Sliding Window Protocol:

```
import java.util.Random;
import java.util.Scanner;
class SharedObject {
    int N;
    volatile int receive=1;
    volatile int flag=0;
    volatile int send_start=0;
    volatile int send_end;
    public SharedObject(int N){
        this.N=N;
        this.send_end=N-1;

    }

    synchronized public void increment_sender(){
        send_start++;
        send_end++;
        return ;
    }
    synchronized public void increment_receive(){
        receive++;
        return;
    }
}



class Sender implements Runnable {

    int N;
    SharedObject so;

    public Sender(int N, SharedObject so) {
        this.N = N;
        this.so = so;
    }

    public void run() {
        while(true) {
            synchronized (so) {
                if (so.flag == 0) {
                    so.increment_sender();
                    System.out.println("Sent data packets from " +
so.send_start+" to "+ so.send_end);
                } else {
                    System.out.println("Resending data packets from " +
so.send_start+" to "+ so.send_end);
                }
                so.flag = 1;
            }
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
```

```
                    System.out.println("Received Acknowledment before time
out ");

            }
        }
    }
}
class Receiver implements Runnable{
    SharedObject so;
    Thread st;
    public Receiver(SharedObject so,Thread st){
        this.so=so;
        this.st=st;
    }
    public void run(){
        while(true) {
            Random r = new Random();
            int num = r.nextInt(2);
            if (num == 0) {
                continue;
            } else {
                int arr[] = {0, 1000};
                int ind = r.nextInt(2);
                try {
                    Thread.sleep(arr[ind]);
                } catch (InterruptedException e) {
                    System.out.println("Receiver thread is interrupted ");
                }
                synchronized(so) {
                    if(so.flag==1) {
                        System.out.println("Received acknowledgement for " +
so.receive);

                        so.flag = 0;
                        so.increment_receive();
                        st.interrupt();

                    }

                }
                try{
                    Thread.sleep(100);
                }
                catch (InterruptedException e){

                }
            }
        }
    }
}
public class SlidingWindowProtocol{
    public static void main(String[] args) {

        Scanner sc= new Scanner(System.in);
        System.out.println("Enter window size ");
        int N=sc.nextInt();
```

```
        SharedObject so= new SharedObject(N);
        Thread sender= new Thread(new Sender(N,so));
        Thread receiver= new Thread(new Receiver(so,sender));
        sender.start();
        receiver.start();
        try{
            sender.join();
            receiver.join();
        }
        catch(InterruptedException e){
            System.out.println("Main thread ends before sender and receiver
threads join ");
        }


    }
}
```

```
Enter window size
5
Sent data packets from 1 to 5
Received acknowledgement for 1
Received Acknowledment before time out
Sent data packets from 2 to 6
Received acknowledgement for 2
Received Acknowledment before time out
Sent data packets from 3 to 7
Resending data packets from 3 to 7
Received acknowledgement for 3
Received Acknowledment before time out
Sent data packets from 4 to 8
Resending data packets from 4 to 8
Received acknowledgement for 4
Received Acknowledment before time out
Sent data packets from 5 to 9
Resending data packets from 5 to 9
Received acknowledgement for 5
Received Acknowledment before time out
Sent data packets from 6 to 10
Resending data packets from 6 to 10
Received acknowledgement for 6
Received Acknowledment before time out
Sent data packets from 7 to 11
Received acknowledgement for 7
Received Acknowledment before time out
Sent data packets from 8 to 12
Received acknowledgement for 8
Received Acknowledment before time out
Sent data packets from 9 to 13
Received acknowledgement for 9
Received Acknowledment before time out
Sent data packets from 10 to 14
```

Socket Programming:

Server Code:

```
import java.io.BufferedReader;
import java.net.ServerSocket;
import java.io.InputStreamReader;
import java.net.Socket;

public class ServSocket {
    public static void main(String[] args) throws Exception {
        System.out.println("Server is started ");
        ServerSocket ss= new ServerSocket(9999);
        System.out.println("server is waiting for client request");
        Socket s= ss.accept();
        System.out.println("Client Connected");
        BufferedReader br= new BufferedReader(new
InputStreamReader(s.getInputStream()));
        String line;
        while ((line = br.readLine()) != null) {
            System.out.println("Client Data: " + line);
        }
        s.close();
        ss.close();

    }
}
```

Client Code:

```
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;

public class ClientSocket {
    public static void main(String[] args) throws Exception {
        String ip="localhost";
        int port=9999;
        Socket s=new Socket(ip,port);
        String str="Request from client";

        OutputStreamWriter os= new OutputStreamWriter(s.getOutputStream());
        PrintWriter out= new PrintWriter(os);
        os.write(str);
        os.flush();
        s.close();
```

```
    }
}
```

```
javac ClientSocket.java

java ClientSocket
```

```
javac ServSocket.java

java ServSocket

Server is started
server is waiting for client request
Client Connected
Client Data: Request from client
```

Client Server using TCP/IP:

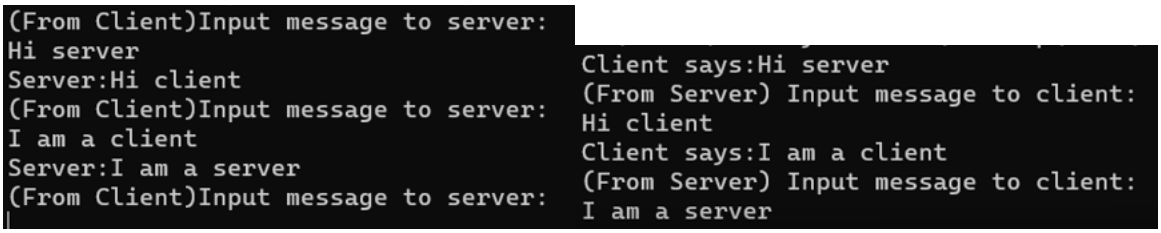
Server Code:

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
public class server_chat {

    public static void main(String[] args) throws IOException {

        ServerSocket serverSocket = new ServerSocket(8000);

        Socket socket = serverSocket.accept();

        DataInputStream inputFromClient = new
DataInputStream(socket.getInputStream());
        DataOutputStream outputToClient = new
```

```
DataOutputStream(socket.getOutputStream());
        Scanner sc = new Scanner(System.in);
        String msg;
        while (true) {
            msg = inputFromClient.readUTF();
            System.out.println("Client says:" + msg);
            System.out.println("(From Server) Input message to client:");
            msg = sc.nextLine();
            outputToClient.writeUTF(msg);

        }

    }

}
```

Client code

```
import java.io.*;
import java.net.Socket;
import java.util.Scanner;

public class client_chat {

    public static void main(String[] args) throws IOException {

        Socket socket = new Socket("localhost", 8000);
        DataInputStream fromServer = new
DataInputStream(socket.getInputStream());
        DataOutputStream toServer = new
DataOutputStream(socket.getOutputStream());
        Scanner sc = new Scanner(System.in);
        String msg;
        while (true) {
            System.out.println("(From Client)Input message to server:");
            msg = sc.nextLine();
            toServer.writeUTF(msg);
            msg = fromServer.readUTF();
            System.out.println("Server:" + msg);

        }

    }

}
```


(From Client)Input message to server:
Hi server
Server:Hi client
(From Client)Input message to server:
I am a client
Server:I am a server
(From Client)Input message to server:
Client says:Hi server
(From Server) Input message to client:
Hi client
Client says:I am a client
(From Server) Input message to client:
I am a server