KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed University Established Under Section 3 of UGC Act, 1956)

Coimbatore - 641 021, India

FACULTY OF ARTS, SCIENCE AND HUMANITIES (FASH)

Department of CS,CA & IT

II B.Sc CS **III SEMESTER** **BATCH : 2016 - 2019**

16CSU313 **COMPUTER NETWORKS – PRACTICAL** **4H – 2C**

Instruction Hours / week: L: 0 T: 0 P: 4 **Marks:** Int : **40** Ext : **60** Total: **100**

Course Objective

- To develop an understanding of modern network architectures from a design and performance perspective
- To give the students experience working in programming teams
- Build an understanding of the fundamental concepts of computer networking.
- To be familiar with contemporary issues in networking technologies.

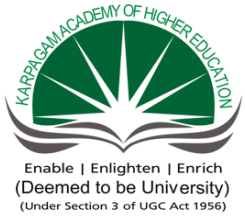
Course Learning Outcome

After completing this course the student must demonstrate the knowledge and ability to:

- To design a network routing for IP networks..
- To explain how a collision occurs and how to solve it
- To impart basic concepts and basic skills for setting up routers
- To be familiar with network tools and network programming.

Practical List

1. Simulate Cyclic Redundancy Check (CRC) error detection algorithm for noisy channel.
2. Simulate and implement stop and wait protocol for noisy channel.
3. Simulate and implement go back n sliding window protocol.
4. Simulate and implement selective repeat sliding window protocol.
5. Simulate and implement distance vector routing algorithm
6. Simulate and implement Dijkstra algorithm for shortest path routing.
7. Simulate and implement Parity Bit Generator



KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act, 1956)

Coimbatore - 641 021, India

FACULTY OF ARTS, SCIENCE AND HUMANITIES (FASH)

Department of CS,CA & IT

II B.Sc CS III SEMESTER BATCH : 2016 – 2019

COMPUTER NETWORKS – PRACTICAL (16CSU313)

Practical List

1. Simulate Cyclic Redundancy Check (CRC) error detection algorithm for noisy channel.
2. Simulate and implement stop and wait protocol for noisy channel.
3. Simulate and implement go back n sliding window protocol.
4. Simulate and implement selective repeat sliding window protocol.
5. Simulate and implement distance vector routing algorithm
6. Simulate and implement Dijkstra algorithm for shortest path routing.
7. Simulate and implement Parity Bit Generator

CRC ERROR DETECTION ALGORITHM

EX NO : 01

AIM:

To simulate cyclic redundancy check (CRC) error Detection algorithm for noisy channel.

ALGORITHM:

Step 1 : Start the process.

Step 2 : Declare the header files and array variables of type char.

Step 3 : Enter the input message in binary format.

Step 4 : Append the message with 16 bit CRC using crc (ip, op, poly,1) function to transmit the message.

Step 5 : Enter the received message in binary format.

Step 6 : Verify the message that is received is the same as the one sent.

Step 7 : Check the condition if(crc(recv,op,poly,0)).

Step 8 : Display the error message.

Step 9 : Stop the process.

PROGRAM

```
#include <iostream.h>
#include <string.h>
#include<conio.h>
int crc(char *ip, char *op, char *poly, int mode)
{
strcpy(op, ip);
if (mode)
{
        for (int i = 1; i < strlen(poly); i++)
        strcat(op, "0");
}
for (int i = 0; i < strlen(ip); i++)
{
if (op[i] == '1')
{
for (int j = 0; j < strlen(poly); j++)
{
        if (op[i + j] == poly[j])
        op[i + j] = '0';
        else
        op[i + j] = '1';
}
}
}
for (i = 0; i < strlen(op); i++)
if (op[i] == '1')
return 0;
return 1;
}
void main()
{
char ip[50], op[50], recv[50];
char poly[] = "10001000000100001";
clrscr();

cout<<"\n IMPLEMENT CYCLIC REDUNDANCY CHECK ERROR DETECTION
ALGORITHM";
cout<<"\n ********************************************************";
```

```
cout <<"\n\nEnter the input message in binary"<< endl;
cin >> ip;
crc(ip, op, poly, 1);
cout << "\nThe transmitted message is: " << ip << op + strlen(ip) << endl;
cout << "\n\nEnter the recevied message in binary" << endl;
cin >> recv;
if (crc(recv, op, poly, 0))
cout << "\nNo error in data" << endl;
else
cout << "\nError in data transmission has occurred" << endl;
getch();
}
```

