the Data word:
2
Please enter the Data word:
1
0
The calculated code word is: 11100
Please enter the received code word:
1
1
1
0
0
The received code word is correct
```

```
for(i=0;i<n,i++)
        {
        scanf("%d",&input[i]);
        }
        I=0;
        while(n>(int)pow(2,i)-(i+1))
        {
        p_n++;
        i++;
        }
        c_l=p_n+n;
        j =k= 0;
        for(i=0;i<c_1;i++)
        {
                if(i == ((int)pow(2,k)-1))
                {
                code[i]=0; k++;
       }
                else
                {
                        code[i]=input[]];
                        j++;
                }
        }
        for(i=0;i<p_n;i++)
        {
                int position = (int)pow(2,i];
                int value = ham_calc(position,c_l);
```

```
            code[position-1]=value;
        }
        printf("\nThe calculated Code Word is: ");
        for(i=0;i<c_l;i++)
            printf("%d",code[i]);
        printf("\n");
        printf("Please enter the received Code Word:\n");
        for(i=0;i<c_l;i++)
            scanf("%d",&code[i]);
        int error_pos = 0;
        for(i=0;i<p_n;i++)
        {
            int position = (int)pow(2,i);
           int value = ham_calc(position,c_I);
            if (value != 0)
                    error_pos+=position;
        }
        if(error_pos == 1)
            printf("The received Code Word is correct. \n");
        else
            printf(" Error at bit position: %d\n", error_pos);
}
int ham_calc(int position,int c_l)
{
        int count=0,i,j;
        i=position-1;
        while(i<c_l)
```

```
        {
                        for(j=i;j<i+position;j++)
                        {
                                if(code[j] == 1)
                                        count++;
}
                        i=i+2*position;
                        }
                        if(count%2 == 0)
                                return 0;
                        else
                                return 1;
        }
```

RESULT:

Thus the C program to implement error detection and correction using hamming code was written, executed and the output is verified.

EX. NO.:4A

DATE:

STOP AND WAIT PROTOCOL

AIM:

To write a C program to implement stop and wait protocol.

SOFTWARE REQUIRED:

- Turbo C++ Software
- Dev C++ Software

THEORY:

Stop-and-wait ARO, commonly known as the alternating bit protocol, refers to a communication technique used to transmit data between two linked devices It makes sure that packets are received in the right order, and that data is not lost as a result of dropped packets.

It is a DDL Protocol that is used to send data through channels with no background noise. It offers unidirectional data transfer, which means that only one of the two operations, data sending or receiving can occur concurrently. Although it offers a flow control system, there is no error control mechanism.

The concept behind using this frame is that after sending one frame, the sender will wait for an acknowledgement before sending another one.

SENDER'S SIDE:

Rule 1: The sender sends one data packet at a time.
Rule 2: The sender only sends the subsequent packet after getting the preceding packet's acknowledgement.

RECEIVER'S SIDE:

Rule 1:Receive the data packet, then consume it.
Rule 2:The receiver provides the sender with an acknowledgment after consuming the data packet.

As a result, the stop and wait protocol's basic tenet on the receiver's end is similarly extremely straight forward, Invent the packet and after it has been consumed, send the acknowledgement this is a mechanism for flowcontrol.

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

OUTPUT:

Enter no of frames: 3

frame 0 has been transmitted

frame 1 has been transmitted

frame 2 has been transmitted

please enter the last acknowledgement received 1

frame 1 has been transmitted

frame 2 has been transmitted

please enter the last acknowledgement received 2

frame 2 has been transmitted

please enter the last acknowledgement received 3

PROGRAM:

```
#include<stdio.h>
int main()
{
int framesize, sent=0,ack,i;
printf("Enter no of frames: \n");
scanf("%d",&framesize);
while(1)
{
for(i=0;i<framesize;i++)
{ printf("frame %d has been transmitted\n",sent);
sent++;
if(sent==framesize)
break; }
printf("please enter the last acknowledgement received\n");
scanf("%d", &ack);
if(ack>=framesize)
break;
else
sent= ack;
}
return 0;
}
```

RESULT:

Thus the C program to implement stop and wait protocol was written, executed and the output was verified.

Ex No.:4B

Date:

SLIDING WINDOW PROTOCOL

AIM:

To write a C program to implement sliding window protocol.

SOFTWARE REQUIRED:

- Turbo C++ Software
- Dev C++ Software

THEORY:

The sliding window is a technique for sending multiple frames at a time. It controls the data packets between the two devices where reliable and gradual delivery of data frames is needed. It is also used in TCP.

In this technique, each frame has sent frame. The sequence numbers are used to find the missing data in the receiver and the purpose of the sliding window technique is to avoid duplicate data, so it uses the sequence number.

Sliding window protocol has two types:

- Go-Back-N ARQ
- Selective Repeat ARQ

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

OUTPUT:

Enter window size: 5

Enter number of frames to transmit: 3

Enter 3 frames:

34

34

65

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

After sending %d frames at each stage sender waits for acknowledgement sent by the receiver

34

34

65

Acknowledgement of above frames sent is received by sender

PROGRAM:

```
#include<stdio.h>
int main()
int w,i,f,frames[50];
printf("Enter window size: ");
scanf("%d",&w);
printf("\nEnter number of frames to transmit: ");
scanf("%d",&f);
printf("\nEnter %d frames: ",f);
for(i=1;i<=f;i++)
        scanf("%d", &frames[i]);
printf("\nWith sliding window protocol the frames will be sent in the following manner
        (assuming no corruption of frames)\n\n");
printf("After sending %d frames at each stage sender waits for acknowledgement sent by the
        receiver\n\n",w);
for(i=1;i<=f;i++)
{
        if(i%w==0)
        {
        printf("%d\n",frames[i]);
        printf("Acknowledgement of above frames sent is received by sender\n\n");
}
else
        printf("%d",frames[i]);
}
if(f%w!=0)
printf("\nAcknowledgement of above frames sent is received by sender\n");
return 0;
}
```

RESULT:

Thus the C program to implement sliding window protocol was written, executed and the output was verified.

Ex No.: 5A

Date:

GO BACK – N PROTOCOL

AIM:

To write a C program to implement Go-Back-N Protocol.

SOFTWARE REQUIRED:

- Turbo C++ Software
- Dev C++ Software

THEORY:

Go-Back-N ARQ Protocol is also known as Go-Back-N Automatic Repeat Request. It is a data link layer protocol that uses a sliding window method. In this, if any frame is corrupted or lost, all subsequent frames have to be sent again.

The Size of the sender window is N is this protocol. For example, Go - Back -8, the size of the sender window, will be 8 the receiver window size is always 1.

If the receiver a corrupted frame, it cancels it. The receiver does not accept a corrupted frame. When the timer expires, the sender sends the correct frame again.

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

PROGRAM:

```
#include<stdio.h>
int main()
{
    int windowsize, sent=0,ack,l;
    printf("Enter window size:\n");
    scanf("%d",&windowsize);
    while(1)
```

OUTPUT:

Enter window size: 3

Frame 0has been transmitted.

Frame 1has been transmitted.

Frame 2has been transmitted.

Please enter the last Acknowledgement received: 1

Frame 1has been transmitted.

Frame 2has been transmitted.

Please enter the last Acknowledgement received: 2

Frame 2has been transmitted.

Please enter the last Acknowledgement received: 3

```
        {
        for(i=0; i < windowsize; i++)
        {
        printf("Frame %d has been transmitted.\n",sent);
        sent++;
        if(sent==windowsize)
        break;
        printf("\nPlease enter the last Acknowledgement received:\n");
        scanf("%d",&ack);
        if(ack==windowsize)
        break;
        else
        sent=ack;
        }
return 0;
}
```

RESULT:

Thus the C program to implement Go-Back-N Protocol was written, executed and the output was verified.

Ex No.: 5B

Date:

SELECTIVE REPEAT PROTOCOL

AIM:

To write a C program to implement selective repeat protocol.

SOFTWARE REQUIRED:

- Turbo C++ Software
- Dev C++ Software

THEORY:

Selective Repeat ARQ is also known as the Selective Repeat Automatic Repeat Request. It is a data link layer protocol that uses a sliding window method. The Go-Back-N Protocol works well if it has fewer errors. But if there is a lot of error in the frame ,lots of Bandwidth loss in sending the frames again. So, we use the selective Repeat ARQ Protocol. In this protocol the size of the sender windows always equal to the size of the receiver window. The size of the sliding window is always greater than 1.

If the receiver receives a corrupt frame, it does not directly discard it. It sends a negative acknowledgement to the sender. The sender sends that frame again as soon as receiving negative acknowledgement. There is no waiting for any time-out to sent that frame.

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int n,r;
struct frame
{
char ack;
int data;
```

OUTPUT:

Enter no. of frames to be sent: 4

Enter data for frame[1]:10

Enter data for frame[2]:20

Enter data for frame[3]:30

Enter data for frame[4]:40

The frame no.3 not received

Resending frame no.3

Received frame is 30

Frames sent successfully.

```
}
frm[10];
int sender(void);
void recvack(void);
void resend(void);
int main()
{
sender();
recvack();
resend();
printf("\n Frames sent successfully");
}
int sender()
{
int i;
printf("\n Enter no. of frames to be sent:");
scanf("%d",&n);
for(i=1; i<=n; i++){
printf("\n Enter data for frame[%d]:",i);
scanf("%d", &frm[i].data);
frm[i].ack='y';
}
return 0;
}
void recvack()
{
int i;
rand();
r=rand()%n;
frm[r].ack=='n';
for(i=1; i<=n; i++){
if(frm[i].ack=='n')
```

```
printf("\n The frame no.%d not received",r);
}
}
void resend()
{
int i;
printf(“\n Resending frame no.%d”,r);
frm[i].ack=’y’;
printf(“\n Received frame is %d”, frm[r].data);
}
```

RESULT:

Thus the C program to implement selective repeat protocol was written, executed and the output was verified.

Ex No.:6A

Date:

DISTANCE VECTOR ROUTING ALGORITHM – ROUTING INFORMATION PROTOCOL

AIM:

To write a C program to implement distance vector routing algorithm using Routing Information Protocol.

