|  |  |
| --- | --- |
| **Ex. No. 3** | **IMPLEMENT FLOW CONTROL MECHANISMS IN DATA LINK CONTROL** |
|  |

### AIM

To study and implement the “STOP & WAIT” protocol.

### PRINCIPLE:

* Protocols in which the sender sends a frame and then waits for an acknowledgement before proceeding are called “STOP & WAIT” protocol.
* The data traffic is simple.
* Frames will travel in both the direction.
* The sender in this protocol simply receives a packet from the network layer copies it into a frame, and then transmit it.
* After transmission, the sender will go to busy waits state until an acknowledgement is received from the receiver.
* The receiver simply waits in a busy state until a frame is received.
* Once a frame is received it passes the data packet to the network layer and sends an acknowledgement for the frame it just received.
* It then loops back to busy waiting and the process continues until the End of File is reached.
* In this protocol, there can be only one outgoing frame at a time so no sequence numbers are required.
* The acknowledgement sent by the receiver to the sender is nothing more than an empty frame.
* Another frame will not be sent until this acknowledgement is received.

### ALGORITHM:

**SERVER SIDE**

1. Initialize server socket
2. Display waiting for connection
3. Initialize the socket and accept the client message
4. Display connected with client
5. Initialize i/p stream
6. Initialize o/p stream
7. Display the message received from client
8. Check the condition
9. Display the message acknowledgement sent to client from client
10. Close all objects
11. Stop

### CLIENT SIDE

1. Open socket with input address ,port
2. Display the message server connected
3. Initialize o/p stream
4. Initialize i/p stream
5. Create sub frame
6. Write message
7. Display the message frame sent to server
8. Check the condition
9. Display the message acknowledgement received from server
10. Close all objects
11. Stop

### STOP AND WAIT PROGRAM SERVER

import java.io.\*; import java.net.\*; public class snws

{

public static void main(String args[])

{

try

{

System.out.println("============== SERVER =============");

String frame = null; String ack = null;

//1. creating a server socket

ServerSocket ss = new ServerSocket(123);

//2. Wait for connection System.out.println("Waiting for connection"); Socket con = ss.accept();

System.out.println("Connected with client - IP : " + con.getInetAddress().getHostAddress());

//3. set Input and output streams

ObjectInputStream in = new ObjectInputStream(con.getInputStream()); ObjectOutputStream out = new ObjectOutputStream(con.getOutputStream());

//4. receive frame length to control for loop

String framelength= (String)in.readObject();

//5. frame receiving and acknowledgment sending process int ackno = 0;

for(int i=0;i<Integer.parseInt(framelength);i++)

{

frame = (String)in.readObject();

System.out.println("Frame Received from Client " + frame);

// swap acknowledge number if(ackno == 0)

ackno = 1; else

ackno = 0;

// compose acknowledge message ack = "ack" + ackno;

// send acknowledgment to client out.writeObject(ack);

System.out.println("Acknowlegement Sent to Client : " + ack);

}

in.close();

out.close();

ss.close();

}

catch(Exception e)

{

System.out.println("Error:" + e);

}

}

}

### STOP AND WAIT PROGRAM CLIENT

import java.io.\*; import java.net.\*; public class snwc

{

public static void main(String args[])

{

try

{

System.out.println("============== CLIENT ==============");

String frame = null; String ack = null;

//1. creating a socket to connect to the server Socket con = new Socket("localhost",123);

System.out.println("Connected with server - IP: "+con.getInetAddress().getHostAddress());

//2. set Output and input streams

ObjectOutputStream out = new ObjectOutputStream(con.getOutputStream()); ObjectInputStream in = new ObjectInputStream(con.getInputStream()); frame = "program";

//3. send the frame length to server to control loop operation in server out.writeObject(Integer.toString(frame.length()));

//4. frame sending and acknowledgment receiving process String subframe = null;

int frameno = 0;

for(int i=0; i< frame.length();i++)

{

subframe = frame.substring(i,i+1); out.writeObject("frame" + frameno + " : "+ subframe );

System.out.println("frame" + frameno + " Sent to Server : " + subframe); if(frameno == 0)

frameno = 1; else

frameno = 0;

ack = (String)in.readObject();

System.out.println("Ack received from Server : " + ack);

}

//5. Close all objects in.close();

out.close();

con.close();

}

catch(Exception e)

{

System.out.println("socket error:"+e);

}

}

}

### OUTPUT:

**CLIENT:**

============================= CLIENT ============================

Connected with server - IP: 127.0.0.1 frame0 Sent to Server : p

Ack received from Server : ack1 frame1 Sent to Server : r

Ack received from Server : ack0 frame0 Sent to Server : o

Ack received from Server : ack1 frame1 Sent to Server : g

Ack received from Server : ack0 frame0 Sent to Server : r

Ack received from Server : ack1 frame1 Sent to Server : a

Ack received from Server : ack0 frame0 Sent to Server : m

Ack received from Server : ack1

### SERVER:

============================ SERVER =============================

Waiting for connection

Connected with client - IP : 127.0.0.1 Frame Received from Client frame0 : p Acknowlegement Sent to Client : ack1 Frame Received from Client frame1 : r Acknowlegement Sent to Client : ack0 Frame Received from Client frame0 : o Acknowlegement Sent to Client : ack1 Frame Received from Client frame1 : g Acknowlegement Sent to Client : ack0 Frame Received from Client frame0 : r Acknowlegement Sent to Client : ack1 Frame Received from Client frame1 : a Acknowlegement Sent to Client : ack0 Frame Received from Client frame0 : m Acknowlegement Sent to Client : ack1

### RESULT

Thus the “STOP AND WAIT” protocol programmed using java was implemented successfully.