CS 580 Client-Server Programming
Spring Semester, 2010
Doc 21 Distributed Objects & RMI
April 27, 2009

References

Pattern-Oriented Software: A System of Patterns, Buschman, Meunier, Rohnert, Sommerlad, Stal, 1996, pp. 323-338 (Client-Dispacther-Server), pp. 263-275 (Proxy)

Design Patterns: Elements of Reusable Object-Oriented Software, Gamma, Helm, Johnson, Vlissides, 1995, pp. 207-218 (Proxy)

CS 696 Emerging Technologies: Java Distributed Computing, Doc 5 Distributed Object Basics, 1999, http://www.eli.sdsu.edu/courses/spring99/cs696/notes/distObj/distObj.html, Doc 6 RMI Intro, Doc 7 Java Security Model

Permissions in the Java SE Development Kit, http://java.sun.com/javase/6/docs/technotes/guides/security/permissions.html

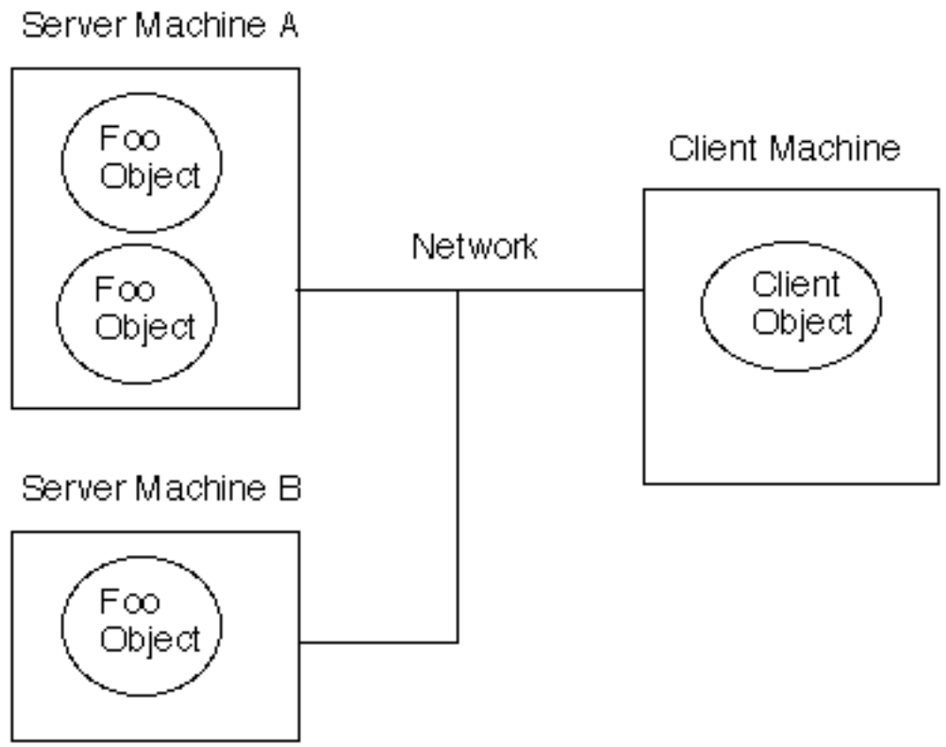
Default Policy Implementation and Policy File Syntax, http://java.sun.com/javase/6/docs/technotes/guides/security/PolicyFiles.html

Distrbuted Objects, http://en.wikipedia.org/wiki/Distributed_object

Java Network Programming 3'rd Ed, Harold, O'Reilly, 2005, Chapter18 Remote Method Invocation

Distributed Objects

System that allows sending of messags to objects on remote machines




```
public class Foo {
   public String hello() { return "Hi there"; }
}
```

```
Foo remote = getRemoteObject();
String message = remote.hello();
```

Some Existing Systems

Java RMI
CORBA
DCOM
Pyro (Python)
dRuby
ReplicaNet (C++)

Local verses Distributed Objects

Life Cycle
References
Request Latency
Parallelism
Communication
Failure
Security

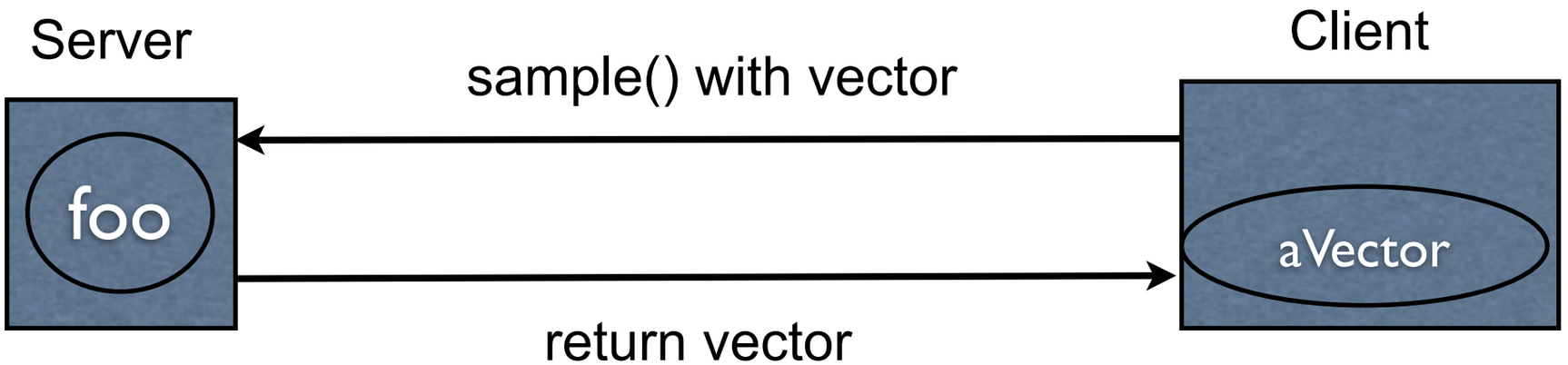
Example (Pseudo Code)