SOFTWARE REQUIRED:

- Turbo C++ Software
- Dev C++ Software

THEORY:

Routing Information Protocol (RIP) is a distance vector protocol that uses hop count as its vector primary metric. RIP defines how routers should share information when moving traffic among an interconnected group of local area networks.

In the enterprise, open shortest path first (OSPE) routing has largely replaced RIP as the most widely used Interior Gateway Protocol. RIP has been supplanted mainly due to its simplicity and inability to scale to very large and complex networks. Border Gateway Protocol (BGP) is another distance vector protocol that is now used to transfer routing information across autonomous systems on the internet.

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

PROGRAM:

```
#include<stdio.h>
#include<string.h>
int main()
{
int count,src_router,i,j,k,w,v,min;
int cost_matrix[100][100], dist[100], last[100];
int flag[100];
```

OUTPUT:

```
Enter the no of routers: 3
Enter the cost matrix values:
0->0:4
0->1:5
0->2:3
1->0:5
1->1:6
1->2:2
2->0:1
2->1:7
2->2:3
Enter the source router: 1
1==>0:path taken=0
<--2
<--11
Shortest path cost:3
1==>1:path taken:1
Shortest path cost:6
1==>2:path taken:2
<--1
Shortest path cost:2
```

```
printf("\n Enter the no of routers:");
scanf("%d",&count);
printf("\nEnter the cost matrix values:");
for(i=0;i<count;i++)
{
for j=0;j<count;j++)
{
printf("\n%d->%d:",i,j);
scanf("%d", &cost_matrix[i][j]);
if(cost_matrix[i][j]<0)
cost_matrix[i][j]=1000;
}
}
printf("Enter the source router:");
scanf("%d", &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++)
{
printf("\n%d==>%d:path taken:%d",src_router,i,i);
w=i;
while(w!=src_router)
{
printf("\n<--%d", last[w]);
w=last[w];
}
printf("\nShortest path cost:%d", dist[i]);
}
}
```

RESULT:

Thus the C program to implement distance vector routing algorithm using Routing Information Protocol was written, executed and the output was verified.

Ex. No.:6B

Date:

DISTANCE VECTOR ROUTING ALGORITHM – BELLMAN FORD PROTOCOL

AIM:

To write a C program to implement distance vector routing algorithm using Bellman-Ford Protocol.

SOFTWARE REQUIRED:

- Turbo C++ Software
- Dev C++ Software

THEORY:

Bellman Ford algorithm is a single-source shortest path algorithm. This algorithm is used to find the shortest distance from the single vertex to all the other vertices of a weighted graph. There are the various other algorithms used to find the shortest path like Dijkstra algorithm etc. If the weighted graph contains the negative weight values, then the Dijkstra algorithm does not confirm whether it produces the correct answer or not. In contrast to Dijkstra algorithm, Bellman Ford algorithm guarantees the correct answer even if the weighted graph contains the negative weight values.

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

PROGRAM:

```
#include<stdio.h>
struct node
{
unsigned dist[20];
unsigned from[20];
}rt[10];
int main()
{
int costmat[20][20];
```

OUTPUT:

```
Enter no of nodes:3
Enter the cost matrix:
023
201
310
for router 1
node 1 via 1 distance 0
node 2 via 2 distance 2
node 3 via 3 distance 3
for router 2
node 1 via 1 distance 2
node 2 via 2 distance 0
node 3 via 3 distance 1
for router 3
node 1 via 1 distance 3
node 2 via 2 distance 1
node 3 via 3 distance 0
```

```
int nodes,i,j,k,count=0;
printf("\nEnter no of nodes:");
scanf("%d", &nodes);
printf("\nEnter the cost matrix:\n");
for(i=0;i<nodes;i++)
{
for(j=0;j<nodes;j++)
{
scanf("%d",&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++)
{
printf("\t\n node %d via %d distance %d", j+1,rt[i].from[j]+1,rt[i].dist[j]);
```

```
}
}
printf("\n\n");
return 0;
}
```

RESULT:

Thus the C program to implement distance vector routing algorithm using Bellman-Ford was Protocol, written, executed and the output was verified.

Ex. No.:7

Date:

LINK STATE ROUTING ALGORITHM

AIM:

To write a C program to implement link state routing algorithm.

SOFTWARE REQUIRED:

- Turbo C++ Software
- Dev C++ Software

THEORY:

Link State routing is a method in which each router shares its neighbourhood's knowledge with every other router in the internetwork. In this algorithm, each router in the network understands the network topology then makes a routing table depend on their topology. Each router will share data about its connection to its neighbour, who will construct a topology of the network. In LSP, each node transmits its signature .Neighbour determine the signature and maintain a record of the combining IP address and the MAC. The Neighbour Lookup Protocol of LSP deserves and maintains the MAC and IP address of every network frame accepted by a node. The extracted data can support the mapping of MACs and IP addresses. The link state flooding algorithm prevents the general issues of broadcast in the existence of loops by having every node maintain database of all LSP message.

