24.370

Telecomm

Lab Exercises

For the lab we designed a server and client based on RMI. Together these programs created a parallel machine that computed the maximum and minimum of an array of numbers. The servers registered a RMI method that calculated the min and max of an array, and the client broke up a large array into pieces that the servers can handle in parallel. The code for the client and server is attached at the back of this report.

Questions

What is the default port number used by the RMI registry?

Port 1099

What information do you need to use a remote object with RMI?

You need the remote host name and the object name that you want to use.

What security implications are there with remote objects and the server? the client? How does the JVM security manager address these issues?

The implications are that a non-trusted client can run methods on a machine running a RMI server thus by-passing the requirements of logging in and such. If this client runs any methods that access sensitive data or execute dangerous commands, this can be a problem. The client really doesn't have much of an issue with security except when the RMI server methods can be dangerously executed back on the client machine. The JVM addresses this issue by using a security manager that runs and watches and makes sure no bad commands are executed.

Why does the server program call the super() constructor? Specifically, why must an RMI class need to have an overloaded constructor, even if it is just to call this constructor?

The server program calls the super() constructor so that the parent class constructor gets called. The parent class is UnicastRemoteObject. This constructor actually creates and exports a new remote object on an anonymous port.

How does CORBA manage remote objects? What differences/advantages are there over RMI?

CORBA uses an Object Request Broker (ORB) to manage the interface between the systems. It provides mechanisms for clients to find objects based either on Name or by Trade (properties). When an object is requested, it is the ORB's duty to find the object. Stubs and skeletons are also used in CORBA just like JAVA. CORBA is improved over RMI because it has support for lifecycle management, security, transactions, and event notification. As well, CORBA allows the communicating systems to be programming language independent, so that one program could be written in C/C++ and another in JAVA.

Another resource for invoking remote methods is RPC - Remote Procedure Calls. Briefly explain this technology. Compare RPC to RMI.

RPC is a specification that allows calls to be made in HTTP using XML. It allows heterogeneous programming languages and operating systems to communicate. It uses stubs to compile the RPC into the client program. The client must specify the name of the service to be invoked across the network, the order and types of the parameters to be sent in the procedure call, and the expected type of the return value. The server must specify the name of the service it is providing, the order and types of parameters expected, and the type of the return value. The server registers with a location server so that the clients may use the service. RMI and RPC are similar except that RPC runs with HTTP and XML where as RMI is TCP based without another protocol above it.

What other remote object/method/procedure mechanisms exist for network computing?

The server program calls the super() constructor so that the parent class constructor gets called. The parent class is UnicastRemoteObject. This constructor actually creates and exports a new remote object on an anonymous port.

How does CORBA manage remote objects? What differences/advantages are there over RMI?

CORBA uses an Object Request Broker (ORB) to manage the interface between the systems. It provides mechanisms for clients to find objects based either on Name or by Trade (properties). When an object is requested, it is the ORB's duty to find the object. Stubs and skeletons are also used in CORBA just like JAVA. CORBA is improved over RMI because it has support for lifecycle management, security, transactions, and event notification. As well, CORBA allows the communicating systems to be programming language independent, so that one program could be written in C/C++ and another in JAVA.

Another resource for invoking remote methods is RPC - Remote Procedure Calls. Briefly explain this technology. Compare RPC to RMI.

RPC is a specification that allows calls to be made in HTTP using XML. It allows heterogeneous programming languages and operating systems to communicate. It uses stubs to compile the RPC into the client program. The client must specify the name of the service to be invoked across the network, the order and types of the parameters to be sent in the procedure call, and the expected type of the return value. The server must specify the name of the service it is providing, the order and types of parameters expected, and the type of the return value. The server registers with a location server so that the clients may use the service. RMI and RPC are similar except that RPC runs with HTTP and XML where as RMI is TCP based without another protocol above it.

What other remote object/method/procedure mechanisms exist for network computing?

There is software libraries called MPI that can be used for network programming. There is also another specification called DCOM which is a Microsoft strategy.

Source code for parallel max, min program

```
/** The RMIserver class:

        This class illustrates all of the components necessary to set up
an object for use as a remote object.
        When the server is run, it creates a new RMIserver object, and
initiallizes it with the constructor. The constructor first calls the
constructor of its parent class (UnicastRemoteObject) and then performs
its own initiallization which is to determine the name of the host it is
running on and assign this to the object data hostname.
        Next it uses an RMI method to bind together this object and a
reference name - RMIserver - in the registry, so that it can be accessed
remotely.

**/

import java.net.*;
import java.rmi.*;
import java.rmi.server.*;

public class RMIserver extends UnicastRemoteObject implements GreetingServer {

        String hostname;


        RMIserver() throws RemoteException {

                // This calls the constructor for the parent class.
                super();

                // Get the host name of the current computer.
                try {
                        hostname = InetAddress.getLocalHost().getHostName();
                }
                catch (java.net.UnknownHostException e) {
                        System.out.println("ERROR 5: " + e);
                        System.exit(0);
                }
        }

        public int[] FindMaxMin(int[] inArray, int arrayLength, int offset) throws RemoteException {
                int curMax = -16000;
                int curMin = 16000;

                int[] ret = new int[2];

                int i;


                for (i = 0; i < arrayLength; i++) {

                        if (inArray[i+offset] > curMax) {

                                curMax = inArray[i+offset];

                                }

                        if (inArray[i+offset] < curMin) {
```

```
                                curMin = inArray[i+offset];

                                }

                        }


                ret[0] = curMax;

                ret[1] = curMin;

                return ret;

                }

        public static void main( String args[]) {

                // Make a copy of this object, and pass it to the RMI registry.
                try {
                        RMIserver server_obj = new RMIserver();
                        Naming.rebind("RMIserver",server_obj);
                        System.out.println("The RMIserver object is ready.");
                }
                catch (java.net.MalformedURLException e) {
                        System.out.println("ERROR 1: " + e);
                        System.exit(0);
                }
                catch (java.rmi.UnknownHostException e) {
                        System.out.println("ERROR 2: " + e);
                        System.exit(0);
                }
                catch (java.rmi.RemoteException e) {
                        System.out.println("ERROR 3: " + e);
                        System.exit(0);
                }
                catch (Exception e) {
                        System.out.println("ERROR 4: " + e);
                        System.exit(0);
                }
        }
}

/** The RMIclient class:

        This class illustrates all of the components necessary instantiate
and use a remote object.
        The first segment of code should be familliar, setting the name of
the (remote) host on which the remote object resides.
        Next we try to create a "stub" - the actual instance used to
communicate over the network with the remote server. To do this we begin
by creating a generic Remote object, and use an RMI method to try and
lookup a class called RMIserver on the remote host. The RMI registry will
return an interface class, if one exists, and this is then compared to the
GreetingServer class.
        If the returned interface is the correct one, then we try to use
it by simply invoking a method defined in the interface - hello().
        This program also performs a variety of exception handling, the final
catch statement used for any unspecified exceptions. While they are
illustrated, we do not use them in any way, and are not really doing any more
than had we just used the global "Exception" as in the last catch. However
they do illustrate several (but not all) examples of specific problems that
could occur, and could be dealt with in ways other then quiting (or crashing)
the program.

**/

import java.util.*;
import java.rmi.*;

public class RMIclient {
```

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

                String shost = "localhost";
                if (args.length < 0) {
                        System.out.println("Not enough parameters!!\n");
                        System.exit(0);
                        }

                // Get remote object from shost, with name RMIserver
                try {
                        Random rand = new Random();
                        GreetingServer[] GreetingServerStub = new GreetingServer[10];
                        int[] MaxResult = new int[10];
                        int[] MinResult = new int[10];
                        int i;

                int[] returnVal = new int[2];
                        int[] inArray = new int[1000];

                        int curMax = -16000;
                        int curMin = 16000;
                        int offset = 0;

                        for (i = 0; i<1000; i++ )
                        {
                                inArray[i] = rand.nextInt(500);
                        }
                        for (i = 0; i < args.length; i++) {
                                GreetingServerStub[i] =
(GreetingServer)Naming.lookup("rmi://" + args[i] + "/RMIserver");
                                returnVal = GreetingServerStub[i].FindMaxMin(inArray,
1000/args.length, offset);
                                offset += 1000/args.length;
                                MaxResult[i] = returnVal[0];
                                MinResult[i] = returnVal[1];
                                System.out.println("Recieved Max of " + MaxResult[i] + "
and Min of " + MinResult[i] + " from " + i);
                        }

                        for (i = 0; i < args.length; i++) {
                                if (MaxResult[i] > curMax) {
                                        curMax = MaxResult[i];
                                }
                                if (MinResult[i] < curMin) {
                                        curMin = MinResult[i];
                                }
                        }
                        System.out.println("The Maximum Value Is " + curMax);
                        System.out.println("The Minimum Value Is " + curMin);
                }
                catch (java.net.MalformedURLException e) {
                        System.out.println("ERROR 1: " + e);
                        System.exit(0);
                }
                catch (java.rmi.UnknownHostException e) {
                        System.out.println("ERROR 2: " + e);
                        System.exit(0);
                }
                catch (java.rmi.NotBoundException e) {
                        System.out.println("ERROR 3: " + e);
                        System.exit(0);
                }
                catch (java.rmi.RemoteException e) {
                        System.out.println("ERROR 4: " + e);
                        System.exit(0);
                }
                catch (Exception e) {
                        System.out.println("ERROR 5: " + e);
                        System.exit(0);
                }
        }
```

Lavina Carter
Administrative Assistant
Canadian Light Source
University of Saskatchewan
107 North Road
Saskatoon SK S7N 5C6

4/4

24.370
Telecommunications Networks

Lab 1

Cameron Melvin
6721510
January 25, 2000

24.370 Laboratory 1 Report

1.3.1

- We can see that the answers form ping did not resolve to the same address, as the ece87ws.ee.umanitoba.ca is not a valid address for the ping command. We can see that the message has not changed as we do this a second time, and this is due to the fact that we have not been allocated a new IP.

1.4.1

- The name returned to us by nslookup for our IP(130.179.11.194) was dh011194.eng.umanitoba.ca. This is different than the one in the last part due to the fact that we are being given the dynamic host.

Table 1

Hostname	Response
yourMachine	130.179.11.194
yourMachine.ee.umanitoba.ca	Bad IP Address
Ic14.ee.umanitoba.ca	130.179.8.80
www.ibm.com	204.146.80.99
Xyz.umanitoba.ca	Bad IP Address

Table 2

Hostname	IP-Address
dh011194.eng.umanitoba.ca	130.179.11.194(our IP address)
eeserv.ee.umanitoba.ca	130.179.8.1
eeserv.ee.umanitoba.ca	130.179.8.1
java.sun.com	204.160.241.93,192.9.48.9,192.18.97.137
terminal.escape.ca	198.163.232.242

IBM.txt file:

```
NeoTrace  Version 1.22 - Shareware
Destination: www.ibm.com

-#-------------Node Name---------------IP Address--------RT*--High---Low---Avg-Tot---D-
Who
 1                           ece84ws 130.179.11.194       0     0     0     0   1    0
-
 2              enrouter.cc.umanitoba.ca 130.179.8.70     0    10     0     0   6    0
1
 3              atrouter.cc.umanitoba.ca 130.179.16.1     0    10     0     0   6    0
1
 4                   mrnet.mbnet.mb.ca 207.161.242.17     0    10     0     3   6    0
2
 5                                     207.161.242.33    10    10    10     6   6    0
-
 6   bxfrt-atm3-0.472.in.bellnexxia.net 192.68.64.5      20    20    10    16   6    0
3
 7                                     0.0.0.0            -    50    50    16   7    6
-
 8   sl-gw4-nyc-6-1-0-t3.sprintlink.net 144.232.171.33   40    50    40    43   6    0
4
```

```
 9        sl-bb10-nyc-2-1.sprintlink.net 144.232.7.45       50    50    40    46   6   0
4
10        sl-bb12-rly-7-0.sprintlink.net 144.232.9.226      51    51    40    50   6   0
4
11       sl-bb1-rly-4-0-0.sprintlink.net 144.232.0.34       60    60    40    50   6   0
4
12       beth1sr2-2-0-0.md.us.prserv.net 165.87.97.226      50    60    40    46   6   0
5
13       beth1ol1-2-0-0.md.us.prserv.net 165.87.29.140     110   110    70    93   6   0
5
14                           www.ibm.com 204.146.81.99     140   140   100   123   6   0
6
-------------------------------------------------------------------------------------------
------
*All times in milliseconds (ms), D=Dropped packets

----------
1:
No match for domain  umanitoba.ca .
----------
2:
No match for domain  mb.ca .
----------
3:

Whois Server Version 1.1
----------
4:

Whois Server Version 1.1
----------
5:

Whois Server Version 1.1
----------
6:

Whois Server Version 1.1
-------------------------------------------------------------------------------------------
--
NeoTrace Copyright ©1997-1998 NeoWorx inc
http://www.neoworx.com
```

ESCAPE.txt

```
NeoTrace  Version 1.22 - Shareware
□
Destination: terminal.escape.ca

-#--------------Node Name---------------IP Address--------RT*--High---Low---Avg-Tot---D-
Who
 1                               ece84ws 130.179.11.194      0     0     0     0   1   0
-
 2              enrouter.cc.umanitoba.ca 130.179.8.70        0    10     0     0  43   0
1
 3              atrouter.cc.umanitoba.ca 130.179.16.1        0    40     0     0  43   0
1
 4                     mrnet.mbnet.mb.ca 207.161.242.17      0    10     0     6  43   0
2
 5                                       207.161.242.33      0   400     0     6  43   0
-
 6                     fibre.mbnet.mb.ca 204.112.54.34      10    91    10    10  43   1
3
 7                         bb2.escape.ca 204.112.225.10     70   301    40    53  43   1
4
 8                      wpg-14.escape.ca 198.163.232.242    70    80    40    57  43   1
4
-------------------------------------------------------------------------------------------
------
*All times in milliseconds (ms), D=Dropped packets
```

```
----------
1:
No match for domain  umanitoba.ca .
----------
2:
No match for domain  mb.ca .
----------
3:
No match for domain  mb.ca .
----------
4:
No match for domain  escape.ca .
------------------------------------------------------------------------------------------
--
NeoTrace Copyright ©1997-1998 NeoWorx inc
http://www.neoworx.com
```

BRAIN.txt

