Lab Assignment No: 02

Problem Statement: Implement Go-Back-N Protocol using Socket

Name: Adnan Sadar Roll No: 2

Class: SY Branch: IT

Batch: B1

**Server Code:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<arpa/inet.h>

#include<unistd.h>

#include<netinet/in.h>

int main()

{

//Server message

char msg[256] = "This is server talking to you";

// socket creation

int s\_socket = socket(AF\_INET,SOCK\_STREAM,0);

if(s\_socket<0)

printf("Socket creation failed!\n");

else

printf("Socket created successfully\n");

//Address specification

struct sockaddr\_in s\_address;

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

s\_address.sin\_port = htons(9003);

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

//Binding Address to Socket

int status = bind(s\_socket,(struct sockaddr \*)&s\_address,sizeof(s\_address));

if(status < 0)

printf("Binging not successful!\n");

else

printf("Successful Binding!\n");

//Listening

int listen\_status;

listen\_status = listen(s\_socket,5);

if(listen\_status <0)

printf("Listening problems\n");

else

printf("Waiting...\n");

//Accepting client sockets

int client\_socket, size = sizeof(struct sockaddr\_in);

struct sockaddr\_in client\_address;

client\_socket = accept(s\_socket,(struct sockaddr \*)&client\_socket,&size);

if(client\_socket < 0)

printf("Accept failed\n");

else

printf("Client accepted\n");

int tp,tt,n,loss;

if(recv(client\_socket, &tp ,sizeof(tp),0) < 0)

{

printf("Couldn't recieve tp");

exit(0);

// break;

}

if(recv(client\_socket, &tt ,sizeof(tt),0) < 0)

{

printf("Couldn't recieve tt");

exit(0);

// break;

}

if(recv(client\_socket, &n ,sizeof(n),0) < 0)

{

printf("Couldn't recieve n");

exit(0);

// break;

}

if(recv(client\_socket, &loss ,sizeof(n),0) < 0)

{

printf("Couldn't recieve loss");

exit(0);

// break;

}

printf("TP: %d ", tp);

printf("TT: %d ", tt);

printf("N: %d\n\n", n);

int packet;

int expected = 0;

int ack,i=0, flag=0, flag1=1,j=0;

int arrived\_time=0, arrived\_time\_copy=0;

int arrived[n];

while(1)

{

if(recv(client\_socket, &packet ,sizeof(packet),0) < 0){

printf("Couldn't recieve packet");

break;

}

if(recv(client\_socket, &arrived\_time ,sizeof(arrived\_time),0) < 0){

printf("Couldn't recieve packet");

break;

}

if(packet == loss){

arrived\_time\_copy = arrived\_time;

for(j=loss+1; j<n; j++){

printf("Dropped Packet: %d\n", j);

arrived\_time\_copy += tt;

printf("Arrival Time: %d\n\n", arrived\_time\_copy);

}

ack = -1;

if(send(client\_socket, &ack ,sizeof(ack),0) < 0){

printf("Couldn't send reply");

break;

}

arrived\_time += 2\*(2\*tp);

if(send(client\_socket, &arrived\_time ,sizeof(arrived\_time),0) < 0){

printf("Couldn't send arrived time");

break;

}

loss = -1;

continue;

}

if(packet == expected) {

arrived[i] = packet;

arrived\_time += tp;

printf("Recieved Packet: %d\n", packet);

printf("Arrival Time: %d\n\n", arrived\_time);

i++;

flag = 1;

expected++;

}

if(expected != n){

ack = 1;

}else {

ack = 0;

}

if(send(client\_socket, &ack ,sizeof(ack),0) < 0){

printf("Couldn't send reply");

break;

}

if(flag == 1){

if(send(client\_socket, &arrived\_time ,sizeof(arrived\_time),0) < 0){

printf("Couldn't send arrived time");

break;

}

}

if(ack == 0){

printf("All packets recieved!!\n");

break;

}

}

close(s\_socket);

return 0;

}

**Client Code:-**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<arpa/inet.h>

#include<netinet/in.h>

#include <math.h>

int power (int a, int b)

{

int res = 1, i;

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

res = res \* a;

}

return res;

}

int logs (int n)

{

return (n > 1) ? 1 + logs(n / 2) : 0;

}

int logbase (int y, int b)

{

int lg;

lg = logs(y)/logs(b);

return(lg);

