

For each mechanism, what are possible causes, if any, of messages being lost?

UDP offers no guarantee of delivery, ordering, or duplicate protection. IP datagram packets may be dropped because of network congestion or error. Congestion can occur when too many messages are sent or when too many users are on the same network. Errors may occur when distances increase as there may be more servers in between, increasing the probability of hardware or software failure. If on a wireless networks, external interference can corrupt the data. Packets can also be dropped when their checksum fail (corrupted).

Our UDP implementation also has a fixed buffer and message size. When incoming transmissions exceed these allocations, the buffer overflows, and begins skipping packets (discussed further in the next section). - no mention to RMI working over TCP

RMI is very reliable. However, the target server can still malfunction (terminating early/breaking the connection). Programming errors can occur during interface implementation or parameter handling. There can also be protocol disagreements between client and server. (Though these errors are not exclusive to RMI).

Are there any patterns in the way messages are lost?

Our RMI server implementation did not lose any of the 1000 messages (RMI 1) or even the 10,000 messages we sent from the client (RMI 2). This indicates a high level of reliability. Nevertheless, we did notice a significant drop in performance (higher latency) which was not present in UDP after the total message count exceeded 2000.

Our UDP server seemed to handle requests below 200 messages well (UDP 1). However, when we tested with 500 messages, missing packets began to surface at 260~ at an interval between 1 and 4, but most frequently between 1 and 2. (UDP 2). Multiple tests indicate that the start of and the number of missing messages vary randomly. Curious, we decided to tweak with the pack and buffer sizes through `(packData=new byte[], recvSoc.setReceiveBufferSize())`, and all messages between 5 and 99 went missing! (UDP 3) This indicates that small buffer or large message sizes can cause significant packet drops until the buffer is flushed, reset after some time, or when the server is rebooted.

What is the relative reliability of the different communication mechanisms?

RMI is significantly more reliable than UDP for many reasons. The request-reply system, duplicate filtering, retransmission of results, and interface abstraction/checking significantly improves the reliability of RMI. On the other hand, beyond the reasons discussed in question one, we were able to send UDP messages without the UDP server even being up. Would that then be considered a 100% drop rate?

Which was easier to program and why?

RMI was easier to program for us mainly due to its high-level abstraction and Java programming syntax. The remote interface abstracts away many tasks like marshalling, garbage collection, registry, and security. The allowance of object reflection and method invocation, makes it easy to manipulate! UDP was longer and required a new understanding of datagram and sockets, which was educational, but more complex.

Figure RMI 1

Terminal 1 (Left)	Terminal 2 (Right)
Received 1000;990	909 sent 910 sent 911 sent 912 sent
Received 1000;991	913 sent 914 sent 915 sent 916 sent
Received 1000;992	917 sent 918 sent 919 sent 920 sent
Received 1000;993	921 sent 922 sent 923 sent 924 sent
Received 1000;994	925 sent 926 sent 927 sent 928 sent
Received 1000;995	929 sent 930 sent 931 sent 932 sent
Received 1000;996	933 sent 934 sent 935 sent 936 sent
Received 1000;997	937 sent 938 sent 939 sent 940 sent
Received 1000;998	941 sent 942 sent 943 sent 944 sent
Received 1000;999	945 sent 946 sent 947 sent 948 sent
Received 1000;1000	949 sent 950 sent 951 sent 952 sent
No message is missing	953 sent 954 sent 955 sent 956 sent
	957 sent 958 sent 959 sent 960 sent
	961 sent 962 sent 963 sent 964 sent
	965 sent 966 sent 967 sent 968 sent
	969 sent 970 sent 971 sent 972 sent
	973 sent 974 sent 975 sent 976 sent
	977 sent 978 sent 979 sent 980 sent
	981 sent 982 sent 983 sent 984 sent
	985 sent 986 sent 987 sent 988 sent
	989 sent 990 sent 991 sent 992 sent
	993 sent 994 sent 995 sent 996 sent
	997 sent 998 sent 999 sent 1000 sent
	as5017@arc10:RMI_UDP\$