```
NeoTrace  Version 1.22 - Shareware
□
Destination: brain.net.pk
□

□
-#--------------Node Name---------------IP Address--------RT*--High---Low---Avg-Tot---D-
Who
□
 1                              ece84ws 130.179.11.194      0     0     0     0   1    0
-
 2              enrouter.cc.umanitoba.ca 130.179.8.70        0     0     0     0   5    0
1
 3              atrouter.cc.umanitoba.ca 130.179.16.1        0     0     0     0   5    0
1
 4                    mrnet.mbnet.mb.ca 207.161.242.17      0    10     0     0   5    0
2
 5                                      207.161.242.33     10    10    10     3   5    0
-
 6     bxfrt-atm3-0.472.in.bellnexxia.net 192.68.64.5        10   200    10    73   5    0
3
 7      bx2blm-atm1.464.in.bellnexxia.net 205.207.238.45     40    50    30    40   5    0
3
 8    sl-gw4-nyc-6-1-0-t3.sprintlink.net 144.232.171.33     60    60    40    50   5    1
4
 9         sl-bb11-nyc-2-1.sprintlink.net 144.232.7.49       50    50    40    46   5    0
4
10         sl-bb10-chi-6-0.sprintlink.net 144.232.9.150      80    80    70    73   5    0
4
11         sl-gw22-chi-9-0.sprintlink.net 144.232.10.14      70    90    60    66   5    0
4
12 sl-abovenet-2-0-0-155m.sprintlink.net 144.232.189.6      70    80    60    76   5    0
4
13            sjc-ord-oc12.sjc2.above.net 207.126.96.118   140   151   120   140   5    0
5
14         core5-core1-oc48.sjc.above.net 216.200.0.177    141   150   120   137   5    0
5
15       main4-core5-oc3-2.sjc.above.net 208.184.102.194  120   180   120   126   5    0
5
16 singtel-abovenet.sjc.us.singtel.com.sg 209.249.140.26   120   141   120   123   5    0
6
17                                      0.0.0.0             -   321   321   107   6    5
-
18                                      202.160.250.137   330   351   320   337   5    0
-
19                                      202.160.250.1     321   341   320   330   5    0
-
20                                      202.160.250.10    330   330   320   323   5    0
-
```

```
21                                     202.160.255.181   341    360    331    344    5    0
-
22                    alpha.brain.net.pk 203.128.7.1      861    911    861    891    4    0
7
23                    brain.brain.net.pk 203.128.7.10     901    901    861    884    4    0
7
------------------------------------------------------------------------------------------
------
*All times in milliseconds (ms), D=Dropped packets

----------
1:
No match for domain  umanitoba.ca .
----------
2:
No match for domain  mb.ca .
----------
3:

Whois Server Version 1.1
----------
4:

Whois Server Version 1.1
----------
5:

Whois Server Version 1.1
----------
6:

*****************************************************************
* Copyright (C) 1998 by SGNIC.                                  *
*                                                               *
* This record is compiled by SGNIC. Whilst SGNIC has made every *
* effort to ensure that the information contained herein is     *
* updated and correct, SGNIC disclaims any liability for damage *
* or loss that may be caused as a result of error or omission.  *
* This record may only be reproduced as part of the intended    *
* search facility (provided by SGNIC) or with SGNIC's consent.  *
* Any such reproduction shall be in its entirety without        *
* modification.                                                 *
*****************************************************************
----------
7:
No match for domain  net.pk .
------------------------------------------------------------------------------------------
--
NeoTrace Copyright ©1997-1998 NeoWorx inc
http://www.neoworx.com
```

NETSCAPE.txt(shortened version of original file)

```
EtherPeek Decoded Packet File
Saved: Tue Jan 11 15:45:26 2000

Packet #1
  Flags:         0x00
  Status:        0x02  Truncated
  Packet Length:64     Slice Length:  54
  Timestamp:     15:42:00.221227 01/11/2000
Ethernet Header
  Destination:  00:10:f6:ad:20:00
  Source:       00:80:c8:2a:f1:ec
  Protocol Type:08-00  IP
IP Header - Internet Protocol Datagram
  Version:              4
  Header Length:        5
  Precedence:           0
  Type of Service:      %0000
```

```
  Unused:               %0
  Total Length:         40
  Identifier:           53346
  Fragmentation Flags:  %010  Do Not Fragment
  Fragment Offset:      0
  Time To Live:         128
  IP Type:              0x06  TCP
  Header Checksum:      0xdd92
  Source IP Address:    130.179.11.194
  Dest. IP Address:     204.160.241.196
  No Internet Datagram Options
TCP - Transport Control Protocol
  Source Port:      1505
  Destination Port: 80  World Wide Web HTTP
  Sequence Number:  92010468
  Ack Number:       4096526278
  Offset:           5
  Reserved:         %000000
  Code:             %010001
          Ack is valid
          FIN (Sender End of Byte Stream)
  Window:           7558
  Checksum:         0x45ef
  Urgent Pointer:   0
Packet is too short for further decode.
Bytes short: 10


Packet #2
  Flags:        0x00
  Status:       0x02  Truncated
  Packet Length:64    Slice Length:  58
  Timestamp:    15:42:00.261234 01/11/2000
Ethernet Header
  Destination:  00:10:f6:ad:20:00
  Source:       00:80:c8:2a:f1:ec
  Protocol Type:08-00  IP
IP Header - Internet Protocol Datagram
  Version:              4
  Header Length:        5
  Precedence:           0
  Type of Service:      %0000
  Unused:               %0
  Total Length:         44
  Identifier:           53602
  Fragmentation Flags:  %010  Do Not Fragment
  Fragment Offset:      0
  Time To Live:         128
  IP Type:              0x06  TCP
  Header Checksum:      0xdcf5
  Source IP Address:    130.179.11.194
  Dest. IP Address:     204.160.241.93
  No Internet Datagram Options
TCP - Transport Control Protocol
  Source Port:      1512
  Destination Port: 80  World Wide Web HTTP
  Sequence Number:  92809109
  Ack Number:       0
  Offset:           6
  Reserved:         %000000
  Code:             %000010
          Synch Sequence
  Window:           8192
  Checksum:         0xf85d
  Urgent Pointer:   0
  TCP Options:
    Option Type:    2  Maximum Segment Size
        Length:     4
        MSS:        1460
Packet is too short for further decode.
Bytes short: 6
```

```
Packet #3
  Flags:        0x00
  Status:       0x00
  Packet Length:64
  Timestamp:    15:42:00.331943 01/11/2000
Ethernet Header
  Destination:  00:80:c8:2a:f1:ec
  Source:       00:10:f6:ad:20:00
  Protocol Type:08-00  IP
IP Header - Internet Protocol Datagram
  Version:              4
  Header Length:        5
  Precedence:           0
  Type of Service:      %0000
  Unused:               %0
  Total Length:         40
  Identifier:           62627
  Fragmentation Flags:  %010  Do Not Fragment
  Fragment Offset:      0
  Time To Live:         92
  IP Type:              0x06  TCP
  Header Checksum:      0xdd51
  Source IP Address:    204.160.241.196
  Dest. IP Address:     130.179.11.194
  No Internet Datagram Options
TCP - Transport Control Protocol
  Source Port:      80  World Wide Web HTTP
  Destination Port: 1505
  Sequence Number:  4096526278
  Ack Number:       0
  Offset:           5
  Reserved:         %000000
  Code:             %000100
            Reset Connection
  Window:           0
  Checksum:         0x60e2
  Urgent Pointer:   0
  No TCP Options
    No More HTTP Data
Extra bytes (Padding):
  ......                00 00 00 00 00 00
Frame Check Sequence:  0x00000000

Packet #4
  Flags:        0x00
  Status:       0x00
  Packet Length:64
  Timestamp:    15:42:00.378762 01/11/2000
Ethernet Header
  Destination:  00:80:c8:2a:f1:ec
  Source:       00:10:f6:ad:20:00
  Protocol Type:08-00  IP
IP Header - Internet Protocol Datagram
  Version:              4
  Header Length:        5
  Precedence:           0
  Type of Service:      %0000
  Unused:               %0
  Total Length:         44
  Identifier:           49657
  Fragmentation Flags:  %010  Do Not Fragment
  Fragment Offset:      0
  Time To Live:         236
  IP Type:              0x06  TCP
  Header Checksum:      0x805e
  Source IP Address:    204.160.241.93
  Dest. IP Address:     130.179.11.194
  No Internet Datagram Options
TCP - Transport Control Protocol
```

```
  Source Port:      80  World Wide Web HTTP
  Destination Port: 1512
  Sequence Number:  1696313818
  Ack Number:       92809110
  Offset:           6
  Reserved:         %000000
  Code:             %010010
            Ack is valid
            Synch Sequence
  Window:           33580
  Checksum:         0x7e2a
  Urgent Pointer:   0
  TCP Options:
    Option Type:    2  Maximum Segment Size
        Length:     4
        MSS:        1460
    No More HTTP Data
Extra bytes (Padding):
  ..                    00 00
Frame Check Sequence:  0x36000000

Packet #5
  Flags:        0x00
  Status:       0x02  Truncated
  Packet Length:64    Slice Length:  54
  Timestamp:    15:42:00.379277 01/11/2000
Ethernet Header
  Destination:  00:10:f6:ad:20:00
  Source:       00:80:c8:2a:f1:ec
  Protocol Type:08-00  IP
IP Header - Internet Protocol Datagram
  Version:              4
  Header Length:        5
  Precedence:           0
  Type of Service:      %0000
  Unused:               %0
  Total Length:         40
  Identifier:           53858
  Fragmentation Flags:  %010  Do Not Fragment
  Fragment Offset:      0
  Time To Live:         128
  IP Type:              0x06  TCP
  Header Checksum:      0xdbf9
  Source IP Address:    130.179.11.194
  Dest. IP Address:     204.160.241.93
  No Internet Datagram Options
TCP - Transport Control Protocol
  Source Port:      1512
  Destination Port: 80  World Wide Web HTTP
  Sequence Number:  92809110
  Ack Number:       1696313819
  Offset:           5
  Reserved:         %000000
  Code:             %010000
            Ack is valid
  Window:           8760
  Checksum:         0xf6db
  Urgent Pointer:   0
Packet is too short for further decode.
Bytes short: 10


Packet #6
  Flags:        0x00
  Status:       0x00
  Packet Length:318
  Timestamp:    15:42:00.383467 01/11/2000
Ethernet Header
  Destination:  00:10:f6:ad:20:00
  Source:       00:80:c8:2a:f1:ec
  Protocol Type:08-00  IP
```

```
IP Header - Internet Protocol Datagram
  Version:              4
  Header Length:        5
  Precedence:           0
  Type of Service:      %0000
  Unused:               %0
  Total Length:         300
  Identifier:           54114
  Fragmentation Flags:  %010  Do Not Fragment
  Fragment Offset:      0
  Time To Live:         128
  IP Type:              0x06  TCP
  Header Checksum:      0xd9f5
  Source IP Address:    130.179.11.194
  Dest. IP Address:     204.160.241.93
  No Internet Datagram Options
TCP - Transport Control Protocol
  Source Port:      1512
  Destination Port: 80  World Wide Web HTTP
  Sequence Number:  92809110
  Ack Number:       1696313819
  Offset:           5
  Reserved:         %000000
  Code:             %011000
            Ack is valid
            Push Request
  Window:           8760
  Checksum:         0xa453
  Urgent Pointer:   0
  No TCP Options
HTTP - HyperText Transfer Protocol
  GET / HTTP/1.0..  47 45 54 20 2f 20 48 54 54 50 2f 31 2e 30 0d 0a
  Connection: Keep  43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 4b 65 65 70
  -Alive..          2d 41 6c 69 76 65 0d 0a
  User-Agent: Mozi  55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69
  lla/4.5 [en] (Wi  6c 6c 61 2f 34 2e 35 20 5b 65 6e 5d 20 28 57 69
  nNT; I)..         6e 4e 54 3b 20 49 29 0d 0a
  Host: java.sun.c  48 6f 73 74 3a 20 6a 61 76 61 2e 73 75 6e 2e 63
  om..              6f 6d 0d 0a
  Accept: image/gi  41 63 63 65 70 74 3a 20 69 6d 61 67 65 2f 67 69
  f, image/x-xbitm  66 2c 20 69 6d 61 67 65 2f 78 2d 78 62 69 74 6d
  ap, image/jpeg,   61 70 2c 20 69 6d 61 67 65 2f 6a 70 65 67 2c 20
  image/pjpeg, ima  69 6d 61 67 65 2f 70 6a 70 65 67 2c 20 69 6d 61
  ge/png, */*..     67 65 2f 70 6e 67 2c 20 2a 2f 2a 0d 0a
  Accept-Encoding:  41 63 63 65 70 74 2d 45 6e 63 6f 64 69 6e 67 3a
   gzip..           20 67 7a 69 70 0d 0a
  Accept-Language:  41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 65 3a
   en..             20 65 6e 0d 0a
  Accept-Charset:   41 63 63 65 70 74 2d 43 68 61 72 73 65 74 3a 20
  iso-8859-1,*,utf  69 73 6f 2d 38 38 35 39 2d 31 2c 2a 2c 75 74 66
  -8....            2d 38 0d 0a 0d 0a
Frame Check Sequence:  0x00000000
```

NETSCAPE.pkt(shortened version of the overall file)

```
Date: Tue, 11 Jan 2000 21:34:40 GMT

Server: Apache/1.3.6 (Unix)
□
Cache-Control: max-age=86400
□
Expires: Wed, 12 Jan 2000 21:34:40 GMT
□
Connection: close
□
Content-Type: text/html
□

□
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
```

```
<HTML>

<HEAD>

        <TITLE>java.sun.com - The Source for Java(TM) Technology</TITLE>

<!------------BEGIN-META TAGS-------->

<meta http-equiv="Expires" CONTENT="Wed, 16 June 1999 00:00:00 GMT">

<meta name="description" CONTENT="Sun Microsystems' Java Technology Home Page - Visit
this site to get the latest Java technologies, news, and products.">
<meta name="keywords" CONTENT="Java, JDK, JVM, activeX bridge, jumpstart, JDC, ava,
Java language"><!-- IRMonitorTag#5618 -->
<meta name="distribution" CONTENT="global">

