**IMPLEMENTATION OF ERROR DETECTION / ERROR CORRECTION TECHNIQUES**

**EXPT NO:**

**DATE:**

**ERROR DETECTION 8 – BIT CRC**

**AIM:**

**APPARATUS REQUIRED:**

**ALGORITHM:**

**FLOW CHART:**

**PROGRAM:**

**1.(a)Generation Of CRC:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

#define poly 0xd8

intcrc(intmsg)

{

intrem,i;

rem=msg;

for(i=8;i>0;--i)

{

if(rem &0x80)

{

rem^=poly;

}

rem=rem<<1;

}

return(rem>>4);

}

void main()

{

intmsg=0xe5,c;

clrscr();

c=crc(msg);

printf("%d",c);

getch();

}

**1.(b).Error Detection using CRC**

#include<stdio.h>

#include<conio.h>

#define N strlen(g)

char t[28],cs[28],g[]="10001000000100001";

inta,e,c;

voidxor()

{

for(c=1;c<N;c++)

cs[c]=((cs[c]==g[c])?'0':'1');

}

voidcrc()

{

for(e=0;e<N;e++)

cs[e]=t[e];

do

{

if(cs[0]=='1')

xor();

for(c=0;c<N-1;c++)

cs[c]=cs[c+1];

cs[c]=t[e++];

}

while(e<=a+N-1);

}

int main()

{

clrscr();

printf("\n enter data");

scanf("%s",t);

printf("\n..............");

printf("\n generating polynomial:%s",g);

a=strlen(t);

for(e=a;e<a+N-1;e++)

t[e]='0';

printf("\n......");

printf("modified data is%s",t);

printf("\n.........");

crc();

printf("\n checksum is:%s",cs);

for(e=a;e<a+N-1;e++)

t[e]=cs[e-a];

printf("\n........");

printf("\n final code word is%s",t);

printf("\n test error detection 0(yes)or(1no)?:");

scanf("%d",&e);

if(e==0)

{

do

{

printf("enter the position where error is to be inserted");

scanf("%d",&e);

}

while(e==0||e>a+N-1);

t[e-1]=(t[e-1]=='0')?'1':'0';

printf("\n...........");

printf("\erroneous data:%s",t);

}

crc();

for (e=0;(e<(N-1))&&(cs[e]!='1');e++);

if(e<(N-1))

printf("\n error detected\n");

else

printf("no error detected\n");

return 0;

}

**OUTPUT:**

**RESULT:**

**ERROR CORRECTION – HAMMING CODE**

**EXPT NO:**

**DATE:**

**AIM:**

**APPARATUS REQUIRED:**

**ALGORITHM:**

**FLOW CHART:**

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

void main()

{

int d[7],r[7],i,c1,c2,c3,c;

clrscr();

printf("THIS WORKS FOR 4 BIT MESSAGE\n\nENTER THE 4 BIT DATA ONE BY ONE\n");

for(i=0;i<3;i++)

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

scanf("%d",&d[4]);

d[6]=d[0]^d[2]^d[4];

d[5]=d[0]^d[1]^d[4];

d[3]=d[0]^d[1]^d[2];

printf("\n\nTHE ENCODED BITS ARE:\n\n");

for(i=0;i<7;i++)

printf("%d\t",d[i]);

printf("\n\nENTER THE RECEIVED DATA:\n\n");

for(i=0;i<7;i++)

scanf("%d",&r[i]);

c1=r[6]^r[4]^r[2]^r[0];

c2=r[5]^r[4]^r[1]^r[0];

c3=r[3]^r[2]^r[1]^r[0];

c=c3\*4+c2\*2+c1;

if(c==0)

printf("\nTHERE IS NO ERROR\n");

else

{

printf("\n\nERROR POSITION IS %d \n\n",c);

printf("THE CORRECTED MESSAGE IS:\n\n");

if(r[7-c]=0)

r[7-c]=1;

else

r[7-c]=0;

for(i=0;i<7;i++)

{

printf("%d \t",r[i]);

}

}

getch();