Figure RMI 2

Terminal 1 (Left)	Terminal 2 (Right)
Received 10000;9989	sent 9930 sent 9931 sent 9932
Received 10000;9990	sent 9933 sent 9934 sent 9935 s
Received 10000;9991	ent 9936 sent 9937 sent 9938 se
Received 10000;9992	t 9939 sent 9940 sent 9941 sen
Received 10000;9993	t 9942 sent 9943 sent 9944 sent
Received 10000;9994	9945 sent 9946 sent 9947 sent
Received 10000;9995	9948 sent 9949 sent 9950 sent 9
Received 10000;9996	951 sent 9952 sent 9953 sent 99
Received 10000;9997	54 sent 9955 sent 9956 sent 995
Received 10000;9998	7 sent 9958 sent 9959 sent 9960
Received 10000;9999	sent 9961 sent 9962 sent 9963
Received 10000;10000	sent 9964 sent 9965 sent 9966 s
	ent 9967 sent 9968 sent 9969 se
	nt 9970 sent 9971 sent 9972 sen
	t 9973 sent 9974 sent 9975 sent
	9976 sent 9977 sent 9978 sent
	9979 sent 9980 sent 9981 sent 9
	982 sent 9983 sent 9984 sent 99
	85 sent 9986 sent 9987 sent 998
	8 sent 9989 sent 9990 sent 9991
	sent 9992 sent 9993 sent 9994
	sent 9995 sent 9996 sent 9997 s
	ent 9998 sent 9999 sent 10000 s
	ent as5017@arc10:RMI_UDP\$

