**EXPERIMENT 8**

**NAME:**Khan Mohd Aaquib Sameer Sameer

**ROLL NO:** 22CO21

**SUBJECT:** Computer Network (CN)

**YEAR:** TE CO

### **AIM:**

To perform stop and wait using programming

### **THEORY:**

The **Stop and Wait** protocol ensures reliable data transmission by sending one packet at a time. The sender waits for an acknowledgment (ACK) from the receiver before sending the next packet. If no ACK is received, the sender retransmits the packet, ensuring no data is lost.

### **CODE (Reciever):**

import java.io.\*;

import java.net.\*;

public class Reciever{

ServerSocket reciever;

Socket connection=null;

ObjectOutputStream out;

ObjectInputStream in;

String packet,ack,data="";

int i=0,sequence=0;

Reciever(){}

public void run(){

try{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

reciever = new ServerSocket(2004,10);

System.out.println("waiting for connection...");

connection=reciever.accept();

sequence=0;

System.out.println("Connection established :");

out=new ObjectOutputStream(connection.getOutputStream());

out.flush();

in=new ObjectInputStream(connection.getInputStream());

out.writeObject("connected .");

do{

try{

packet=(String)in.readObject();

if(Integer.valueOf(packet.substring(0,1))==sequence){

data+=packet.substring(1);

sequence=(sequence==0)?1:0;

System.out.println("\n\nreceiver>"+packet);

}

else

{

System.out.println("\n\nreceiver>"+packet +" duplicate data");

}

if(i<3){

out.writeObject(String.valueOf(sequence));i++;

}

else{

out.writeObject(String.valueOf((sequence+1)%2));

i=0;

}

}

catch(Exception e){}

}while(!packet.equals("end"));

System.out.println("Data recived="+data);

out.writeObject("connection ended .");

}

catch(Exception e){}

finally{

try{

in.close();

out.close();

reciever.close();

}

catch(Exception e){}

}

}

public static void main(String args[]){

Reciever s=new Reciever();

while(true){

s.run();

}

}

}

### 

### 

### **CODE (Sender):**

import java.io.\*;

import java.net.\*;

public class Sender{

Socket sender;

ObjectOutputStream out;

ObjectInputStream in;

String packet,ack,str, msg;

int n,i=0,sequence=0;

Sender(){}

public void run(){

try{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Waiting for Connection....");

sender = new Socket("localhost",2004);

sequence=0;

out=new ObjectOutputStream(sender.getOutputStream());

out.flush();

in=new ObjectInputStream(sender.getInputStream());

str=(String)in.readObject();

System.out.println("reciver > "+str);

System.out.println("Enter the data to send....");

packet=br.readLine();

n=packet.length();

do{

try{

if(i<n){

msg=String.valueOf(sequence);

msg=msg.concat(packet.substring(i,i+1));

}

else if(i==n){

msg="end";out.writeObject(msg);break;

}

out.writeObject(msg);

sequence=(sequence==0)?1:0;

out.flush();

System.out.println("data sent>"+msg);

ack=(String)in.readObject();

System.out.println("waiting for ack.....\n\n");

if(ack.equals(String.valueOf(sequence))){

i++;

System.out.println("receiver > "+" packet recieved\n\n");

}

else{

System.out.println("Time out resending data....\n\n");

sequence=(sequence==0)?1:0;

}

}catch(Exception e){}

}while(i<n+1);

System.out.println("All data sent. exiting.");

}catch(Exception e){}

finally{

try{

in.close();

out.close();

sender.close();

}

catch(Exception e){}

}

}

public static void main(String args[]){

Sender s=new Sender();

s.run();

}

}

### **Output (Sender):**

PS C:\Users\moidin\Desktop\Termwork\CN>

waiting for connection...

Connection established :

receiver>0h

receiver>1i

Data recived=hi

waiting for connection…

### **Output (Reciever):**

PS C:\Users\moidin\Desktop\Termwork\CN>

Connection....

reciver > connected .

Enter the data to send....

hi

data sent>0h

waiting for ack.....

receiver > packet recieved

data sent>1i

waiting for ack.....

receiver > packet recieved

All data sent. exiting.

### **CONCLUSION:**

The program successfully demonstrates the Stop and Wait protocol. The sender sends data, and the receiver acknowledges it. If the acknowledgment is lost, the sender retransmits, ensuring reliable data transfer.