}

**OUTPUT:**

**RESULT:**

**STOP & WAIT PROTOCOL USING C PROGRAM**

**EXPT NO:**

**DATE:**

**AIM:**

**APPARATUS REQUIRED:**

**ALGOROTHM:**

**FLOW CHART:**

**PROGRAM:**

#include <conio.h>

#include <dos.h>

#include <stdio.h>

#include <stdlib.h>

#define TIMEOUT 5

#define MAX\_SEQ 3

#define TOT\_PACKETS 8

#define inc(k) if(k<MAX\_SEQ) k++; else k=0;

typedefstruct

{

int data;

}packet;

typedefstruct

{

int kind;

intseq;

intack;

packet info;

int err;

}frame;

frame DATA;

typedefenum{frame\_arrival,err,timeout,no\_event} event\_type;

voidfrom\_network\_layer(packet \*);

voidto\_network\_layer(packet \*);

voidto\_physical\_layer(frame \*);

voidfrom\_physical\_layer(frame \*);

voidwait\_for\_event\_sender(event\_type \*);

voidwait\_for\_event\_reciever(event\_type \*);

voidreciever();

void sender();

inti=1; //Data to be sent by sender

char turn; //r , s

int DISCONNECT=0;

/\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

void main()

{

clrscr();

randomize();

while(!DISCONNECT)

{

sender();

delay(400);

reciever();

}

getch();

}

/\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

void sender()

{

staticintframe\_to\_send=0;

static frame s;

packet buffer;

event\_type event;

staticint 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);

}

}

}

/\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

voidreciever()

{

staticintframe\_expected=0;

framer,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

}

}

/\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

voidfrom\_network\_layer(packet \*buffer)

{

(\*buffer).data = i;

i++;

}

/\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

voidto\_physical\_layer(frame \*s)

{ // 0 means error

s->err = random(4); //non zero means no error

DATA = \*s; //probability of error = 1/4

}

/\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

voidto\_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");

}

}

/\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

voidfrom\_physical\_layer(frame \*buffer)

{

\*buffer = DATA;

}

/\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

voidwait\_for\_event\_sender(event\_type \* e)

{

staticint 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;

}

}

}

/\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

voidwait\_for\_event\_reciever(event\_type \* e)

{

if(turn=='r')

{

if(DATA.err==0)

\*e = err;

else

\*e = frame\_arrival;

}

}

**OUTPUT:**

**RESULT:**

**SLIDING WINDOW ARQ USING ‘C’**

**EXP NO:**

**DATE:**

**AIM:**

**APPARATUS REQUIRED:**

**ALGORITHM:**

**FLOW CHART:**

**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

intn,r;

struct frame

{

charack;

int data;

}frm[10];

int sender(void);

voidrecvfrm(void);

void resend(void);

void resend1(void);

voidgoback(void);

void selective(void);

int main()

{

int c;

clrscr();

do

{

printf("\n\n1.Selective repeat ARQ\n2.Goback ARQ\n3.exit");

printf("\nEnterur choice:");

scanf("%d",&c);

switch(c)

{

case 1:

selective();

break;

case 2:

goback();

break;

case 3:

exit(0);

break;

}

}while(c!=4);

}

voidgoback()

{

sender();

recvfrm();

resend1();

printf("\n all packets sent successfully\n");

}

void selective()

{

sender();

recvfrm();

resend();

printf("\nAll packets sent successfully");

}

int sender()

{

inti;

printf("\nEnter the no. of packets to be sent:");

scanf("%d",&n);

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

{

printf("\nEnter data for packets[%d]",i);

scanf("%d",&frm[i].data);

frm[i].ack='y';

}

return 0;

}

voidrecvfrm()

{

inti;

rand();

r=rand()%n;

frm[r].ack='n';

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

{

if(frm[i].ack=='n')

printf("\nThe packet number %d is not received\n",r);

}

}

void resend()

{

printf("\nresending packet %d",r);

sleep(2);

frm[r].ack='y';

printf("\nThe received packet is %d",frm[r].data);

}

void resend1()

{

inti;

printf("\n resending from packet %d",r);

for(i=r;i<n;i++)

{

sleep(2);

frm[i].ack='y';

printf("\nReceived data of parent %d is %d",i,frm[i].data);

