### 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).

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 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 (pacData=**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.

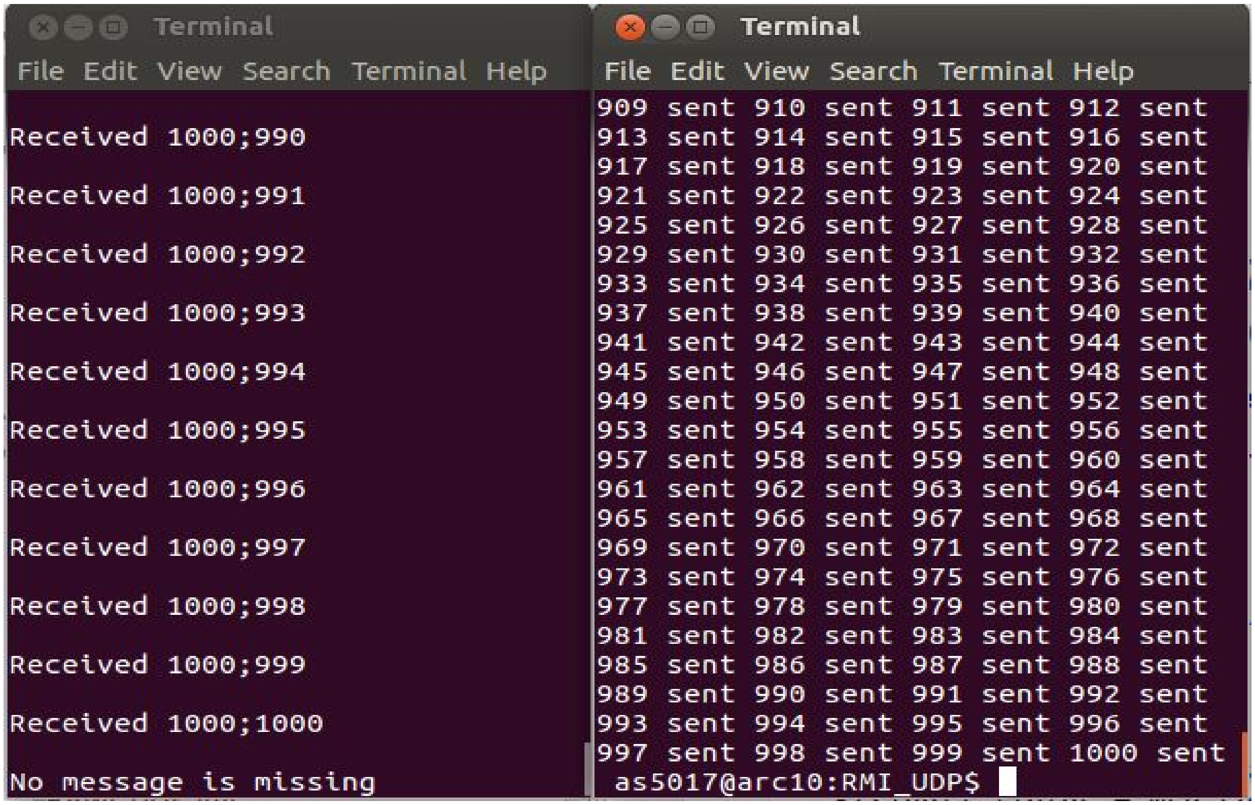
### What is the relative reliability of the different communication mechanisms?

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.