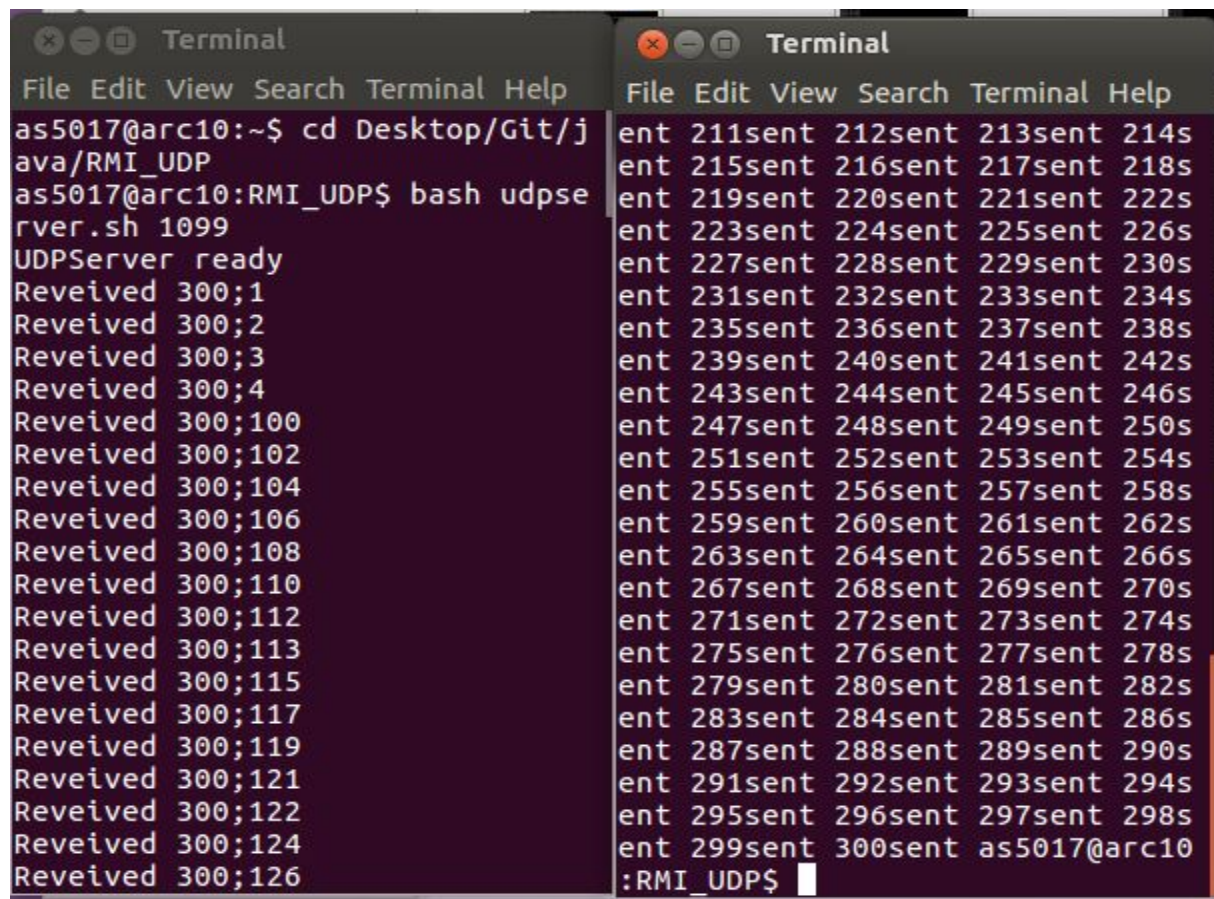
Figure UDP 1

Terminal 1 (Left)	Terminal 2 (Right)
Reveived 200;179	ent 111sent 112sent 113sent 114s
Reveived 200;180	ent 115sent 116sent 117sent 118s
Reveived 200;181	ent 119sent 120sent 121sent 122s
Reveived 200;182	ent 123sent 124sent 125sent 126s
Reveived 200;183	ent 127sent 128sent 129sent 130s
Reveived 200;184	ent 131sent 132sent 133sent 134s
Reveived 200;185	ent 135sent 136sent 137sent 138s
Reveived 200;186	ent 139sent 140sent 141sent 142s
Reveived 200;187	ent 143sent 144sent 145sent 146s
Reveived 200;188	ent 147sent 148sent 149sent 150s
Reveived 200;189	ent 151sent 152sent 153sent 154s
Reveived 200;190	ent 155sent 156sent 157sent 158s
Reveived 200;191	ent 159sent 160sent 161sent 162s
Reveived 200;192	ent 163sent 164sent 165sent 166s
Reveived 200;193	ent 167sent 168sent 169sent 170s
Reveived 200;194	ent 171sent 172sent 173sent 174s
Reveived 200;195	ent 175sent 176sent 177sent 178s
Reveived 200;196	ent 179sent 180sent 181sent 182s
Reveived 200;197	ent 183sent 184sent 185sent 186s
Reveived 200;198	ent 187sent 188sent 189sent 190s
Reveived 200;199	ent 191sent 192sent 193sent 194s
Reveived 200;200	ent 195sent 196sent 197sent 198s
No message is missing	ent 199sent 200sent as5017@arc10
as5017@arc10:RMI_UDP\$:RMI_UDP\$

Figure UDP 2

Terminal 1 (Left)	Terminal 2 (Right)
389missing 391missing 392missing	ent 411sent 412sent 413sent 414s
394missing 396missing 398missing	ent 415sent 416sent 417sent 418s
399missing 401missing 402missing	ent 419sent 420sent 421sent 422s
404missing 406missing 407missing	ent 423sent 424sent 425sent 426s
409missing 411missing 413missing	ent 427sent 428sent 429sent 430s
414missing 416missing 418missing	ent 431sent 432sent 433sent 434s
420missing 421missing 423missing	ent 435sent 436sent 437sent 438s
425missing 427missing 429missing	ent 439sent 440sent 441sent 442s
431missing 432missing 434missing	ent 443sent 444sent 445sent 446s
436missing 438missing 439missing	ent 447sent 448sent 449sent 450s
441missing 443missing 445missing	ent 451sent 452sent 453sent 454s
447missing 448missing 450missing	ent 455sent 456sent 457sent 458s
452missing 453missing 455missing	ent 459sent 460sent 461sent 462s
457missing 458missing 460missing	ent 463sent 464sent 465sent 466s
461missing 463missing 464missing	ent 467sent 468sent 469sent 470s
466missing 468missing 470missing	ent 471sent 472sent 473sent 474s
471missing 473missing 475missing	ent 475sent 476sent 477sent 478s
476missing 478missing 480missing	ent 479sent 480sent 481sent 482s
482missing 483missing 485missing	ent 483sent 484sent 485sent 486s
487missing 489missing 490missing	ent 487sent 488sent 489sent 490s
492missing 494missing 496missing	ent 491sent 492sent 493sent 494s
498missing 499missing	ent 495sent 496sent 497sent 498s
366/500 received	ent 499sent 500sent as5017@line1
as5017@line14:RMI_UDP\$	4:RMI_UDP\$

Figure UDP 3



```
Terminal
File Edit View Search Terminal Help
as5017@arc10:~$ cd Desktop/Git/java/RMI_UDP
as5017@arc10:RMI_UDP$ bash udpserver.sh 1099
UDPServer ready
Received 300;1
Received 300;2
Received 300;3
Received 300;4
Received 300;100
Received 300;102
Received 300;104
Received 300;106
Received 300;108
Received 300;110
Received 300;112
Received 300;113
Received 300;115
Received 300;117
Received 300;119
Received 300;121
Received 300;122
Received 300;124
Received 300;126

Terminal
File Edit View Search Terminal Help
ent 211sent 212sent 213sent 214s
ent 215sent 216sent 217sent 218s
ent 219sent 220sent 221sent 222s
ent 223sent 224sent 225sent 226s
ent 227sent 228sent 229sent 230s
ent 231sent 232sent 233sent 234s
ent 235sent 236sent 237sent 238s
ent 239sent 240sent 241sent 242s
ent 243sent 244sent 245sent 246s
ent 247sent 248sent 249sent 250s
ent 251sent 252sent 253sent 254s
ent 255sent 256sent 257sent 258s
ent 259sent 260sent 261sent 262s
ent 263sent 264sent 265sent 266s
ent 267sent 268sent 269sent 270s
ent 271sent 272sent 273sent 274s
ent 275sent 276sent 277sent 278s
ent 279sent 280sent 281sent 282s
ent 283sent 284sent 285sent 286s
ent 287sent 288sent 289sent 290s
ent 291sent 292sent 293sent 294s
ent 295sent 296sent 297sent 298s
ent 299sent 300sent as5017@arc10
:RMI_UDP$
```

```
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]);
```

```
        // Initialise Security Manager
```

