RMI

After testing our RMI code with two computers located in different parts of London, we concluded that RMI communication mechanism is very reliable.

From our observations, we found that the RMI communication mechanism has minimal instances of messages being lost. This could be because a connection has already been established between the client and server. This means that the server and client are already bound before messages are sent over from client to server.

During our testing with RMI server, we observed that not a single message has been lost during transmission when we despite the computers being in different parts of London.

UDP

UDP is unreliable as it does not utilize TCP. Packets are sent over the network without first establishing a connection between the client and the server. This results in some loss of packets and the order of receipt being jumbled up.

Causes of lost messages

RMI: Messages can be lost when the network is congested and the buffer becomes full. Subsequent packets are denied entry into the buffer and are not received properly. However, RMI implements TCP congestion control to deal with this situation. This explains why messages were hardly lost even when sending a large amount of messages. However, this also took a toll on the performance as the messages were sent much more slowly due to client-server communication for each packet.

UDP: Messages also can be lost when the network becomes congested. However, there is no in-built congestion control system so messages are more easily lost. Furthermore, the lack of client-server communication results in faster sending rate of packets, causing the network to become congested more quickly when the number of messages is sufficiently large.

Comparison:

In our opinion, we believe that UDP was easier to program. To us, RMI was more difficult because we had to take into consideration other factors like the security manager as well as client and serving binding. On the other hand, UDP did not require such methods.

That being said, the simplicity of UDP meant that it was not as reliable in sending information from the client and receiving information from the server. As the server and the client are not bound, there is a much higher chance of messages not being sent from client to server.

RMI Client

/\*

 \* Created on 01-Mar-2016

 \*/

package rmi;

import java.rmi.Naming;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.rmi.RMISecurityManager;

import java.rmi.registry.\*;

import common.MessageInfo;

public class RMIClient {

    public static *void* main(*String*[] *args*) {

*RMIServerI* iRMIServer = null;

        // Check arguments for Server host and number of messages

        if (args.length < 2){

            System.out.println("Needs 2 arguments: ServerHostName/IPAddress, TotalMessageCount");

            System.exit(-1);

        }

*String* urlServer = new String("rmi://" + args[0] + "/RMIServer");

*int* numMessages = Integer.parseInt(args[1]);

        try{

            if(System.getSecurityManager() == null){

                System.setSecurityManager (new RMISecurityManager ());

            }

        // TO-DO: Initialise Security Manager

*RMIServerI* remobj = (RMIServerI)Naming.lookup(urlServer);

        // TO-DO: Bind to RMIServer

            for(*int* i=0; i<numMessages; i++){

*MessageInfo* msg = new MessageInfo(numMessages, i);

                System.out.println("Message Sent: " + msg.toString());

                remobj.receiveMessage(msg);

            }

        // TO-DO: Attempt to send messages the specified number of times

        }

        catch(*Exception* *e*){

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

        }

    }

}

RMI Server

/\*

 \* Created on 01-Mar-2016

 \*/

package rmi;

import java.rmi.Naming;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import java.rmi.NotBoundException;

import java.rmi.RMISecurityManager;

import java.rmi.registry.\*;

import java.net.MalformedURLException;

import java.util.Arrays;

import common.\*;

public class RMIServer extends *UnicastRemoteObject* implements *RMIServerI* {

    private *int* totalMessages = -1;

    private *int*[] receivedMessages;

    public RMIServer() throws *RemoteException* {

        super();

    }

    public *void* receiveMessage(*MessageInfo* *msg*) throws *RemoteException* {

        if(totalMessages == -1){

            receivedMessages = new *int*[msg.totalMessages];

            totalMessages = msg.totalMessages;

            System.out.println("First Message Received! Message Content: " + msg.toString());

        }

        else{

            System.out.println("Message Received! Message Content: " + msg.toString());

        }

        // TO-DO: On receipt of first message, initialise the receive buffer

        receivedMessages[msg.messageNum] = msg.messageNum + 1;

        // TO-DO: Log receipt of the message

        if(msg.messageNum == totalMessages - 1){

            System.out.println("Last Message Received!");

            totalMessages = -1;

*int* missingmsg = 0;

            for(*int* i=0; i<msg.totalMessages; i++){

                if(receivedMessages[i] == 0){

                    missingmsg = missingmsg + 1;

                }

            }

            System.out.println("Number of Missing Messages: " + missingmsg);

        }

        // TO-DO: If this is the last expected message, then identify

        //        any missing messages

    }

    public static *void* main(*String*[] *args*) {

// TO-DO: Initialise Security Manager

*RMIServer* rmis = null;

        if(System.getSecurityManager() == null){

            System.setSecurityManager (new RMISecurityManager ());

        }

        try{

// TO-DO: Instantiate the server class

*RMIServer* s = new RMIServer();

            // TO-DO: Bind to RMI registry

            rebindServer("rmi://localhost/RMIServer", s);

        }

        catch(*Exception* *e*){

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

        }

    }

    protected static *void* rebindServer(*String* *serverURL*, *RMIServer* *server*) {

        try{

            Naming.rebind(serverURL, server);

        }

        catch(*Exception* *e*){

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

        }

        // TO-DO:

        // Start / find the registry (hint use LocateRegistry.createRegistry(...)

        // If we \*know\* the registry is running we could skip this (eg run rmiregistry in the start script)

        // TO-DO:

        // Now rebind the server to the registry (rebind replaces any existing servers bound to the serverURL)

        // Note - Registry.rebind (as returned by createRegistry / getRegistry) does something similar but

        // expects different things from the URL field.

    }

}