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

PROGRAM:

```
#include<stdio.h>

#include<string.h>

int main()

{

    int count,src_router,i,j,k,w,v,min;

    int cost_matrix[100][100],dist[100],last[100];
```

OUTPUT:

Enter the no of routers:3

Enter the cost matrix values:

0→0:1

0→1:2

0→2:3

1→0:4

1→1:5

2→0:7

2→1:9

2→2:8

Enter the source router:3

3➔0:path taken=0

🡐3

Shortest path cost=0

3➔1:path taken=1

🡐3

Shortest path cost=0

3➔2:path taken=2

🡐3

Shortest path cost=0

```
intflag[100];
 printf("\nEnter the no of routers:");
scanf("%d", &count);
printf("\nEnter the cost matrix values:");
for(i=0;i<count;i++){
        for(j=0;j<count;j++){
                printf("\n%d→%d",i,j);
                scanf("%d", &cost_matrix[i][j]);
                if(cost_matrix[i][j] <0)
                cost_matrix[i][j] = 1000
        }
}
printf("\nEnter the source router:");
scanf("%d", &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(min+cost_matrix[v][w]<dist[w])
                    {
                        dist[w]=min+cost_matrix[v][w];
                        last[w]=v;
                    }
                }
}
for(i=0;i<count;i++)
{
printf("\n%d==>%d:path taken:%d",src_router,i,i);

                    w=i;
                    while(w!=src_router)
                    {
                        printf("\n<--%d",last[w]);w=last[w];
                    }
                    printf("\nShortest path cost:%d",dist[i]);
                }
        }
```

RESULT:

Thus the C program to implement Link State Routing algorithm was written, executed and the output was obtained.

Ex. No.:8

Date:

DATA ENCRYPTION AND DECRYPTION

USING DATA ENCRYPTION STANDARD ALGORITHM

AIM:

To write a C program for performing encryption and decryption of given data using DES algorithm.

SOFTWARE REQUIRED:

- Turbo C++ Software
- Dev C++ Software

THEORY:

Data encryption standard (DES) has been found vulnerable to very powerful attacks and therefore, the popularity of DES has been found slightly on the decline. DES is a block cipher and encrypts data in blocks of size of 64 bits each, which means 64 bits of plain text go as the input to DES, which produces 64 bits of cipher text. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits.

Let us now discuss the broad level step in DES.

- In the first step, the 64-bit plain text block is handed over to an initial permutation (IP).
- Next the initial permutation products two halves of the permuted block, saying Left Plain Text (LPT) and Right Plain Text (RPT).
- Now each LPT and RPT go through 16 rounds of the encryption process.
- In the end, LPT and RPT are rejoined and a Final permutation (FP) is performed on the combined. block
- The result of this process produces 64-bit cipher text.

ALGORITHM:

1. Ask the user to enter two prime numbers and validate them.
2. Store the prime numbers in variables.
3. Compute n=pq.
4. Compute $\lambda(n)=(p-1)(q-1)$.
5. Choose a random number as a relatively prime number to $\lambda(n)$ and $1<e<\lambda(n)$.
6. Compute d=e-mod $\lambda(n)$.
7. Print the public and private keys.
8. Ask the user to enter a message and store it in a variable.
9. Encrypt the message using the public key.
10. Decrypt the message using the private key.
11. Print the encrypted and decrypted message.

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

PROGRAM:

```
#include<stdio.h>
int main()
{
int i,x;
char str[100];
printf("\n Please enter the string:");
gets(str);
printf("\n Please choose following options\n");
printf("1-encrypt the string\n2-decrypt the string\n");
scanf("%d", &x);
switch(x)
{
        case 1:
                for(i=0;(i<100&&str[i]!='\0');i++)
                        str[i]=str[i]+3;
                        printf("\n Encrypted string:%s\n",str);
                break;
        case 2:
                for(i=0;(i<100&&str[i]!='\0');i++)
                        str[i]=str[i]-3;
                        printf("\n Decrypted string:%s\n", str);
                break;
        default:
                printf("\nerror\n");
```

OUTPUT 1:

Encryption

Please enter the string: hello

Please choose following options

1-encrypt the string

2-decrypt the string

Encrypted string: khoor

OUTPUT 2:

Decryption

Please enter the string: khoor

Please choose following options

1-encrypt the string

2-decrypt the string

Decrypted string: hello

```
    }
    return 0;
}
```

RESULT:

Thus a C program for performing encryption and decryption of given data using DES Algorithm was written, executed and the output was verified.

Ex. No.:9

Date:

DATA ENCRYPTION AND DECRYPTION USING RSA ALGORITHM

AIM:

To write a C program to implement data encryption and decryption using RSA algorithm.

SOFTWARE REQUIRED:

- Turbo C++ Software
- Dev C++ Software

THEORY:

RSA is the most common public key algorithm named after its inventors Rivest, Shamir and Adelman, public key encryption algorithm is also called the Asymmetric algorithm. Asymmetric algorithms are those algorithms in which sender and receiver use different keys for encryption. Each sender is assigned a pair of keys

- public key
- private key

The public key is used for encryption, and the private key using a public key. The two keys are linked, but the private key cannot be derived from the public key the public key is well known, but the private key is secret and it is known only to the user who owns the key. It means that everybody can sends a message to the user using user's public key. But only user can decrypt the message using his private key.

