1. Write a client server program to communicate between them (two way).

Ans:

Client

#include<stdio.h>

#include<unistd.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<string.h>

int main()

{

struct sockaddr\_in c\_addr;

int c\_fd,c\_len;

char buff[100];

if((c\_fd=socket(AF\_INET,SOCK\_STREAM,0))==-1)

printf("[-]Error in Socket\n");

printf("[+]Client Socket created\n");

c\_addr.sin\_family=AF\_INET;

c\_addr.sin\_addr.s\_addr=INADDR\_ANY;

c\_addr.sin\_port=3452;

c\_len=sizeof(c\_addr);

if(connect(c\_fd,(struct sockaddr\*)& c\_addr,c\_len)==-1)

printf("[-]Error in Connect\n");

printf("[+]Connected to the Server: \n\n");

while(1)

{

read(c\_fd,buff,100);

printf("From Server: %s\n",buff);

printf("Client Message: ");

fgets(buff,sizeof(buff),stdin);

if (strcmp(buff, "Exit\n") == 0){

printf("[-]Disconneted from Server");

write(c\_fd,"Disconnected\n",100);

break;

}

else{

write(c\_fd,buff,100);

}

}

close(c\_fd);

return 0;

}

Server

#include<stdio.h>

#include<unistd.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<sys/socket.h>

#include<sys/types.h>

int main()

{

struct sockaddr\_in s\_addr,c\_addr;

int s\_fd,c\_fd,s\_len,c\_len;

if((s\_fd=socket(AF\_INET,SOCK\_STREAM,0))==-1)

printf("[-]Error in Socket\n");

printf("[+]Server Socket created\n");

s\_addr.sin\_family = AF\_INET;

s\_addr.sin\_port = 3452;

s\_len=sizeof(s\_addr);

if(bind(s\_fd,(struct sockaddr\*)& s\_addr,s\_len)==-1)

printf("[-]Error in binding\n");

printf("[+]Binding successfull\n");

if(listen(s\_fd,5)==-1)

printf("[-]Error in listen\n");

printf("[+]Listening... \n");

c\_len=sizeof(c\_addr);

if((c\_fd=accept(s\_fd,(struct sockaddr\*)&c\_addr,&c\_len))==-1)

printf("\n[-]Error in accepting\n");

printf("[+]New Client Connected!!!: \n\n");

char buff[100];

while(1)

{

printf("Server Message: ");

fgets(buff,sizeof(buff),stdin);

write(c\_fd,buff,100);

read(c\_fd,buff,100);

printf("From Client: %s\n",buff);

}

close(c\_fd);

return 0;

}

Output

2. Write a program that will ask the client program to enter two numbers and the server will display the addition, subtraction, and multiplication of that two numbers.

Ans:

Client

#include <stdio.h>

#include <unistd.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <string.h>

#include <stdlib.h>

int main()

{

struct sockaddr\_in s\_addr, c\_addr;

int s\_fd, c\_fd, s\_len, c\_len;

if ((s\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == -1)

printf("[-]Error in Socket\n");

printf("[+]Server Socket created\n");

s\_addr.sin\_family = AF\_INET;

s\_addr.sin\_port = 3452;

s\_len = sizeof(s\_addr);

if (bind(s\_fd, (struct sockaddr \*)&s\_addr, s\_len) == -1)

printf("[-]Error in binding\n");

printf("[+]Binding successfull\n");

if (listen(s\_fd, 5) == -1)

printf("[-]Error in listen\n");

printf("[+]Listening... \n");

int nums[2], n = 0;

c\_len = sizeof(c\_addr);

if ((c\_fd = accept(s\_fd, (struct sockaddr \*)&c\_addr, &c\_len)) == -1)

printf("\n[-]Error in accepting\n");

printf("[+]New Client Connected!!!: \n\n");

while (n < 2)

{

char buff[100];

read(c\_fd, buff, 100);

printf("From Client Number %d is %s", n + 1, buff);

nums[n++] = atoi(buff);

}

printf("\nAddition of %d and %d is %d\n", nums[0], nums[1], nums[0] + nums[1]);

printf("Subtaction of %d and %d is %d\n", nums[0], nums[1], nums[0] - nums[1]);

printf("Multiplication of %d and %d is %d\n", nums[0], nums[1], nums[0] \* nums[1]);

close(c\_fd);

return 0;