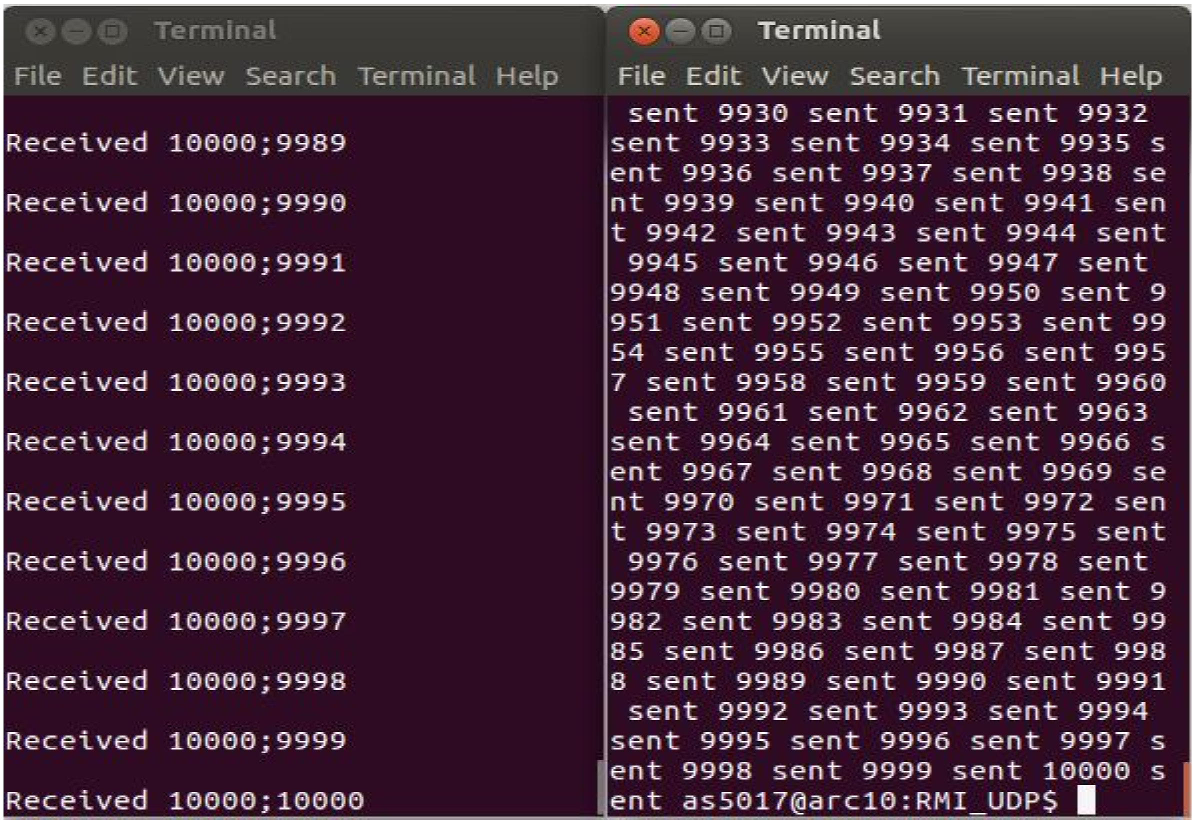
**Figure RMI 1**


Terminal
File Edit View Search Terminal Help
Terminal
File Edit View Search Terminal Help

```

Received 1000;990

Received 1000;991

Received 1000;992

Received 1000;993

Received 1000;994

Received 1000;995

Received 1000;996

Received 1000;997

Received 1000;998

Received 1000;999

Received 1000;1000

No message is missing
```

```
909 sent 910 sent 911 sent 912 sent
913 sent 914 sent 915 sent 916 sent
917 sent 918 sent 919 sent 920 sent
921 sent 922 sent 923 sent 924 sent
925 sent 926 sent 927 sent 928 sent
929 sent 930 sent 931 sent 932 sent
933 sent 934 sent 935 sent 936 sent
937 sent 938 sent 939 sent 940 sent
941 sent 942 sent 943 sent 944 sent
945 sent 946 sent 947 sent 948 sent
949 sent 950 sent 951 sent 952 sent
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$
```