ALGORITHM:

- Ask the user two prime numbers and validate them.
- Store the prime numbers in variables
- Compute n=pq.
- Compute $\lambda(n) = (p-1)(q-1)$.
- choose a random number as a relatively prime number to $\lambda(n)$ and $1<e<\lambda(n)$.
- Compute $d=e-1 \bmod \lambda(n)$.
- Print the public and private keys.
- Ask the user to enter a message and store it in available
- Encrypt the message using the public key.
- Decrypt the message using the private key.
- Print the encrypted and decrypted message.

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
long int p, q, n, t, flag, e[100], d[100], temp[100], j, m[100], en[100], i;
char msg[100];
int prime(long int);
void ce();
long int cd(long int);
void encrypt();
void decrypt();
int main()
{
printf("ENTER FIRST PRIME NUMBER: ");
scanf("%ld", &p);
flag = prime(p);
if (flag == 0 || p == 1) {
printf("WRONG INPUT\n");
exit(1);
   }
printf("ENTER ANOTHER PRIME NUMBER: ");
scanf("%ld", &q);
flag = prime(q);
if (flag == 0 || q == 1 || p == q) {
printf("WRONG INPUT\n");
exit(1);
   }
```

OUTPUT:

ENTER FIRST PRIME NUMBER: 17

ENTER ANOTHER PRIME NUMBER: 11

ENTER MESSAGE: hello

POSSIBLE VALUES OF e AND d ARE:

7	23
11	7
13	11
17	17
19	19
23	7
29	5
31	11
37	29
41	5
43	37
47	23
53	5
59	5
61	53
67	53
71	47
73	73
79	47
83	59
89	5
97	5

```
fgets(msg, sizeof(msg), stdin);
msg[strcspn(msg, "\n")] = '\0'; // remove newline if present
for (i = 0; i < strlen(msg); i++)
m[i] = msg[i];
    n = p * q;
  t = (p - 1) * (q - 1);
ce();
printf("\nPOSSIBLE VALUES OF e AND d ARE:\n");
for (i = 0; i < j; i++)
printf("%ld\t%ld\n", e[i], d[i]);
encrypt();
decrypt();
return 0;
}
int prime(long int pr)
{
int i;
if (pr == 1)
return 0;
for (i = 2; i <= sqrt(pr); i++) {
if (pr % i == 0)
return 0;
   }
return 1;
}
void ce()
{
int k = 0;
for (i = 2; i < t; i++) {
if (t % i == 0)
continue;
flag = prime(i);
```

THE ENCRYPTED MESSAGE IS:

ÍÝÝØØ

THE DECRYPTED MESSAGE IS:

hello

```
if (flag == 1 && i != p && i != q) {
e[k] = i;
flag = cd(e[k]);
if (flag > 0) {
d[k] = flag;
k++;
          }
       }
if (k == 99)
break;
  }
   j = k;
}
long int cd(long int x)
{
long int k = 1;
while (1) {
       k = k + t;
if (k % x == 0)
return (k / x);
   }
}
void encrypt()
{
long int pt, ct, key = e[0], k, len;
   i = 0;
len = strlen(msg);
printf("\nTHE ENCRYPTED MESSAGE IS:\n");
while (i < len) {
pt = m[i];
pt = pt - 96;
       k = 1;
```

```
for (j = 0; j < key; j++) {
            k = k * pt;
            k = k % n;
        }
temp[i] = k;
ct = k + 96;
en[i] = ct;
printf("%c", (char)en[i]);
i++;
    }
en[i] = -1; // Mark end of encrypted array
}
void decrypt()
{
long int pt, ct, key = d[0], k;
    i = 0;
printf("\n\nTHE DECRYPTED MESSAGE IS:\n");
while (en[i] != -1) {
ct = temp[i];
        k = 1;
for (j = 0; j < key; j++) {
            k = k * ct;
            k = k % n;
        }
pt = k + 96;
m[i] = pt;
printf("%c", (char)m[i]);
i++;
    }
m[i] = -1;
}
```

RESULT:

Thus the C program to implement data encryption and decryption using RSA algorithm was written, executed and the output was obtained.

Ex. No.:10

Date:

CLIENT SERVER MODEL USING FTP PROTOCOL

AIM:

To write a C program to implement client server model using FTP Protocol.

SOFTWARE REQUIRED:

- Turbo C++ Software
- Dev C++ Software

THEORY:

FTP is a network protocol for transmitting files between computers over transmission control protocol/Internet protocol connections within the TCP/IP suite, FTP is considered an application layer protocol. In an FTP transaction, the end user's computer is typically called the end user's computer is typically called the local host. The second computer involved in FTP is a remote host, which is usually a server. Both Computers need to be connected via a network and configured properly to transfer files via FTP servers and the client must have FTP software installed to access these services.

Although many file transfers can be conducted using Hypertext Transfer Protocol another protocol using TCP/IP suite.FTP is still commonly used to transfer files behind the scenes for other applications, such as banking services, It is also sometimes used to download new applications via web browser.

PROCEDURE:

Step 1: Open the required software Turbo C++/Dev C++.

Step 2: Open a new file.

Step 3: Type the given program.

Step 4: Save the source file with .c extension.

Step 5: Compile and run the program.

Step 6: Give the input and obtain the output.

PROGRAM 1:

```
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <stdio.h>
#include <net/if_arp.h>
```

OUTPUT 1:

Socket is created

Binded successfully

Receiving file: example.txt

The file has been transferred

```
#include <unistd.h> // For close()
int main()
{
int sd, b, cd;
char fname[50], op[1000];
struct sockaddr_in caddr, saddr;
   FILE *fp;
socklen_t clen = sizeof(caddr);
   // Create socket
sd = socket(AF_INET, SOCK_STREAM, 0);
if (sd != -1)
printf("Socket is created\n");
else {
printf("Socket is not created\n");
return 1;
   }
   // Setup server address
   saddr.sin_family = AF_INET;
   saddr.sin_port = htons(2500);
   saddr.sin_addr.s_addr = htonl(INADDR_ANY); // Corrected htoni -> htonl
   // Bind
  b = bind(sd, (struct sockaddr*)&saddr, sizeof(saddr));
if (b == 0)
printf("Binded successfully\n");
else {
printf("Binding failed\n");
return 1;
   }
   // Listen
listen(sd, 5);
   // Accept
cd = accept(sd, (struct sockaddr*)&caddr, &clen);
```

OUTPUT 2:

Enter the server IP address: 127.0.0.1

Socket created

Connected to server

Enter the file name: sample.txt

Enter the server IP address: 127.0.0.1

Socket created

Connected to server

Enter the file name: missing.txt

File not found!

Connection failed

```
if (cd < 0) {
printf("Accept failed\n");
return 1;
   }
   // Receive filename
recv(cd, fname, sizeof(fname), 0);
printf("Receiving file: %s\n", fname);
   // Open file to write
fp = fopen(fname, "w");
if (fp == NULL) {
printf("Failed to open file\n");
return 1;
   }
   // Receive data and write to file
int n;
while ((n = recv(cd, op, sizeof(op), 0)) > 0) {
fwrite(op, 1, n, fp);
   }
printf("The file has been transferred\n");
   // Close everything
fclose(fp);
close(cd);
close(sd);
return 0;
}
```

PROGRAM 2:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
```

```
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
int main() {
int sd, c;
char fname[50], sip[25], op[1000];
struct sockaddr_in caddr;
struct hostent *he;
   FILE *fp;
int n;

printf("Enter the server IP address: ");
scanf("%s", sip);
he = gethostbyname(sip);
if (he == NULL) {
printf("gethostbyname failed\n");
return 1;
   }
sd = socket(AF_INET, SOCK_STREAM, 0);
if (sd != -1)
printf("Socket created\n");
else {
printf("Socket is not created\n");
return 1;
   }
memset(&caddr, 0, sizeof(caddr));
   caddr.sin_family = AF_INET;
   caddr.sin_port = htons(2500);
   caddr.sin_addr = *((struct in_addr*)he->h_addr);
   c = connect(sd, (struct sockaddr*)&caddr, sizeof(caddr));
if (c == 0)
printf("Connected to server\n");
```

```
else {
printf("Connection failed\n");
close(sd);
return 1;
   }
printf("Enter the file name: ");
scanf("%s", fname);
send(sd, fname, sizeof(fname), 0);
fp = fopen(fname, "r");
if (fp == NULL) {
printf("File not found!\n");
close(sd);
return 1;
   }
while ((n = fread(op, 1, sizeof(op), fp)) > 0) {
send(sd, op, n, 0);
   }
fclose(fp);
close(sd);
return 0;
}
```

RESULT:

Thus the C program to implement client server model using FTP Protocol was written, executed and the output was verified.

Ex. No.:11

Date:

NETWORK TOPOLOGY – STAR, BUS AND RING USING NS2

AIM:

To study the network topologies – Star, Bus and Ring using NS2.

SOFTWARE REQUIRED:

- NS – 2

THEORY:

Bus Topology:

Token bus is a LAN protocol operating in the MAC layer. Token bus is standardized per IEEE 802.4. Token can operate at speeds of 5mbps, 10mbps and 20mbps. The operation of token bus is as follows; Unlike token ring, in token bus the ring topology is virtually created and maintained by the protocol. A node can receive data even if it is not part of the virtual ring, a node joins the virtual ring only if it has data to transmit. In token bus data is transmitted to the destination node only where as other control frames is hop to hop. After each data transmission there is a solicit successor control frame transmitted which reduces the performance of the protocol.

Ring Topology:

Token ring is a LAN protocol operating in MAC layer. Token ring is standardized as per IEEE 802.5. Token ring can operate at speeds of 4 mbps and 16mbps. The operation of token ring is as follow. When there is no traffic on the network a simple 3-byte token circulates the ring. If the token is free then the station may size the token and start sending the data frame. As the frame travels around the data frame. As the frame travels around the ring each Station examines the destination address and is either Forwarded or copied. After the frame makes a round trip the sender receives the frame and releases a new token onto the ring.