<!-------------END-META TAGS--------->

<SCRIPT LANGUAGE="javascript">

<!-- Begin
function Start(page) {
window.self.name="jsctarget";
OpenWin = this.open(page, "CtrlWindow",
"toolbar=no,menubar=no,location=no,scrollbars=no,resize=yes,height=410,width=235");
}
function gotoFunctionShortcuts() {
        self.location =
document.shortcutsGoto.link.options[document.shortcutsGoto.link.selectedIndex].value;
}
function gotoFunctionPlatform() {
        self.location =
document.links.link.options[document.links.link.selectedIndex].value;
}
function gotoFunctionBusiness() {
        self.location =
document.businessGoto.link.options[document.businessGoto.link.selectedIndex].value;
}
function gotoFunctionInê□□□□Š□î□□□□□
Ä-¦ñi‡□□È*ñi□□ö- □□□E□□ÜÁu@□ì□z«Ì ñ],³
Â□P□èe□·□□^(šP□úõ
O□□dustry() {
        self.location =
document.industryGoto.link.options[document.industryGoto.link.selectedIndex].value;
}

// End -->
</SCRIPT>

</HEAD>

<BODY BGCOLOR="#FFFFFF" TEXT="#000000">

<!-- server side include for top part of page -->

<TABLE BORDER="0" CELLSPACING="0" CELLPADDING="0" WIDTH="100%">
<TR ALIGN="CENTER" VALIGN="TOP">
<TD WIDTH="157" ROWSPAN="2" ALIGN="LEFT">
<IMG SRC="/images/pixel.gif" HEIGHT="40" WIDTH="40" ALT="">
<A HREF="/"><IMG SRC="/images/logos/javalogo52x88.gif" WIDTH="52" HEIGHT="88" ALT="Java
Technology Home Page" BORDER="0"></A><BR>
<IMG SRC="/images/pixel.gif" WIDTH="157" HEIGHT="1" ALT=""></TD>

<TD>

<!-- Begin Header Segment -->

<FORM method=GET action="http://search.java.sun.com/query.html">
<input type=hidden name=col value="jsun">
<input type=hidden name=pw value="100%">
<input type=hidden name=ws value="0">
```

```
<input type=hidden name=nh value="10">
<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0" WIDTH="100%">
<TR VALIGN="CENTER">
<TD WIDTH="100%">
  <TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0" WIDTH="100%">
  <TR><TD ROWSPAN="7" ALIGN="RIGHT"><IMG SRC="/images/banners/stripelt.gif" WIDTH="6"
HEIGHT="14" ALT=""></TD>
    <TD BGCOLOR="#CC9966" WIDTH="100%"><IMG SRC="/images/pixel.gif" WIDTH="1" HEIGHT="2"
ALT=""></TD>
    <TD ROWSPAN="7" ALIGN="LEFT"><IMG SRC="/images/banners/stripert.gif" WIDTH="6"
HEIGHT="14" ALT=""></TD></TR>
  <TR><TD><IMG SRC="/images/pixel.gif" WIDTH="1" HEIGHT="2" ALT=""></6□□□□Š□@□6□□□
Ă-¦ñr□□□ö- □□□È*ñì□□E□□(Ôb@□□□Ùù,³
ÂÌ ñ]□è□P□^(še□½CP□"8êo□□□Š□î□□□□□
Ă-¦óU□□□È*ñì□□ö- □□□E□□ŮÁý@□i□zªÌ ñ],³
Â□P□èe□½C□^(šP□úð—¼□□TD></TR>
  <TR><TD BGCOLOR="#CC9966"><IMG SRC="/images/pixel.gif" WIDTH="1" HEIGHT="2"
ALT=""></TD></TR>
  <TR><TD><IMG SRC="/images/pixel.gif" WIDTH="1" HEIGHT="2" ALT=""></TD></TR>
  <TR><TD BGCOLOR="#CC9966"><IMG SRC="/images/pixel.gif" WIDTH="1" HEIGHT="2"
ALT=""></TD></TR>
  <TR><TD><IMG SRC="/images/pixel.gif" WIDTH="1" HEIGHT="2" ALT=""></TD></TR>
  <TR><TD BGCOLOR="#CC9966"><IMG SRC="/images/pixel.gif" WIDTH="1" HEIGHT="2"
ALT=""></TD></TR>
  </TABLE></TD>
<TD><A HREF="/a-z/"><IMG SRC="/images/banners/azindex.gif" BORDER="0" WIDTH="72"
HEIGHT="11" ALT="A-Z Index"></A></TD>
<TD><FONT FACE="Helvetica" SIZE="1"><INPUT TYPE="text" SIZE="15" MAXLENGTH="128"
NAME=qt></FONT></TD>
<TD><INPUT TYPE="image" SRC="/images/banners/search.button.gif" WIDTH="55" HEIGHT="14"
BORDER="0" value="search" NAME="Search" ALT="Search"></TD></TR></TABLE>
</FORM><P>


<TABLE BORDER="0" WIDTH="100%" CELLPADDING="0" CELLSPACING="0">
<TR VALIGN="TOP">
<TD WIDTH="100%"><IMG SRC="/images/pixel.gif" HEIGHT="18" WIDTH="20" ALT=""></TD>
<TD ROWSPAN="3" ALIGN="RIGHT">
<A HREF="/"><IMG SRC="/images/banners/thesourceforjava.gif" WIDTH="389" HEIGHT="42"
ALT="The Source for Java Technology" BORDER=0></A></TD></TR>
<TR VALIGN="TOP">
<TD BGCOLOR="#CC9966" HEIGHT="1" WIDTH="100%">
 <IMG SRC="/images/pixel.gif" HEIGHT="1" WIDTH="1" ALT=""></TD></TR>
<TR VALIGN="TOP">
<TD WIDTH="100%"><IMG SRC="/images/pixel.gif" WIDTH="20" HEIGHT="20" ALT=""></TD>
</TR>
</TABLE>


</TD>`□□□□Š□d□□□□□
Ă-¦ó\â□□È*ñì□□ö- □□□E□□RÁþ@□ì□□3Ì ñ],³
Â□P□èe□Â÷□^(šP□úð□Z□□
</TR>

<TR><TD>&nbsp;</TD></TR>
</TABLE>
ê□□□□Š□î□□□□□
Ă-¦ó^□□□È*ñì□□ö- □□□E□□ŮÁý@□ì□z¨Ì ñ],³
Â□P□èe□Ã!□^(šP□úð«z□□

<!-- end server side include for top of page -->

<!-- begin table for main body navigation and content -->

<TABLE BORDER="0" CELLSPACING="0" CELLPADDING="0" WIDTH="100%">
<TR ALIGN="LEFT" VALIGN="TOP">
<TD WIDTH="157">

<!-- include for global java.sun.com navigation -->
```

```
    <TABLE BORDER="0" CELLSPACING="0" CELLPADDING="3" WIDTH="157">

<!-- the following spacer ensures that the -->
<!--   table is at least 157 pixels wide   -->

<TR><TD></TD></TR>

<!-- tab categories -->

<TR><TD></TD>
<TD><A HREF="/products/">
<IMG SRC="/images/navbar/side.tab.products.gif" HEIGHT="15" WIDTH="130"
BORDER="0" ALT="Downloads, APIs, Documentation"></A></TD>
</TR>

<TR><TD></TD>
<TD><A HREF="/jdc/">
<IMG SRC="/images/navbar/side.tab.developer.gif" HEIGHT="15" WIDTH="130"
BORDER="0" ALT="Java Developer Connection"></A></TD>
</TR>

<TR><TD></TD>
<TD><A HREF="/infodocs/"><IMG SRC="/images/navbar/side.tab.docs.gif" HEIGHT="15"
WIDTH="130"
BORDER="0" ALT="Docs, Tutorials, Tech Articles, Training"></A></TD>
</TR>

<TR><TD></TD>
<TD><A HREF="/support/"><IMG SRC="/images/navbar/side.tab.support.gif" HEIGHT="15"
WIDTH="130"
BORDER="0" ALT="Online Support"></A></TD>
</TR>

<TR><TD></TD>
<TD><A HREF="/community/">
<IMG SRC="/images/navbar/side.tab.community.gif" HEIGHT="15" WIDTH="130" BORDER="0"
ALT="Community Discussion"></A></TD>
</TR>

<TR><TD></TD>
<TD><A HREF="/industry">
<IMG SRC="/images/navbar/side.tab.news.gif" HEIGHT="15" WIDTH="130"
BORDER="0" ALT6□□□□Š□@□6□□□
Ä-¦óf□□□ö- □□□È*ñì□□E□□(Õb@□□□Øù,³
ÂÌ ñ]□è□P□ˆ(še□ÈÕP□"8ÞÝ□□□Š□î□□□□□
Ä-¦õ,™□□È*ñì□□ö- □□□E□□ÜÂ□@□ì□z§Ì ñ],³
Â□P□èe□ÈÕ□ˆ(šP□úð¯-□□="News & Events from Everywhere"></A></TD>
</TR>

<TR><TD></TD>
<TD><A HREF="/solutions/">
<IMG SRC="/images/navbar/side.tab.solutions.gif" HEIGHT="15" WIDTH="130"
BORDER="0" ALT="Products from Everywhere"></A></TD>
</TR>

<TR><TD></TD>
<TD><A HREF="/casestudies/">
<IMG SRC="/images/navbar/side.tab.case.gif" HEIGHT="15" WIDTH="130"
BORDER="0" ALT="How Java Technology is Used Worldwide"></A></TD>
</TR>

<TR><TD></TD><TD></TD></TR>

</TABLE>


<!-- end include for main java.sun.com navigation -->

<TABLE BORDER="0" CELLSPACING="0" CELLPADDING="3" WIDTH="157">

<TR>
```

```
<TD></TD>

<TD WIDTH="100%">

<IMG SRC="/images/applet.launch.duke.gif" WIDTH="15" HEIGHT="12" BORDER="0"
ALT="Launch Applet">
<FONT SIZE="-2"><A
href="javascript:Start('/share/classes/sitemenuapplet/index.html')";
STYLE="text-decoration: none">Site Menu Applet</A></FONT>

</TD>

</TR>
```

NEOTRACE.txt

```
NeoTrace  Version 1.22 - Shareware
□
Destination: java.sun.com

-#--------------Node Name---------------IP Address--------RT*--High---Low---Avg-Tot---
Dropped
 1                              ece84ws 130.179.11.194      0     0     0     0   1   0
 2                enrouter.cc.umanitoba.ca 130.179.8.70     0     0     0     0   1   0
 3                atrouter.cc.umanitoba.ca 130.179.16.1    10    10    10    10   1   0
 4                       mrnet.mbnet.mb.ca 207.161.242.17  10    10    10    10   1   0
 5                                       207.161.242.33    10    10    10    10   1   0
 6     bxfrt-atm3-0.472.in.bellnexxia.net 192.68.64.5      10    10    10    10   1   0
 7      bx2blm-atm1.464.in.bellnexxia.net 205.207.238.45   40    40    40    40   1   0
 8 bordercore3-hssi0-0-0.westorange.cw.net 166.48.9.249      60    60    60    60   1
0
 9          corerouter1.westorange.cw.net 204.70.9.138     50    50    50    50   1   0
10                    xcore3.boston.cw.net 204.70.150.81    60    60    60    60   1   0
11        sun-micro-system.boston.cw.net 204.70.179.102   81    81    81    81   1   0
12               softwarema.usec.sun.com 192.9.48.9       50    80    50    65   2   0
------------------------------------------------------------------------------------------
------
*All times in milliseconds (ms), D=Dropped packets

------------------------------------------------------------------------------------------
--
NeoTrace Copyright ©1997-1998 NeoWorx inc
http://www.neoworx.com
```

ETHERPEEK.txt(shortened version of the original file)

```
EtherPeek Decoded Packet File
□
Saved: Tue Jan 11 15:53:57 2000
□

□
Packet #1
□
  Flags:         0x00
□
  Status:        0x02  Truncated
□
  Packet Length:64     Slice Length:  51
□
  Timestamp:     15:53:32.914987 01/11/2000
Ethernet Header
  Destination:   00:10:f6:ad:20:00
  Source:        00:80:c8:2a:f1:ec
  Protocol Type:08-00  IP
IP Header - Internet Protocol Datagram
  Version:              4
```

```
  Header Length:        5
  Precedence:           0
  Type of Service:      %0000
  Unused:               %0
  Total Length:         37
  Identifier:           39531
  Fragmentation Flags:  %000
  Fragment Offset:      0
  Time To Live:         1
  IP Type:              0x01  ICMP
  Header Checksum:      0xa0e5
  Source IP Address:    130.179.11.194
  Dest. IP Address:     192.9.48.9
  No Internet Datagram Options
ICMP - Internet Control Messages Protocol
  ICMP Type:            8  Echo Request
  Code:                 0
  Checksum:             0x4f39
  Identifier:           0x0100
  Sequence Number:      5190
  ICMP Data Area:       No more data.
Extra bytes (Padding):
  NeoTrace.             4e 65 6f 54 72 61 63 65 00
Packet is too short for further decode.
Bytes short: 13


Packet #2
  Flags:        0x00
  Status:       0x00
  Packet Length:74
  Timestamp:    15:53:32.915927 01/11/2000
Ethernet Header
  Destination:  00:80:c8:2a:f1:ec
  Source:       00:10:f6:ad:20:00
  Protocol Type:08-00  IP
IP Header - Internet Protocol Datagram
  Version:              4
  Header Length:        5
  Precedence:           6
  Type of Service:      %0000
  Unused:               %0
  Total Length:         56
  Identifier:           57967
  Fragmentation Flags:  %000
  Fragment Offset:      0
  Time To Live:         255
  IP Type:              0x01  ICMP
  Header Checksum:      0xbf26
  Source IP Address:    130.179.8.70
  Dest. IP Address:     130.179.11.194
  No Internet Datagram Options
ICMP - Internet Control Messages Protocol
  ICMP Type:            11  Time Exceeded
  Code:                 0  Time to Live count exceeded
  Checksum:             0x8880
  Unused (must be zero):0x00000000

Header of packet that caused error follows.
IP Header - Internet Protocol Datagram
  Version:              4
  Header Length:        5
  Precedence:           0
  Type of Service:      %0000
  Unused:               %0
  Total Length:         37
  Identifier:           39531
  Fragmentation Flags:  %000
  Fragment Offset:      0
  Time To Live:         1
  IP Type:              0x01  ICMP
```

```
  Header Checksum:      0xa0e5
  Source IP Address:    130.179.11.194
  Dest. IP Address:     192.9.48.9
  No Internet Datagram Options
ICMP - Internet Control Messages Protocol
  ICMP Type:            8  Echo Request
  Code:                 0
  Checksum:             0x4f39
  Identifier:           0x0100
  Sequence Number:      5190
  ICMP Data Area:
  ....                  00 00 00 00