OUTPUT

```
IMPLEMENT CYCLIC REDUNDANCY CHECK ERROR DETECTION ALGORITHM
**********************************************************

Enter the input message in binary
1111

The transmitted message is: 11111111000111101111

Enter the recevied message in binary
1010

Error in data transmission has occurred



IMPLEMENT CYCLIC REDUNDANCY CHECK ERROR DETECTION ALGORITHM

**********************************************************

Enter the input message in binary

5

The transmitted message is: 50000000000000000

Enter the recevied message in binary

5

No error in data
```

RESULT:

The above program has been executed successfully and output was verified.

STOP AND WAIT PROTOCOL

EX NO : 02

AIM:

To simulate and implement Stop and wait protocol for noisy channel.

ALGORITHM:

Step 1 : Start the process.

Step 2 : Declare the header files and create structures.

Step 3 : Generate a random that gives total no of frame to be transmitted.

Step 4 : Transmit the first frame.

Step 5 : Receive the acknowledgement for the first frame.

Step 6 : Transmit the next frame.

Step 7 : Find the remaining frames to be sent.

Step 8 : If an acknowledgement is not received for particular frame retransmit that frame alone again.

Step 9 : Repeat the step 5 to 7 till the no of remaining frames to be sent becomes zero.

Step 10 : Display the results.

Step 11 : Stop the process.

PROGRAM

```
#include<iostream.h>
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<dos.h>
#define time 5
#define max_seq 1
#define tot_pack 5
int randn(int n)
{
   return rand()%n + 1;
}
typedef struct
{
     int data;
}packet;
typedef struct
{
      int kind;
      int seq;
      int ack;
      packet info;
}frame;
typedef enum{ frame_arrival,error,time_out}event_type;
frame data1;

void from_network_layer(packet *);
void to_physical_layer(frame *);
void to_network_layer(packet *);
void from_physical_layer(frame*);
void sender();
void receiver();
void wait_for_event_sender(event_type *);
void wait_for_event_receiver(event_type *);
#define inc(k) if(k<max_seq)k++;else k=0;
int i=1;
char turn;
int disc=0;
```

```
void main()
{
clrscr();
  cout<<"\n STOP AND WAIT PROTOCOL FOR NOISY CHANNEL";
  cout<<"\n**************************************";
  while(!disc)
  {  sender();
    // delay(400);
     receiver();
  }
   getchar();
   getch();
}
void sender()
{
    static int frame_to_send=0;
    static frame s;
    packet buffer;
    event_type event;
    static int flag=0;      //first place
    if (flag==0)
    {
  from_network_layer(&buffer);
  s.info=buffer;
  s.seq=frame_to_send;
  cout<<"\nsender information \t"<<s.info.data<<"\n";
  cout<<"\nsequence no. \t"<<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;
```

```
     cout<<"\nsender information \t"<<s.info.data<<"\n";
     cout<<"\nsequence no. \t"<<s.seq<<"\n";
     getch();
     turn='r';
     to_physical_layer(&s);
     }
}
}              //end of sender function
void from_network_layer(packet *buffer)
{
    (*buffer).data=i;
    i++;
}                    //end of from network layer function
void to_physical_layer(frame *s)
{
    data1=*s;
}          //end of to physical layer function
void wait_for_event_sender(event_type *e)
{
     static int timer=0;
     if(turn=='s')
     {   timer++;
   //timer=0;
   return ;
 }
 else                    //event is frame arrival
   {
     timer=0;
     *e=frame_arrival;
   }
}          //end of wait for event function
void receiver()
{
    static int frame_expected=0;
    frame s,r;
    event_type event;
    wait_for_event_receiver(&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
   cout<<"\nReceiver :Acknowledgement resent \n";
   getch();
   turn='s';
   to_physical_layer(&s);
    }
    }
}                //end of receiver function
void wait_for_event_receiver(event_type *e)
{
    if(turn=='r')
    {
  *e=frame_arrival;
    }
}
void from_physical_layer(frame *buffer)
{
   *buffer=data1;
}
void to_network_layer(packet *buffer)
{
    cout<<"\nReceiver : packet received \t"<< i-1;
    cout<<"\n Acknowledgement  sent \t";
    getch();
    if(i>tot_pack)
     { disc=1;
          cout<<"\ndiscontinue\n";
     }
}
```