Server

```
public class Foo {
   public Vector sample( Vector input ) {
      input.add( "Hello" );
      return input;
   }
}
```

Client

```
Foo remote = getRemoteObject();
Vector local = new Vector();
Vector b = remote.sample(local);
String message = b.get(0);
```

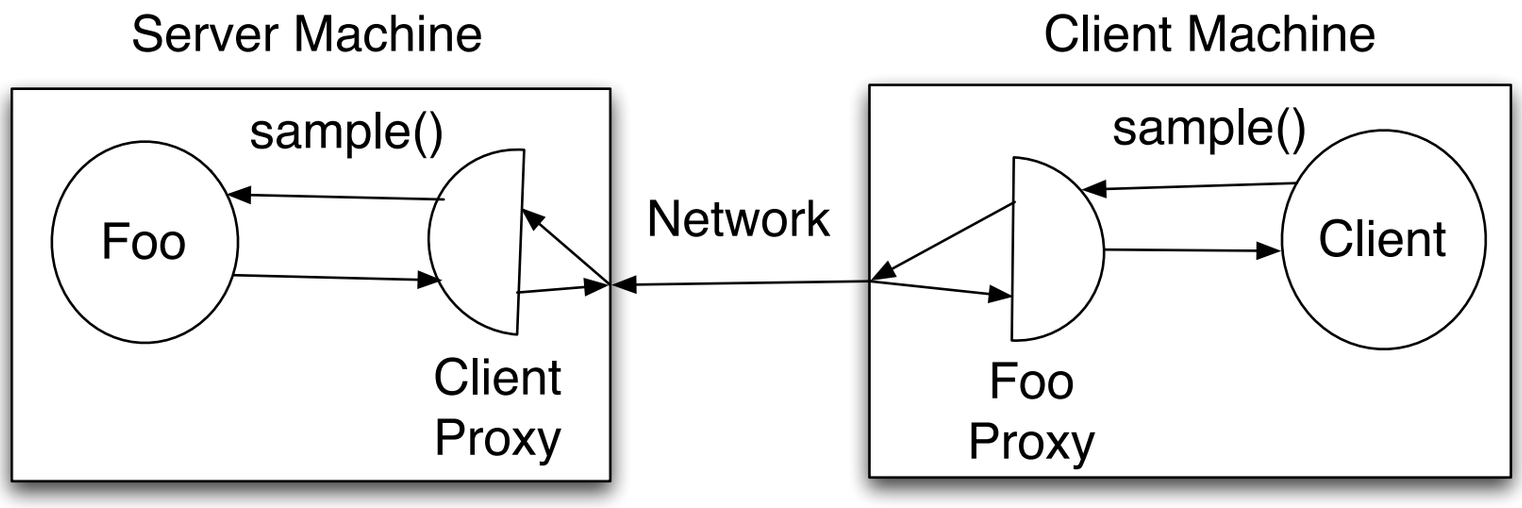


Remote Proxy

The actual object is on a remote machine (remote address space)

Hide real details of accessing the object

Used in CORBA, Java RMI



Issues

How does the client find the server?
Network connections must be made but to where and by who?

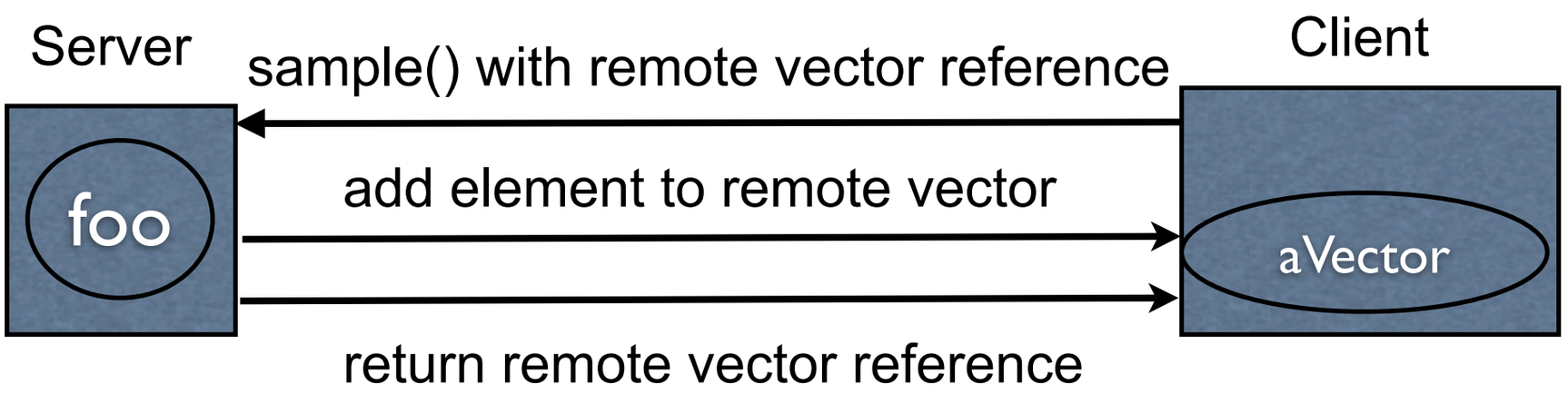
How does the client object interact with the server object?
How can we send a method request across the network?