Packet #3
  Flags:        0x00
  Status:       0x02  Truncated
  Packet Length:64    Slice Length:  51
  Timestamp:    15:53:32.920429 01/11/2000
Ethernet Header
  Destination:  00:10:f6:ad:20:00
  Source:       00:80:c8:2a:f1:ec
  Protocol Type:08-00  IP
IP Header - Internet Protocol Datagram
  Version:              4
  Header Length:        5
  Precedence:           0
  Type of Service:      %0000
  Unused:               %0
  Total Length:         37
  Identifier:           39787
  Fragmentation Flags:  %000
  Fragment Offset:      0
  Time To Live:         2
  IP Type:              0x01  ICMP
  Header Checksum:      0x9ee5
  Source IP Address:    130.179.11.194
  Dest. IP Address:     192.9.48.9
  No Internet Datagram Options
ICMP - Internet Control Messages Protocol
  ICMP Type:            8  Echo Request
  Code:                 0
  Checksum:             0x4e39
  Identifier:           0x0100
  Sequence Number:      5446
  ICMP Data Area:       No more data.
Extra bytes (Padding):
  NeoTrace.             4e 65 6f 54 72 61 63 65 00
Packet is too short for further decode.
Bytes short: 13


Packet #4
  Flags:        0x00
  Status:       0x00
  Packet Length:74
  Timestamp:    15:53:32.921316 01/11/2000
Ethernet Header
  Destination:  00:80:c8:2a:f1:ec
  Source:       00:10:f6:ad:20:00
  Protocol Type:08-00  IP
IP Header - Internet Protocol Datagram
  Version:              4
  Header Length:        5
  Precedence:           6
  Type of Service:      %0000
  Unused:               %0
  Total Length:         56
  Identifier:           44114
  Fragmentation Flags:  %000
  Fragment Offset:      0
  Time To Live:         254
```

```
  IP Type:               0x01  ICMP
  Header Checksum:       0xee88
  Source IP Address:     130.179.16.1
  Dest. IP Address:      130.179.11.194
  No Internet Datagram Options
ICMP - Internet Control Messages Protocol
  ICMP Type:             11  Time Exceeded
  Code:                  0  Time to Live count exceeded
  Checksum:              0x8880
  Unused (must be zero):0x00000000

Header of packet that caused error follows.
IP Header - Internet Protocol Datagram
  Version:               4
  Header Length:         5
  Precedence:            0
  Type of Service:       %0000
  Unused:                %0
  Total Length:          37
  Identifier:            39787
  Fragmentation Flags:   %000
  Fragment Offset:       0
  Time To Live:          1
  IP Type:               0x01  ICMP
  Header Checksum:       0x9fe5
  Source IP Address:     130.179.11.194
  Dest. IP Address:      192.9.48.9
  No Internet Datagram Options
ICMP - Internet Control Messages Protocol
  ICMP Type:             8  Echo Request
  Code:                  0
  Checksum:              0x4e39
  Identifier:            0x0100
  Sequence Number:       5446
  ICMP Data Area:
  ....                   00 00 00 00

Packet #5
  Flags:        0x00
  Status:       0x02  Truncated
  Packet Length:64    Slice Length:  51
  Timestamp:    15:53:32.948929 01/11/2000
Ethernet Header
  Destination:  00:10:f6:ad:20:00
  Source:       00:80:c8:2a:f1:ec
  Protocol Type:08-00  IP
IP Header - Internet Protocol Datagram
  Version:               4
  Header Length:         5
  Precedence:            0
  Type of Service:       %0000
  Unused:                %0
  Total Length:          37
  Identifier:            40043
  Fragmentation Flags:   %000
  Fragment Offset:       0
  Time To Live:          3
  IP Type:               0x01  ICMP
  Header Checksum:       0x9ce5
  Source IP Address:     130.179.11.194
  Dest. IP Address:      192.9.48.9
  No Internet Datagram Options
ICMP - Internet Control Messages Protocol
  ICMP Type:             8  Echo Request
  Code:                  0
  Checksum:              0x4d39
  Identifier:            0x0100
  Sequence Number:       5702
  ICMP Data Area:        No more data.
Extra bytes (Padding):
  NeoTrace.              4e 65 6f 54 72 61 63 65 00
```

```
Packet is too short for further decode.
Bytes short: 13
```

PROTOCOL.txt

```
# This is an EtherPeek statistics file created:
□
# Tue Jan 11 16:07:36 2000
# The format of this file is:
# Indent    Protocol    Packets
0       "Ethernet Type 2"      0
1       "IP"    0
2       "ICMP"  0
3       "Echo Req"      12
3       "Time Exceed"   10
3       "Echo Reply"    2
```

REVIEW QUESTIONS:

1. The warning about not logging into Unix is because other people that are capturing packets in the network and would be able to pick the usernames and passwords.
2. A service like DNS is important because it enables people to use computer names and not having to remember long and hard to remember IP addresses.
3. Possible uses of arp, ping and nslookup are the following: Arp can used to determine your IP address; Ping can be used for determining if certain computers are still active; and lastly Nslookup can be used to connect to a DNS server.
4. The packet(trace) to java.sun.com did not take the same route as demonstrated above, and this is because there are many different routes avaliable. This could be attributed to the usage of certain routes and servers, also net traffic, and it also depends on the way the routers determine your best way of getting there. We did encounter some loops called 'No Answer' and I belive that this means that the route was seen as viable to the router, however, the nodes(or computers) themselves were not currently available.
5. The TTL field is the 'Time To Live' field. It is used to determine the path taken to reach a defined target destination, and this is why it is used in the traceroute program. The way it is shown to the user is that it starts with a TTL field of 1 and then progresses by one until the destination is reached.
6. The return packets did not use the same number of hops returning from java.sun.com and this is due to dynamic allocation of the path.
7. By tracing several U of M Unix machines we were able to notice that they all pass through enrouter to connect to them. Thus the same router is used.
8. In the traceroute packets, the ethernet and IP addresses

   ____

   ____

   ____

   ____

   ____

9. If we were using a Pentium computer with the division bug, the result would still be correct because of 'Virtual division'. This is a good thing because it is platform independent and thus all computers will work, even ones that have the bug.

10. Sure there are other ‘hardware’ bugs that we never see, especially since more and more operations and devices are put into computers nowadays.

tep@ee.umanitoba.ca.

login: lab 1
psod: sajid#3

Laboratory #2 Client/Server, Telnet, RIP, and DNS

Part 1: Network Programming

Client/Server Programming:

On the *lab web* page there is a simple java *client* and *server* example. In this example, the client program uses a socket class (sockets are used in windows to create a tcp connection) to connect to the server. By default the client tries to access the server on the same computer using port number 9100. You can also specify an IP address when you run the client.

i. Download the two programs, view them and determine what is being done by each application. Compile and run the programs (you will need two DOS shells to run both programs at once).

   - ○ Did they perform as you expected, if not go back over the code, and find out why.
   - ○ What is the IP address of '*localhost*'? 127.0.0.1

   Without using EtherPeek, do you think any network traffic is being generated. Use EtherPeek to verify your suspicions, set the capture filter to only capture IP traffic from your computers IP address (remember to verify your machines address first). Using your client, specify a neighbours IP address and try to access their server.

   Joel: 130.179.9.52
   Ours: 130.179.11.194

ii. Edit the client and server programs so that;

   The server application:

   - ○ accepts a connection from the client, –
   - ○ prompts the client for a userid,
   - ○ reads the userid from the client,
   - ○ prompts the client for a password,
   - ○ reads the password from the client,
   - ○ send a message to the client, and
   - ○ closes the connection.

   and the client application that interacts with the server:

   - ○ initiates a connection to the server,
   - ○ reads a prompt from the server (userid),
   - ○ respond to the prompt (userid),
   - ○ reads a prompt from the server (password),
   - ○ respond to the prompt (password),
   - ○ read and display the message from the server, and
   - ○ closes the connection.

   Run the modified client/server applications.

iii. Use your client to connect to the server running on a neighbours machine and capture the traffic using Etherpeek.

   - ○ identify the three way handshake to establish the tcp connection and show the sequence and acknowledge number of the communication during the connection.

Laboratory #2 Client\Server, Telnet, RIP, and DNS

Part 1: Network Programming

Client\Server Programming:

On the *lab* web page there is a simple java *client and server* example. In this example, the client program uses a socket class (sockets are used in windows to create a tcp connection) to connect to the server. By default the client tries to access the server on the same computer using port number 9100. You can also specify an IP address when you run the client.

i. Download the two programs, view them and determine what is being done by each application. Compile and run the programs (you will need two DOS shells to run both programs at once).

  - ○ Did they perform as you expected, if not go back over the code, and find out why.
  - ○ What is the IP address of '*localhost*'?

Without using EtherPeek, do you think any network traffic is being generated. Use EtherPeek to verify your suspicions, set the capture filter to only capture IP traffic from your computers IP address (remember to verify your machines address first). Using your client, specify a neighbours IP address and try to access their server.

ii. Edit the client and server programs so that:

The server application:

  - ○ accepts a connection from the client,
  - ○ prompts the client for a userid,
  - ○ reads the userid from the client,
  - ○ prompts the client for a password,
  - ○ reads the password from the client,
  - ○ send a message to the client, and
  - ○ closes the connection.

and the client application that interacts with the server:

  - ○ initiates a connection to the server,
  - ○ reads a prompt from the server (userid),
  - ○ respond to the prompt (userid),
  - ○ reads a prompt from the server (password),
  - ○ respond to the prompt (password),
  - ○ read and display the message from the server, and
  - ○ closes the connection.

Run the modified client\server applications.

iii. Use your client to connect to the server running on a neighbours machine and capture the traffic using Etherpeck.

  - ○ identify the three way handshake to establish the tcp connection and show the sequence and acknowledge number of the communication during the connection.

○ match the observed packets with the operations in your program.
○ identify the port numbers used in the tcp packets, and comment on their values, and
○ identify the termination sequence of the connection.
○ Did this connection behave as you expected? Comment on potential or actual problems.

Part 2: TCP/IP Networks

Telnet:

Telnet is a program used to establish character based communication with internet servers via TCP/IP. Telnet is most commonly used to login to another computer (via a telnet server, port 23). However, telnet may also be used to communicate with other services that use a text based protocol, for example your modified server program.

1. Use telnet to interact with your modified server program, and capture the network traffic that is generated.
   i. Begin a capture session using EtherPeek, then start your server (if it is not still running).
   ii. Telnet to your server (port 9100):
      - At the command prompt type: `telnet IP_Address 9100`
   iii. Interact with the server using text input:
      - Do you notice anything different from a normal telnet session? Explain.
   iv. Stop Etherpeek and compare the captured session to the session using the client application.

2. The Simple Mail Transfer Protocol (SMTP - RFC 821), is another service that can be accessed directly with telnet using tcp port 25. This is the port that mail servers (such as the sendmail program on UNIX) uses to accept email messages.

   i. telnet to port 25 on any UNIX workstation, for example:

      mail.cc.umanitoba.ca 25

      telnet ~~ouzo.ee.umanitoba.ca 25~~

   ii. To determine what commands are available enter help, or for a specific command enter:

   ```
   help command_name
   ```

   iii. Use the following commands to email yourself. Note that you can enter anything you wish as the sender information (this is one method used to send fake email):

   ```
   helo, mail, rcpt, data, and quit
   ```

   These commands should be used in the order given, and the help facility can be used to find out the proper syntax.

   iv. Send an email message to your TA, with subject c24422-lab2, from a phony email address. (remember to indicate who you are in the message to receive lab credit).

- ○ match the observed packets with the operations in your program,
- ○ identify the port numbers used in the tcp packets, and comment on their values, and
- ○ identify the termination sequence of the connection.
- ○ Did this connection behave as you expected? Comment on potential or actual problems.

Part 2: TCP/IP Networks

Telnet

Telnet is a program used to establish character based communication with internet servers via TCP/IP. Telnet is most commonly used to login to another computer (via a telnet server, port 23). However, telnet may also be used to communicate with other services that use a text based protocol, for example your modified server program.

1. Use telnet to interact with your modified server program, and capture the network traffic that is generated.
   i. Begin a capture session using EtherPeek, then start your server (if it is not still running).
   ii. Telnet to your server (port 9100):
      - ■ At the command prompt type: `telnet IP_Address 9100`
   iii. Interact with the server using text input:
      - ■ Do you notice anything different from a normal telnet session? Explain.
   iv. Stop Etherpeek and compare the captured session to the session using the client application.

2. The Simple Mail Transfer Protocol (SMTP - RFC 821), is another service that can be accessed directly with telnet using tcp port 25. This is the port that mail servers (such as the sendmail program on UNIX) uses to accept email messages.

   i. telnet to port 25 on any UNIX workstation, for example:

   ```
   telnet ouzo.ee.umanitoba.ca 25
   ```

   ii. To determine what commands are available enter help, or for a specific command enter:

   ```
   help command_name
   ```

   iii. Use the following commands to email yourself. Note that you can enter anything you wish as the sender information (this is one method used to send fake email):

   ```
   helo, mail, rcpt, data, and quit
   ```

   These commands should be used in the order given, and the help facility can be used to find out the proper syntax.

   iv. Send an email message to your TA, with subject c24422-lab2, from a phony email address. (remember to indicate who you are in the message to receive lab credit).

3. Mail access protocol (POP3) (RFC 1939) It is a very simple mail access protocol.
   - ○ POP3 usage
     - i. telnet smtp.cc.umanitoba.ca 110
     - ii. user *lab1* (temporary user for the lab)
     - iii. pass <*passord*> (ask the password from your TA)
     - iv. list: to see the messages
     - v. retr <*message number*> to retrieve the message
     - vi. dele <*message number*> to delete the message
     - vii. quit: to quit the session
   - ○ Use Ethepeek to filter the traffic of POP3 protocol.

4. Telnet can connect with other text-based services such as echo (port 7), and HTTP (port 80 WWW). Connect to port 80 on www.cc.umanitoba.ca (or any other web server you like). Once connected, enter the command:

   ```
   GET / HTTP/1.0
   ```

   and hit return twice. The returned information will be a mime encoded web page (HTML).

   - ○ Describe the application header received by your request.

5. Port scan tool
   Write a java program to identify the ports used on any machine.
   - ○ Try using Etherpeek to find out how port scan works.

RIP Protocol:

The Routing Information Protocol (RIP) is a simple routing algorithm that exchanges routing information on a regular basis using UDP broadcast packets.

6. Use EtherPeek to capture all UDP packets on the ethernet.
   - i. Set your filter to IP/ UDP and capture all packets until RIP packets are found. To be certain capture at least two full sets of RIP packets. 130.179.8.70
     - ■ What router(s) are on the network?
     - ■ What networks are in this routing domain?
   - ii. Once you find some different networks, try using the name scan tool in one of the IP toolkits to find hosts on a remote network in the umanitoba.ca domain.
   - iii. List the routing table for your current host, open a DOS shell, and use the command:

     ```
     netstat -r
     ```

     - ■ Does the local routing table use any information in the RIP packets?
   - iv. Use NeoTrace to find the path to the computer `ic18-atm.ee.umanitoba.ca`, and comment on the path. Compare it to just using `ic18`, or `ic18.ee.umanitoba.ca`.
     - ■ Speculate on why these paths occur.

DNS (Domain Name Service):

3. Mail access protocol (POP3) (RFC 1939) **It is a very simple mail access protocol.**
   ○ POP3 usage
      i. telnet smtp.cc.umanitoba.ca 110
      ii. user *lab1* (temporary user for the lab)
      iii. pass <*password*> (ask the password from your TA)
      iv. list: to see the messages
      v. retr <*message number*> to retrieve the message
      vi. dele <*message number*> to delete the message
      vii. quit: to quit the session
   ○ Use Etherpeek to filter the traffic of POP3 protocol.

4. Telnet can connect with other text-based services such as echo (port 7), and HTTP (port 80 WWW). Connect to port 80 on www.cc.umanitoba.ca (or any other web server you like). Once connected, enter the command:

```
GET / HTTP/1.0
```

and hit return twice. The returned information will be a mime encoded web page (HTML).

   ○ Describe the application header received by your request.

5. Port scan tool
   Write a java program to identify the ports used on any machine.
   ○ Try using Etherpeck to find out how port scan works.

RIP Protocol:

The Routing Information Protocol (RIP) is a simple routing algorithm that exchanges routing information on a regular basis using UDP broadcast packets.

6. Use EtherPeek to capture all UDP packets on the ethernet.
   i. Set your filter to IP/UDP and capture all packets until RIP packets are found. To be certain capture at least two full sets of RIP packets.
      ■ What router(s) are on the network?
      ■ What networks are in this routing domain?
   ii. Once you find some different networks, try using the name scan tool in one of the IP toolkits to find hosts on a remote network in the umanitoba.ca domain.
   iii. List the routing table for your current host, open a DOS shell, and use the command:

```
netstat -r
```

      ■ Does the local routing table use any information in the RIP packets?

   iv. Use NeoTrace to find the path to the computer `ic18-atm.ee.umanitoba.ca`, and comment on the path. Compare it to just using `ic18`, or `ic18.ee.umanitoba.ca`.

      ■ Speculate on why these paths occur.

DNS (Domain Name Service):

DNS is a required service that converts between IP addresses and IP names. DNS itself is hierarchical (like IP names) and at each level a DNS server is responsible to know the hosts and networks below itself, and the network(s) above itself.

7. To explore this service we will use the nslookup utility (for more details see the man pages).
   i. Open a DOS shell and type nslookup to automatically connect to the default DNS server:

```
Default Server: eeserv.ee.umanitoba.ca
Address: 130.179.8.1
>
```

   ii. Use it to find out about the computer ic18.ee.umanitoba.ca:

```
> ic18.ee.umanitoba.ca
Server: eeserv.ee.umanitoba.ca
Address: 130.179.8.1

Name: ic18.ee.umanitoba.ca
Address: 130.179.8.98
```

- Alternatively a IP address can be entered, and a hostname return

```
>130.179.8.98
```

   iii. List all computers in the ee.umnaitoba.ca domain

```
>ls ee.umanitoba.ca
```

- Try listing non local computer names (eg www.win.trlabs.ca).
- To list all computers in a domain served by another DNS server, we must first connect to that server. To connect to the ca domain server type:

```
>server 192.73.5.1
```

- We could now enter ls ca to see all names in that domain (it would be a large list).
- Find all of the computers in the mbnet domain, their server is:

```
access.mbnet.mb.ca
```

   iv. Try turning on the debug option to see more info about how the result is obtained:

 Set
>~~all~~ debug

nslookup will now return more info. about the different queries it makes to resolve a name.

Part 3: Questions:

Describe in terms of functionality the difference between a client and a server application. Extend this to describe the Socket vs. the ServerSocket classes in Java.

DNS is a required service that converts between IP addresses and IP names. DNS itself is hierarchical (like IP names) and at each level a DNS server is responsible to know the hosts and networks below itself, and the network(s) above itself.

7. To explore this service we will use the nslookup utility (for more details see the man pages).

   i. Open a DOS shell and type nslookup to automatically connect to the default DNS server:

```
Default Server: eeserv.ee.umanitoba.ca
Address: 130.179.8.1
>
```

   ii. Use it to find out about the computer ic18.ee.umanitoba.ca:

```
> ic18.ee.umanitoba.ca
Server: eeserv.ee.umanitoba.ca
Address: 130.179.8.1

Name: ic18.ee.umanitoba.ca
Address: 130.179.8.59
```

- ■ Alternatively a IP address can be entered, and a hostname return

```
>130.179.8.99
```

   iii. List all computers in the ee.umanitoba.ca domain

```
>ls ee.umanitoba.ca
```

- ■ Try listing non local computer names (eg www.win.trlabs.ca).
- ■ To list all computers in a domain served by another DNS server, we must first connect to that server. To connect to the ca domain server type:

```
>server 192.73.5.1
```

- ■ We could now enter ls ca to see all names in that domain (it would be a large list).
- ■ Find all of the computers in the mbnet domain, their server is:

```
access.mbnet.mb.ca
```

   iv. Try turning on the debug option to see more info about how the result is obtained:

```
>?set debug
```

nslookup will now return more info, about the different queries it makes to resolve a name.

Part 3: Questions:

Describe in terms of functionality the difference between a client and a server application. Extend this to describe the Socket vs. the ServerSocket classes in Java.

Can telnet communicate with any service? What, if any, are the requirements?
What is the purpose of using telnet to connect to other services?

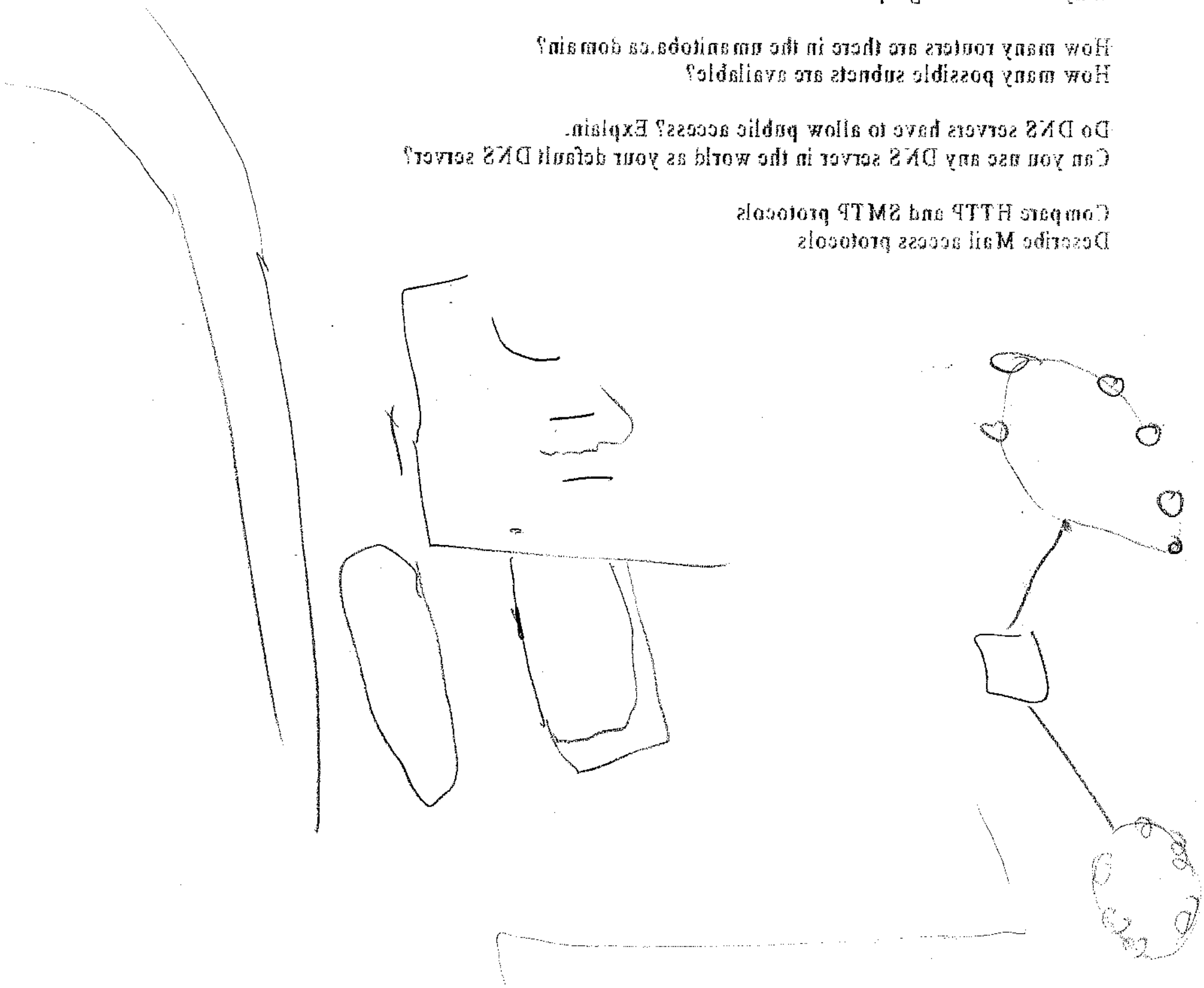
What is a port number?
Do TCP and UDP share the same port numbers? What about other protocols above IP?
Why is it important to have pre-defined port numbers for different services?
Why would running a port scan or name scan of a network be considered an attack?

How many routers are there in the umanitoba.ca domain?
How many possible subnets are available?

Do DNS servers have to allow public access? Explain.
Can you use any DNS server in the world as your default DNS server?

Compare HTTP and SMTP protocols
Describe Mail access protocols

redirect to text.



```
/* Simple Java Client.

- The simple client tries to establish a connection to a waiting server.

*/

import java.io.*;
import java.net.*;

class client {

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

            int         port = 9100;
            String      host = "localhost";
            String      message1 = "from server:\n";

            switch (args.length){
                  case 2:{
                        Integer portNum = new Integer(args[1]);
                        port = portNum.intValue();
                  }
                  case 1:{
                        host = args[0];
                  }
            }

            System.out.println("The client is connecting to port " + port + ", on " + host
+ ".\n");

            try{
                  Socket sock = new Socket(host,port);

            InputStreamReader inr = new InputStreamReader(sock.getInputStream());
            BufferedReader    in  = new BufferedReader(inr);
            String fromServer = in.readLine();

            System.out.println(message1);
            System.out.println("\t" + fromServer + "\n");

            PrintWriter out = new PrintWriter(sock.getOutputStream());
            out.println("Bond, James Bond");
            out.flush();

            System.out.println(message1);
            fromServer = in.readLine();
            System.out.println("\t" + fromServer);
            fromServer = in.readLine();
            System.out.println("\t" + fromServer);
            fromServer = in.readLine();
            System.out.println("\t" + fromServer);
            sock.close();

            System.out.println();
            System.out.println("Done\n");
            }
            catch (ConnectException e){
                  System.out.println("Could not establish the desired connection.");
                  System.out.println(e);
            }
      }
}
```

```
/* Simple Java Client.
- The simple client tries to establish a connection to a waiting server.
*/

import java.io.*;
import java.net.*;

class client {

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

        int         port = 1100;
        String      host = "localhost";
        String      message1 = "from server:\n";

        switch (args.length){
            case 2:{
                Integer portNum = new Integer(args[1]);
                port = portNum.intValue();
            }
            case 1:{
                host = args[0];
            }
        }

        System.out.println("The client is connecting to port " + port + ", on " + host
+ ".\n");

        try{
            Socket sock = new Socket(host,port);

        InputStreamReader inr = new InputStreamReader(sock.getInputStream());
        BufferedReader    in  = new BufferedReader(inr);
        String fromServer = in.readLine();

        System.out.println(message1);
        System.out.println("\t" + fromServer + "\n");

        PrintWriter out = new PrintWriter(sock.getOutputStream());
        out.println("Bond, James Bond");
        out.flush();

        System.out.println(message1);
        fromServer = in.readLine();
        System.out.println("\t" + fromServer);
        fromServer = in.readLine();
        System.out.println("\t" + fromServer);
        fromServer = in.readLine();
        System.out.println("\t" + fromServer);
        sock.close();

        System.out.println();
        System.out.println("Done\n");
            }
        catch (ConnectException e){
            System.out.println("Could not establish the desired connection.");
            System.out.println(e);
        }
    }
}
```

```
/* Simple Java Server.

- The simple server uses the ServerSocket class to listen for a client
and then create a socket connection to that client (with the Socket class).
- Notice the usage of two objects called 'out'. The first is qualified
with the reference to the System class. The second is locally defined and
used directly as just 'out'.