OUTPUT

```
STOP AND WAIT PROTOCOL FOR NOISY CHANNEL
**************************************
Sender information  1
Sequence no: 0
Receiver: packet received  1
Acknowledgement sent
Sender information  2

Sequence no: 1
Receiver: packet received  2
Acknowledgement sent
Sender information  3

Sequence no: 0
Receiver: packet received  3
Acknowledgement sent
Sender information  4

Sequence no: 1
Receiver: packet received  4
Acknowledgement sent
Sender information  5

Sequence no: 0
Receiver: packet received  5
Acknowledgement sent
Discontinue
```

RESULT:

The above program has been executed successfully and output was verified.

GO BACK N SLIDING WINDOW PROTOCOL

EX NO : 03

AIM:

To simulate and implement go back n sliding window protocol.

ALGORITHM :

Step 1 : Start the process

Step 2 : Declare the header file and Variable I, winsize of type int

Step 3 : Declare the array sender [50] and receiver[50] of type char.

Step 4 : Enter the window size from the user.

Step 5 : Get the sender information by using gets(sender) method.

Step 6 : Use the for loop to execute the condition.

Step 7 : Display the receiver information using puts(receiver) method.

Step 8 : Display the result.

Step 9: Stop the process.

PROGRAM

```
#include<stdio.h>
#include<iostream.h>
#include<conio.h>
void main()
{
char sender[50],receiver[50];
int i,winsize;
clrscr();
cout<<"\n IMPLEMENT GO BACK N SLIDING WINDOW PROTOCOL";
cout<<"\n ****************************************";
cout<<"\n\n Enter the windows size : ";
cin>>winsize;
cout<<"\n Sender window is expanded to store message or window \n";
cout<<"\n Enter the data to be sent: ";
fflush(stdin);
gets(sender);
for(i=0;i<winsize;i++)
receiver[i]=sender[i];
receiver[i]=NULL;
cout<<"\n Message send by the sender:";
puts(sender);
cout<<"\n Window size of receiver is expanded\n";
cout<<"\n Acknowledgement from receiver \n";
for(i=0;i<winsize;i++)
cout<<"\n Ack:"<<i;
cout<<"\n Message received by receiver is : ";
puts(receiver);
cout<<"\n Window size of receiver is shrinked \n";
getch();
}
```

OUTPUT

IMPLEMENT GO BACK N SLIDING WINDOW PROTOCOL

****

Enter the windows size : 5

Sender window is expanded to store message or window

Enter the data to be sent: 10

Message send by the sender: 10

Window size of receiver is expanded

Acknowledgement from receiver
Ack: 0
Ack: 1
Ack: 2
Ack: 3
Ack: 4

Message received by receiver is : 10

Window size of receiver is shrinked

RESULT:

The above program has been executed successfully and output was verified.

SELECTIVE REPEAT SLIDING WINDOW PROTOCOL

EX NO : 04

AIM:

To simulate and implement selective repeat sliding window protocol.

ALGORITHM:

Step 1 : Start the process.

Step 2 : Declare the header file.

Step 3 : In main function, declare the variable as temp 1,2,3,4,5,winsize=8, Noframes, moreframe of type int.

Step 4 : Declare the functions receiver(),simulate(),nack() of type int.

Step 5 : Using the for loop to send the frame

Step 6 : Receive the acknowledgement by the receiver() method.

Step 7 : Retransmit the frame if Acknowledgement is not received

Step 8 : Display the results.

Step 9 : Stop the process.