Remote Method Parameters

Don’t move the parameters, just send them messages remotely

```
public class Foo {
   public Vector sample( Vector input ) {
      input.add( "Hello" );
      return input;
   }
}
```

```
Foo remote = getRemoteObject();
Vector local = new Vector();
local.magicToKeepItLocal();
Vector b = remote.sample(local);
String message = b.get(0);
```



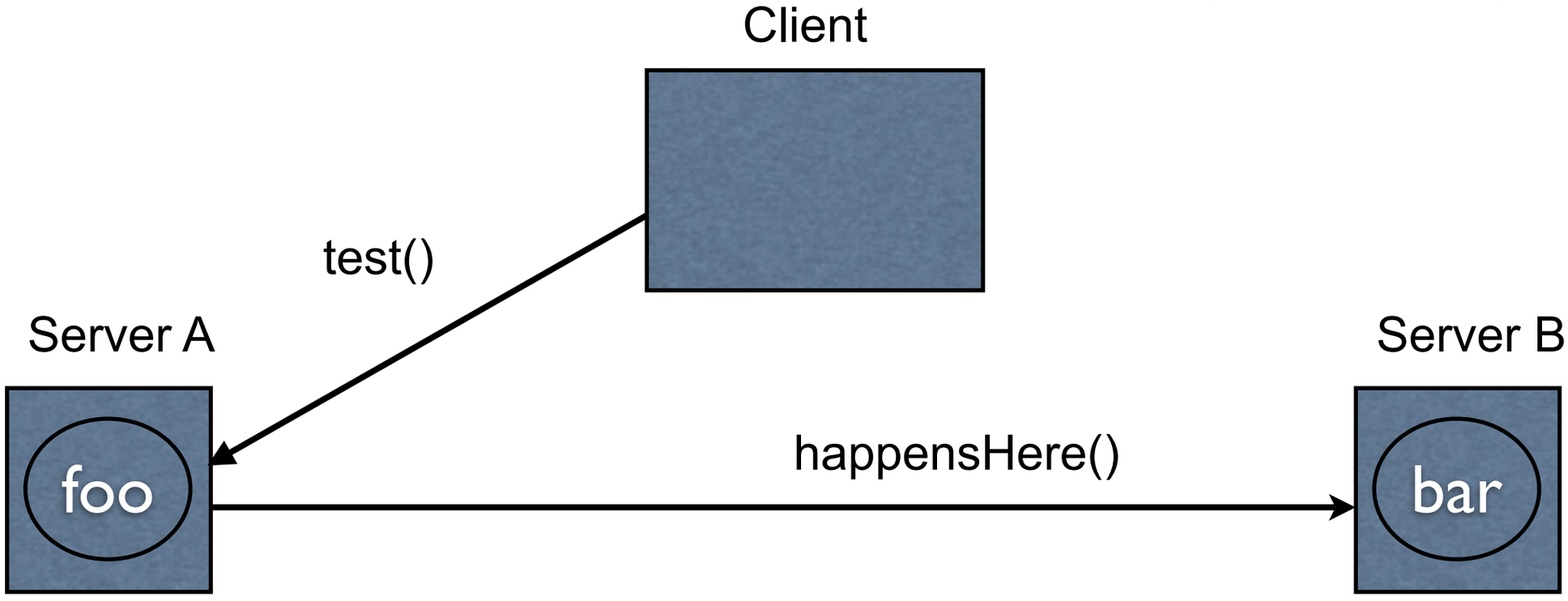
Passing Remote References

Server A

```
public class Foo {
   public void test(Bar what) {
     what.happensHere():
   }
}
```

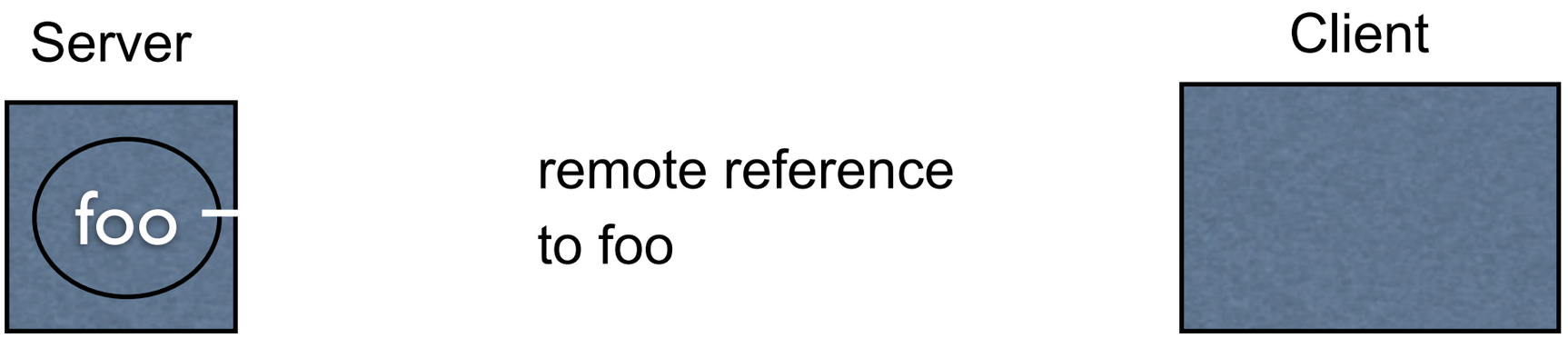
Client

```
Foo remote = getRemoteObject();
Bar otherRemote = getOtherRemote():
remote.test(otherRemote);
```



Remote References & Garbage Collection

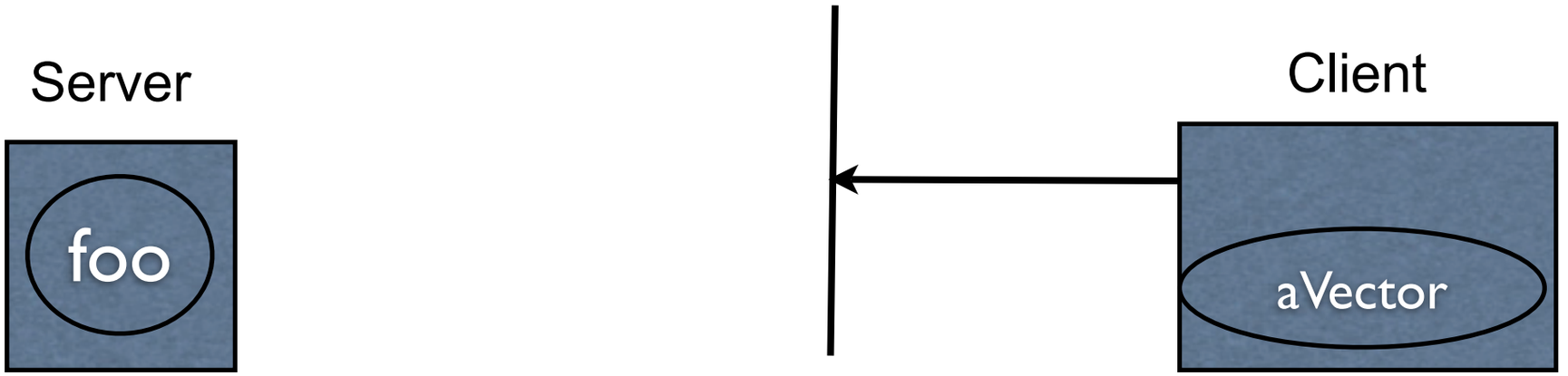
When can foo be garbage collected?



