#include <conio.h>

#include <dos.h>

#include <stdio.h>

#include <stdlib.h>

#define TIMEOUT 5

#define MAX\_SEQ 1

#define TOT\_PACKETS 8

#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; //r , s

int DISCONNECT=0;

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

void main()

{

clrscr();

randomize();

while(!DISCONNECT)

{

sender();

delay(400);

reciever();

}

getch();

}

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

void sender()

{

static int frame\_to\_send=0;

static frame s;

packet buffer;

event\_type event;

static int flag=0;

if(flag==0)

{

from\_network\_layer(&buffer);

s.info = buffer;

s.seq = frame\_to\_send;

printf("SENDER : Info = %d Seq No = %d ",s.info,s.seq);

turn = 'r';

to\_physical\_layer(&s);

flag = 1;

}

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

if(turn=='s')

{

if(event==frame\_arrival)

{

from\_network\_layer(&buffer);

inc(frame\_to\_send);

s.info = buffer;

s.seq = frame\_to\_send;

printf("SENDER : Info = %d Seq No = %d ",s.info,s.seq);

turn = 'r';

to\_physical\_layer(&s);

}

if(event==timeout)

{

printf("SENDER : Resending Frame ");

turn = 'r';

to\_physical\_layer(&s);

}

}

}

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

void reciever()

{

static int frame\_expected=0;

frame r,s;

event\_type event;

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

if(turn=='r')

{

if(event==frame\_arrival)

{

from\_physical\_layer(&r);

if(r.seq==frame\_expected)

{

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

inc(frame\_expected);

}

else

printf("RECIEVER : Acknowledgement Resent\n");

turn = 's';

to\_physical\_layer(&s);

}

if(event==err)

{

printf("RECIEVER : Garbled Frame\n");

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

} //sender shold send it again

}

}

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

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

{

(\*buffer).data = i;

i++;

}

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

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

{ // 0 means error

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

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

}

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

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

{

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

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

{

DISCONNECT = 1;

printf("\nDISCONNECTED");

}

}

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

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

{

\*buffer = DATA;

}

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

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

{

static int timer=0;

if(turn=='s')

{

timer++;

if(timer==TIMEOUT)

{

\*e = timeout;

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

timer = 0;

return;

}

if(DATA.err==0)

\*e = err;

else

{

timer = 0;

\*e = frame\_arrival;

}

}

}

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

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

{

if(turn=='r')

{

if(DATA.err==0)

\*e = err;

else

\*e = frame\_arrival;

}

}

OUTPUT:-