}

}

**OUTPUT:**

**RESULT:**

**IMPLEMENTATION OF GO BACK N AND SELECTIVE REPEAT PROTOCOL**

**EXPT NO:**

**DATE:**

**AIM:**

**APPARATUS REQUIRED:**

**ALGORITHM:**

**FLOW CHART:**

**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

intn,r;

struct frame

{

charack;

int data;

}frm[10];

int sender(void);

voidrecvfrm(void);

void resend(void);

void resend1(void);

voidgoback(void);

void selective(void);

int main()

{

int c;

clrscr();

do

{

printf("\n\n1.Selective repeat ARQ\n2.Goback ARQ\n3.exit");

printf("\nEnterur choice:");

scanf("%d",&c);

switch(c)

{

case 1:

selective();

break;

case 2:

goback();

break;

case 3:

exit(0);

break;

}

}while(c!=4);

}

voidgoback()

{

sender();

recvfrm();

resend1();

printf("\n all packets sent successfully\n");

}

void selective()

{

sender();

recvfrm();

resend();

printf("\nAll packets sent successfully");

}

int sender()

{

inti;

printf("\nEnter the no. of packets to be sent:");

scanf("%d",&n);

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

{

printf("\nEnter data for packets[%d]",i);

scanf("%d",&frm[i].data);

frm[i].ack='y';

}

return 0;

}

voidrecvfrm()

{

inti;

rand();

r=rand()%n;

frm[r].ack='n';

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

{

if(frm[i].ack=='n')

printf("\nThe packet number %d is not received\n",r);

}

}

void resend()

{

printf("\nresending packet %d",r);

sleep(2);

frm[r].ack='y';

printf("\nThe received packet is %d",frm[r].data);

}

void resend1()

{

inti;

printf("\n resending from packet %d",r);

for(i=r;i<n;i++)

{

sleep(2);

frm[i].ack='y';

printf("\nReceived data of parent %d is %d",i,frm[i].data);

}

}

**OUTPUT:**

**RESULT:**

**IMPLEMENTATION OF HIGH LEVEL DATA LINK CONTROL (HDLC) USING C**

**EXPT NO:**

**AIM:**

**APPARATUS REQUIRED:**

**ALGORITHM:**

**FLOW CHART:**

**PROGRAM:**

#include <stdio.h>

#include <string.h>

void main()

{

inti, j,count=0,nl;

charstr[100];

printf("enter the bit string: ");

gets(str);

for (i=0;i<strlen(str);i++)

{

count=0;

for (j=i;j<=(i+5);j++)

{

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

{

count++;

}

}

if(count==6)

{

nl=strlen(str)+2;

for (;nl>=(i+5);nl--)

{

str[nl]=str[nl-1];

}

str[i+5]='0';

i=i+7;

}

}

puts(str);

getch();

}

**OUTPUT:**

**RESULT:**

**Study of Client – Server model**

**EXPT. NO:**

**DATE:**

**AIM:**

**APPARATUS REQUIRED:**

**THEORY:**

A socket is a software entity that provides the basic building block for inter-process communications. It is also an interface between the application and the network. Once the socket is configured, the application can pass data to the socket for network transmission and receive the data from the socket from the other host. A socket communication can be connection oriented (TCP sockets) or connectionless (UDP sockets) There is a receiver (TCP/UDP) server, which listens for the sender (TCP/UDP) clients to communicate. There can be two-way communication.

**ALGORITHM:**

**PROGRAM:**

//server

#include<stdio.h>

#include<unistd.h>

#include<fcntl.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<stdlib.h>intmain()

{ intcs,ns,fd,n;intbufsize=1024;

char \*buffer=malloc(bufsize);structsockaddr\_inaddress;char fname[255];address.sin\_family=AF\_INET;address.sin\_port=htons(15000);

address.sin\_addr.s\_addr=INADDR\_ANY;cs=socket(AF\_INET,SOCK\_STREAM,0);

bind(cs,(structsockaddr \*)&address,sizeof(address));listen(cs,3);

ns=accept(cs,(structsockaddr \*)NULL,NULL);recv(ns,fname,255,0);

fd=open(fname,O\_RDONLY);n=read(fd,buffer,bufsize);send(ns,buffer,n,0);

close(ns);return close(cs);

