Java RMI

Different Approaches to Distributed Computation

- High-performance, parallel scientific apps
  - e.g. MPI
- Connecting via sockets
  - custom protocols for each application
  - text or binary protocol
- RPC/DCOM/CORBA/RMI
  - make what looks like a normal function call
  - function is actually invoked on another machine
  - Arguments are marshalled for transport
  - return value is marshalled as well

Remote Method Invocation

- Easy way to get distributed computation
- Have stub for remote object
  - calls to stub get translated into network call
- Arguments can be passed over network

Remote Objects and Interfaces

- Remote Objects are those that can be referenced remotely
  - extends java.rmi.UnicastRemoteObject
  - constructor throws java.rmi.RemoteException
- Remote interfaces describe services that can be provided remotely
  - extends java.rmi.Remote interface
  - all methods throw java.rmi.RemoteException

RMIC - RMI Compiler

- Generates stub code for a class
  - For 1.1, also generates skeleton class
  - skeleton not needed for 1.2+
- Generates stubs for all methods declared in remote interfaces
  - other methods don’t get a stub

Passing arguments

- Can pass arbitrary values as arguments
- Can return arbitrary values as results
- To pass a value, it must either be
  - Serializable, or
  - Remote
- Passing the same Serializable object in different calls
  - will materialize different objects at receiver
**Downloading code**

- When you pass a ref to remote class
  - receiver needs stub class
- When you pass a ref to serializable class
  - receiver needs class
- Annotate ref’s with RMI codebase
  - where code can be loaded from

**SecurityManager**

- Must install some Security Manager to allow download of classes from RMI codebase
- Can use RMISecurityManager
  
  ```java
  System.setSecurityManager(new RMISecurityManager());
  
  // Modify policy file to grant permissions
  ```

**Naming.lookup**

- Naming.lookup is used to bootstrap RMI communication
  - Get your first reference to a remote object
- Run an RMIRegistry
  - a separate Java VM
  - listens to a particular port (default 1099)
- Can bind/unbind/rebind name on localhost
- Can lookup name on any host

**RMI Chat server**

- Server
  - runs the chat room
- Client
  - participant in chat room
  - receives messages from others in room
- Connection
  - uniquely identifies a client
  - used to speak in chat room

**Server**

```java
interface Server extends Remote {
  Connection logon(String name, Client c)
      throws RemoteException;
}
```

**Connection**

```java
interface Connection extends Remote {
  /** Say to everyone */
  void say(String msg)
      throws RemoteException;
  /** Say to one person */
  void say(String who, String msg)
      throws RemoteException;
  String[] who()
      throws RemoteException;
  void logoff()
      throws RemoteException;
}
```
Client

- interface Client extends Remote {
  void said(String who, String msg) throws RemoteException;
  void whoChanged(String [] who) throws RemoteException;
}

Remote Object creation

Server s = new ServerImpl();

ServerImpl

Client creation

Client c = new ClientImpl();

Server Impl

RMI Registry

Naming.rebind("ChatServer", s);

ServerImpl

ClientImpl

Server Impl Stub

ServerImpl

ServerImpl Stub

ServerImpl

ServerImpl

ServerImpl Stub

ServerImpl

Invoke message on Server

prepare call on client

Connection conn = s.logon("Bill", c);

ClientImpl

ServerImpl

ServerImpl Stub

ServerImpl

ServerImpl Stub

ServerImpl

ClientImpl

ServerImpl

ServerImpl Stub

ServerImpl

ServerImpl Stub
Unmarshalling arguments on Server

**target**
- method: `logon`
- Marshalled arguments
  - String "Bill"
  - Serialized Stub for `c`

**ServerImpl**
- `logon`
- "Bill"
- ClientImpl Stub

Execution on Server

**Hosted Remote Objects**
- "Bill"
- ClientImpl Stub
- Connection Impl

Unmarshalling return value

**Hosted Remote Objects**
- Marshalled return value: Serialized Stub for conn

Marshalling return value

**Hosted Remote Objects**
- Marshalled return value: Serialized Stub for conn

**ServerImpl**
- ConnImpl Stub