*/

import java.io.*;
import java.net.*;

class server {

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

                int             queue = 20;
                int             port  = 9100;
                String          query = "Hello. What is your name?";
                Socket          sock;

                if (args.length > 0){
                        Integer portNum = new Integer(args[0]);
                        port = portNum.intValue();
                }

                try{
                        ServerSocket servSock = new ServerSocket(port,queue);

                        System.out.println("The server is using port " + port + ".");

                        while (true){
                                sock = servSock.accept();
                                InputStreamReader inr = new
InputStreamReader(sock.getInputStream());
                                BufferedReader    in  = new BufferedReader(inr);

                                PrintWriter out = new PrintWriter(sock.getOutputStream());
                                out.println(query);
                                out.flush();

                                String reply = in.readLine();

                                out.println("Hello " + reply + ",");
                                out.println("It is nice to meet you.");
                                out.println("Let's talk again soon.\n");
                                out.flush();

                                sock.close();
                        }
                }
                catch (BindException e){
                        System.out.println("Could not establish server on port" + port + "./n");
                }

}
}
```

Mail FROM: < from address >

rcpt2 TO: < to address >

data

3.8/4

24.370 Telecommunication Networking - Lab 2

by: Cory Jeffrey (6725759)

- **Describe in terms of functionality the difference between a client and a server application?**

The client application initiates the connection with a server application already running. The server application, if the request is valid, will create the connection. After the connection is made, the server responds to commands or requests issued by the client.

- **Extend this to describe the Socket vs. ServerSocket classes in Java?**

Socket classes in Java are "client classes" in that they are responsible for initiating any connection with another application (or another instance of a ServerSocket), and for passing any information to the server. A ServerSocket is thus responsible for accepting connections with Sockets, as well as the transfer of information through to the client socket(s).

- **Can Telnet communicate with any service? If so, what are the requirements?**

While Telnet is able to connect to a server application running on any port of a machine, the format used in communicating may not be one that Telnet can handle. For example, a server must use ASCII text, and be running a compatible terminal type for Telnet's output to be intelligible.

- **What is the purpose of using Telnet to connect to other services?**

Telnet is used for low-level manipulation of a service, and offers more information about the actions being performed than is normally available through client applications. Telnet can also be used for more discrete purposes, such as sending anonymous e-mails to lab TAs. Of its more legitimate purposes, it is also a easy, high-speed & commonly used terminal program.

interactive

- **What is a port number?**

A port number is an unsigned 16-bit integer, each of which can represent a service running on a machine. The port number is included in each packet to Port numbers are unique to a service, in that they cannot be used by multiple services at the same time (not that they can never be changed for a particular service).

- **Do TCP and UDP share the same port numbers? What about other protocols above IP?**

Yes TCP and UDP use the same port numbers for the same services, however they are unable to intercommunicate. For example, even though HTTP (which is running above IP) is running on port 80 on both TCP & UDP, one cannot use a UDP web browser to access a TCP web server. Thus, any protocol above IP must be classified as either TCP & UDP, and will share the same port number for ~~both base~~ protocols.

not really !

- **Why is it important to have pre-defined port numbers for different services?**

Pre-defined port numbers are extremely important when communicating between two machines to ensure that any packet you are sending out (for whatever service) reaches the proper port on the other machine. Without pre-defined ports, one would have to send out a port-request to the other machine to determine the proper destination port. In this case however, we would still have to have a pre-defined constant to send along with the port-request packet.

- **Why would running a port scan or name scan of a network be considered an attack?**

Name scanning on a network can be considered an attack as it offers a means of discovering valid

hostnames/IPs of machines running on the network. A portscan can then be run on these machines, allowing the "attacker" to discover the services being run on that particular machine. An attack (eg: packet flooding) can then occur on an open port of a machine. ✓

- **How many routers are there in the umanitoba.ca domain?** 83

  There is one router in the umanitoba.ca domain, enrouter.cc.umanitoba.ca. X

- **How many possible subnets are available?** 16384 possible in a class B or 64

  active

  There are 68 possible subnets, as listed in the routing table broadcast by enrouter.cc.umanitoba.ca. x $\frac{1}{2}$

- **Do DNS servers have to allow public access? Explain.**

  No, they don't have to allow public access. Access can be restricted to a certain subnet or set of subnets. However, if one wishes the "outside world" to be able to resolve local names to IP addresses, then the DNS server must be public. This is because other DNS servers will contact it to see if it has the information they want, and if it doesn't give it to them, no one will be able to resolve the name. ✓

- **Can you use any DNS server in the world as your default DNS server?**

  As long as the IP address of the DNS server is known, it isn't behind a firewall, and it is public, it should work as a default DNS server. Choosing a distant DNS server will likely result in poor performance as it will not have many (if any) of your local hostnames. ✓

- **Compare HTTP and SMTP protocols & describe mail access protocols**

  SMTP is a protocol for sending e-mail between mail servers, as well as sending e-mail from an e-mail client to a mail server. HTTP is a protocol used by the WWW to define how file transfers take place, and how servers respond to various commands. The two protocols really have very few similarities at all, other than the fact that they are both above IP and are both text-based. ✓

  Mail access protocols are protocols used by mail clients and servers to exchange e-mail (possibly in an encrypted format), or for server-server communications. There are simply too many protocols (POP, IMAP, SMTP, etc) to discuss all of them (after all, that's what RFCs are for!). so lazy! X $\frac{1}{2}$

24.370 Telecommunications Networks

Lab 2: Client/Server, Telnet, RIP, and DNS:

$\frac{3.4}{4}$

Cameron Melvin
6721510
Tuesday, February 8, 2000

BAD FONT!

Lab 2 Cameron Melvin

Part 3 Questions

Describe in terms of functionality the difference between a client and a server application.

The purpose of a server application is to supply a connection to clients who request it. The client initiates the connection with a request, and the server just waits for requests and responds. ✓

Extend this to describe the Socket vs. the ServerSocket classes in Java.

The Socket class is designed to initiate connections, whereas the ServerSocket class is designed to receive requests for connections. The Socket class, then is used for client apps and the ServerSocket class is used for server apps. ✓

Can telnet communicate with any service? What, if any, are the requirements?

Telnet can only communicate with services that have telnet servers. The user can specify any service on any port, but telnet will only be able to communicate successfully if the service is telnet enabled. X

What is the purpose of using telnet to connect to other services?

Telnet gives users quick, simple, text-based access to any (text) information on the server end. 1/2

What is a port number?

service

A port number is a 16 bit integer that identifies which process on the server machine is being requested. Each service has an port number that it uses. Eg HTTP port # is 80 ✓

Do TCP and UDP share the same port numbers? What about other protocols above IP? X TCP, UDP

TCP and UDP are higher level protocols which utilize the lower level protocol IP. IP has 2^{16} port numbers, each with a TCP and UDP stack. All protocols on the levels above TCP and UDP must use TCP and UDP, each of which generally have their own port numbers (eg HTTP, FTP, SMTP, etc).

Why is it important to have pre-defined port numbers for different services?

This is important so that applications know which port to use. For example, an FTP application must know which port to use when it connects to an FTP server. ✓

Why would running a port scan or name scan of a network be considered an attack?

Port scans are usually considered attacks because they tell the scanner which ports are open, and therefore which ports can be used. ✓

How many routers are there in the umanitoba.ca domain?

There is one router on the umanitoba.ca domain. X 83

How many possible subnets are available?

There are 68 possible subnets available. X 16384 or 64

Do DNS servers have to allow public access? Explain.

Yes, DNS servers have to allow public access. They must respond to all requests for domain name translation. They DO NOT allow public access to their tables. ? ✓

Can you use any DNS server in the world as your default DNS server?

You can use any DNS that you know of (have an IP address for). and if they allow you to. $\frac{1}{2}$

Compare HTTP and SMTP protocols

HTTP (Hyper Text Transfer Protocol) is designed for the transfer of HTML documents for web browsers, while SMTP (Simple Mail Transfer Protocol) is designed for sending and receiving e-mail messages. ✓

Describe Mail access protocols

POP/IMAP

The mail access protocols are POP and IMAP. Several versions of each protocol have been developed. POP (Post Office Protocol) and IMAP both ~~send~~ and receive messages to/from an SMTP server. The most recent version of the popular POP is POP3, which can function without a connection to the SMTP server. The most recent version of IMAP is IMAP4, which does require a connection to the SMTP server. X

130.179.9.166

Lab 3 - Network Programming and Web Server

In this lab we will investigate a more sophisticated web server from our introductory client / server application, and add multi-threading capability to it. To begin however we will examine two introductory programs using threads: a simple threads program, and a more interesting `chat' like program.

1. Multi-Threaded Programming examples:

On the course web page there are two example applications that run multiple threads. The first is a simple example with threads called:

thread_1.java

and a more interesting chat program (using three classes) will be examined next.

Save this program, and take a look at their source code. In particular thread_1.java and chat_server.java, as these instantiate the multiple threads. In general there are two methods for implementing threads in java. We are only looking at the first such method - creating a class that extends the Threads class. A new thread can be started by creating a new object of this extended class type and then using that object to invoke the start() method of the Threads class. This method will start a new thread and execute the code in the class method run(). You create a new run() method in the extended class, replacing the default method of the Threads class, and that acts like the main() method for a new thread.

Compile and run the thread_1 program (you may need to redirect the output of thread_1.java to a file for slower examination). To redirect output try:

```
java thread_1 > a.txt
```

The `>' sign indicates to redirect standard I/O to the specified file (a.txt).

Review this file, and determine if the two threads ran separately, or linearly, in the program.

They ran seperately sharing CPU time. It does not seem like the CPU is dividing its time evenly as they do not interupts each other in any sort of pattern

2. Asynchronous Communication

In the client server architecture used by the WWW, the client first sends a request, and the server responds. Only one process is writing the socket at any instance. Often there is no agreed upon protocol for who will have access to write to a socket. In this case, a two threaded program is created in which one threads reads from the socket, and the other thread writes to a socket.
A simple chat server program is provided:

chat_server.java
ChatListner.java

ChatTalker.java

For the chat program run the server, and then telnet into the server. To get the server to display a line, the program actually waits for the current line to be entered. Play with this, and try to explain why this occurs. What happens with programs like talk and IRC? How do they get around this problem?

These programs get around this problem by reading in a character at a time and not waiting for and end of line character to send the data

Modify the chat server program to implement a chat client. What happens now?

3. Simple Web Server

The World Wide Web (WWW) is based on a simple client server architecture. A web browser is the client, and a web server is the server. The communication protocol used is http (hyper text transfer protocol). Information is encoded in HTML (hypertext markup language).

Details about HTTP can be found in RFC 1945. In addition the behaviour of the http can be observed by direct observation of web browsers and servers.

A very simple web server written in Java is available in the labs directory:

- WebServer.java
- RequestHandler.java
- SendLocalFile.java

The WebServer is the multi-threaded web server. It uses the class RequestHandler to manage the requests from the clients. The RequestHandler uses SendLocalFile class to send the file to the client. This web runs on port 9100 by default, although this value can be overridden on the command line.

Save, view, compile and run the web server. A port number may be specified on command line

```
java WebServer 9100
```

Use a web browser (Netscape or Internet Explorer) to talk to the web server. Use URL:

```
http://your_computer_name:9100
```

The Web server provides text files. Modify the RequestHandler to view image files too. Your RequestHandler should be able to sense the type of the file. Try to view different files (image and text).

Add Comments in the java files and use javadoc to generate the API for your web server.

Use EtherPeek to capture the packets of your HTTP protocol session. Analyse the packets and briefly describe the HTTP protocol. WebServer.txt

Explain the purpose of if-Modfied-Since tag

```
if (url. toUpperCase(). indexOf(".JPG")!=-1
    || url.toUpperCase(). indexOf(".JPEG")!=-1)
    outClientTextStream.println("Content.type:
                                  image/jpeg");
else
  if (url.toUpperCase(). indexOf(".GIF")=-1)
        outClientTextStream.println("Content.type:
                                     image/gif");
else
    outClientTextStream.println("Content.type:
                                 text/html");
```

Laboratory #4 - Encryption

RSA in Java:

1. Use the example code for RSA available on the course web page (rsa.java) to investigate the operation of the RSA algorithm for data encryption.

- You should first become familiar with the **BigInteger class library**. This will be very important to the remainder of the lab.

How does it work?
RSA works on the basis of the difficulty of factoring very large numbers. Enough information is made public to allow a message to be encrypted in such a way that it may later be decrypted using the rest of the information.

For example, if Bob wishes to send a message to Alice:

- Alice first generates two large prime numbers p and q - 100 digits minimum.
- Alice creates a new large number n = p X q.
- Alice chooses a private key d such that d is relatively prime to PHI(n) = (p-1)(q-1).
- Alice then calculates a public key e such that e X d = 1 mod PHI(n) => $e = d^{-1}$ mod PHI(n). that is: $e = d^{-1}$ mod (p-1)(q-1).
- Alice sends to Bob (and anyone inbetween) the values n and e. Thus e is called the public key.
- Bob can now encrypt a message M for Alice: $C = M^e$ mod n.
- Bob sends C to Alice (which again anyone can see).
- Only Alice can decrypt the cipher message C with her private key: $M = C^d$ mod n.

2. Build a proper stand-alone RSA class in java. The class should contain at least the required data and methods to initialize an object of type RSA, which would include passing a key size to the RSA constructor, and constructing the required data - public and private keys as well as `n'. In addition methods will be required to encrypt and decrypt a text string (String), as well as to return the values of the data arguments for an RSA object (BigInteger).

- You will need to set the values if they are passed to you as public information.
- Do not forget the relationship between the string size that can be encoded and `n'. To be safe ensure that any text string is divided into segments no larger than the key size before encryption. This implies that the encryption method should return an array of BigIntegers.
- Consequently, decryption should accept an array of BigIntegers and rebuild the string if it was segmented.
- If too difficult, start by assuming the message is always smaller than `n'. This can be modified later, time permitting. Under what conditions would this be a valid assumption to make?

Secure Client/Server:

3. Modify the client/server programs from lab 2 to incorporate RSA encryption using your new RSA class. The encryption should be used to request and transfer the username and password over the

network.

Both the client and server will create an RSA object, and initialize it to a given key size.

When the client first makes contact with the server, the server will send its public key to the client. The client will use the server's public key to both encrypt all messages before sending them to the server as well as to decode all messages from the server.

Similarly the server will encrypt and decrypt all messages with the servers private key.

In your programs, measure the time (use the java.Util.Date class) required to calculate keys, and to encrypt, and decrypt messages for different size keys: 64, 128, 256, 512, 1024, and 2048 bits. Estimate the maximum data transfer rates for the different key lengths based on these times, neglecting any networking overhead.

- Run the client/server on different machines.
- Observe the client/server communication with Etherpeek.

Questions:

- In RSA two keys are calculated. Does it matter which key is made public and which is kept private? If so, why do you think the second key is chosen as the public key?

- In your RSA class which data items did you choose to include? What methods did you create? Which were private? Were any public?
- After trying to use your RSA class did you need to make any changes? If so, what were they? Did you need any methods not specified, or did you find any that would be more useful - such as an over-ridden constructor that accepts a public key and `n', or perhaps an over-ridden encryption method which accepts a public key and `n' as an argument?
- What is the relationship between the message size and the key size?

- Using Etherpeek to sniff the ethernet, what do you think about what you saw?
- What problem, if any, is there with the manner in which we implemented RSA in the client/server programs?
- How would you correct these problems?
- Can you think of any other problems with using RSA in a client/server architecture such as the one described in this lab.

- Is there a relationship between time and keysize? If so, what is it?
- Based on the approximations of the maximum rate of data transfer, do you think that it is feasible to use RSA to encrypt large amounts of data for either storage or transmission.

Note: Submit your RSA class, and modified client/server programs along with your lab on a floppy disk or as attached files via email.

24.370 Lab 4

Lukasz Filipecki

6713649

March 28, 2000

Questions for laboratory #4:

- In RSA two keys are calculated. Does it matter which key is made public and which is kept private? If so, why do you think the second key is chosen as the public key?

We can see that it does not matter which one of the keys is made public and which is kept private. I think that the second key is chosen as the public key simply due to convenience and also due to possible less complex computation.

- In your RSA class which data items did you choose to include? What methods did you create? Which were private? Were any public?

In our RSA class the data items that we did not include any data members. The methods that we created were the following and the description of whether they were private or public is listed below:
- method that created keys " n e d".
- method that encrypts a passed string from keys "n e", and returns an encrypted string as output.
- method that decrypts a passed string from keys "n d", and returns a decrypted string as output.

These outlined methods were all made public.

- After trying to use your RSA class did you need to make any changes? If so, what were they?

When we tried to use RSA class we were first made add code that actually worked. Thus debugging was added and correct code was generated. We added an over-ridden constructor that allowed the user to specify the key size as it became apparent that it would be needed.

Did you need any methods not specified, or did you find any that would be more useful - such as an over-ridden constructor that accepts a public key and `n', or perhaps an over-ridden encryption method which accepts a public key and `n' as an argument?
We did not need any methods not specified.

- What is the relationship between the message size and the key size?

The relationship between message size and the key size is one of the fact that the message size should be smaller than the key size.

- Using Etherpeek to sniff the ethernet, what do you think about what you saw?

Using Etherpeek to sniff Ethernet the main thing that we saw was the fact that we observed the public keys being transmitted, along with the messages that were broken up into packets.

- What problem, if any, is there with the manner in which we implemented RSA in the client/server programs?

The problem that I believe would come out of the way that we implemented the RSA in the client server program would be one of security. This would be evident when we are going from the server-to-client. The server encrypts the messages using its private key, but the messages are decrypted by the client using the public key that was sent out over the network. This is clearly a problem as anyone could have obtained this and then been able to decrypt the message.

- How would you correct these problems?

This problem could be resolved if both the client and the server generate keys. This way we could have the client send the public keys to the server for encryption, and thus the messages could only be decrypted by the client.

- Can you think of any other problems with using RSA in a client/server architecture such as the one described in this lab.

Actually to be honest I cannot see any other ways of the client/server architecture can be insecure unless some psycho were to use some crazy supercomputer and attack the client/server by brute force.

- Is there a relationship between time and keysize? If so, what is it?

There appears to be an exponential relationship between the time and the keysize.

- Based on the approximations of the maximum rate of data transfer, do you think that it is feasible to use RSA to encrypt large amounts of data for either storage or transmission.

Based on the approximations of the maximum rate of data transfer we can see that it would really not be practical to use RSA to encrypt larger amounts of data for either storage or transmission. Since there is a growing exponential relationship between the time and keysize we can see that if the files are large it would simply take too long for people to appreciate RSA. Thus it is feasible, however not practical for large files.

24.370
Lab 4 – Encryption

Jake Szot
6724814
March 28, 2000

24.370
Lab 4 – Encryption

Jake Szot
6724814

1. In RSA two keys are calculated. Does it matter which key is kept private?

Yes.

2. If so, why do you think that the second key is chosen as the public key?

The second key is released to the public because it is mathematically easier to obtain than the first key. That is, given n and D it is easier to obtain E than given n and E, trying to guess D.

3. In your RSA class which data items did you choose to include?