}

Server

#include <stdio.h>

#include <unistd.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <string.h>

#include <stdlib.h>

int main()

{

struct sockaddr\_in c\_addr;

int c\_fd, c\_len;

char buff[100];

if ((c\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == -1)

printf("[-]Error in Socket\n");

printf("[+]Client Socket created\n");

c\_addr.sin\_family = AF\_INET;

c\_addr.sin\_addr.s\_addr = INADDR\_ANY;

c\_addr.sin\_port = 3452;

c\_len = sizeof(c\_addr);

if (connect(c\_fd, (struct sockaddr \*)&c\_addr, c\_len) == -1)

printf("[-]Error in Connect\n");

printf("[+]Connected to the Server: \n\n");

int n = 0;

while (n < 2)

{

printf("Enter Number %d: ", n + 1);

fgets(buff, sizeof(buff), stdin);

write(c\_fd, buff, 100);

n++;

}

close(c\_fd);

return 0;

}

Output

3. Write a program to transfer a text file from the client to server and vice-versa.

Ans:

Client

#include <arpa/inet.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h>

#define IP\_PROTOCOL 0

#define IP\_ADDRESS "127.0.0.1" // localhost

#define PORT\_NO 15050

#define NET\_BUF\_SIZE 32

#define cipherKey 'S'

#define sendrecvflag 0

// function to clear buffer

void clearBuf(char\* b)

{

int i;

for (i = 0; i < NET\_BUF\_SIZE; i++)

b[i] = '\0';

}

// function for decryption

char Cipher(char ch)

{

return ch ^ cipherKey;

}

// function to receive file

int recvFile(char\* buf, int s)

{

int i;

char ch;

for (i = 0; i < s; i++) {

ch = buf[i];

ch = Cipher(ch);

if (ch == EOF)

return 1;

else

printf("%c", ch);

}

return 0;

}

// driver code

int main()

{

int sockfd, nBytes;

struct sockaddr\_in addr\_con;

int addrlen = sizeof(addr\_con);

addr\_con.sin\_family = AF\_INET;

addr\_con.sin\_port = htons(PORT\_NO);

addr\_con.sin\_addr.s\_addr = inet\_addr(IP\_ADDRESS);

char net\_buf[NET\_BUF\_SIZE];

FILE\* fp;

// socket()

sockfd = socket(AF\_INET, SOCK\_DGRAM,

IP\_PROTOCOL);

if (sockfd < 0)

printf("\nfile descriptor not received!!\n");

else

printf("\nfile descriptor %d received\n", sockfd);

while (1) {

printf("\nPlease enter file name to receive:\n");

scanf("%s", net\_buf);

sendto(sockfd, net\_buf, NET\_BUF\_SIZE,

sendrecvflag, (struct sockaddr\*)&addr\_con,

addrlen);

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

while (1) {

// receive

clearBuf(net\_buf);

nBytes = recvfrom(sockfd, net\_buf, NET\_BUF\_SIZE,

sendrecvflag, (struct sockaddr\*)&addr\_con,

&addrlen);

// process

if (recvFile(net\_buf, NET\_BUF\_SIZE)) {

break;

}

}

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

}

return 0;

}

Server

#include <arpa/inet.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h>

#define IP\_PROTOCOL 0

#define PORT\_NO 15050

#define NET\_BUF\_SIZE 32

#define cipherKey 'S'

#define sendrecvflag 0

#define nofile "File Not Found!"

// function to clear buffer

void clearBuf(char\* b)

{

int i;

for (i = 0; i < NET\_BUF\_SIZE; i++)

b[i] = '\0';

}

// function to encrypt

char Cipher(char ch)

{

return ch ^ cipherKey;

}

// function sending file

int sendFile(FILE\* fp, char\* buf, int s)

{

int i, len;

if (fp == NULL) {

strcpy(buf, nofile);

len = strlen(nofile);

buf[len] = EOF;

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

buf[i] = Cipher(buf[i]);

return 1;

}

char ch, ch2;

for (i = 0; i < s; i++) {

ch = fgetc(fp);

ch2 = Cipher(ch);

buf[i] = ch2;

if (ch == EOF)

return 1;

}

return 0;

}

// driver code

int main()