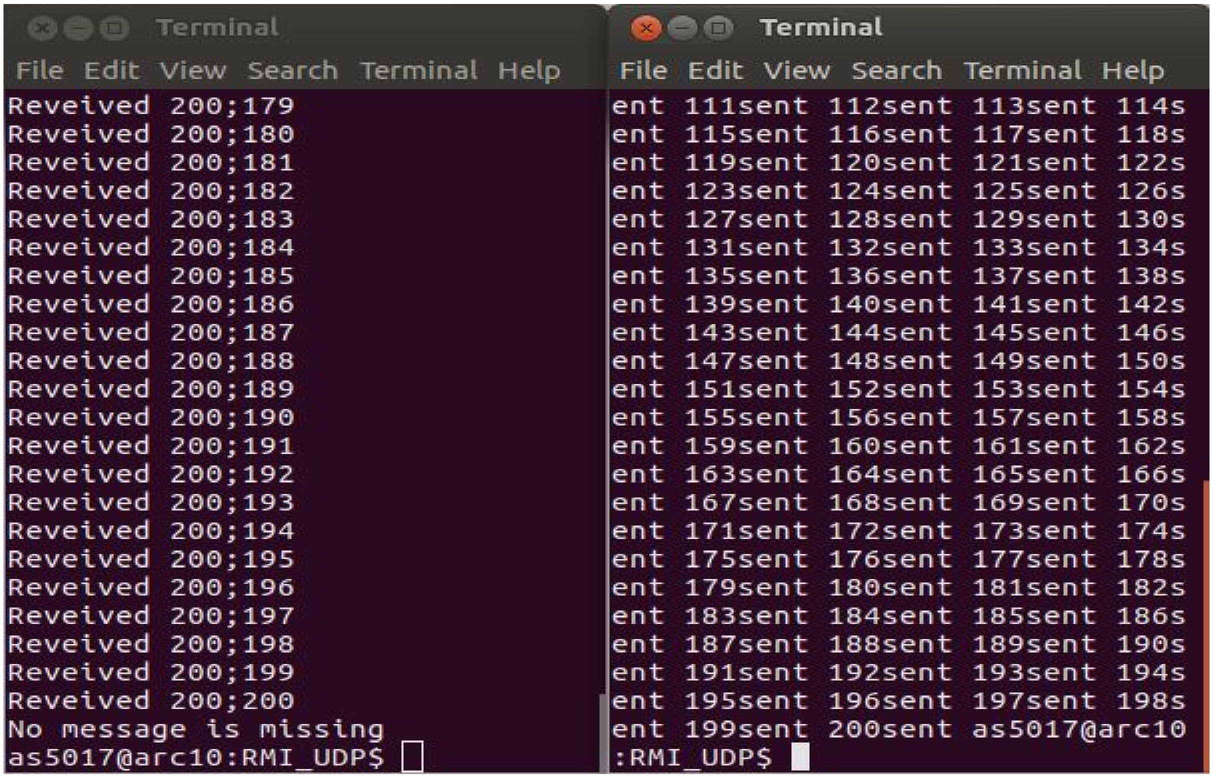
**Figure RMI 2**


Terminal
File Edit View Search Terminal Help
Terminal
File Edit View Search Terminal Help

```

Received 10000;9989

Received 10000;9990

Received 10000;9991

Received 10000;9992

Received 10000;9993

Received 10000;9994

Received 10000;9995

Received 10000;9996

Received 10000;9997

Received 10000;9998

Received 10000;9999

Received 10000;10000
```

```
 sent 9930 sent 9931 sent 9932
sent 9933 sent 9934 sent 9935 s
ent 9936 sent 9937 sent 9938 se
nt 9939 sent 9940 sent 9941 sen
t 9942 sent 9943 sent 9944 sent
 9945 sent 9946 sent 9947 sent
9948 sent 9949 sent 9950 sent 9
951 sent 9952 sent 9953 sent 99
54 sent 9955 sent 9956 sent 995
7 sent 9958 sent 9959 sent 9960
 sent 9961 sent 9962 sent 9963
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$
```