Star Topology:

Star networks are one of the most common computer network topologies. In its simple form a star network consists of one central switch, hub or computer, which acts as acondant to transmit message. This consists of a central node, to which all other nodes connected, this central node provides a common connection point for all nodes through a hub. In Star topology, every node is connected to a central node called a hub or switch.

The Switch is the server and the peripheral are the clients. Thus, the hub and leaf nodes, and the transmission lines between them, form a graph with the topology of a star. If the central node is passive, the originating node must be able to is passive, the originating node must be able to tolerate the reception of an echo of its own transmission delayed by the two way transmission time plus any delay generated in the central node. An active star network has an active central node that usually has the means to prevent echo-related problems. The Star topology reduces the damage caused by line failure by connecting all of the systems to a central node. When applied to a bus based network, this central hub rebroadcasts all transmissions to, and receiving from the modernly. The failure of a transmission line

OUTPUT 1: Bus Topology

n0 --- n1 --- n2 --- n3 --- n4

linking any peripheral node to the central node will result in the isolation of that peripheral node from all others, but the rest of the systems will be unaffected.

PROCEDURE:

Step 1: Create a simulator object.

Step 2: Define different colours for different data.

Step 3: Open a NAM trace file and define finish procedure then close the trace file, and execute NAM on trace file.

Step 4: Create five nodes that forms a network numbered forms a network to 4.

Step 5: Create duplex links between the nodes and add orientation to the nodes for setting a LAN topology.

Step 6: Setup TCP connection between n(1) and n(3).

Step 7: Apply CBR Traffic over TCP.

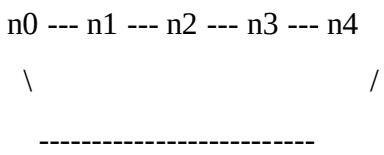
Step 8: Schedule events and run the program.

PROGRAM:

Bus Topology:

```
# Bus Topology in NS2

set ns [new Simulator]

# Open Trace File

set nf [open out.tr w]

$ns trace-all $nf

# Create Nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

# Create Links in a Line (Bus Simulation)

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

$ns duplex-link $n1 $n2 1Mb 10ms DropTail

$ns duplex-link $n2 $n3 1Mb 10ms DropTail

$ns duplex-link $n3 $n4 1Mb 10ms DropTail

# Traffic between ends
```

OUTPUT 2: Ring Topology

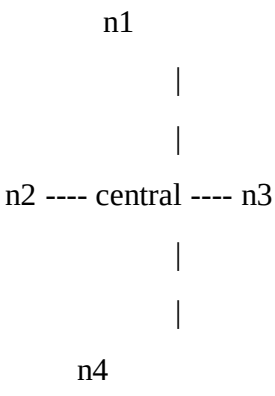


```
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
set null [new Agent/Null]
$ns attach-agent $n4 $null
$ns connect $udp $null
set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ 500
$cbr set interval_ 0.005
$cbr attach-agent $udp
$cbr start
# Finish
$ns at 5.0 "finish"
proc finish {} {
global ns nf
   $ns flush-trace
close $nf
exit 0
}
$ns run
```

Ring Topology:

```
# Ring Topology in NS2
set ns [new Simulator]
# Open Trace File
set nf [open out.tr w]
$ns trace-all $nf
# Create Nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
# Create Links in a Ring
```

OUTPUT 3: Star Topology



```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n0 1Mb 10ms DropTail ;# Connect back to form a ring
# Traffic from one node to another
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ 500
$cbr set interval_ 0.005
$cbr attach-agent $udp
$cbr start
# Finish
$ns at 5.0 "finish"
proc finish {} {
global ns nf
   $ns flush-trace
close $nf
exit 0
}
$ns run
```

Star Topology:

```
# Star Topology in NS2
set ns [new Simulator]
# Open Trace File
set nf [open out.tr w]
$ns trace-all $nf
# Create Nodes
```

```
set central [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
# Create Links (All nodes connect to central node)
$ns duplex-link $central $n1 1Mb 10ms DropTail
$ns duplex-link $central $n2 1Mb 10ms DropTail
$ns duplex-link $central $n3 1Mb 10ms DropTail
$ns duplex-link $central $n4 1Mb 10ms DropTail
# Create Traffic
set tcp [new Agent/TCP]
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n2 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp start
# Finish
$ns at 5.0 "finish"
proc finish {} {
global ns nf
   $ns flush-trace
close $nf
exit 0
}
$ns run]
```

RESULT:

Thus the network topologies star, bus and ring were studied using NS2.

Ex. No.:12

Date:

OPERATION OF CSMA/CD AND CSMA/CA USING NS2

AIM:

To implement and perform the operation of CSMA/CD and CSMA/CA using NS2.

SOFTWARE REQUIRED:

NS2

THEORY:

Ethernet is a LAN protocol operating at the MAC layer. Ethernet has been standardized per IEEE 802.3. The underlying protocol in Ethernet is known as the CSMA/CD-Carrier Sense Multiple Access/Collision Detection. The working of the Ethernet protocol is an explained below, A node which has data 60 transmit senses the channel. If the Channel is idle then, the data is transmitted. If the channel is busy then, the data station defers transmission until the channel is sensed to be idle and then immediately transmitted. If more than one nodes starts data transmission at the same data collides. This collision is heard by the transmitting nodes which enter into contention node phase. The contending nodes resolve contention using an algorithm called truncated binary exponential back off.