{

int sockfd, nBytes;

struct sockaddr\_in addr\_con;

int addrlen = sizeof(addr\_con);

addr\_con.sin\_family = AF\_INET;

addr\_con.sin\_port = htons(PORT\_NO);

addr\_con.sin\_addr.s\_addr = INADDR\_ANY;

char net\_buf[NET\_BUF\_SIZE];

FILE\* fp;

// socket()

sockfd = socket(AF\_INET, SOCK\_DGRAM, IP\_PROTOCOL);

if (sockfd < 0)

printf("\nfile descriptor not received!!\n");

else

printf("\nfile descriptor %d received\n", sockfd);

// bind()

if (bind(sockfd, (struct sockaddr\*)&addr\_con, sizeof(addr\_con)) == 0)

printf("\nSuccessfully binded!\n");

else

printf("\nBinding Failed!\n");

while (1) {

printf("\nWaiting for file name...\n");

// receive file name

clearBuf(net\_buf);

nBytes = recvfrom(sockfd, net\_buf,

NET\_BUF\_SIZE, sendrecvflag,

(struct sockaddr\*)&addr\_con, &addrlen);

fp = fopen(net\_buf, "r");

printf("\nFile Name Received: %s\n", net\_buf);

if (fp == NULL)

printf("\nFile open failed!\n");

else

printf("\nFile Successfully opened!\n");

while (1) {

// process

if (sendFile(fp, net\_buf, NET\_BUF\_SIZE)) {

sendto(sockfd, net\_buf, NET\_BUF\_SIZE,

sendrecvflag,

(struct sockaddr\*)&addr\_con, addrlen);

break;

}

// send

sendto(sockfd, net\_buf, NET\_BUF\_SIZE,

sendrecvflag,

(struct sockaddr\*)&addr\_con, addrlen);

clearBuf(net\_buf);

}

if (fp != NULL)

fclose(fp);

}

return 0;

}

Output

4. A database is created with the following fields: roll no, student name, address, marks1, marks2. The database will be stored in the server and the client will fetch the information of a student by sending the roll no of a particular student. Implement this scenario using client server program.

Ans:

Client

#include<stdio.h>

#include<unistd.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<string.h>

#include<stdlib.h>

int main()

{

struct sockaddr\_in c\_addr;

int c\_fd,c\_len;

char buff[100];

if((c\_fd=socket(AF\_INET,SOCK\_STREAM,0))==-1){

printf("[-]Error in Socket\n");

exit(0);

}

printf("[+]Client Socket created\n");

c\_addr.sin\_family=AF\_INET;

c\_addr.sin\_addr.s\_addr=INADDR\_ANY;

c\_addr.sin\_port=3452;

c\_len=sizeof(c\_addr);

if(connect(c\_fd,(struct sockaddr\*)& c\_addr,c\_len)==-1) {

printf("[-]Error in Connect\n");

exit(0);

}

printf("[+]Connected to the Server: \n\n");

while(1)

{

printf("\nEnter Roll No of Student: \n");

fgets(buff,sizeof(buff),stdin);

if (strcmp(buff, "Exit\n") == 0){

printf("[-]Disconneted from Server");

write(c\_fd,"Disconnected\n",100);

break;

}

else{

write(c\_fd,buff,100);

}

read(c\_fd,buff,100);

printf("%s",buff);

}

close(c\_fd);

return 0;

}

Server

#include<stdio.h>

#include<unistd.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<stdlib.h>

struct StudentDetails

{

int roll\_no;

char\* name;

char\* address;

float marks1;

float marks2;

};

int main()

{

struct sockaddr\_in s\_addr,c\_addr;

int s\_fd,c\_fd,s\_len,c\_len;

struct StudentDetails studentdet[13];

studentdet[0].roll\_no = 1;

studentdet[0].name = "Indraneel";

studentdet[0].address = "CGEC";

studentdet[0].marks1 = 89.0;

studentdet[0].marks2 = 99.0;

studentdet[1].roll\_no = 2;

studentdet[1].name = "Sachin";

studentdet[1].address = "CGEC";

studentdet[1].marks1 = 81.0;

studentdet[1].marks2 = 92.0;

studentdet[2].roll\_no = 3;

studentdet[2].name = "Saurabh";

studentdet[2].address = "CGEC";

studentdet[2].marks1 = 76.0;

studentdet[2].marks2 = 87.0;