What happens when the network is disrupted?

```
public class Foo {
   public Vector sample( Vector input, int data ) {
      Integer remoteInt = new Integer( data );
      input.addElement( remoteInt );
      return input;
   }
}
```

```
Foo remote = getRemoteObject();
Vector b = remote.sample(aVector);
```



Some Background

Finding servers - Client-Dispatcher-Server Pattern

Making remote method calls - Proxy Pattern

Security - Java Security features

Client-Dispatcher-Server Pattern

Problems

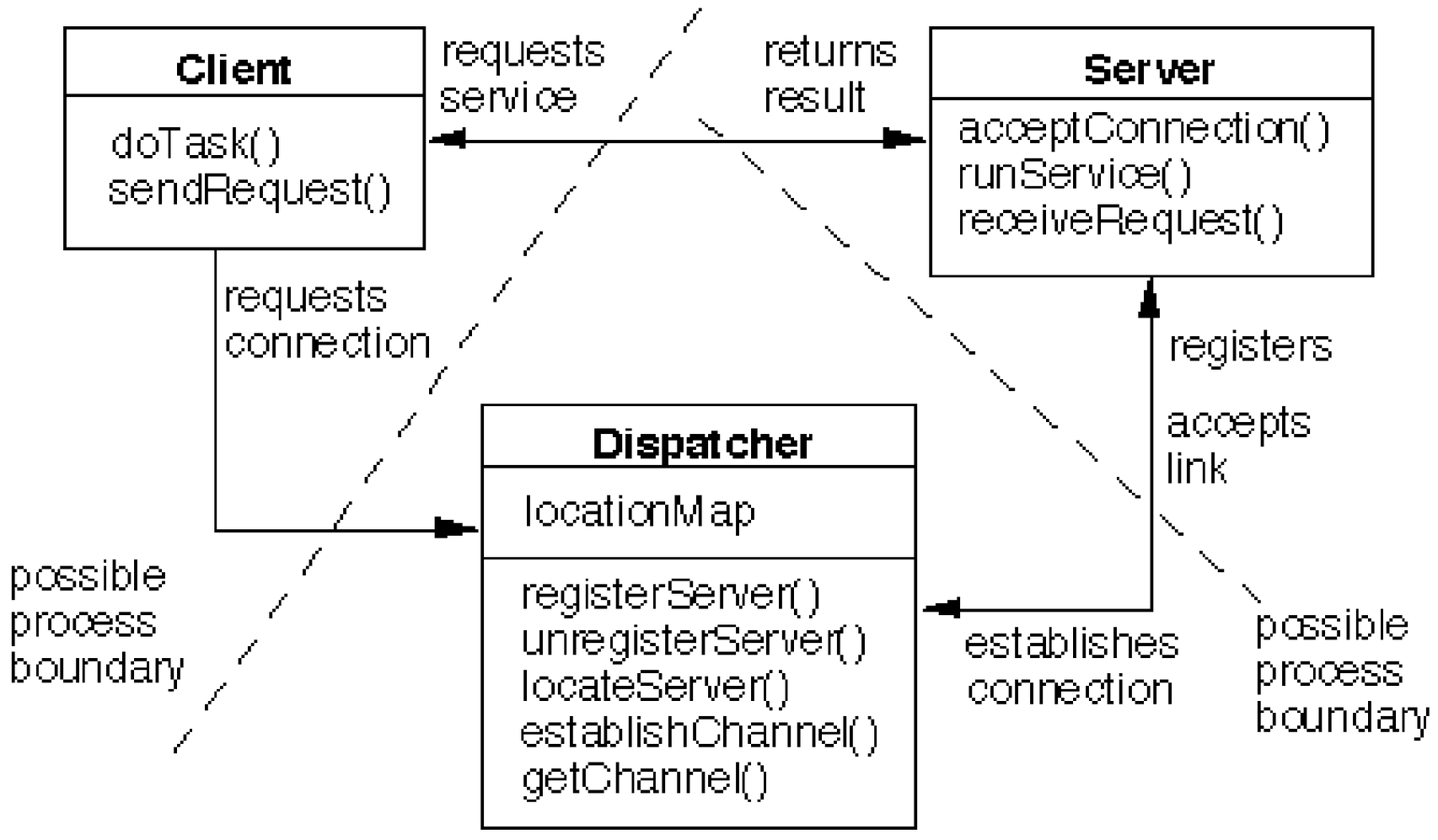
How to find the servers?