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

```
            System.setSecurityManager(new SecurityManager());
```

```
        try{
```

```
            //Bind to RMIServer
```

```
            Registry registry = LocateRegistry.getRegistry(args[0]);
```

```
            RMIServerI server = (RMIServerI) registry.lookup("RMIServer");
```

```
            // Attempt to send messages the specified number of times
```

```
            for (int msgNum = 1; msgNum <= numMessages; msgNum++) {
```

```
                System.out.print(msgNum+" sent ");
```

```
                MessageInfo msg = new MessageInfo(numMessages, msgNum);
```

```
                server.receiveMessage(msg);
```

```
            }
```

```
        } catch (Exception e){
```

- should handle different exceptions separately

```
            e.printStackTrace();
```

```
            System.exit(1);
```

```
        }
```

```
    }
```

```
}
```


public class RMIServer **extends** UnicastRemoteObject **implements**

RMIServerI {

private int totalMessages = -1;

private int[] receivedMessages;

private int msg_index = 0;

private boolean[] miss_msg;

public RMIServer() **throws** RemoteException {}

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

// On receipt of first message, initialise the receive buffer

if(totalMessages == -1) {

totalMessages = msg.totalMessages;

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

miss_msg = **new boolean**[totalMessages];

for(**int** index=0; index<totalMessages; index++)

miss_msg[index] = **true**;

}

// Log receipt of the message

System.out.println("**Received** "+msg);

miss_msg[msg.messageNum-1] = **false**;

receivedMessages[msg_index++] = msg.messageNum;

// If this is the last expected message, then identify any missing messages

if (msg.messageNum == totalMessages){

boolean any_miss = **false**;

for (**int** index=0; index<totalMessages; index++)

if (miss_msg[index] == **true**){

any_miss = **true**;

System.out.println("**missing** "+index+1);

}

if(!any_miss)

System.out.println("**No message is missing**");

totalMessages = -1;

msg_index = 0;

```
}  
}
```

```
public static void main(String[] args) {  
    // Initialise Security Manager  
    if(System.getSecurityManager() == null)  
        System.setSecurityManager(new SecurityManager());
```

```
    try{  
        // Instantiate the server class  
        String serverURL = "rmi://localhost/RMIServer";  
        RMIServer rmis = new RMIServer();  
        // Bind to RMI registry  
        rebindServer(serverURL, rmis);  
        System.out.println("RMIServer Bound");  
    } catch ( Exception e){  
        e.printStackTrace();  
        System.exit(1);  
    }  
}
```

```
protected static void rebindServer(String serverURL, RMIServer server) {  
    try {  
        // Start / find the registry  
        LocateRegistry.createRegistry(1099);  
        // Rebind the server to the registry  
        Naming.rebind(serverURL, server);  
    } catch (Exception e){  
        e.printStackTrace();           - should handle different exceptions separately  
        System.exit(1);  
    }  
}
```

```
public class UDPClient{
```

```
    private DatagramSocket sendSoc;
```