studentdet[3].roll\_no = 4;

studentdet[3].name = "Indra";

studentdet[3].address = "CGEC";

studentdet[3].marks1 = 87.0;

studentdet[3].marks2 = 94.0;

studentdet[4].roll\_no = 5;

studentdet[4].name = "Rajesh";

studentdet[4].address = "CGEC";

studentdet[4].marks1 = 77.0;

studentdet[4].marks2 = 77.0;

studentdet[5].roll\_no = 6;

studentdet[5].name = "king";

studentdet[5].address = "CGEC";

studentdet[5].marks1 = 89.0;

studentdet[5].marks2 = 91.0;

studentdet[6].roll\_no = 7;

studentdet[6].name = "Raj";

studentdet[6].address = "CGEC";

studentdet[6].marks1 = 81.0;

studentdet[6].marks2 = 91.0;

studentdet[7].roll\_no = 8;

studentdet[7].name = "Ravi";

studentdet[7].address = "CGEC";

studentdet[7].marks1 = 86.0;

studentdet[7].marks2 = 90.0;

studentdet[8].roll\_no = 9;

studentdet[8].name = "sourav";

studentdet[8].address = "CGEC";

studentdet[8].marks1 = 90.0;

studentdet[8].marks2 = 90.0;

studentdet[9].roll\_no = 10;

studentdet[9].name = "Neel";

studentdet[9].address = "CGEC";

studentdet[9].marks1 = 89.0;

studentdet[9].marks2 = 92.0;

if((s\_fd=socket(AF\_INET,SOCK\_STREAM,0))==-1){

printf("[-]Error in Socket\n");

exit(0);

}

printf("[+]Server Socket created\n");

s\_addr.sin\_family = AF\_INET;

s\_addr.sin\_port = 3452;

s\_len=sizeof(s\_addr);

if(bind(s\_fd,(struct sockaddr\*)& s\_addr,s\_len)==-1){

printf("[-]Error in binding\n");

exit(0);

}

printf("[+]Binding successfull\n");

if(listen(s\_fd,5)==-1){

printf("[-]Error in listen\n");

exit(0);

}

printf("[+]Listening... \n");

c\_len=sizeof(c\_addr);

if((c\_fd=accept(s\_fd,(struct sockaddr\*)&c\_addr,&c\_len))==-1){

printf("\n[-]Error in accepting\n");

exit(0);

}

printf("[+]New Client Connected!!!: \n\n");

char buff[100];

while(1)

{

int roll;

read(c\_fd,buff,100);

roll = atoi(buff);

for(int i = 0; i < 10; i++){

if(studentdet[i].roll\_no == roll){

snprintf(buff, 100, "\nRoll No: %d\nName: %s\nAddress: %s\nMarks 1: %2.f\nMarks 2: %2.f\n"

,studentdet[i].roll\_no, studentdet[i].name,studentdet[i].address,

studentdet[i].marks1, studentdet[i].marks2);

write(c\_fd,buff,100);

printf("[+]Detail found and sent to client successfully!");

break;

}

else if(studentdet[i].roll\_no != roll && i == 9){

printf("[-]Details not found.");

write(c\_fd,"\n[-]Details not found.\n",100);

}

}

}

close(c\_fd);

return 0;

}

Output

5. Write a program to implement Stop & Wait ARQ method.

Ans:

Client

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <unistd.h>

#include <string.h>

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/stat.h>

#define MAXLINE 80

char buffer[255] ;

#define SERVER\_ADDR "127.0.0.1"

#define SERVER\_PORT\_NUM 6000

#define t\_max\_retransmission 3

typedef int bit\_32\_var ;

typedef char bit\_8\_var ;

typedef struct pdu\_field

{

bit\_32\_var SN ;

bit\_8\_var data[MAXLINE] ;

bit\_8\_var status ;

}PDU\_FIELD;

void str\_cli ( FILE \*fp , bit\_32\_var sfd )

{

bit\_32\_var no\_of\_data , counter\_1 , counter\_2 ;

static bit\_32\_var retransmission\_counter , intial\_readycheck\_counter ;

PDU\_FIELD send\_data[MAXLINE] ,recv\_data[MAXLINE] ;

printf ( "enter how many data you have to send : " ) ;

scanf ( "%d", &no\_of\_data ) ;

for ( counter\_1 = 0 ; counter\_1 < no\_of\_data ; counter\_1++ )