Clients should be able to use a server independent its location

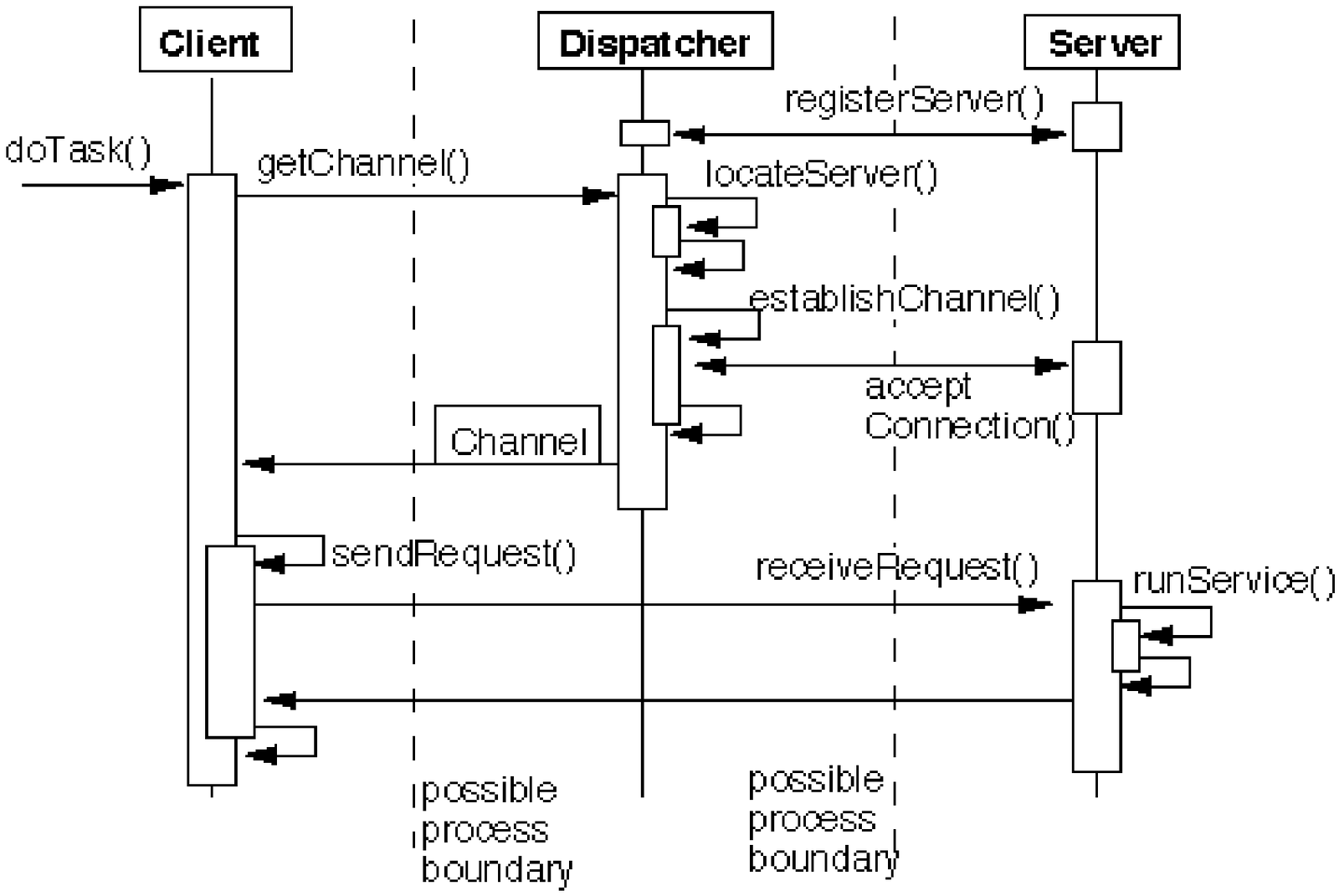
Separate functional core of a client from network code

Solution

Provide a dispatcher components to act as an intermediate layer between clients and servers



Dynamics


Client
Dispatcher
Server
registerServer()
doTask()
getChannel()
locateServer()
establishChannel()
accept
Connection()
Channel
sendRequest()
receiveRequest()
runService()
possible
process
boundary
possible
process
boundary

Variant - Distributed Dispatchers

Each machine has its own dispatcher

When a client needs to connection to a server on a remote machine:

The client connections to its local dispatcher

The local dispatcher connects to the remote dispatcher

The remote dispatcher returns the server communication channel to the local dispatcher

The local dispatcher returns to server communication channel to the client

The client now uses the communication channel to interact with the remote server

Variant - Communication Managed by clients

The dispatcher returns the physical server location to the client

The client manages all the communication with the server including opening the communication channel

Variant - Client-Dispatcher-Service

Clients request a service from the dispatcher

The dispatcher looks up all servers that provide that service and opens a communication channel to one of those servers

Consequences

Benefits

Exchangeability of servers

Location and migration transparency

Re-configuration

Fault Tolerance

Liabilities

Dispatcher overhead

Sensitivity to change in the interfaces of the dispatcher

Know Uses

Sun's RPC

Java's RMI

OMG Corba

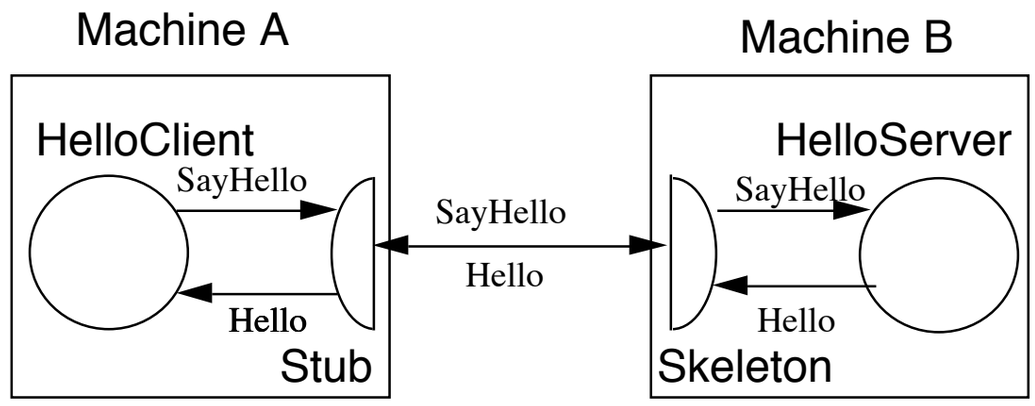
RMI - Hello World Example

Implement a server with the method sayHello()

