Java RMI

Adapted from:
Paul Krzyzanowski
pxk@cs.rutgers.edu
ds@pk.org
Java RMI

• Java language had no mechanism for invoking remote methods

• 1995: Sun added extension
  - Remote Method Invocation (RMI)
  - Allow programmer to create distributed applications where methods of remote objects can be invoked from other JVMs
RMI components

**Client**
- Invokes method on remote object

**Server**
- Process that owns the remote object

**Object registry**
- Name server that relates objects with names
Interoperability

RMI is built for Java only!
- No goal of OS interoperability (as CORBA)
- No language interoperability (goals of SUN, DCE, and CORBA)
- No architecture interoperability

No need for external data representation
- All sides run a JVM

Benefit: simple and clean design
RMI similarities

Similar to local objects

- References to remote objects can be passed as parameters (not really)
- Objects can be passed as parameters to remote methods (but not as a reference)
- Object can be cast to any of the set of interfaces supported by the implementation
  - Operations can be invoked on these objects
RMI differences

• Non-remote arguments/results passed to/from a remote method by copy

• Remote object passed by reference, not by copying remote implementation

• Extra exceptions
New classes

• **remote class:**
  - One whose instances can be used remotely
  - Within its address space: regular object
  - Other address spaces: can be referenced with an **object handle**

• **serializable class:**
  - Object that can be marshaled
  - If object is passed as parameter or return value of a remote method invocation, the value will be copied from one address space to another
    • If remote object is passed, only the object handle is copied between address spaces
New classes

• **remote class:**
  - One whose instances can be used remotely
  - Within its address space: regular object
  - Other address spaces: can be referenced with an object handle

• **serializable class:**
  - Object that can be marshaled
  - If object is passed as parameter or return value of a remote method invocation, the value will be copied from one address space to another
  - If remote object is passed, only the object handle is copied between address spaces
Stubs

Generated by separate compiler

rmic

- Produces Stubs and skeletons for the remote interfaces are generated (class files)
Naming service

Need a remote object reference to perform remote object invocations

Object registry does this: rmiregistry
Server

Register object(s) with Object Registry

Stuff obj = new Stuff();
Naming.bind("MyStuff", obj);
Contact `rmiregistry` to lookup name

```java
MyInterface test = (MyInterface) Naming.lookup("rmi://www.pk.org/MyStuff");
```

`rmiregistry` returns a remote object reference. `lookup` gives reference to local stub.

Invoke remote method(s):

```java
test.func(1, 2, "hi");
```
Java RMI infrastructure

client application

lookup
remote reference

f(args)
return/exception

stub

marshal stream

registry

bind

remote object implementation
remote interface

f(args)
return/exception

skeleton

(Skeletons not needed as of Java 2, RMI uses straight Java Reflection)
RMI Distributed Garbage Collection

- Two operations: dirty and free

- Local JVM sends a dirty call to the server JVM when the object is in use
  - The dirty call is refreshed based on the lease time given by the server

- Local JVM sends a clean call when there are no more local references to the object