{

printf ( "enter %d'th data : ",counter\_1 ) ;

scanf ( "%s", send\_data[counter\_1].data ) ;

send\_data[counter\_1].SN = counter\_1 ;

send\_data[counter\_1].status = 0 ;

}

for ( counter\_1 = 0 ; counter\_1 < no\_of\_data ; counter\_1++ )

{

write ( sfd , &send\_data[counter\_1] , sizeof ( send\_data[counter\_1] ) ) ;

read ( sfd , &recv\_data[counter\_1] , sizeof ( recv\_data[counter\_1] ) ) ;

if ( counter\_1 == 0 && ( strcmp ( recv\_data[counter\_1].data , "yes" ) != 0 ) )

{

if ( intial\_readycheck\_counter == t\_max\_retransmission )

{

printf ( "client : receiver not there , better to exit : \n" ) ;

exit ( 1 ) ;

}

intial\_readycheck\_counter++ ;

printf ( "client : receiver is not ready : wait 10sec... \n" ) ;

sleep ( 10 ) ;

counter\_1 = counter\_1- 1 ;

}

if ( counter\_1 > 0 )

{

if ( recv\_data[counter\_1].SN == send\_data[counter\_1].SN + 1 )

printf ( "\n server responding = %s ", recv\_data[counter\_1].data ) ;

else

{

if ( retransmission\_counter < t\_max\_retransmission )

{

printf ( "incorrect ack - sending the same data - \n" ) ;

counter\_1 -= 1 ;

retransmission\_counter++ ;

}

else

{

printf ( "Time out : sending next data \n " ) ;

retransmission\_counter = 0 ;

}

}

}

}

write ( sfd , "Nothingtotransmit:\n" , 10 ) ;

printf ( "Transmission finished : \n" ) ;

exit ( 1 ) ;

}

int main ()

{

bit\_32\_var socket\_fd ;

struct sockaddr\_in config\_client ;

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

if ( socket\_fd < 0 )

{

printf ( "client : failed to create socket \n" ) ;

exit ( 1 ) ;

}

memset ( &config\_client , 0 , sizeof ( struct sockaddr\_in )) ;

config\_client.sin\_family = AF\_INET ;

config\_client.sin\_port = htons ( SERVER\_PORT\_NUM ) ;

inet\_aton ( SERVER\_ADDR , &config\_client.sin\_addr ) ;

connect ( socket\_fd , (struct sockaddr \*)&config\_client , sizeof ( config\_client ) ) ;

printf ( "connect successfully\n" ) ;

system ( "clear" ) ;

str\_cli ( stdin , socket\_fd ) ;

exit ( 0 );

}

Server

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#define SUCCESS 1

#define FAILURE 0

#define NO\_OF\_FRAMES 10

typedef int BIT\_32\_VAR\_INT ;

typedef char BIT\_8\_VAR\_CHAR ;

void error ( BIT\_8\_VAR\_CHAR \*msg )

{

perror ( msg ) ;

exit ( 1 ) ;

}

typedef struct pdu\_data

{

BIT\_32\_VAR\_INT SN ;

BIT\_8\_VAR\_CHAR data[80] ;

BIT\_32\_VAR\_INT status ;

}PDU\_DATA;

int main () {

PDU\_DATA send\_data[10] , recv\_data[10] ;

BIT\_32\_VAR\_INT socket\_fd , newsocket\_fd , port\_no , client\_len ;

// char buffer[256] ;

struct sockaddr\_in serv\_addr , cli\_addr ;

int sizeof\_read\_data , counter\_1 ;

//char buf[256] ;

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

if ( socket\_fd < 0 )

{

error ( "error in socket opening " ) ;

}

bzero ( ( char \* ) &serv\_addr , sizeof ( serv\_addr ) ) ;

port\_no = atoi ( "7100" ) ;

serv\_addr.sin\_family = AF\_INET ;

serv\_addr.sin\_addr.s\_addr = INADDR\_ANY ;

serv\_addr.sin\_port = htons ( 6000 ) ;

if ( bind ( socket\_fd , ( struct sockaddr \* ) &serv\_addr , sizeof ( serv\_addr ) ) < 0 )

error ( "error in binding" ) ;

listen ( socket\_fd , 5 ) ;

client\_len = sizeof ( cli\_addr ) ;

newsocket\_fd = accept ( socket\_fd , ( struct sockaddr \* ) &cli\_addr , &client\_len ) ;

if ( newsocket\_fd < 0 )

error ( "error in accept " ) ;