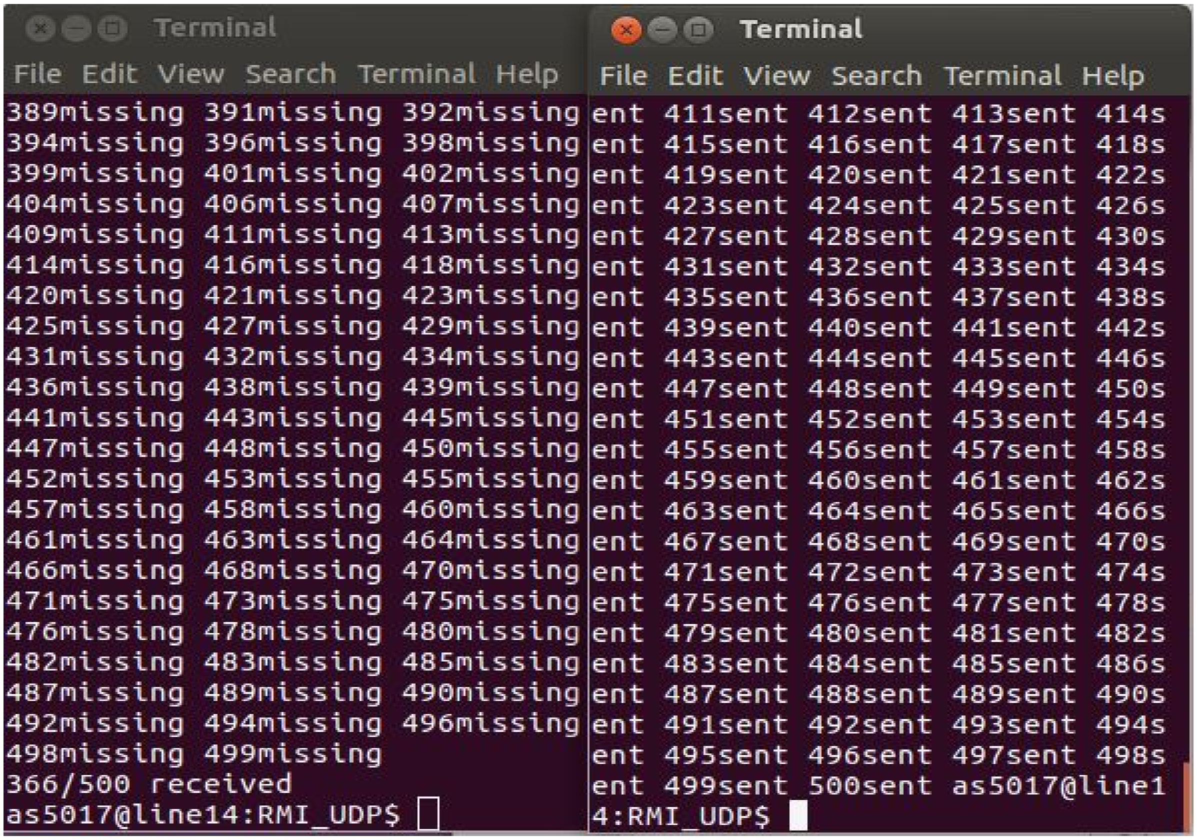
**Figure UDP 1**


Terminal
File Edit View Search Terminal Help
Terminal
File Edit View Search Terminal Help

```
Reveived 200;179
Reveived 200;180
Reveived 200;181
Reveived 200;182
Reveived 200;183
Reveived 200;184
Reveived 200;185
Reveived 200;186
Reveived 200;187
Reveived 200;188
Reveived 200;189
Reveived 200;190
Reveived 200;191
Reveived 200;192
Reveived 200;193
Reveived 200;194
Reveived 200;195
Reveived 200;196
Reveived 200;197
Reveived 200;198
Reveived 200;199
Reveived 200;200
No message is missing
as5017@arc10:RMI_UDP$
```

```
ent 111sent 112sent 113sent 114s
ent 115sent 116sent 117sent 118s
ent 119sent 120sent 121sent 122s
ent 123sent 124sent 125sent 126s
ent 127sent 128sent 129sent 130s
ent 131sent 132sent 133sent 134s
ent 135sent 136sent 137sent 138s
ent 139sent 140sent 141sent 142s
ent 143sent 144sent 145sent 146s
ent 147sent 148sent 149sent 150s
ent 151sent 152sent 153sent 154s
ent 155sent 156sent 157sent 158s
ent 159sent 160sent 161sent 162s
ent 163sent 164sent 165sent 166s
ent 167sent 168sent 169sent 170s
ent 171sent 172sent 173sent 174s
ent 175sent 176sent 177sent 178s
ent 179sent 180sent 181sent 182s
ent 183sent 184sent 185sent 186s
ent 187sent 188sent 189sent 190s
ent 191sent 192sent 193sent 194s
ent 195sent 196sent 197sent 198s
ent 199sent 200sent as5017@arc10
:RMI_UDP$
```