```
public BigInteger n, e;
private BigInteger p, q, phi, d, gcd;
public int keySize;
private BigInteger one = new BigInteger("1");
```

4. What methods did you create? Which were private?

```
private void createKeys()
private BigInteger encryptBigInteger(BigInteger i)
private BigInteger decryptBigInteger(BigInteger i)
```

5. Were any public?

```
public RSA()
public RSA(int keySize)
public void changeKeySize(int keySize)
private void createKeys()
public BigInteger encryptString(String str, BigInteger e, BigInteger n)
public BigInteger encryptString(String str)
public String decryptString(BigInteger i)
public BigInteger [] encryptStringArray(String str, BigInteger e, BigInteger n)
public String decryptString(BigInteger [] bigInt)
```

6. After trying to use your RSA class did you need to make any changes?

No.

7. Did you need any methods not specified.

Yes, see above.

8. What is the relationship between the message size and the key size?

The size of the message must be less than n.

9. Using Etherpeek to sniff the ethernet, what do you think about what you saw?

The data was indeed encrypted.

10. Can you think of any other problems with using RSA in a client/server architecture such as the one described in this lab?

No.

11. Is there a relationship between time and keysize?

Yes.

12. If so what is it?

Exponential.

13. Based on the approximations of the maximum rate of data transfer, do you think that it is feasible to use RSA to encrypt large amounts of data for either storage or transmission?

Yes.

Telecommunication
24.370
Lab 5

Raymon Hung
67129791

Question 1: What is the default port number used by the RMI registry?

The default port number used by the RMI registry is port number 1099.

Question 2: What information do you need to use a remote object with RMI?

The client must obtain a reference to the remote object, in terms of IP address or URL, and have a defined task to be performed by the computing engine.

Question 3: What security implications are there with remote objects and the server? the client? How does the JVM security manager address these issues?

Without any security manager, all downloaded code would execute and have access to the systems resources. Thus, untrusted code could be invoked on the server side. For the client, code could be downloaded from the RMI server to the client based on its workpool. This code could come from an untrusted source and contain methods that may harm the client's system. The RMISecurityManager determines whether the downloaded code has access to local file systems or perform operations. Otherwise, RMI will not download classes for objects received as parameters, return values, or exceptions in remote method calls.

Question 4: Why does the server program call the super() constructor? Specifically why must an RMI class need to have an overloaded constructor, even if it is just to call this constructor?

The server program calls the super() constructor for the no-argument constructor superclass, UnicastRemoteObject, to declare the exception RemoteException in its throws clause. This way, the Compute-Engine constructor must also declare that it can throw RemoteException in the event that, during construction, an attempt to export the object fails.

Question 5: How does CORBA manage remote objects? What differences / advantages are there over RMI?

Everything in the CORBA architecture depends on an Object Request Broker (ORB) running the protocol Internet Inter-ORB Protocol or IIOP. The ORB acts as a central Object Bus over which each CORBA object interacts transparently with other CORBA objects located either locally or remotely. Some of the pros and cons of CORBA over RMI are listed as follows:

PROS

Services can be written in many different languages, executed on many different platforms, and accessed by any language with an interface definition language (IDL) mapping.

With IDL, the interface is clearly separated from implementation, and developers can create different implementations based on the same interface.

CORBA supports primitive data types, and a wide range of data structures, as parameters

CORBA is ideally suited to use with legacy systems written in C++, Ada, Fortran, Cobol etc.

CONS

Describing services require the use of an IDL that must be learned.

IDL to language mapping tools create code stubs based on the interface. Thus, some tools may not integrate new changes with existing code.

CORBA does not support the transfer of objects, or code.

Question 6: Another resource for invoking remote methods is RPC - Remote Procedure Calls. Briefly explain this technology. Compare RPC to RMI.

RPC is a technique for constructing distributed, client-server based applications based on extending the notion of conventional, or local procedure calling. Thus, the called procedure need not exist in the same address space as the calling procedure. Describing RFC can be likened to a function call. Like a function call, when an RPC is made, the calling arguments are passed to the remote procedure and the caller waits for a response to be returned from the remote procedure. The client makes a procedure call that sends a request to the server and waits. The thread is blocked from processing until either a reply is received, or it times out. When the request arrives, the server calls a dispatch routine that performs the requested service, and sends the reply to the client. After the RPC call is completed, the client program continues. RPC and RMI are similar but the major difference is that RPC sends only procedure calls over the wire, with arguments either passed along or described in such a way that they can reconstructed at either end. RMI actually passes whole objects back and forth over the net and is better suited for a fully object-oriented distributed object model.

Question 7: What other remote object / method / procedure mechanisms exist for network computing?

There is another method called DCOM, otherwise known as 'COM on the wire', that supports remoting objects by running on a protocol called the Object Remote Procedure Call (ORPC). A DCOM server is a body of code that is capable of serving up objects of a particular type at runtime. Each DCOM server object can support multiple interfaces each representing a different behavior of the object. A DCOM client calls into the exposed methods of a DCOM server by acquiring a pointer to one of the server object's interfaces. The client object then starts calling the server object's exposed methods through the acquired interface pointer as if the server object resided in the client's address space. As specified by COM, a server object's memory layout conforms to the C++ vtable layout. Since the COM specification is at the binary level it allows DCOM server components to be written in diverse programming languages like C++, Java, Object Pascal (Delphi), Visual Basic and even COBOL. As long as a platform supports COM services, DCOM can be used on that platform. DCOM is now heavily used on the Windows platform. Companies like Software AG provide COM service implementations through their EntireX product for UNIX, Linux and mainframe platforms; Digital for the Open VMS platform and Microsoft for Windows and Solaris platforms.

24.370

Telecommunication Networks

Lab #5

Cory Jeffrey
6725759

Cory Jeffrey
6725759

Questions

- **What is the default port number used by the RMI registry?**

  The default port number used is 1099.

- **What information do you need, to use a remote object with RMI?**

  To use a remote object with RMI you must know the hostname, along with the available objects/methods.

- **What security implications are there with remote objects and the server? The client? How does the JVM security manager address these issues?**

  The most important security matters involving RMI and distributed computing include: object access (the server must only allow access to specific objects); communication integrity (communications musn't be intercepted or tampered with); client authentication (the client must prove its identity to the server so that the server can only perform tasks the client is authorized to perform); server authentication (the server must prove its identity to the client so that the client can trust the privacy of its information); delegation (the server must be able to authenticate as the client when dealing with 3rd party remote servers); resource abuse (the server must have a method available to limit resource abuse by over-zealous clients)

  The JVM security manager partially addresses these issues in that it guarantees that only allowed classes are loaded from trusted sources. When loading local classes (from CLASSPATH) a security manager isn't necessary. However, when attempting to run a class over the network, a security manager must be present or an exception is thrown. To guarantee the other security issues, one must use the Java RMI Security Extension from Sun or another related product.

- **Why does the server program call the super() constructor? Specifically, why must an RMI class need to have an overloaded constructor, even if it is just to call this constructor?**

  A remote object instance must be exported, which makes it available to accept incoming remote method requests, by listening for incoming calls. Since we are extending our server class from the java.rmi.server.UnicastRemoteObject, we needn't call super() ourselves, as it is called by default. By calling super (explicitly or implicitly), the UnicastRemoteObject constructor ensures that our object is exported. If we were to extend our server from another class, we would be required to explicitly call the UnicastRemoteObject.exportObject() or Activatable.exportObject() methods.

- **How does CORBA manage remote objects? What differences/advantages are there over RMI?**

CORBA(Common Object Request Broker Architecture) is an architecture standard for creating heterogeneous distributed systems, or language-independent systems. CORBA differs from RMI in that there are no direct client-server communications. All communications occur through an ORB (Object Request Broker), which, after intercepting the communications, is responsible for finding an object that can implement the request, pass it the parameters, invoke its method, and return the results. The client doesn't need to know the location of the object, what language it is in, or even the operating system it is running on. Communications between clients/servers and ORBs are handled using the General Inter-ORB Protocol (GIOP) or the Internet Inter-ORB Protocol (IIOP) (which RMI uses).

Though RMI and CORBA aren't really competing systems, CORBA does have its advantages. CORBA allows distributed systems to perform over any supported language on any supported operating system. This is a much larger system base than RMI has, due to its dependence on Java. However, RMI does have its advantages as well, including distributed garbage collection and full Java semantics. As CORBA API uses IDL, distributed systems must be developed using common semantics.

- **Another resource for invoking remote methods is RPC – Remote Procedure Calls. Briefly explain this technology. Compare RPC to RMI.**

RPC is virtually identical to RMI (as RMI development was based on RPC), however, as its name implies, RPC allows programmers to access remote procedures on a distributed system. As such, RPC applies only to procedure-oriented languages such as C, while RMI applies only to Java (which is object-oriented). As well, RPC uses a daemon and fixed ports for communication and data exchange.

RMI is essentially the Java RPC standard, and therefore is quite identical to RPC. The main disadvantage is it is available only to Java objects.

- **What other remote object/method/procedure mechanisms exist for network computing?**

  Other distributed network computing mechanisms include the lower-level Distributed Computing Environment (DCE), the basis for RPC, RMI, & CORBA, Remote Programming Agents, and DCOM. These agents work on a different principle that reduces the amount of network communications. The client first ships the full procedure, including the algorithm, to the server by means of an agent. Then, when the server has finished computing, it ships the result back to the client, again by means of an agent. Obviously, this can significantly reduce the amount of network communications required. However, there are no standards as of yet defined for Remote Programming Agents. As such, there are no existing libraries and code must therefore be written at a comparatively lower level. Examples of commonly used Remote Programming Agents are Voyager & Aglets. Another product, DCOM, is quite similar to CORBA, but is a competing version presented by Microsoft.

Program Code

Program code for Part 3: Parallel Distributed Programming, can be found in Appendices A-C.

Appendix A: AddServer.java

```
import java.net.*;
import java.rmi.*;
import java.rmi.server.*;

public class AddServer extends UnicastRemoteObject implements AddServerInt {

    String hostname;

    AddServer() throws RemoteException {

        // This calls the constructor for the parent class.
        super();

        // Get the host name of the current computer.
        try {
            hostname = InetAddress.getLocalHost().getHostName();
        }
        catch (java.net.UnknownHostException e) {
            System.out.println("ERROR 5: " + e);
            System.exit(0);
        }
    }


    public int add(int a[]) throws RemoteException {

                int sum = 0;
                for (int i = 0; i < 100; i++)
                        sum += a[i];
      return sum;
    }

    public static void main( String args[]) {

        // Make a copy of this object, and pass it to the RMI registry.
        try {
            AddServer server_obj = new AddServer();
            Naming.rebind("AddServer",server_obj);
            System.out.println("The AddServer object is ready.");
        }
        catch (java.net.MalformedURLException e) {
            System.out.println("ERROR 1: " + e);
            System.exit(0);
        }
        catch (java.rmi.UnknownHostException e) {
            System.out.println("ERROR 2: " + e);
            System.exit(0);
        }
        catch (java.rmi.RemoteException e) {
            System.out.println("ERROR 3: " + e);
            System.exit(0);
        }
        catch (Exception e) {
            System.out.println("ERROR 4: " + e);
            System.exit(0);
        }
    }
}
```

Appendix B: AddServerInt.java