}

###### //client

#include<stdio.h>

#include<unistd.h>

#include<fcntl.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<stdlib.h>

int main(intargc,char \*\*argv)

{

intcs,n;

intbufsize=1024;

char \*buffer=malloc(bufsize);char fname[255];

structsockaddr\_in address;

address.sin\_family=AF\_INET;address.sin\_port=htons(15000);inet\_pton(AF\_INET,argv[1],&address.sin\_addr);

cs=socket(AF\_INET,SOCK\_STREAM,0);

connect(cs,(structsockaddr \*)&address,sizeof(address));

printf("\nEnter filename: ");scanf("%s",fname);send(cs,fname,255,0);

while((recv(cs,buffer,bufsize,0))>0)printf("%s",buffer);printf("\nEOF\n");

return close(cs);

}

## OUTPUT:

**RESULT:**

**STUDY THE PERFORMANCE OF NETWORK WITH CSMA / CA**

**PROTOCOL AND COMPARE WITH CSMA/CD PROTOCOLS**

**EXPT. NO:**

**DATE:**

**AIM:**

**Hardware / Software Requirements:**

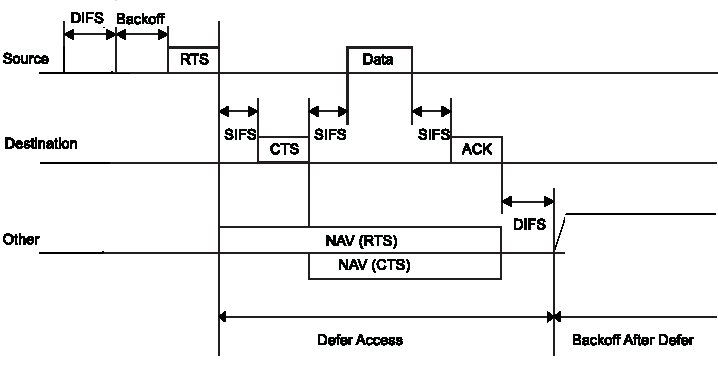
1. NetSim software (for simulation)

**Theory:**

Wireless LAN is basically a LAN that transmits data over air, without any physical connection between devices. The transmission medium is a form of electromagnetic radiation. Wireless LAN is ratified by IEEE in the IEEE 802.11 standard. The underlying algorithm used in Wireless LAN is known as the CSMA / CA – Carrier Sense Multiple Access / Collision Avoidance algorithm. The working of CSMA / CA algorithm is given below

* The node which has data to transmit senses the medium. If the medium has been idle for longer than the DIFS (DCF Inter Frame Space), it finishes its back off interval & transmits Request To Send (RTS) signal immediately.
* The access point responds with Clear to Send (CTS) signal .Now the node has reserved the medium and transmits data.
* If the medium is busy, the node waits for the channel to become idle for the DIFS.
* If two nodes sense the medium at the same time & transmit RTS simultaneously, RTS collision occurs and the transmission is retried. Hence data collision is avoided.
* For each retransmission, contention window increases exponentially hence back off time is selected from larger contention window.

This is explained in the timing diagram given below:



**Formulae:**

**(i) Throughput (%)**

Fraction of link’s capacity devoted to carrying frames.

****

**(ii) Normalized Throughput (%)**

Also called as Goodput(%)

Fraction of link’s capacity devoted to carrying non-retransmitted frames excluding bytes due to protocol overhead, collision and retransmission.

****

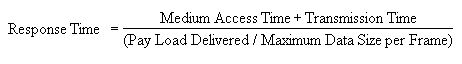
**(iii) Mean delay** (Micro Second/Frame)

Mean time a frame waits at a station before being successfully transmitted (queuing time and medium access time) and the transmission time per frame.

**mean**

**(iv) Response Time**(Micro Second / Frame)

Also called as Service Time; Sum of medium access time & transmission time per frame.

****

**(v) Collision count.**

Total number of collisions in the network.