}

int min(int a, int b){

return a > b ? b : a;

}

int main()

{

int client\_socket = socket(AF\_INET,SOCK\_STREAM,0);

if(client\_socket <0){

printf("Cannot create the socket\n");

} else {

printf("Socket created\n");

}

//Creating sever Address

struct sockaddr\_in server\_address;

server\_address.sin\_family = AF\_INET;

server\_address.sin\_addr.s\_addr = INADDR\_ANY;

server\_address.sin\_port = htons(9003);

int connection\_staus = connect(client\_socket,(struct sockaddr \*) &server\_address, sizeof(server\_address));

if(connection\_staus <0){

printf("Can't connect to sever\n");

} else {

printf("connection established\n");

}

int data;

char msg[1000];

int server\_reply;

int tp=5, tt=1, n=10, loss=2;

printf("Enter TP: ");

scanf("%d", &tp);

printf("Enter TT: ");

scanf("%d", &tt);

printf("Enter Number packets to transmit: ");

scanf("%d", &n);

printf("Enter packet to be lost: ");

scanf("%d", &loss);

int i,j;

int a = tp/tt;

int max\_packet = 1+(2\*a);

int no\_of\_bits = power(2, logbase(max\_packet,2));

int ws = min(max\_packet, no\_of\_bits);

printf("Window Size: %d\n\n", ws);

int window[100], outstanding[100];

int packets[n];

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

packets[i] = i;

}

if(send(client\_socket, &tp ,sizeof(tp),0) < 0){

printf("Couldn't send tp");

exit(0);

// break;

}

if(send(client\_socket, &tt ,sizeof(tt),0) < 0){

printf("Couldn't send tt");

exit(0);

}

if(send(client\_socket, &n ,sizeof(n),0) < 0){

printf("Couldn't send n");

exit(0);

}

if(send(client\_socket, &loss ,sizeof(n),0) < 0){

printf("Couldn't send loss");

exit(0);

}

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

window[i] = packets[i];

}

int nCopy = i;

for(i=0; i<(n-ws); i++){

outstanding[i]=packets[nCopy];

nCopy++;

}

int transmission\_time=0, ack\_time = 0,arrived\_time, counter=0, temp = tt;

int flag=0;

while(1) {

printf("Packet: %d\n",window[0]);

if(counter == 0){

transmission\_time = temp;

}else {

transmission\_time += tt;

}

printf("Transmission Time: %d\n",transmission\_time);

if(send(client\_socket, &window[0] ,sizeof(window[0]),0) < 0){

printf("Couldn't send packet");

break;

}

if(send(client\_socket, &transmission\_time ,sizeof(transmission\_time),0) < 0){

printf("Couldn't send packet");

break;

}

if(recv(client\_socket, &server\_reply ,sizeof(server\_reply),0) < 0){

printf("Couldn't receive server reply");

break;

}

if(server\_reply == 1){

for(i=0; i<ws-1; i++){

window[i] = window[i+1];

}

window[i] = outstanding[0];

int nCopy = i+1;

for(i=0; i<(n-ws)-1; i++){

outstanding[i]=outstanding[i+1];

nCopy++;

}

if(recv(client\_socket, &arrived\_time ,sizeof(arrived\_time),0) < 0){

printf("Couldn't recieve arrived time");

break;

}

ack\_time = arrived\_time + tp;

temp = ack\_time;

printf("Acknowledge Time: %d\n\n",ack\_time);

if(counter == 0)

temp = ack\_time;

counter++;

if(counter == ws)

counter = 0;

} else if(server\_reply == 0) {

if(recv(client\_socket, &arrived\_time ,sizeof(arrived\_time),0) < 0){

printf("Couldn't receive arrived time");

break;

}

ack\_time = arrived\_time + tp;

temp = ack\_time;

printf("Acknowledge Time: %d\n\n",ack\_time);

printf("All packets are transmitted!!\n");

break;

} else if(server\_reply == -1){

if(recv(client\_socket, &arrived\_time ,sizeof(arrived\_time),0) < 0){

printf("Couldn't receive arrived time");

break;

}

ack\_time = arrived\_time + (2\*tp);

temp = ack\_time;

transmission\_time = ack\_time;

printf("Waited Time: %d\n\n",ack\_time);

}

}

close(client\_socket);

return 0;

}

**Output :**