```
import java.rmi.*;

public interface AddServerInt extends Remote {
     public int add(int a[]) throws RemoteException;
}
```

Appendix C: AddClient.java

```
import java.rmi.*;

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

        String shost = "localhost";
        if (args.length > 0) {
            shost = args[0];
        }

        // Get remote object from shost, with name RMIserver.
        // The remote object must be initialized outside the try ststement.
        Remote remoteObject = null;
                  Remote remoteObject2 = null;
        try {
            remoteObject = Naming.lookup("rmi://" + shost + "/AddServer");
                                remoteObject2 = Naming.lookup("rmi://ece86ws/AddServer");
        }
        catch (java.net.MalformedURLException e) {
            System.out.println("ERROR 1: " + e);
            System.exit(0);
        }
        catch (java.rmi.UnknownHostException e) {
            System.out.println("ERROR 2: " + e);
            System.exit(0);
        }
        catch (java.rmi.NotBoundException e) {
            System.out.println("ERROR 3: " + e);
            System.exit(0);
        }
        catch (java.rmi.RemoteException e) {
            System.out.println("ERROR 4: " + e);
            System.exit(0);
        }
        catch (Exception e) {
            System.out.println("ERROR 5: " + e);
            System.exit(0);
        }

        AddServerInt AddServerIntStub,AddServerIntStub2;
        if (remoteObject instanceof AddServerInt) {
            AddServerIntStub = (AddServerInt) remoteObject;
                                AddServerIntStub2 = (AddServerInt) remoteObject2;
                                int a[] = new int[100];
                                int b[] = new int[100];

                                for (int i = 0; i < 100; i++)
                                {
                                        a[i] = 3*i;
                                        b[i] = 6*i;
                                }
            try {
                                        System.out.print("Array to be summed: {");
                                        for (int i = 0; i < 100; i++)
                                                System.out.print(a[i]+ ",");
                                        for (int i = 0; i < 100; i++)
                                                System.out.print(b[i]+ ",");
                                        System.out.println("}");
                                        int sum = 0;
                                        sum += AddServerIntStub.add(a);
                                        sum += AddServerIntStub2.add(b);
                                        System.out.println("The sum is: " + sum);
            }
            catch (java.rmi.RemoteException e) {
                System.out.println("ERROR 6: " + e);
                System.exit(0);
            }
            catch (Exception e) {
                System.out.println("ERROR 7: " + e);
                System.exit(0);
            }
        } else {
            System.out.println("ERROR 8: invalid remote object");
            System.exit(0);
        }
    }

}
```

24.370 Telecom
Lab 5: Distributed Objects

Jake Szot
6724814
April 4, 2000

24.370 Telecom
Lab 5: Distributed Objects

Jake Szot
6724814

1. What is the default port number used by the RMI registry?

1099.

2. What information do you need to use a remote object with RMI?

You need to know the names of the remote host (url) and the object (methods).

3. What security implications are there with remote objects and the server?

A client now has the ability to execute a method on the machine that is running the RMI server. The client does not need to authenticate itself by logging in. A malicious client could obtain sensitive information or execute compromising methods.

4. What security implications are there with remote objects and the client?

The RMI server has the ability to execute methods on the client. A malicious server could execute compromising methods on the client. The client could be installed by way of a trojan horse.

5. How does the JVM security manager address these issues?

The security manager is a virtual machine. This means that one does not actually have access to the machine itself so that the virtual machine can place restrictions on what is done. It ensures that only trusted hosts are allowed to instantiate classes.

6. Why does the server program call the super() constructor.

The super constructor allows the parent class to be called.

7. Specifically, why must an RMI class need to have an overloaded constructor, even if it is just to call this constructor?

The overloaded constructor exports the UnicastRemoteObject that sets up the RMI object.

8. How does Corba manage remote objects?

Corba (Common Object Request Broker Architecture) utilizes an interface called an General Inter-ORB Protocol (GIOP). Cobra's GIOP utilizes stubs and skeletons in a similar manner to Java. GIOP transfers calls and receives results using TCP. Sun has provided a method to map GIOP (IIOP) to RMI.

9. What advantages are there over RMI?

Cobra is language independent. This allows development to be done in other languages such as C++ instead of restricting it to Java. In addition, Corba does not use direct client/server communication.

10. Briefly explain RPC (Remote Procedure Call).

RPC is a protocol that permits a client to request a service from a server. The compiling of RPC statements results in a stub that represents the remote procedure. When the procedure is called the stub sends the commands to the client at runtime. In a similar manner, the client utilizes a stub that is responsible for interfacing with the client at runtime. [1]

11. Compare RPC to RMI

RPC and RMI are very similar. RMI is the java version of RPC with the ability to pass objects. Both RMI and RPC utilize the transport layer.

12. What other remote object/method/procedure mechanisms exist for network computing?

Microsoft has released an effort called the Distributed Component Object Model (DCOM)

References

[1] http://www.whatis.com/rpc.htm

24.370

Lab 5 - Distributed Objects

Marc Desjardine
6705906
April 2000

Lab Exercises

For the lab we designed a server and client based on RMI. Together these programs created a parallel machine that computed the maximum and minimum of an array of numbers. The servers registered a RMI method that calculated the min and max of an array, and the client broke up a large array into pieces that the servers can handle in parallel. The code for the client and server is attached at the back of this report.

Questions

What is the default port number used by the RMI registry?

Port 1099

What information do you need to use a remote object with RMI?

You need the remote host name and the object name that you want to use.

What security implications are there with remote objects and the server? the client? How does the JVM security manager address these issues?

The implications are that a non-trusted client can run methods on a machine running a RMI server thus by-passing the requirements of logging in and such. If this client runs any methods that access sensitive data or execute dangerous commands, this can be a problem. The client really doesn't have much of an issue with security except when the RMI server methods can be dangerously executed back on the client machine. The JVM addresses this issue by using a security manager that runs and watches and makes sure no bad commands are executed.

Why does the server program call the super() constructor? Specifically, why must an RMI class need to have an overloaded constructor, even if it is just to call this constructor?

The server program calls the super() constructor so that the parent class constructor gets called. The parent class is UnicastRemoteObject. This constructor actually creates and exports a new remote object on an anonymous port.

How does CORBA manage remote objects? What differences/advantages are there over RMI?

CORBA uses an Object Request Broker (ORB) to manage the interface between the systems. It provides mechanisms for clients to find objects based either on Name or by Trade (properties). When an object is requested, it is the ORB's duty to find the object. Stubs and skeletons are also used in CORBA just like JAVA. CORBA is improved over RMI because it has support for lifecycle management, security, transactions, and event notification. As well, CORBA allows the communicating systems to be programming language independent, so that one program could be written in C/C++ and another in JAVA.

Another resource for invoking remote methods is RPC - Remote Procedure Calls. Briefly explain this technology. Compare RPC to RMI.

RPC is a specification that allows calls to be made in HTTP using XML. It allows heterogeneous programming languages and operating systems to communicate. It uses stubs to compile the RPC into the client program. The client must specify the name of the service to be invoked across the network, the order and types of the parameters to be sent in the procedure call, and the expected type of the return value. The server must specify the name of the service it is providing, the order and types of parameters expected, and the type of the return value. The server registers with a location server so that the clients may use the service. RMI and RPC are similar except that RPC runs with HTTP and XML where as RMI is TCP based without another protocol above it.

What other remote object/method/procedure mechanisms exist for network computing?

The server program calls the super() constructor so that the parent class constructor gets called. The parent class is UnicastRemoteObject. This constructor actually creates and exports a new remote object on an anonymous port.

How does CORBA manage remote objects? What differences/advantages are there over RMI?

CORBA uses an Object Request Broker (ORB) to manage the interface between the systems. It provides mechanisms for clients to find objects based either on Name or by Trade (properties). When an object is requested, it is the ORB's duty to find the object. Stubs and skeletons are also used in CORBA just like JAVA. CORBA is improved over RMI because it has support for lifecycle management, security, transactions, and event notification. As well, CORBA allows the communicating systems to be programming language independent, so that one program could be written in C/C++ and another in JAVA.

Another resource for invoking remote methods is RPC - Remote Procedure Calls. Briefly explain this technology. Compare RPC to RMI.

RPC is a specification that allows calls to be made in HTTP using XML. It allows heterogeneous programming languages and operating systems to communicate. It uses stubs to compile the RPC into the client program. The client must specify the name of the service to be invoked across the network, the order and types of the parameters to be sent in the procedure call, and the expected type of the return value. The server must specify the name of the service it is providing, the order and types of parameters expected, and the type of the return value. The server registers with a location server so that the clients may use the service. RMI and RPC are similar except that RPC runs with HTTP and XML where as RMI is TCP based without another protocol above it.

What other remote object/method/procedure mechanisms exist for network computing?

There is software libraries called MPI that can be used for network programming. There is also another specification called DCOM which is a Microsoft strategy.

Source code for parallel max, min program

```
/** The RMIserver class:

        This class illustrates all of the components necessary to set up
an object for use as a remote object.
        When the server is run, it creates a new RMIserver object, and
initiallizes it with the constructor. The constructor first calls the
constructor of its parent class (UnicastRemoteObject) and then performs
its own initiallization which is to determine the name of the host it is
running on and assign this to the object data hostname.
        Next it uses an RMI method to bind together this object and a
reference name - RMIserver - in the registry, so that it can be accessed
remotely.

**/

import java.net.*;
import java.rmi.*;
import java.rmi.server.*;

public class RMIserver extends UnicastRemoteObject implements GreetingServer {

        String hostname;


        RMIserver() throws RemoteException {

                // This calls the constructor for the parent class.
                super();

                // Get the host name of the current computer.
                try {
                        hostname = InetAddress.getLocalHost().getHostName();
                }
                catch (java.net.UnknownHostException e) {
                        System.out.println("ERROR 5: " + e);
                        System.exit(0);
                }
        }

        public int[] FindMaxMin(int[] inArray, int arrayLength, int offset) throws RemoteException {
                int curMax = -16000;
                int curMin = 16000;

                int[] ret = new int[2];

                int i;



                for (i = 0; i < arrayLength; i++) {

                        if (inArray[i+offset] > curMax) {

                                curMax = inArray[i+offset];

                                }

                        if (inArray[i+offset] < curMin) {
```

```
                                curMin = inArray[i+offset];

                                }

                        }



                ret[0] = curMax;

                ret[1] = curMin;

                return ret;

                }

        public static void main( String args[]) {

                // Make a copy of this object, and pass it to the RMI registry.
                try {
                        RMIserver server_obj = new RMIserver();
                        Naming.rebind("RMIserver",server_obj);
                        System.out.println("The RMIserver object is ready.");
                }
                catch (java.net.MalformedURLException e) {
                        System.out.println("ERROR 1: " + e);
                        System.exit(0);
                }
                catch (java.rmi.UnknownHostException e) {
                        System.out.println("ERROR 2: " + e);
                        System.exit(0);
                }
                catch (java.rmi.RemoteException e) {
                        System.out.println("ERROR 3: " + e);
                        System.exit(0);
                }
                catch (Exception e) {
                        System.out.println("ERROR 4: " + e);
                        System.exit(0);
                }
        }
}

/** The RMIclient class:

        This class illustrates all of the components necessary instantiate
and use a remote object.
        The first segment of code should be familliar, setting the name of
the (remote) host on which the remote object resides.
        Next we try to create a "stub" - the actual instance used to
communicate over the network with the remote server. To do this we begin
by creating a generic Remote object, and use an RMI method to try and
lookup a class called RMIserver on the remote host. The RMI registry will
return an interface class, if one exists, and this is then compared to the
GreetingServer class.
        If the returned interface is the correct one, then we try to use
it by simply invoking a method defined in the interface - hello().
        This program also performs a variety of exception handling, the final
catch statement used for any unspecified exceptions. While they are
illustrated, we do not use them in any way, and are not really doing any more
than had we just used the global "Exception" as in the last catch. However
they do illustrate several (but not all) examples of specific problems that
could occur, and could be dealt with in ways other then quiting (or crashing)
the program.

**/

import java.util.*;
import java.rmi.*;

public class RMIclient {
```

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

                String shost = "localhost";
                if (args.length < 0) {
                        System.out.println("Not enough parameters!!\n");
                        System.exit(0);
                        }

                // Get remote object from shost, with name RMIserver
                try {
                        Random rand = new Random();
                        GreetingServer[] GreetingServerStub = new GreetingServer[10];
                        int[] MaxResult = new int[10];
                        int[] MinResult = new int[10];
                        int i;

                int[] returnVal = new int[2];
                        int[] inArray = new int[1000];

                        int curMax = -16000;
                        int curMin = 16000;
                        int offset = 0;

                        for (i = 0; i<1000; i++ )
                        {
                                inArray[i] = rand.nextInt(500);
                        }
                        for (i = 0; i < args.length; i++) {
                                GreetingServerStub[i] =
(GreetingServer)Naming.lookup("rmi://" + args[i] + "/RMIserver");
                                returnVal = GreetingServerStub[i].FindMaxMin(inArray,
1000/args.length, offset);
                                offset += 1000/args.length;
                                MaxResult[i] = returnVal[0];
                                MinResult[i] = returnVal[1];
                                System.out.println("Recieved Max of " + MaxResult[i] + "
and Min of " + MinResult[i] + " from " + i);
                        }

                        for (i = 0; i < args.length; i++) {
                                if (MaxResult[i] > curMax) {
                                        curMax = MaxResult[i];
                                }
                                if (MinResult[i] < curMin) {
                                        curMin = MinResult[i];
                                }
                        }
                        System.out.println("The Maximum Value Is " + curMax);
                        System.out.println("The Minimum Value Is " + curMin);
                }
                catch (java.net.MalformedURLException e) {
                        System.out.println("ERROR 1: " + e);
                        System.exit(0);
                }
                catch (java.rmi.UnknownHostException e) {
                        System.out.println("ERROR 2: " + e);
                        System.exit(0);
                }
                catch (java.rmi.NotBoundException e) {
                        System.out.println("ERROR 3: " + e);
                        System.exit(0);
                }
                catch (java.rmi.RemoteException e) {
                        System.out.println("ERROR 4: " + e);
                        System.exit(0);
                }
                catch (Exception e) {
                        System.out.println("ERROR 5: " + e);
                        System.exit(0);
                }
        }
```