**(a) LAN (unicast vs broadcast):**

**Network Diagram:**

Hub 1

*Figure (1) LAN*



1

6

5

4

3

2

AP 1

**Specifications:**

|  |  |  |
| --- | --- | --- |
| **Sl No** | **Parameters** | **Value** |
|  | Network | **Wireless LAN** |
|  | Topology | **Star-Hub** |
|  | Traffic | **Broadcast / Point to Point** |

**Procedure:**

1. To begin with the experiment, run **NetSim**.
2. Click on **Simulation**, select **LAN** network and **Wireless LAN** from the sub menu. The simulation environment is now open.
3. Drag and drop the **BSS** and the **Hub** on the environment. Connect the BSS and the hub.
4. Drag and drop the **node** near to the BSS only. Two is the minimum number of nodes required for transmission to take place.
5. Right click on the 1st node and select **Broadcast** from the **transmission option**.
6. Click **Configure** and **Simulate** - you will see the performance metrics screen.
7. **Close** and **save** the experiment
8. After saving the experiment, click **File** and **Open** from the sub menu on the file panel.
9. Select **LAN option**. Select **Wireless LAN - Protocol**, **Star-Hub Topology** and the **experiment** you have saved. Click on the **accept** button.
10. Set the **Broadcast** traffic on one additional node (total of 2 nodes) Click **Configure** and **Simulate**.
11. Follow the same procedure increasing the number of **Broadcast** nodes by 1 every time, till all **6 nodes** broadcast.
12. For **unicast**, set the traffic (in the traffic Generator) to **Point to Point** and follow the procedure as given for broadcast. (Refer Analysis for generating plot)

**Model graph:**



**Inference:**

For unicast (point to point traffic), the nodes use RTS/CTS mechanism to avoid data collision. But in broadcast the frames are simply sent without RTS/CTS .The reduction in protocol overheads results in better throughput, for broadcast transmission.

**(b) CSMA / CD vs. CSMA / CA protocols**

**Network Diagram:**

Node 1

Node 2

Hub 1

*Figure 2.a.CSMA/CD*

Node 3

Node 5

Node 4

Node 6

Hub 1



1

6

5

4

3

2

AP 1

*Figure 2.b.CSMA-CA*

**Specifications:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Parameters** | **CSMA/CD** | **CSMA/CA** |
|  | Network | **LAN** | **LAN** |
|  | Protocol | **Ethernet - Star** | **Wireless LAN** |
|  | Topology/ Device | **Hub** | **Star-Hub** |
|  | Traffic | **Broadcast** | **Point to Point** |

**Procedure:**

1. To begin with the experiment, run **NetSim**.
2. Click on **Simulation**, select **LAN** network and **Wireless LAN** from the sub menu. The simulation environment is now open.
3. Drag and drop the **BSS** and the **Hub** on the environment. Connect the BSS and the hub.
4. Drag and drop the **node** near to the BSS only. Two is the minimum number of nodes required for transmission to take place.
5. Right click on the 1st node and select **Point to Point** from the **transmission option**. Click **Accept** and **Close.**
6. Click **Configure** and **Simulate** - you will see the performance metrics screen.
7. **Close** and **save** the experiment
8. After saving the experiment, click **File** and **Open** from the sub menu on the file panel.
9. Select **LAN option**. Select **Wireless LAN - Protocol**, **Star-Hub Topology** and the **experiment** you have saved. Click on the **accept** button.
10. Set the traffic to **Point to Point** on one additional node (total of 2 nodes) Click **Configure** and **Simulate**.
11. Follow the same procedure increasing the number of nodes by 1 every time, till all **6 nodes** transmit Point to Point.
12. Follow the appropriate procedure given for **Ethernet Star – Hub** with 6 nodes broadcast.
13. Analyze and compare the results between the experiment done in CSMA/CA and one done with CSMA/CD.

**Analysis:**

1. Select **Simulation** and **Analysis** from the sub menu in the file panel.
2. Select the required details such as **User, Network, Protocol** and the **Sample**
3. Add all the samples to the table.
4. After all the samples are added, select the required **input and output axis** from the browse button. Select either the **Bar Chart** or the **Line Chart** to view the graphs.
5. User can also E**xport to Excel** and plot the graph. Click close to exit.