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

```
        InetAddress  serverAddr = null;
```

```
        int recvPort;
```

```
        int countTo;
```

```
        String message;
```

```
        // Get the parameters
```

```
        if (args.length < 3) {
```

```
            System.err.println("Arguments required: server name/IP, recv port, message count");
```

```
            System.exit(-1);
```

```
        }
```

```
        try {
```

```
            serverAddr = InetAddress.getByName(args[0]);
```

```
        } catch (UnknownHostException e) {
```

```
            System.out.println("Bad server address in UDPClient, " + args[0] + " caused an unknown host exception " + e);
```

```
            System.exit(-1);
```

```
        }
```

```
        recvPort = Integer.parseInt(args[1]);
```

```
        countTo = Integer.parseInt(args[2]);
```

```
        // Construct UDP client class and try to send messages
```

```
        UDPClient udpClient = new UDPClient();
```

```
        udpClient.testLoop(serverAddr, recvPort, countTo);
```

```
    }
```

```
    // Initialise the UDP socket for sending data
```

```
    public UDPClient() {
```

```
        try {
```

```
            sendSoc = new DatagramSocket();
```

```
        } catch (Exception e) {
```

```
            e.printStackTrace();
```



```

        System.exit(1);
    }
}

// Send the messages to the server
private void testLoop(InetAddress serverAddr, int recvPort, int countTo) {
    int tries = 0;
    while(tries < countTo) {
        send(countTo+";"+(tries+1), serverAddr, recvPort);
        System.out.print((tries+1)+"sent ");
        tries++;
    }
}

// Build the datagram packet and send it to the server
private void send(String payload, InetAddress destAddr, int destPort) {
    byte[] pktData = payload.getBytes();
    int payloadSize = pktData.length;
    DatagramPacket pkt = new DatagramPacket(pktData, payloadSize, destAddr,
destPort);

    try {
        sendSoc.send(pkt);
    } catch (Exception e) {
        e.printStackTrace();
        System.exit(1);
    }
}
}

```

```
public class UDPServer {
```

```
    private DatagramSocket recvSoc;  
    private int totalMessages = -1;  
    private int[] receivedMessages;  
    boolean[] miss_msg;  
    private boolean close;  
    private int msg_index = 0;  
    private int count=0;
```

```
// Use a timeout (e.g. 30 secs) to ensure the program doesn't block forever
```

```
    private void run() {
```

```
        int pacSize;
```

```
        byte[] pacData;
```

```
        pacData = new byte[32];
```

```
        DatagramPacket pac = new DatagramPacket(pacData, pacData.length);
```

```
        close = false;
```

```
// Receive the messages and process them by calling processMessage(...)
```

```
    while(!close) {
```

```
        try {
```

```
            recvSoc.receive(pac);
```

```
            String data = new String(pac.getData(), 0, pac.getLength());
```

```
            processMessage(data);
```

```
        } catch (SocketTimeoutException tm_out_exc) {
```

```
            System.out.println("timed out!");    - not printing received/lost msgs in case of timeout
```

```
        } catch (Exception e){
```

```
            e.printStackTrace();
```

```
            System.exit(1);
```

```
        }
```

```
    }
```

```
}
```

```
public void processMessage(String data) {
```

```
// Use the data to construct a new MessageInfo object
```

```
    MessageInfo msg = null;
```

```
    try{
```

```
        msg = new MessageInfo(data);
```

```

    } catch (Exception e) {
        e.printStackTrace();
        System.exit(1);
    }

    // Log receipt of the message
    if(totalMessages == -1) {
        close = false;
        totalMessages = msg.totalMessages;
        receivedMessages = new int[totalMessages];
        miss_msg = new boolean[totalMessages];
        for(int index = 0; index<totalMessages; index++)
            miss_msg[index] = true;
    }

    // If this is the last expected message, then identify any missing messages
    System.out.print("Received "+msg);
    miss_msg[msg.messageNum-1] = false;
    receivedMessages[msg_index++] = msg.messageNum;
    if (msg.messageNum == totalMessages){
        boolean any_miss = false;
        for (int index=0; index<totalMessages; index++)
            if (miss_msg[index]){
                any_miss = true;
                count+=1;
                System.out.print((index+1)+"missing ");
            }
        System.out.println("\n"+(totalMessages-count)+"/"+totalMessages+" received");
        if(!any_miss)
            System.out.println("No message is missing");
        close = true;
        System.exit(0);
    }
}

public UDPServer(int rp) {
    try {
        // Initialise UDP socket for receiving data
        rcvSoc = new DatagramSocket(rp);
    }
}

```

```

    // Use a timeout (e.g. 30 secs) to ensure the program doesn't block forever
    recvSoc.setSoTimeout(30000);
    // On receipt of first message, initialise the receive buffer
    recvSoc.setReceiveBufferSize(51200);
} catch (Exception e) {
    e.printStackTrace();
    System.exit(1);
}
// Done Initialisation
System.out.println("UDPServer ready");
}

public static void main(String args[]) {
    int    recvPort;

    // Get the parameters from command line
    if (args.length < 1) {
        System.err.println("Arguments required: recv port");
        System.exit(-1);
    }
    recvPort = Integer.parseInt(args[0]);
    // Construct Server object and start it by calling run().
    UDPServer udpServer = new UDPServer(recvPort);
    udpServer.run();
}
}

```