PROGRAM

```
#include<stdio.h>
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int temp1,temp2,temp3,temp4,temp5,i,winsize=8,noframes,moreframes;
char c;
int receiver(int);
int simulate(int);
int nack(int);
clrscr();
temp4=0,temp1=0,temp2=0,temp3=0,temp5=0;
for(i=0;i<2500;i++)
rand();
noframes=rand()/2500;
cout<<"\n Number of frames is "<<noframes;
getch();
moreframes=noframes;
while(moreframes>=0)
{
temp1=simulate(winsize);
winsize=temp1;
temp4+=temp1;
if(temp4>noframes)
temp4=noframes;
for(i=noframes-moreframes;i<=temp4;i++)
cout<<"\n Sending frame "<<i;
getch();
temp2=receiver(temp1);
temp3+=temp2;
if(temp3>noframes)
temp3=noframes;
temp2=nack(temp1);
temp5+=temp2;
if(temp5!=0)
{
cout<<"\n No acknowledgement for the frames "<<temp5;
```

```
getch();
for(i=1;i<temp5;i++)
;
cout<<"\n Retransmitting frame "<<temp5;
getch();
}
moreframes-=temp1;
if(winsize<=0)
winsize=8;
}
cout<<"\n End of sliding window protocol selective reject ";
getch();
}

int receiver(int temp1)
{
int i;
for(i=1;i<100;i++)
rand();
i=rand()%temp1;
return i;
}

int nack(int temp1)
{
int i;
for(i=1;i<100;i++)
rand();
i=rand()%temp1;
return i;
}

int simulate(int winsize)
{
int temp1,i;
for(i=1;i<50;i++)
temp1=rand();
if(temp1==0)
temp1=simulate(winsize);
i=temp1%winsize;
```

```
if(i==0)
return winsize;
else
return temp1%winsize;
}
```

OUTPUT

```
Number of frames is 10
Sending frame 0
Sending frame 1
Sending frame 2
Sending frame 3
Sending frame 4
Sending frame 5
No acknowledgement for the frames 3
Retransmitting frame 3

Sending frame 5
Sending frame 6
Sending frame 7
Sending frame 8
No acknowledgement for the frames 4
Retransmitting frame 4

Sending frame 8
Sending frame 9
Sending frame 10
No acknowledgement for the frames 5
Retransmitting frame 5

End of sliding window protocol selective reject
```

RESULT:

The above program has been executed successfully and output was verified.

DISTANCE VECTOR ROUTING ALGORITHM

EX NO : 05

AIM:

To write a program to simulate and implement distance vector routing algorithm.

ALGORITHM:

Step 1 : Start the process.

Step 2 : Declare the header file.

Step 3 : Create struct node and Variable dist,from of type unsigned.

Step 4 : In main function declare the array costmat [6][6] and Variable node

,I,j,k of type int.

Step 5 : Enter the no of nodes and their cost matrix.

Step 6 : Calculate the distance using if condition (DVR[i].dist[j] >costmat[i][k]

+ DVR[k].dist[j])

Step 7 : Display the results.

Step 8 : Stop the process.

PROGRAM

```
#include<stdio.h>
#include<iostream.h>
#include<conio.h>
struct node
{
   unsigned dist[6];
   unsigned from[6];
}DVR[10];
void main()
{
   clrscr();
   cout<<"\n\n IMPLEMENT DISTANCE VECTOR ROUTING ALGORITHM ";
   cout<<"\n ****************************************";
   int costmat[6][6];
   int nodes, i, j, k;
   cout<<"\n\n Enter the number of nodes : ";
   cin>>nodes;
   cout<<"\n Enter the cost matrix : \n" ;
   for(i = 0; i < nodes; i++)
    {
      for(j = 0; j < nodes; j++)
          {
             cin>>costmat[i][j];
           costmat[i][i] = 0;
            DVR[i].dist[j] = costmat[i][j];
            DVR[i].from[j] = j;
      }
 }
   for(i = 0; i < nodes; i++)
   for(j = i+1; j < nodes; j++)
   for(k = 0; k < nodes; k++)
  if(DVR[i].dist[j] > costmat[i][k] + DVR[k].dist[j])
    {
       DVR[i].dist[j] = DVR[i].dist[k] + DVR[k].dist[j];
      DVR[j].dist[i] = DVR[i].dist[j];
      DVR[i].from[j] = k;
      DVR[j].from[i] = k;
          }
for(i = 0; i < nodes; i++)
```

```
{
  cout<<"\n\n For router: "<<i+1;
  for(j = 0; j < nodes; j++)
cout<<"\t\n node "<<j+1<<" via "<<DVR[i].from[j]+1<<" Distance "<<DVR[i].dist[j];
           }
    cout<<" \n\n ";
    getch();
}
```

OUTPUT

```
IMPLEMENT DISTANCE VECTOR ROUTING ALGORITHM
*******************************************

Enter the number of nodes : 3

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


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 : 4
node 2 via 2 Distance : 0
node 3 via 3 Distance : 6

For router: 3
node 1 via 1 Distance : 7
node 2 via 2 Distance : 8
node 3 via 3 Distance : 0
```

RESULT:

The above program has been executed successfully and output was verified.