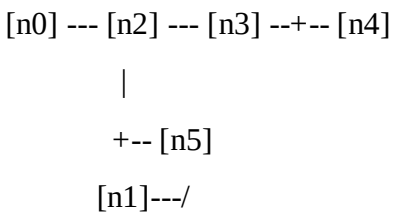
PROCEDURE:

1. Create a simulator object.
2. Define different colours for different data flows.
3. Open a NAM trace file, define finish procedure then close trace file, execute NAM on trace file.
4. Create six nodes that forms a network numbered from 0 to 5.
5. Create duplex links between the nodes, add orientation to nodes for setting a LAN topology.
6. Setup TCP connection between n(0) and n(4).

PROGRAM 1:CSMA/CA

```
# Create a simulator instance

set ns [new Simulator]

# Define different colors for data flows (for NAM)

$ns color 1 Blue

$ns color 2 Red

# Open the Trace files

set file1 [open out.tr w]

set winfile [open WinFile w]
```

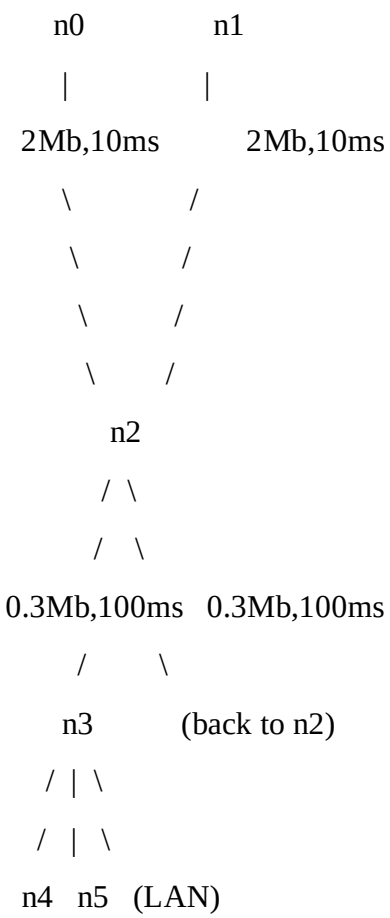
OUTPUT 1:



```
$ns trace-all $file1
# Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2
# Define a 'finish' procedure
proc finish {} {
global ns file1 file2
   $ns flush-trace
close $file1
close $file2
exec nam out.nam &
exit 0
}
# Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
# (Optional) set colors or shapes for NAM display (requires extensions)
#$n1 color red
#$n1 shape box
# Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
# Create LAN
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Ca Channel]
# Setup a TCP connection
```

```
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552
# Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type FTP
# Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_ 2
# Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type CBR
$cbr set packet_size 1000
$cbr set rate 0.01Mb
$cbr set random_ false
# Schedule events
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 124.0 "$ftp stop"
$ns at 124.5 "$cbr stop"
# Define the plotWindow procedure
```

OUTPUT:


n0
n1
2Mb,10ms
2Mb,10ms
n2
0.3Mb,100ms
0.3Mb,100ms
n3
(back to n2)
n4
n5
(LAN)

```
proc plotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
   $ns at [expr $now + $time] "plotWindow $tcpSource $file"
}
$ns at 0.1 "plotWindow $tcp $winfile"
# Annotations for NAM
$ns at 5.0 "$ns trace-annotate \"packet drop\""
# Finish simulation
$ns at 125.0 "finish"
# Run simulation
$ns run
```

PROGRAM 2:

```
# Create the simulator object
set ns [new Simulator]
# Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
# Open trace files
set file1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $file1
# Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2
# Define a 'finish' procedure
proc finish {} {
global ns file1 file2
```

```
  $ns flush-trace
close $file1
close $file2
exec nam out.nam &
exit 0
}
# Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
# Set properties for nodes
$n1 color red
$n1 shape box
# Create links between nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
# Create a LAN between n3, n4, and n5
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]
# Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
```

```
# $tcp set packetSize_ 552
# Setup a FTP application over TCP
set ftp [new Application/FTP]
$ftp attach-agent $tcp
# $ftp set type_ FTP
# Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
# $udp set fid_ 2
# Setup a CBR application over UDP
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0.01Mb
$cbr set random_ false
# Schedule application starts and stops
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 124.0 "$ftp stop"
$ns at 124.5 "$cbr stop"
# Define a procedure to plot congestion window
proc plotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
   $ns at [expr $now+$time] "plotWindow $tcpSource $file"
```

```
}
$ns at 0.1 "plotWindow $tcp $winfile"
# Schedule an annotation for packet drop
$ns at 5.0 "$ns trace-annotate \"Packet Drop\""
# Finish simulation
$ns at 125.0 "finish"
# Run simulation
$ns run
```

RESULT:

Thus the operation of CSMA/CA and CSMA/CS were implemented using NS2.