// bzero ( buffer , 256 ) ;

while ( 1 )

{

for ( counter\_1 = 0 ; counter\_1 < NO\_OF\_FRAMES ; counter\_1++ )

{

sizeof\_read\_data = read ( newsocket\_fd , &recv\_data[counter\_1] , sizeof ( recv\_data[counter\_1]) ) ;

if ( sizeof\_read\_data < 0 )

error ( "error in reading from socket " ) ;

printf ( "client sended : %s with SN %d :\n" , recv\_data[counter\_1].data , recv\_data[counter\_1].SN) ;

printf ( "enter a reply to client : " ) ;

scanf ( "%s",send\_data[counter\_1].data ) ;

printf ( "\nenter a SN for next you want to receive : " ) ;

scanf ( "%d",&send\_data[counter\_1].SN ) ;

if ( send\_data[counter\_1].SN <= recv\_data[counter\_1].SN )

counter\_1 -= 1 ;

send\_data[counter\_1].status = SUCCESS ;

write ( newsocket\_fd , &send\_data[counter\_1] , sizeof ( send\_data[counter\_1]) ) ;

}

bzero ( &recv\_data[counter\_1].data , sizeof ( recv\_data[counter\_1].data ) ) ;

read ( newsocket\_fd , &recv\_data[counter\_1] , sizeof ( recv\_data[counter\_1] ) ) ;

printf ( "it's end Thank YOu : \n" ) ;

exit ( 1 ) ;

}

return 0 ;

}

6. Write a program to implement Go-Back-N ARQ method.

Ans:

#include<bits/stdc++.h>

#include<ctime>

#define ll long long int

using namespace std;

void transmission(ll & i, ll & N, ll & tf, ll & tt) {

while (i <= tf) {

int z = 0;

for (int k = i; k < i + N && k <= tf; k++) {

cout << "Sending Frame " << k << "..." << endl;

tt++;

}

for (int k = i; k < i + N && k <= tf; k++) {

int f = rand() % 2;

if (!f) {

cout << "Acknowledgment for Frame " << k << "..." << endl;

z++;

} else {

cout << "Timeout!! Frame Number : " << k << " Not Received" << endl;

cout << "Retransmitting Window..." << endl;

break;

}

}

cout << "\n";

i = i + z;

}

}

int main() {

ll tf, N, tt = 0;

srand(time(NULL));

cout << "Enter the Total number of frames : ";

cin >> tf;

cout << "Enter the Window Size : ";

cin >> N;

ll i = 1;

transmission(i, N, tf, tt);

cout << "Total number of frames which were sent and resent are : " << tt <<

endl;

return 0;

}

Output

7. Write a program to implement Selective Repeat ARQ method.

Ans:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <unistd.h>

int n,r;

struct frame{

char ack;

int data;

}frm[10];

int sender(void);

void recvack(void);

void resend\_sr(void);

void resend\_gb(void);

void goback(void);

void selective(void);

int main(){

int c;

do{

printf("1:Selective Repeat protocol\n2:Go Back N protocol\n0:Exit\n");

printf("Enter choice:");

scanf("%d",&c);

switch(c)

{

case 1:selective();

break;

case 2:goback();

case 0: exit(0);

break;

}

}while(c>=1);

} void goback(){

sender();

recvack();

resend\_gb();

printf("\nAll frames sent successfully!\n");

}

void selective(){

sender();

recvack();

resend\_sr();

printf("\nAll frames sent successfully!\n");

}

int sender(){

int i;

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

scanf("%d",&n);

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

printf("\n Enter data for frame [%d] ",i);

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

frm[i].ack='y';

}

return 0;

}

void recvack(){

int i;

rand();

r=rand()%n;

frm[r].ack='n';

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

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

printf("\n The frame number %d is not received\n",r);

}

}

}

void resend\_sr(){

printf("\nResending the frame %d",r);

sleep(2);

frm[r].ack='y';

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

}

void resend\_gb(){

int i;

printf("\nResending from frame %d",r);

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

sleep(2);

frm[i].ack='y';

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

}

}

Output: