STOP AND WAIT PROTOCOL IMPLEMENTATION IN C

Aim:-

To write a program to simulate stop – and – wait ARQ for noisy channel.

Description of Stop-and-Wait Protocol

Stop-and-wait Protocol is a flow control protocol used in the data link layer for transmission of data in noiseless channels. Sender keeps on sending messages to the Receiver. In order to prevent the receiver from overwhelming, there is a need to tell the sender to slow down the transmission of frames. We can make use of feedback from the receiver to the sender. Frames 0 sends to receiver, ACK 1 will be sent back to sender; frame 1 goes to receiver, ACK 0 will be back to sender, and so on.

Algorithm

1. Start the program

2. Generate a random number that gives the total number of frames to be transmitted.

3. Transmit the first frame

4. Receive the acknowledgement for the first frame

5. Transmit the next frame

6. Find the remaining frames to be sent.

7. If an acknowledgement is not received for a particular frame, retransmit that frame alone again.

8. Repeat the steps 5 to 7 till the number of remaining frames to be sent becomes zero.

9. Stop the program.

**Algorithms for both Sender and Receiver sides**

Algorithm to write in record

**STOP and Wait ARQ for NOISY CHANNEL**

**Stand alone Code** – stopandwait.c

#include <stdio.h>

#include <stdlib.h>

struct frame

{ int info;

int seq;

} ;

int ak;

int t=5,k;

int disconnect=0;

struct frame p;

char turn='s'; //Initialize first turn as sender

int errorframe=1; //no Error

int errorack=1;

void sender();

void receiver();

void main()

{ p.info=0;//data part

p.seq=0;//sequence number

while(!disconnect)

{ sender(); //call sender

for(k=1;k<=10000000;k++);

//After a finite amount of time call receiver

receiver();

}

}

void sender()

{ static int flag=0;

if(turn=='s') //sender turn

{ if(errorack==0) //Ack didn't arrive

{ printf("SENDER: sent packet with seq NO:%d\n",p.seq);

errorframe=rand()%4; //randomly pick Error frame as 4

printf("%s\n",(errorframe==0?"Error While sending Packet":""));

turn='r';

}

else

{

if (flag==1) printf("SENDER: Received ACK for packet %d\n",ak);

if (p.seq==5){ disconnect=1; return;}

p.info=p.info+1;

p.seq=p.seq+1;

printf("SENDER: sent packet with seq NO:%d\n",p.seq);

errorframe=rand()%4;

//Message below is printed only if Error

//occured while sending Packet

printf("%s\n",(errorframe==0?"Error While sending Packet":""));

turn='r';

//Set next turn as Receiver for transmission

flag=1;

}

}

else

{ t--;

printf("SENDER time reducing\n");

if(t==0)

{turn='s';

errorack=0;

t=5;

}

}

}

void receiver()

{ static int frexp=1;

if(turn=='r')

{

if (errorframe!=0)

{ if(p.seq==frexp)

//if frame sequence number is eq to frexp

{ printf("RECEIVER: Received packet with seq %d\n",p.seq);

//note sequence number of frame arrived

//to send acknowledgement

ak=p.seq;

//increment the frame sequence number

frexp=frexp+1;

//Set next turn as sender

turn='s';

//Send acknowledgement error for frame number 4

errorack=rand()%4;

printf("%s\n",(errorack==0?"Error While sending ACK":""));

}

else

{

//Receiver received Duplicated frame for lost frame after Resending

printf("RECEIVER: Duplicated packet with seq %d\n",frexp-1);

//Note down acknowledgement number of frame

ak=frexp-1;

//next turn sender

turn='s';

errorack=rand()%4;

printf("%s\n",(errorack==0?"Error While sending ACK":""));

}

}

}

}

**Output**

labb04@labb04:~/Desktop$ gcc stopandwait.c

labb04@labb04:~/Desktop$ ./a.out

SENDER: sent packet with seq NO:1

RECEIVER: Received packet with seq 1

SENDER: Received ACK for packet 1

SENDER: sent packet with seq NO:2

RECEIVER: Received packet with seq 2

SENDER: Received ACK for packet 2

SENDER: sent packet with seq NO:3

RECEIVER: Received packet with seq 3

SENDER: Received ACK for packet 3

SENDER: sent packet with seq NO:4

RECEIVER: Received packet with seq 4

Error While sending ACK

SENDER: sent packet with seq NO:4

RECEIVER: Duplicated packet with seq 4

SENDER: Received ACK for packet 4

SENDER: sent packet with seq NO:5

RECEIVER: Received packet with seq 5

SENDER: Received ACK for packet 5