**Model graph:**



**Inference:**

The collision avoidance concept in wireless LAN protocol, avoids the possibility of collision of data frames. The working of wireless LAN is such that when a node transmits to the destination the data reaches the access point (coordination point) and from there it is transmitted to the destination node. Due to its nature of double transmission and protocol overheads, the throughput is reduced by 50% when compared to Ethernet.

.

**RESULT:**

**BUS TOPOLOGY**

**EXPT NO:**

**DATE:**

**AIM**:

**ALGORITHM:**

**PROGRAM:**

#Create a simulator object

set ns [new Simulator]

#Open the nam trace file

setnf [open out.nam w]

$ns namtrace-all $nf

#Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

#Close the trace file

close $nf

#Executenam on the trace file

execnamout.nam&

exit 0

}

#Create four nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

#CreateLanbetween the nodes

set lan0 [$ns newLan "$n0 $n1 $n2 $n3 $n4" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]

#Create a TCP agent and attach it to node n0

set tcp0 [new Agent/TCP]

$tcp0 set class\_ 1

$ns attach-agent $n1 $tcp0

#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3

set sink0 [new Agent/TCPSink]

$ns attach-agent $n3 $sink0

#Connect the traffic sources with the traffic sink

$ns connect $tcp0 $sink0

# Create a CBR traffic source and attach it to tcp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.01

$cbr0 attach-agent $tcp0

#Schedule events for the CBR agents

$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation

$ns

OUTPUT:

**RESULT:**

**STAR TOPOLOGY**

**EXPT NO:**

**DATE:**

**AIM**:

ALGORITHM:

**PROGRAM:**

#Create a simulator object

set ns [new Simulator]

#Open the nam trace file

setnf [open out.nam w]

$ns namtrace-all $nf

#Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

#Close the trace file

close $nf

#Executenam on the trace file

execnamout.nam&

exit0

}

#Create four 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]

#Change the shape of center node in a star topology

$n0 shape square

#Create links between the nodes

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

$ns duplex-link $n0 $n2 1Mb 10ms DropTail

$ns duplex-link $n0 $n3 1Mb 10ms DropTail

$ns duplex-link $n0 $n4 1Mb 10ms DropTail

$ns duplex-link $n0 $n5 1Mb 10ms DropTail

#Create a TCP agent and attach it to node n0

set tcp0 [new Agent/TCP]

$tcp0 set class\_ 1

$ns attach-agent $n1 $tcp0

#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3

set sink0 [new Agent/TCPSink]

$ns attach-agent $n3 $sink0

#Connect the traffic sources with the traffic sink

$ns connect $tcp0 $sink0

# Create a CBR traffic source and attach it to tcp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.01

$cbr0 attach-agent $tcp0

#Schedule events for the CBR agents

$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation

$ns run

OUTPUT:

**RESULT:**

**RING TOPOLOGY**

**EXPT:**

**DATE:**

**AIM**:

**APPARATUS REQUIRED:**

**ALGORITHM:**

**PROGRAM:**

#Create a simulator object

set ns [new Simulator]

#Open the nam trace file

setnf [open out.nam w]

$ns namtrace-all $nf

#Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

#Close the trace file

close $nf

#Executenam on the trace file

execnamout.nam&

exit0

}

#Create four 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]

#Create links between the nodes

$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 $n5 1Mb 10ms DropTail

$ns duplex-link $n5 $n0 1Mb 10ms DropTail

#Create a TCP agent and attach it to node n0

set tcp0 [new Agent/TCP]

$tcp0 set class\_ 1

$ns attach-agent $n1 $tcp0

#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3

set sink0 [new Agent/TCPSink]

$ns attach-agent $n3 $sink0

#Connect the traffic sources with the traffic sink

$ns connect $tcp0 $sink0

# Create a CBR traffic source and attach it to tcp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.01

$cbr0 attach-agent $tcp0

#Schedule events for the CBR agents

$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation

$ns run

OUTPUT:

**RESULT:**

**DISTANCE VECTOR ROUTING ALGOROTHM**

**EXPT NO:**

**DATE:**

**AIM:**

**APPARATUS REQUIRED:**

**ALGORITHM:**

**FLOWCHART:**

**PROGRAM:**

#include<stdlib.h>

#define NUL 1000

#define NODES 10

struct node

{

int t[NODES][3];

};

struct node n[NODES];

typedefstruct node NOD;

void main()

{

voidinit(int,int);

voidinp(int,int);

void caller(int,int);

void op1(int,int,int);

void find(int,int);

inti,j,x,y,no;

clrscr();

do{

printf("\n Enter the no of nodes required:");

scanf("%d",&no);

}while(no>10||no<0);

for(i=0;i<no;i++)

{

init(no,i);

inp(no,i);

}

printf("\nThe configuration of the nodes after initalization is as follows:");

for(i=0;i<no;i++)

op1(no,i,0);

for(j=0;j<no;j++)

{

for(i=0;i<no;i++)

caller(no,i);

}

printf("\nTheconfig of the nodes after the comp of the paths is as follows:");

for(i=0;i<no;i++)

op1(no,i,1);

while(1)

{

printf("\n Enter 0 to exit or any other key to find the shortest path:");

scanf("%d",&j);

if(!j)

break;

clrscr();

do{

printf("\n Enter the nodes btn which path is to be found :");

scanf("%d%d",&x,&y);

}while((x<0||x>no) && (y<0||y>no));

printf("\nThe most suitable route from node %d to %d is as follows\n",x,y);

find(x,y);

printf("%d",y);

printf("\nThelengthoftheshortestpathbtnnode%d& %dis%d",x,y,n[x-1].t[y-1][2]);

}

}

voidinit(intno,int x)

{

int i;

for(i=0;i<no;i++)

{

n[x].t[i][1]=i;

n[x].t[i][2]=999;

n[x].t[i][3]=NUL;

}

n[x].t[x][2]=0;

n[x].t[x][3]=x;

}

voidinp(intno,int x)

{

int i;

printf("\nEnter the dists from the nodes %d to other node...",x+1);

printf("\nPls enter 999 if there is no direct \n");

for(i=0;i<no;i++)

{

if(i!=x)

{

do

{

printf("\n Enter dist to node %d=",i+1);

scanf("%d",&n[x].t[i][2]);

}while(n[x].t[i][2]<0|| n[x].t[i][2]>999);

if(n[x].t[i][2]!=999)

n[x].t[i][3]=i;

} }

}

void caller(intno,int x)

{

voidcompar(int,int,int);

int i;

for(i=0;i<no;i++)

{

if(n[x].t[i][2]!=999 && n[x].t[i][2]!=0)

{

compar(x,i,no);

}

}

}

voidcompar(intx,inty,int no)

{

inti,z;

for(i=0;i<no;i++)

{

z=n[x].t[y][2]+n[y].t[i][2];

if(n[x].t[i][2]>z)

{

n[x].t[i][2]=z;

n[x].t[i][3]=y;

}

}

}

void op1(intno,intx,int z)

{

inti,j;

printf("\n The routing table for node no %d is as follows",x+1);

printf("\n\n\t\t\tDESTINATION\tDISTANCE\tNEXT\_HOP");

for(i=0;i<no;i++)

{

if((!z && n[x].t[i][2]>=999) ||(n[x].t[i][2]>=(999\*no)))

printf("\n\t\t\t %d \tNO LINK \t NO HOP",n[x].t[i][1]+1);

else

if(n[x].t[i][3]==NUL)

printf("\n\t\t\t %d \t %d \t NO OP",n[x].t[i][1]+1,n[x].t[i][2]);

else

printf("\n\t\t\t %d \t %d \t%d",n[x].t[i][1]+1,n[x].t[i][2],n[x].t[i][3]+1);

}

getch();

}

void find(intx,int y)

{

inti,j;

i=x-1;

j=y-1;

printf("%d-->",x);

if(n[i].t[j][3]!=j)

{

find(n[i].t[j][3]+1,y);

return;

}

}

**OUTPUT:**

**RESULT:**

**OPEN SHORTEST PATH FIRST (OSPF) – LINK STATE ROUTING**

**EXPT NO:**

**DATE:**

**AIM:**

**APPARATUS REQUIRED:**

**ALGORITHM:**

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

int a[5][5],n,i,j;

void main()

{

voidgetdata();

void shortest();

void display();

clrscr();

getdata();

shortest();

display();

getch();

}

voidgetdata()

{

printf("\nEnter the no. of host in the graph\t");

scanf("%d",&n);

printf("\nIf there is no direction path\nHighest distance value 1000\n");

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

{

a[i][j]=0;

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

{

if(i!=j)

{

printf("\nEnter the distance b/w (%d %d): ",i+1,j+1);

scanf("%d",&a[i][j]);

if(a[i][j]==0)

a[i][j]=1000;

}

}

}

}

void shortest()

{

inti,j,k;

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

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

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

{

if(a[i][k]+a[k][j]<a[i][j])

a[i][j]=a[i][k]+a[k][j];

}

}

void display()

{

inti,j;

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

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

if(i!=j)

{

printf("\nShortest path is:(%d %d): %d\n",i+1,j+1,a[i][j]);

}

}

**OUTPUT:**

**RESULT:**

**CONGESTION CONTROL USING LEAKY BUCKET ALGORITHM**

**EXPT NO:**

**DATE:**

**AIM:**

**APPARATUS REQUIRED:**

**ALGORITHM:**

**PROGRAM:**

#include<stdio.h>

void main(){

int incoming, outgoing, buck\_size, n, store = 0;

printf("Enter bucket size, outgoing rate and no of inputs: ");

scanf("%d %d %d", &buck\_size, &outgoing, &n);

while (n != 0) {

printf("Enter the incoming packet size : ");

scanf("%d", &incoming);

printf("Incoming packet size %d\n", incoming);

if (incoming <= (buck\_size - store)){

store += incoming;

printf("Bucket buffer size %d out of %d\n", store, buck\_size);

**}** else {

printf("Dropped %d no of packets\n", incoming - (buck\_size - store));

printf("Bucket buffer size %d out of %d\n", store, buck\_size);

store = buck\_size;

}

store = store - outgoing;

printf("After outgoing %d packets left out of %d in buffer\n", store, buck\_size);

n--;

}

getch();

}

**OUTPUT:**

**RESULT:**

**ENCRYPTION & DECRYPTION USING RSA ALGORITHM**

**EXPT NO:**

**DATE:**

**AIM:**

**APPARATUS REQUIRED:**

**ALGORITHM:**

**FLOW CHART:**

**PROGRAM:**

#include<stdio.h>

#include<math.h>

double min(double x, double y)

{

return(x<y?x:y);

}

double max(double x,double y)

{

return(x>y?x:y);

}

doublegcd(double x,double y)

{

if(x==y)

return(x);

else

return(gcd(min(x,y),max(x,y)-min(x,y)));

}

long double modexp(long double a,long double x,long double n)

{

long double r=1;

while(x>0)

{

if ((int)(fmodl(x,2))==1)

{

r=fmodl((r\*a),n);

}

a=fmodl((a\*a),n);

x/=2;

}

return(r);

}

void main()

{

long double p,q,phi,n,e,d,cp,cq,dp,dq,mp,mq,sp,sq,rp,rq,qInv,h;

long double ms,es,ds;

do{

printf("\n Enter prime numbers p and q:");

scanf(" %Lf %Lf",&p,&q);

}

while(p==q);

n=p\*q;

phi=(p-1)\*(q-1);

do{

printf("\n Enter prime value of e:");

scanf(" %Lf",&e);

}

while((gcd(e,phi)!=1)&&e>phi); /\*for e being relatively prime to phi \*/

for(d=1;d<phi;++d)

{

if(fmod((e\*d),phi)==1)

break;

}

printf("\n D within main = %Lf",d);

/\* public key is {n,e} private key is d \*/

printf("\n Enter the message:");

scanf(" %Lf",&ms);

es=modexp(ms,e,n);

ds=modexp(es,d,n);

printf("\n Original Message : %Lf",ms);

printf("\n Encrypted Message : %Lf",es);

printf("\n Decrypted Message : %Lf\n",ds);

getch();

}

**OUTPUT:**

**RESULT:**