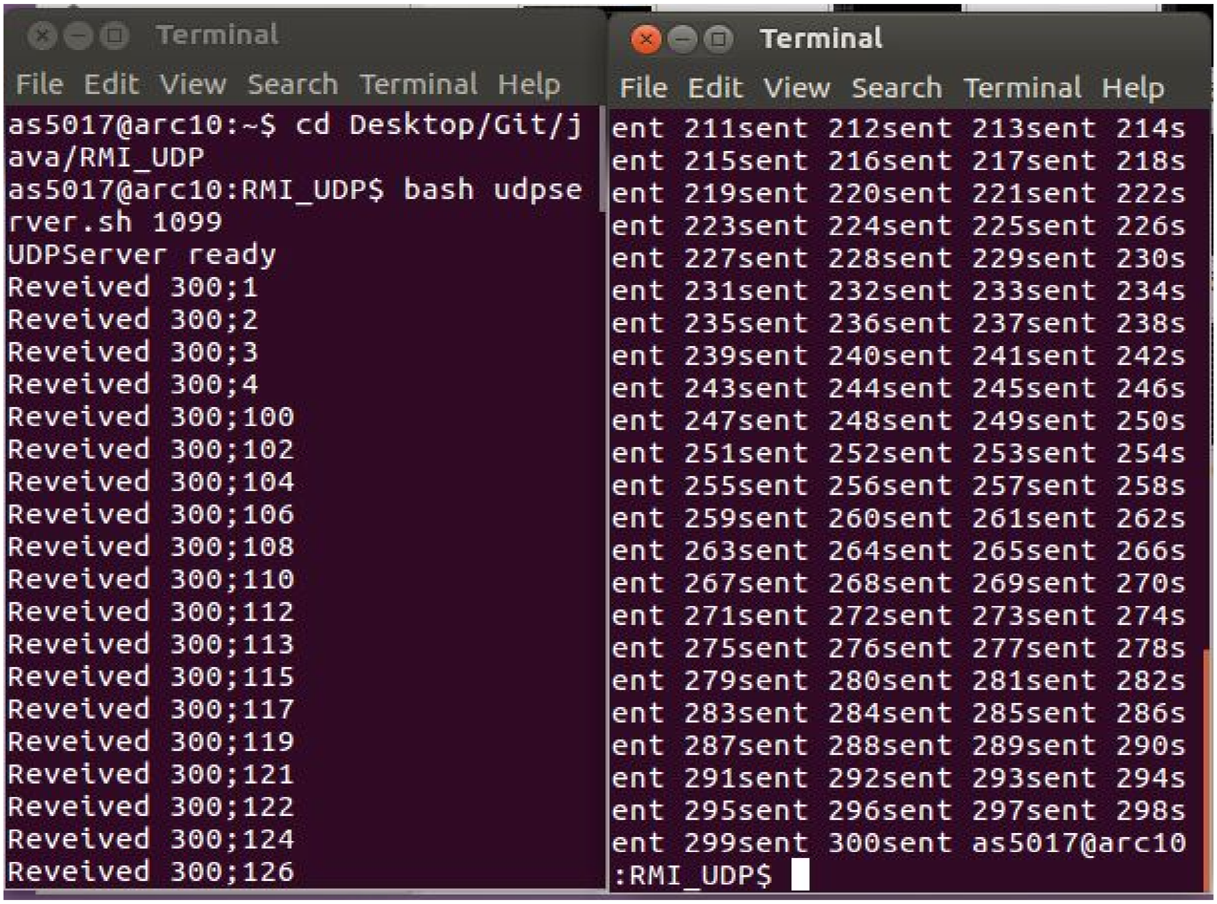
**Figure UDP 2**


Terminal
File Edit View Search Terminal Help
Terminal
File Edit View Search Terminal Help

```
389missing 391missing 392missing
394missing 396missing 398missing
399missing 401missing 402missing
404missing 406missing 407missing
409missing 411missing 413missing
414missing 416missing 418missing
420missing 421missing 423missing
425missing 427missing 429missing
431missing 432missing 434missing
436missing 438missing 439missing
441missing 443missing 445missing
447missing 448missing 450missing
452missing 453missing 455missing
457missing 458missing 460missing
461missing 463missing 464missing
466missing 468missing 470missing
471missing 473missing 475missing
476missing 478missing 480missing
482missing 483missing 485missing
487missing 489missing 490missing
492missing 494missing 496missing
498missing 499missing
366/500 received
as5017@line14:RMI_UDP$
```

```
ent 411sent 412sent 413sent 414s
ent 415sent 416sent 417sent 418s
ent 419sent 420sent 421sent 422s
ent 423sent 424sent 425sent 426s
ent 427sent 428sent 429sent 430s
ent 431sent 432sent 433sent 434s
ent 435sent 436sent 437sent 438s
ent 439sent 440sent 441sent 442s
ent 443sent 444sent 445sent 446s
ent 447sent 448sent 449sent 450s
ent 451sent 452sent 453sent 454s
ent 455sent 456sent 457sent 458s
ent 459sent 460sent 461sent 462s
ent 463sent 464sent 465sent 466s
ent 467sent 468sent 469sent 470s
ent 471sent 472sent 473sent 474s
ent 475sent 476sent 477sent 478s
ent 479sent 480sent 481sent 482s
ent 483sent 484sent 485sent 486s
ent 487sent 488sent 489sent 490s
ent 491sent 492sent 493sent 494s
ent 495sent 496sent 497sent 498s
ent 499sent 500sent as5017@line1
4:RMI_UDP$
```

Figure UDP 3


Terminal
File Edit View Search Terminal Help
Terminal
File Edit View Search Terminal Help

```
as5017@arc10:~$ cd Desktop/Git/j
ava/RMI_UDP
as5017@arc10:RMI_UDP$ bash udpse
rver.sh 1099
UDPServer ready
Reveived 300;1
Reveived 300;2
Reveived 300;3
Reveived 300;4
Reveived 300;100
Reveived 300;102
Reveived 300;104
Reveived 300;106
Reveived 300;108
Reveived 300;110
Reveived 300;112
Reveived 300;113
Reveived 300;115
Reveived 300;117
Reveived 300;119
Reveived 300;121
Reveived 300;122
Reveived 300;124
Reveived 300;126
```

```
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){
      e.printStackTrace();
      System.exit(1);
    }
  }
}
```

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

```java
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!");
     } 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
      recvSoc = 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();
  }
}
```