Parts needed

- Hello interface
- Client code
- Server Code
- rmiregistery
- Proxy classes

(Permission file)



The Remote Interface

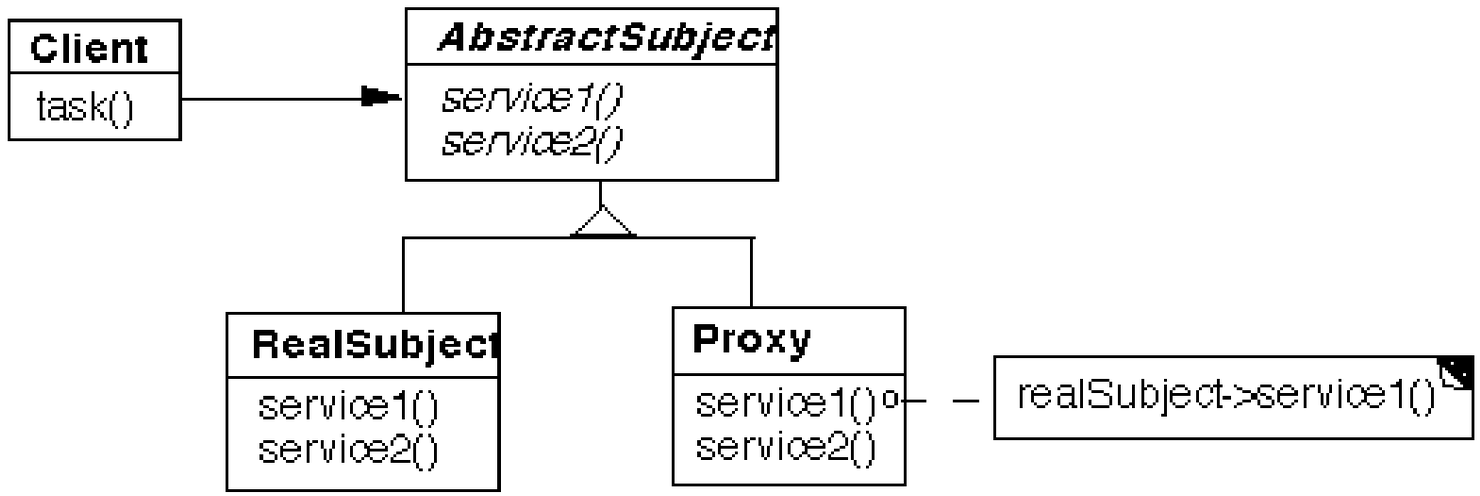
```
public interface Hello extends java.rmi.Remote
        {
        String sayHello() throws java.rmi.RemoteException;
        }
```

Why The Interface?

Client interacts with proxy for the server

The proxy implements the Hello interface

Client does not know difference



HelloServer

```
import java.net.InetAddress;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class HelloServer extends UnicastRemoteObject implements Hello {

        public HelloServer() throws RemoteException {  }

        public String sayHello() { return  "Hello World from " + getHostName(); }

        protected static String getHostName() {
                try {
                        return InetAddress.getLocalHost().getHostName();
                }
                catch (java.net.UnknownHostException who)  {
                        return "Unknown";
                }
        }
```


Example modified from Getting Started Using Java RMI, http://java.sun.com/javase/6/docs/technotes/guides/rmi/hello/hello-world.html

HelloServer Continued

```
        public static void main(String args[]) {
                try {
                        Server helloServer = new Server();
                        Hello stub = (Hello) UnicastRemoteObject.exportObject(helloServer,
0);

                        // Bind the remote object's stub in the registry
                        Registry registry = LocateRegistry.getRegistry();
                        registry.bind("Hello", stub);

                        System.err.println("Server ready");
                } catch (Exception e) {
                        System.err.println("Server exception: " + e.toString());
                        e.printStackTrace();
                }
        }
```

Why the Null Server Constructor?

public HelloServer() throws RemoteException { }

HelloServer's parent class constructor throws RemoteException

Since a constructor calls is parent class constructor, RemoteException can be thrown in HelloServer's constructor

One does not need the "throws RemoteException". I added it to make sure I talked about it.

Server Exits

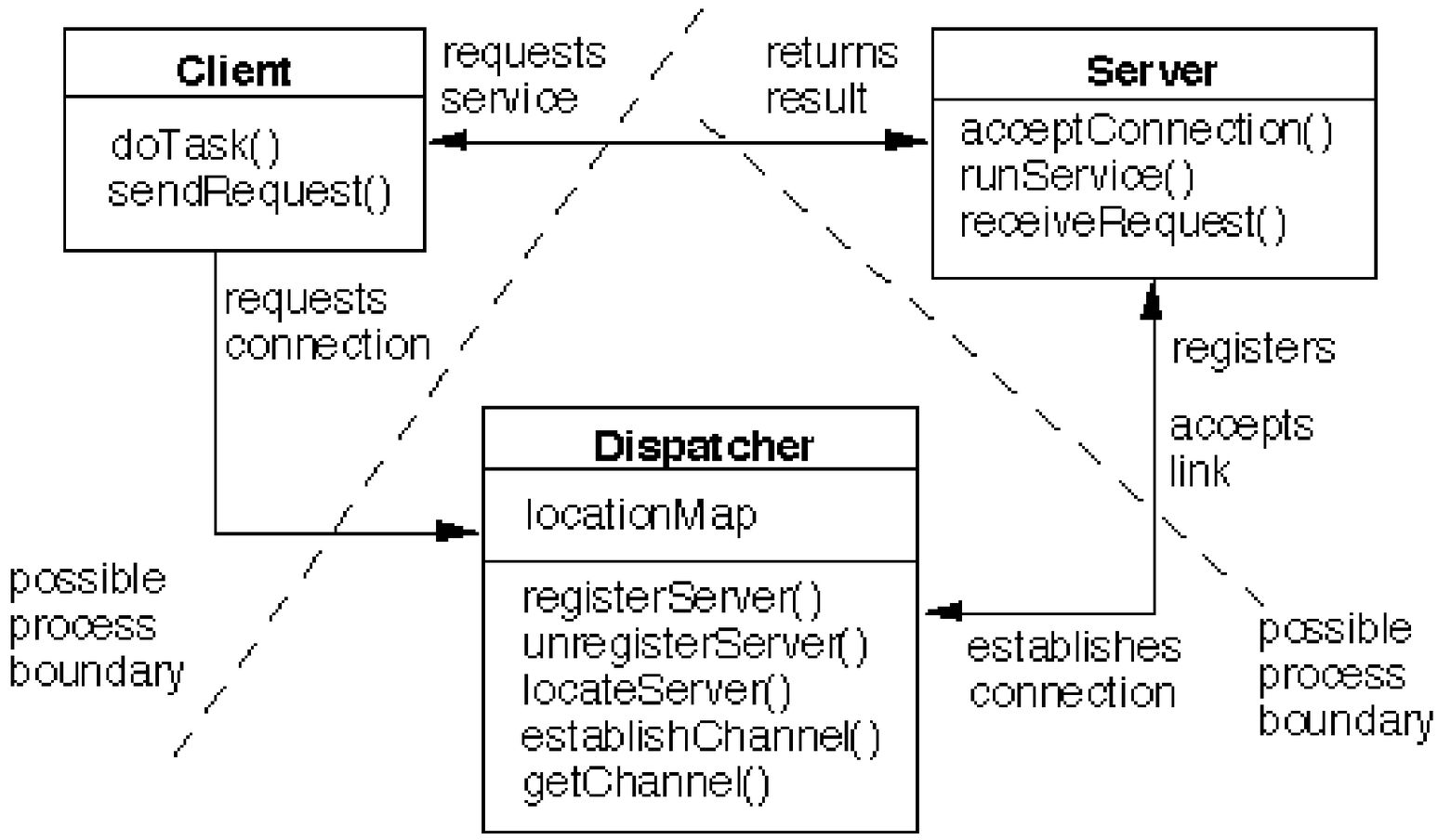
So how does it work?

```
public static void main(String args[]) {
        try {
                Server helloServer = new Server();
                Hello stub = (Hello) UnicastRemoteObject.exportObject(helloServer, 0);

                // Bind the remote object's stub in the registry
                Registry registry = LocateRegistry.getRegistry();
                registry.bind("Hello", stub);

                System.err.println("Server ready");
        } catch (Exception e) {
                System.err.println("Server exception: " + e.toString());
                e.printStackTrace();
        }
}
```


UnicastRemoteObject.exportObject(helloServer, 0); keeps the Server object alive.

What is registry.bind("Hello", stub)?

Registers server object with dispatcher (rmiregistry)



Registry Methods

bind(String name, Remote obj)

Binds a remote reference to the specified name in this registry.

String[] list()

Returns an array of the names bound in this registry.

Remote lookup(String name)

Returns the remote reference bound to the specified name in this registry.

void rebind(String name, Remote obj)

Replaces the binding for the specified name

void unbind(String name)

Removes the binding for the specified name in this registry.

HelloClient

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Client {

    public static void main(String[] args) {

        String host = (args.length < 1) ? "localhost" : args[0];
        try {
            Registry registry = LocateRegistry.getRegistry(host);
            Hello stub = (Hello) registry.lookup("Hello");
            String response = stub.sayHello();
            System.out.println("response: " + response);
        } catch (Exception e) {
            System.err.println("Client exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```


Example from Getting Started Using Java RMI, http://java.sun.com/javase/6/docs/technotes/guides/rmi/hello/hello-world.html

Compiling The Example

Server Side

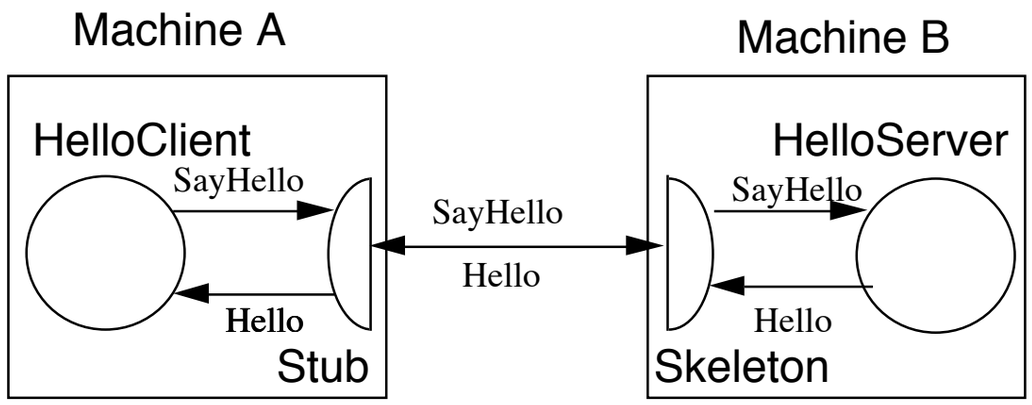
Compile source code

javac Hello.java Server.java

Generate Stub (Client side proxy)

rmic Server (only needed if supporting pre Java 5 clients)

Stub & Skeleton



Stub

Server proxy

Used by client

Java 1.5 can auto-generate

if server is subclass of UnicastRemoteObject

Skeleton

client proxy

Used by server

Generated automatically

Generated Stub class

```
public final class Server_Stub
    extends java.rmi.server.RemoteStub
    implements Hello, java.rmi.Remote {
    private static final long serialVersionUID = 2;
    private static java.lang.reflect.Method $method_sayHello_0;

    static {
        try {  $method_sayHello_0 = Hello.class.getMethod("sayHello", new java.lang.Class[] {});
        } catch (java.lang.NoSuchMethodException e) {
            throw new java.lang.NoSuchMethodError(
                "stub class initialization failed");
        }
    }
}

public HelloServer_Stub(java.rmi.server.RemoteRef ref) {  super(ref);}

 public java.lang.String sayHello() throws java.rmi.RemoteException  {
    try {
        Object $result = ref.invoke(this, $method_sayHello_0, null, 6043973830760146143L);
        return ((java.lang.String) $result);
    } catch (java.lang.RuntimeException e) { throw e;
    } catch (java.rmi.RemoteException e) { throw e;
    } catch (java.lang.Exception e) {
        throw new java.rmi.UnexpectedException("undeclared checked exception", e);
    }}}
```