DIJKSTRA ALGORITHM FOR SHORTEST PATH ROUTING

EX NO : 06

AIM:

To simulate and implement Dijkstra algorithm for shortest path routing.

ALGORITHM:

Step 1 : Start the process.

Step 2 : Declare the header file and create function shortest (int ,int).

Step 3 : Declare the variable cost [20][20],dist[20],I,j,k,m,n,p,s[20],v, totcost, path[20] of type int.

Step 4 : Enter the no of vertices,edges,and edge cost.

Step 5 : Enter initial vertex and call the function shortest (v,n).

Step 6 : In function definition to calculate the shortest path routing.

Step 7 : Display the results.

Step 8 : Stop the process.

PROGRAM

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
int shortest(int ,int);
int cost[10][10],dist[20],i,j,n,k,m,S[20],v,totcost,path[20],p;
void main()
{
int c;
clrscr();
cout<<"\n IMPLEMENT DIJKESHITRA ALGORITHM FOR SHORTEST PATH ROUTING";
cout<<"\n**************************************************";
cout <<"\n\n\n Enter no of vertices   : ";
cin >> n;
cout <<"\nEnter no of edges      : ";
cin >>m;
cout <<"\nEnter the EDGE Cost    : \n";
for(k=1;k<=m;k++)
{
cin >> i >> j >>c;
cost[i][j]=c;
}
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
if(cost[i][j]==0)
cost[i][j]=31999;
cout <<"\nEnter initial vertex  : ";
cin >>v;
cout << v<<"\n";
shortest(v,n);
getch();
}
int shortest(int v,int n)
{
int min;
for(i=1;i<=n;i++)
{
S[i]=0;
dist[i]=cost[v][i];
}
```

```
path[++p]=v;
S[v]=1;
dist[v]=0;
for(i=2;i<=n-1;i++)
{
k=-1;
min=31999;
for(j=1;j<=n;j++)
{
if(dist[j]<min && S[j]!=1)
{
min=dist[j];
k=j;
}
}
if(cost[v][k]<=dist[k])
p=1;
path[++p]=k;
for(j=1;j<=p;j++)
cout<<path[j];
cout <<"\n";
//cout <<k;
S[k]=1;
for(j=1;j<=n;j++)
if(cost[k][j]!=31999 && dist[j]>=dist[k]+cost[k][j] && S[j]!=1)
 dist[j]=dist[k]+cost[k][j];
}
return 0;
}
```

OUTPUT

```
IMPLEMENT DIJKESHITRA ALGORITHM FOR SHORTEST PATH ROUTING
**********************************************************
Enter no of vertices    : 5
Enter no of edges       : 7
Enter
EDGE cost
1
2
10
1
3
2
1
5
100
2
4
3
3
2
5
4
5
5
4
3
15
Enter Initial vertex 1
1
1 3
1 3  2
1 3  2  4
1 3  2  4  5
```

RESULT:

The above program has been executed successfully and output was verified.

PARITY GENERATOR

EX NO : 07

AIM:

To write a C++ program to implement parity generator code from a **g**iven bit pattern.

ALGORITHM:

Step 1 : Start the process.

Step 2 : Declare the header file

Step 3 : In main function, declare the variable as a[20],i,n, flag=0 of type int.

Step 4 : Enter the no of bits from the user.

Step 5 : Enter **the** data **i**n binary format.

Step 6 : Calculate the parity generator by using if(flag%2).

Step 7 : Display the results.

Step 8 : Stop the process.

PROGRAM

```
#include<stdio.h>
#include<iostream.h>
#include<conio.h>
void main()
{
int a[20],n,i,flag=0;
clrscr();
cout<<"\n Enter no of bits: " ;
cin>>n;
cout<<"\n Enter data: ";
for(i=0;i<n;i++)
cin>>a[i];
for(i=0;i<n;i++)
{if(a[i])
flag++;
}//cout<<"\n  "<<flag;
if(flag%2)
{
a[n]=1;
cout<<"\nThe Given Code is odd Parity";
}
else
{
a[n]=0;
cout<<"\n The Given Code is even Parity";
}
cout<<"\n\n\n Parity generator code: \n";
for(i=0;i<=n;i++)
cout<<a[i]<<" ";
getch();
}
```

OUTPUT

Enter no of bits:
4

Enter data:
1010
1100
1001
1111

The Given Code is even Parity

Parity generator code:

1010 1100 1001 1111 0

RESULT:

The above program has been executed successfully and output was verified.