**Computer Network Lab Manual**

**Part B**

**1. Write a C/C++ program to implement the data link layer framing methods.**

1. **bit stuffing**

#include<stdio.h>

#include<string.h>

void main()

{

char data[50], stuff[50];

int i, j, count, len;

printf("enter the data\n");

scanf("%s",data);

len=strlen(data);

count= 0;

j=0;

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

{

if(data[i]=='1') count++;

else count=0;

stuff[j]=data[i];

j++;

if(count==5 && data[i+1]=='1')

{

stuff[j]='0';

j++;

count=0;

}

}

printf("Stuffed data is : \n 01111110 %s 01111110", stuff);

getch();

}

Output

Enter the data:

01111110110

Stuffed data is:

01111110 011111010110 01111110

**B) Character stuffing**

#include<stdio.h>

#include<string.h>

void main(){

char frame[50][50],str[50][50];

char flag[10];

strcpy(flag,"flag");

char esc[10];

strcpy(esc,"esc");

int i,j,k=0,n;

strcpy(frame[k++],"flag");

printf("Enter no.of String :\t");

scanf("%d",&n);

printf("Enter String \n");

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

{

gets(str[i]);

}

printf("You entered :\n");

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

{

puts(str[i]);

}

printf("\n");

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

{

if(strcmp(str[i],flag)!=0 && strcmp(str[i],esc)!=0)

{

strcpy(frame[k++],str[i]);

}

else

{

strcpy(frame[k++],"esc");

strcpy(frame[k++],str[i]);

}

}

strcpy(frame[k++],"flag");

printf("------------------------------\n");

printf("Byte stuffing at sender side:\n\n");

printf("------------------------------\n");

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

{

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

}

}

**2. Write a C/C++program to implement Distance Vector Routing Algorithm.**

#include<stdio.h>

struct routing\_table

{

int dist[10], nexthop[10];

};

struct routing\_table network nodes[10];

void init(int n)

{

int i,j;

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

{

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

{

if(i!=j) {

nodes[i].dist[j]=999;

nodes[i].nexthop[j]= -20;

}

nodes[i].dist[i]=0;

nodes[i].nexthop[i]= -20;

}

}

}

void update(int i,int j,int k)

{

nodes[i].nexthop[j] = k;

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

}

void dvr(int n)

{

int i, j, k;

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

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

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

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

{ update(i,j,k); }

}

void main()

{

int n, i, j;

printf("enter the num of nodes\n");

scanf("%d",&n);

init(n);

printf("enter the distances metric \n");

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

{

printf("enter the node %c routing table \n",65+i);

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

scanf("%d",&nodes[i].dist[j]);

}

dvr(n);

printf("distance vector routing algorithm\n");

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

{

printf(" Updated node %c table\n",65+i);

printf("\t DEST \t DIST \t HOP\n");

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

if(i!=j)

{

printf("\t %c \t %d \t %c\n",65+j,nodes[i].dist[j],65+nodes[i].nexthop[j]);

}

}

}

**Output**

enter the num of nodes

5

enter the distances metric

enter the node A routing table

0 5 2 3 99

enter the node B routing table

5 0 4 99 3

enter the node C routing table

2 4 0 99 4

enter the node D routing table

3 99 99 0 99

enter the node E routing table

99 3 4 99 0

distance vector routing algorithm

Updated node A table

DEST DIST HOP

B 5 -

C 2 -

D 3 -

E 6 C

Updated node B table

DEST DIST HOP

A 5 -

C 4 -

D 8 A

E 3 -

Updated node C table

DEST DIST HOP

A 2 -

B 4 -

D 5 A

E 4 -

Updated node D table

DEST DIST HOP

A 3 -

B 8 A

C 5 A

E 9 A

Updated node E table

DEST DIST HOP

A 6 C

B 3 -

C 4 -

D 9 C

**3. Write a C/C++ Program To Implement Stop and Wait Flow Control Protocol.**

#include <time.h>

#include <stdio.h>

#include <stdlib.h>

#define TIMEOUT 5

#define MAX\_SEQ 1

#define TOT\_PACKETS 3

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

typedef struct

{

int data;

}packet;

typedef struct

{

int kind;

int seq;

int ack;

packet info;

int err;

}frame;

frame DATA;

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

void from\_network\_layer(packet \*);

void to\_network\_layer(packet \*);

void to\_physical\_layer(frame \*);

void from\_physical\_layer(frame \*);

void wait\_for\_event\_sender(event\_type \*);

void wait\_for\_event\_reciever(event\_type \*);

void reciever();

void sender();

int i=1; //Data to be sent by sender

char turn='s'; //r , s

int DISCONNECT=0;

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

int main()

{

//clrscr();

// rand();

while(!DISCONNECT)

{

sender();

for(long int i=0;i<1000000;i++); //delay(40);

reciever();

}

return 0;

}

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

void sender()

{

static int Sn=0;

//static frame s;

static frame r,s;

packet buffer;

event\_type event;

static int flag=0;

if(flag ==0)

{

from\_network\_layer(&buffer);

s.info = buffer;

s.seq = Sn;

printf("\n SENDER : Info = %d seq no = %d ",s.info,s.seq);

inc(Sn);

turn = 'r';

to\_physical\_layer(&s);

flag=1;

}

wait\_for\_event\_sender(&event);

if(turn=='s')

{

if(event==frame\_arrival) // ack recieved

{ from\_physical\_layer(&r);

if(r.seq==Sn)

flag=0;

}

if (event==timeout)

{

printf("\nSENDER : Resending Frame ");

turn = 'r';

to\_physical\_layer(&s);

}

}

}

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

void reciever()

{

static int Rn=0;

static frame fr,fs;

event\_type event;

wait\_for\_event\_reciever(&event);

if(turn=='r')

{

if(event==frame\_arrival)

{

from\_physical\_layer(&fr);

if(fr.seq==Rn)

{ inc(Rn);

to\_network\_layer(&fr.info);

printf(" \n\t ACK SENT %d ",Rn);

}

else

printf("\nRECIEVER : Duplicate Frame.... Acknowledgement Resent\n");

turn = 's';

fs.seq=Rn;

to\_physical\_layer(&fs);

}

if(event==err)

{

printf("\nRECIEVER : Corrupted Frame\n");

turn = 's'; //if frame not recieved

} //sender shold send it again

}// if turn

}//void receiver

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

void from\_network\_layer(packet \*buffer)

{

(\*buffer).data = i;

i++;

}

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

void to\_physical\_layer(frame \*s)

{ static int count =1; // 0 means error

s->err = rand()%2; //non zero means no error

printf("\n\terror rate =%d",s->err);

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

}

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

void to\_network\_layer(packet \*buffer)

{

printf("\nRECIEVER :Packet %d recieved \n",(\*buffer).data);

if(i>TOT\_PACKETS) //if all packets recieved then disconnect

{

DISCONNECT = 1;

printf("\nDISCONNECTED");

}

}

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

void from\_physical\_layer(frame \*buffer)

{

\*buffer = DATA;

}

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

void wait\_for\_event\_sender(event\_type \* e)

{

static int timer=0;

if(turn=='s')

{

timer++;

if(timer==TIMEOUT)

{

\*e = timeout;

printf("\nSENDER : Ack not recieved=> TIMEOUT\n");

timer = 0;

return;

}

if(DATA.err==0) \*e = err;

else

{ timer = 0; \*e = frame\_arrival; //ack

}

}

}

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

void wait\_for\_event\_reciever(event\_type \* e)

{

if(turn=='r')

{

if(DATA.err==0)

\*e = err;

else

\*e = frame\_arrival;

}

}

**Sample Output 1 :**

SENDER : Info = 1 seq No = 0

error rate = 1 RECIEVER :Packet 1 recieved , Ack Sent

error rate = 1 SENDER : Info = 2 seq no = 1

error rate = 0 RECIEVER : Corrupted Frame

SENDER : Ack not recieved=> TIMEOUT

SENDER : Resending Frame

error rate = 1 RECIEVER :Packet 2 recieved , Ack Sent

error rate = 1 SENDER : Info = 3 seq no = 0 error rate =1

RECIEVER :Packet 3 recieved , Ack Sent

DISCONNECTED

**Sample Output 2**

SENDER : Info = 1 seq no = 0

error rate =1

RECIEVER :Packet 1 recieved

ACK SENT 1

error rate =2

SENDER : Info = 2 seq no = 1

error rate =3

RECIEVER :Packet 2 recieved

ACK SENT 0

error rate =0

SENDER : Ack not recieved=> TIMEOUT

SENDER : Resending Frame

error rate =1

RECIEVER : Duplicate Frame.... Acknowledgement Resent

error rate =2

SENDER : Info = 3 seq no = 0

error rate =3

RECIEVER :Packet 3 recieved

DISCONNECTED

ACK SENT 1

error rate =0

**4. Write a C/++ Program for ERROR detecting code using CRC-CCITT (16bit).**

#include<stdio.h>

#include<string.h>

#include<conio.h>

#define N strlen(g)

char t[128], cs[128], g[]="10001000000100001";

int a, e, c;

void xor() {

for(c=1;c<N;c++) cs[c]=((cs[c]==g[c])?'0':'1');

}

void crc() {

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("\nEnter poly : "); scanf("%s",t);

printf("\nGenerating Polynomial is : %s",g);

a=strlen(t);

for(e=a;e<a+N-1;e++) t[e]='0';

printf("\nModified t[u] is : %s",t);

crc();

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

for(e=a;e<a+N-1;e++) t[e]=cs[e-a];

printf("\nFinal Codeword is : %s",t);

printf("\nTest Error detection 0(yes) 1(no) ? : ");

scanf("%d",&e);

if(e==0) {

printf("Enter position where error is to inserted : ");

scanf("%d",&e);

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

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

}

crc();

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

if(e<N-1) printf("Error detected.");

else printf("No Error Detected.");

//getch();

return 0;

}

**Example Output :**

Enter poly : 1011

Generating Polynomial is : 10001000000100001

Modified t[u] is : 10110000000000000000

Checksum is : 1011000101101011

Final Codeword is : 10111011000101101011

Test Error detection 0(yes) 1(no) ? : 0

Enter position where error is to inserted : 3

Errorneous data : 10101011000101101011

Error detected.

**5. Write a C/C++ Program for Congestion control using Leaky Bucket Algorithm.**

#include<stdio.h>

#include<stdlib.h>

struct packet

{

int time;

int size;

}p[50];

int main()

{

int i,n,m,k=0;

int bsize,bfilled,outrate;

printf("Enter the number of packets: ");

scanf("%d",&n);

printf("Enter packets in the order of they are arrival time\n");

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

{

printf("Enter the time and size: ");

scanf("%d%d",&p[i].time,&p[i].size);

}

printf("Enter the bucket size: ");

scanf("%d",&bsize);

printf("Enter the output rate: ");

scanf("%d",&outrate);

m=p[n-1].time;

i=1;

k=0;

bfilled=0;

while(i<=m || bfilled!=0)

{

printf("\n\nAt time %d",i);

if(p[k].time==i )

{

if(bsize>=bfilled + p[k].size)

{

bfilled=bfilled + p[k].size;

printf("\n%d byte packet is inserted",p[k].size);

k=k+1;

}

else

{

printf("\n%d byte packet is discarded",p[k].size);

k=k+1;

}

}

if(bfilled==0)

{

printf("\nNo packets to transmitte");

}

else if(bfilled>=outrate)

{

bfilled=bfilled-outrate;

printf("\n%d bytes transfered",outrate);

}

else

{

printf("\n%d bytes transfered",bfilled);

bfilled=0;

}

printf("\nPackets in the bucket %d byte",bfilled);

i++;

}

return 0;

}

**Example Output1 :**

Enter the number of packets: 3

Enter packets in the order of they are arrival time

Enter the time and size: 1 100

Enter the time and size: 2 400

Enter the time and size: 3 600

Enter the bucket size: 500

Enter the output rate: 200

At time 1

100 byte packet is inserted

100 bytes transfered

Packets in the bucket 0 byte

At time 2

400 byte packet is inserted

200 bytes transfered

Packets in the bucket 200 byte

At time 3

600 byte packet is discarded

200 bytes transfered

Packets in the bucket 0 byte

**Example Output 2**

Enter the time and size: 1 100

Enter the time and size: 3 200

Enter the time and size: 2 400

Enter the time and size: 4 600

Enter the bucket size: 200

Enter the output rate: 100

At time 1

100 byte packet is inserted

100 bytes transfered

Packets in the bucket 0 byte

At time 2

No packets to transmitted

Packets in the bucket 0 byte

At time 3

200 byte packet is inserted

100 bytes transfered

Packets in the bucket 100 byte

At time 4

100 bytes transfered

Packets in the bucket 0 byte