

Java RMI (Remote Method Invocation)

CSE 794R/ECE 694R

1

Java RMI

- The Java Remote Method Invocation (RMI) system allows an object running in one Java Virtual Machine (VM) to invoke methods on an object running in another Java VM
- RMI provides for remote communication between programs written in the Java programming language

CSE 794R/ECE 694R

2

Overview

- RMI applications are often comprised of two separate programs: a server and a client
- A typical server application creates some remote objects, makes references to them accessible, and waits for clients to invoke methods on these remote objects
- A typical client application gets a remote reference to one or more remote objects in the server and then invokes methods on them
- RMI provides the mechanism by which the server and the client communicate and pass information back and forth

CSE 794R/ECE 694R

3

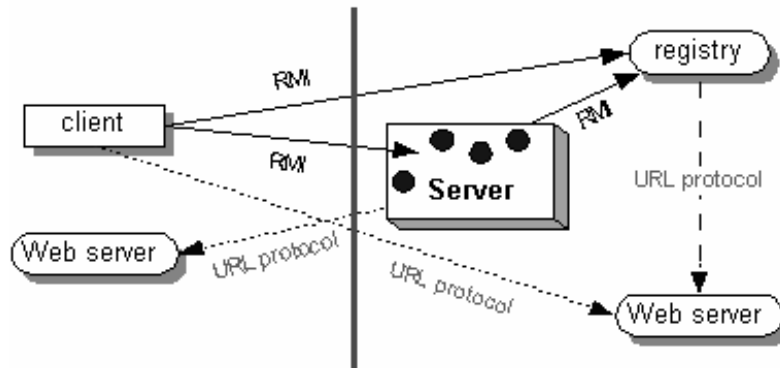
Distributed Object Application

- Applications using RMI need to:
 - Locate remote objects
 - rmiregistry (RMI's simple naming facility)
 - pass and return remote object references
 - Communicate with remote objects
 - details of communication between remote objects are handled by RMI
 - remote communication looks like a standard Java method invocation
 - Load class bytecodes for objects that are passed around
 - RMI allows a caller to pass objects to remote objects
 - RMI provides the necessary mechanisms for loading an object's code, as well as for transmitting its data

CSE 794R/ECE 694R

4

RMi Distributed Application



CSE 794R/ECE 694R

5

Remote Objects

- Objects that have methods that can be called across virtual machines are *remote objects*
- An object becomes remote by implementing a *remote interface*, which has the following characteristics:
 - it extends the interface `java.rmi.Remote`
 - each method of the interface declares `java.rmi.RemoteException` in its throws clause (in addition to any application-specific exceptions)

CSE 794R/ECE 694R

6

Remote Objects cont.

- `java.rmi.Remote` is a *marker* (empty) interface
- remote interface implementation usually extends class `UnicastRemoteObject`
 - must already be executing to service client
 - object is not replicable (one remote object serves all clients)
 - each client executes as a separate thread

CSE 794R/ECE 694R

7

RMI Applications

- When you use RMI to develop a distributed application, you follow these general steps:
 - Design and implement the components of your distributed application
 - Compile sources (and generate stubs)
 - Make classes network accessible
 - Start the application (name server, server, client, ...)

CSE 794R/ECE 694R

8

Naming Service

- Naming class provides methods for storing and obtaining references to remote objects in a remote object registry
- Each method of the Naming class takes a URL (without the scheme component) of the form:

`//host:port/name`

where host is the host (remote or local) where the registry is located, port is the port number on which the registry accepts calls, and where name is a simple string uninterpreted by the registry

- Both host and port are optional. If host is omitted, the host defaults to the local host. If port is omitted, then the port defaults to 1099, the "well-known" port that RMI's registry, **rmiregistry**, uses.

CSE 794R/ECE 694R

9

Accessing Naming Service

- Server:
 - `Naming.rebind(name, object)`
- Client:
 - `Naming.lookup(rmi_url)`

CSE 794R/ECE 694R

10