Compiling Client Code

Compile client code

javac Client.java

Running the Server

Step 1. Insure that the RMI Registry is running

For the default port number

```
rmiregistry &
```

For a specific port number

```
rmiregistry portNumber &
```

Must be running on same machine as server

Server class files must be in rmiregistry's classpath

Running the Server

Step 2. Run the server

java example.hello.Server

Actually this just registers the server object with the rmiregistry

Running the Client

java example.hello.Client 127.0.0.1

RMI Clients & Downloading Code

An RMI client can

request a remote object

interact with parameters of remote object

without having the remote object's code

RMI will attempt to download the needed code from Server

Download Code

Server must provide

ftp site

or

http site

containing classfiles

java example.hello.Server -Djava.rmi.server.codebase="http://webfront/myStuff.jar"

Security Issue

Downloading code is dangerous!

Use Java Security Manager to download only code you trust

```
public class Client {

    public static void main(String[] args) {
System.setSecurityManager(new SecurityManager());
        String host = (args.length < 1) ? "localhost" : args[0];
        try {
            Registry registry = LocateRegistry.getRegistry(host);
            Hello stub = (Hello) registry.lookup("Hello");
            String response = stub.sayHello();
            System.out.println("response: " + response);
```

Java Access Control

Controlling what Java code can do

All Java programs subject to security checks

Fine-grained access control

Easily configurable security policy

Uses

- Security Manager
- Permissions file

Security Manager

Checks all attempts to use system resources

Only one security manager per program

In java.io.File class:

```
public boolean canRead() {
        SecurityManager security = System.getSecurityManager();
        if (security != null) {
                security.checkRead(path);
        }
        return canRead0();
}
```

In constructor for Socket:

```
SecurityManager security = System.getSecurityManager();
if (security != null) {
        security.checkConnect(address.getHostAddress(), port);
}
```

Permissions

Indicate what is allows and by who

Placed in java.policy file

- Global policy file
- Individual user policy file

General Format

```
grant [signedBy "signer_names"] [, codeBase "URL" ]{
      permission permission_class_name "target_name", "action"
            [, signedBy "signer_names"];
      ....
      permission permission_class_name "target_name" , "action"
            [, signedBy "signer_names"];
};
```

Existing Permissions

java.security.AllPermission
java.security.SecurityPermission
java.security.UnresolvedPermission
java.awt.AWTPermission
java.io.FilePermission
java.io.SerializablePermission
java.lang.reflect.ReflectPermission
java.lang.RuntimePermission
java.net.NetPermission
java.net.SocketPermission
java.sql.SQLPermission
java.util.PropertyPermission
java.util.logging.LoggingPermission
javax.net.ssl.SSLPermission
javax.security.auth.AuthPermission
javax.security.auth.PrivateCredentialPermission
javax.security.auth.kerberos.DelegationPermission
javax.security.auth.kerberos.ServicePermission
javax.sound.sampled.AudioPermission

Fine Grain Control

FilePermission

On a per file basis can indicate type of access is allowed

- read
- write
- execute
- delete

Different parts of program can have difference access

SocketPermission

General Format:

grant {
permission java.net.SocketPermission "host", "actions";
};

where:

host = (hostname | IPaddress) [:portrange]
portrange = portnumber | -portnumber |
portnumber-portnumber | portnumber-

actions = action | action,actions
action = accept | connect | listen | resolve

Host

host can be:
a DNS name
“localhost” for the local machine
a DNS name with wildcard character "*"

MacOS 10.6 Default Permission File

```
grant codeBase "file:${{java.ext.dirs}}/*" {permission java.security.AllPermission;};
grant codeBase "file:${user.home}/Library/Java/Extensions/*" {permission java.security.AllPermission;};
grant codeBase "file:/Library/Java/Extensions/*" {permission java.security.AllPermission;};
grant codeBase "file:/System/Library/Java/Extensions/*" {permission java.security.AllPermission;};
grant codeBase "file:/Network/Library/Java/Extensions/*" {permission java.security.AllPermission;};
grant {
        permission java.lang.RuntimePermission "stopThread";
        permission java.net.SocketPermission "localhost:1024-", "listen";
        permission java.util.PropertyPermission "java.version", "read";
        permission java.util.PropertyPermission "java.vendor", "read";
        permission java.util.PropertyPermission "java.vendor.url", "read";
        permission java.util.PropertyPermission "java.class.version", "read";
        permission java.util.PropertyPermission "os.name", "read";
        permission java.util.PropertyPermission "os.version", "read";
        permission java.util.PropertyPermission "os.arch", "read";
        permission java.util.PropertyPermission "file.separator", "read";
        permission java.util.PropertyPermission "path.separator", "read";
        permission java.util.PropertyPermission "line.separator", "read";

        permission java.util.PropertyPermission "java.specification.version", "read";
        permission java.util.PropertyPermission "java.specification.vendor", "read";
        permission java.util.PropertyPermission "java.specification.name", "read";
```


See jre/lib/security/java.policy on your installation

Loading Policy Files

System wide Policy File

Located in {JavaHome}/jre/lib/security/ java.policy

Loaded before user policy file

User Policy File

Default location is {UserHomeDirectory}/.java.policy

Command line policy file

java –Djava.security.policy=policyFileURL anApp

Load only Command line policy

java –Djava.security.policy==policyFileURL anApp