|  |  |
| --- | --- |
| **EX.NO:1** | **Implementationof Stop and Wait Protocol and SlidingWindowProtocol** |
| **DATE :** |

**PROGRAM :(slidsender)**

import java.net.\*;

import java.io.\*;

importjava.rmi.\*;

public class slidsender

{public static void main(String a[])throws Exception{

ServerSocketser=new ServerSocket(10);

Socket s=ser.accept();

DataInputStream in=new DataInputStream(System.in);

DataInputStream in1=new DataInputStream(s.getInputStream());

String sbuff[]=new String[8];

PrintStream p;

intsptr=0,sws=8,nf,ano,i;

String ch;

do{

p=new PrintStream(s.getOutputStream());

System.out.print("Enter the no. of frames : ");

nf=Integer.parseInt(in.readLine());

p.println(nf);

if(nf<=sws-1){

System.out.println("Enter "+nf+" Messages to be send\n");

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

sbuff[sptr]=in.readLine();

p.println(sbuff[sptr]);

sptr=++sptr%8;

}

sws-=nf;

System.out.print("Acknowledgment received");

ano=Integer.parseInt(in1.readLine());

System.out.println(" for "+ano+" frames");

sws+=nf;

}else{

System.out.println("The no. of frames exceeds window size");

break;

}System.out.print("\nDo you wants to send some more frames : ");

ch=in.readLine();

p.println(ch);

} while(ch.equals("yes"));

s.close();}}

**PROGRAM : (slidreceiver)**

**i**mport java.net.\*;

import java.io.\*;

classslidreceiver

{

public static void main(String a[])throws Exception

{

Socket s=new Socket(InetAddress.getLocalHost(),10);

DataInputStream in=new

DataInputStream(s.getInputStream());

PrintStream p=new

PrintStream(s.getOutputStream());

int i=0,rptr=-1,nf,rws=8;

String rbuf[]=new String[8];

String ch;

System.out.println();

do

{nf=Integer.parseInt(in.readLine());

if(nf<=rws-1)

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

{ rptr=++rptr%8;

rbuf[rptr]=in.readLine();

System.out.println("The received Frame " +rptr+" is : "+rbuf[rptr]);

}rws-=nf;

System.out.println("\nAcknowledgment sent\n");

p.println(rptr+1);

rws+=nf;

}else break;

ch=in.readLine();

}while(ch.equals("yes"));

}}

**OUTPUT:**

**//SENDER OUTPUT**

Enter theno. offrames :4

Enter 4 Messages to besend

hiii

how ru

i am fine

how is evryone

Acknowledgment received for4 frames

Doyou wants to send some moreframes :

**//RECEIVER OUTPUT**

The receivedFrame0 is :hiii

The receivedFrame1 is :how ru

The receivedFrame2 is : iam fine

The receivedFrame3 is :how iseveryone

**Result:**

|  |  |
| --- | --- |
| **EX.NO:2** | **Study of Socket Programming and Client–Server model** |
| **Date:** |

**SOCKET PROGRAMMING**

## Socket-Definition:

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent.

## TCP/IP and UDP/IP communications:

There are two communication protocols that one can use for socket programming: datagram communication and stream communication.

**Socket layer:**

|  |
| --- |
| Application Layer |
| Sockets |
| Transport Layer |
| Network Layer |
| Data-Link Layer |
| Physical Layer |

Figure 2 shows the TCP/IP protocol stack, and shows where the “Socket layer” may be placed. Again, please be advised that this is just a representation to indicate the level at which we operate when we write network programs using sockets. As shown in the figure, sockets make use of the lower level network protocols, and provide the application developer with an interface to the lower level network protocols. A library of system calls are provided by the socket layer, and are termed as the “Socket API”. These system calls can be used in writing socket programs. In the sections that ensue, we will study those system calls in detail.

Figure 2. TCP/IP protocol stack

## Datagram communication:

The datagram communication protocol, known as UDP (user datagram protocol), is a connectionless protocol, meaning that each time you send datagrams, you also need to send the local socket descriptor and the receiving socket's address. As you can tell, additional data must be sent each time a communication is made.

**CLIENT-SERVER MODEL**

**TCP/IP and UDP/IP communications:**

There are two communication protocols that one can use for socket programming: datagram communication and stream communication.

* TCP/IP allows two application programs to pass data back and forth
  + The applications can execute on the same machine or on different machines
* How to organize the application programs?
  + Client-server paradigm is the most popular model
    - The communication applications are divided into two categories: client and server
* Client-server paradigm uses the direction of initiation to categorize whether a program is a client or server.
* An application that initiates the communication is called a client
* E.g., Web browser, FTP client, Telnet client, Email client
* The client contacts a server, sends a request, and awaits a response
* An application that waits for incoming communication requests from clients is called a server
* E.g., Web server, FTP server, Telnet server, SMTP server
* The server receives a client’s request, performs the necessary computation, and returns the result to the client
* **Remark**: A server is usually designed to provide service to multiple clients

