

ECE 566- Parallel and Distributed Technology

Distributed Objects: CORBA/Java RMI

Manish Parashar

parashar@ece.rutgers.edu

Department of Electrical &
Computer Engineering

Rutgers University

Distributed Objects: Goals

- Let any object reside anywhere in the network, and allow an application to interact with these objects exactly in the same way as they do with a local objects
- Provide the ability to construct an object on one host and transmit it to another host
- Enable an agent on one host to create a new object on another host.

Distributed Objects: Operations

- Locate remote object
- Creating remote objects
- Invoke methods on remote objects
- Obtain results from a remote method invocation
- Delete remote object

Distributed Objects: Properties

- Object locator
- Communication
 - data type
 - data representation
 - synchronous/asynchronous
- State Persistence
- Security
- Reliability/Availability
- Load Balancing

Java RMI

- Components
 - Object Interfaces
 - extends the *Remote* interface
 - Server side implementation of the interface
 - extends the *java.rmi.server.UnicastRemoteObject*
 - RMI registry
 - *rmic* compiler for stub and skeleton generation
- Others Issues
 - Serialization
 - Exception handling
 - Constructors

Java RMI: Object Interfaces

```
import java.rmi.Remote;  
import java.rmi.RemoteException;  
  
public interface DataRetrieval extends Remote  
{  
    String GetData()  
        throws RemoteException;  
}
```

Java RMI: Server Implementation

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

```
public class DataRetrievalImpl
    extends UnicastRemoteObject
    implements DataRetrieval
{
    private int call;

    public DataRetrievalImpl()
        throws RemoteException
    {
        super();
        call = 0;
    }
}
```

```
public String GetData()
    throws RemoteException
{
    String client = null;

    try
    {
        client = getClientHost();
    }
    catch(Exception e)
    {
        System.out.println("exception: " + e);
        client = "unknown";
    }

    call++;
    System.out.println(client + "
        request, data is:" + call);
    return "data is:" + call;
}
```

Java RMI: Server Implementation

```
public static void main(String args[])
{
    System.out.println("Started server...");
    try
    {
        InetAddress host = InetAddress.getLocalHost();
        String name = "/" + host.getHostName() + "/DataRetrieval";
        System.out.println("binding to " + name);
        System.setSecurityManager(new RMISecurityManager());
        DataRetrievalImpl data = new DataRetrievalImpl();
        Naming.rebind(name,data);
    }
    catch(Exception e)
    {
        System.out.println("exception: " + e);
    }
}
```

Java RMI: Client

```
import DataRetrieval;
import java.rmi.Naming;

public class DataRetrievalClient
{
    public static void main(String[] args)
    {
        if (args.length != 1)
        {
            System.out.println("Please give server name");
            return;
        }
        try
        {
            DataRetrieval data = null;
            String name = "rmi://" + args[0] + "/DataRetrieval";
            data = (DataRetrieval)Naming.lookup(name);
            String reply;
            reply = data.GetData();
            System.out.println("Got " + reply);
        }
        catch(Exception e)
        {
            System.out.println("Exception " + e);
        }
        System.exit(0);
    }
}
```

Java RMI: Registry/rmic

- RMI registry serves the role of the object manager
 - % rmiregistry
 - % rmiregistry <port #> // default is 1099

- Addressing a remote object

MyObject obj1 =

```
(MyObject)Naming.lookup("rmi://objecthost.myorg.com/Object1);
```

MyObject obj1 =

```
(MyObject)Naming.lookup("rmi://objecthost.myorg.com:2099/Object1);
```

- rmic compiler
 - rmic *serversideimpl*
 - generates client *stub* and server *skeleton*
 - bootstraps off java bytecode

CORBA

- Object Request Broker (ORB)
 - enable clients and server objects to interact
 - provide servers
 - naming services, security services, ...
- Interface Definition Language (IDL)
- Dynamic Invocation Interface (DII)
 - Interface repository
- Internet Inter-Orb Protocol (IIOP)

CORBA Solver

- Generate IDL description
- Generate client stub
 - `idltojava -f client Solver.idl`
 - generates *Base Interface*, *holder* classes & client stubs

```
package DCJ.examples;  
public class _SolverStub  
    extends org.omg.CORBA.portable.ObjectImpl  
    implements DCJ.examples.Solver {
```

```
package DCJ.examples;  
public class _ProblemSetStub  
    extends org.omg.CORBA.portable.ObjectImpl  
    implements DCJ.examples.ProblemSet {
```

CORBA Solver

- Generate server skeleton and implementation
 - `idltojava -f server Solver.idl`
 - generates abstract base class for remote implementation

```
package DCJ.examples;  
public abstract class _ProblemSetImplBase extends org.omg.CORBA.portable.ObjectImpl  
    implements DCJ.examples.ProblemSet,  
               org.omg.CORBA.portable.Skeleton {
```

```
package DCJ.examples;  
public abstract class _SolverImplBase extends org.omg.CORBA.portable.ObjectImpl  
    implements DCJ.examples.Solver,  
               org.omg.CORBA.portable.Skeleton {
```

CORBA Solver

- Implement remote object
- Implement client
- Start name server
 - `nameserve -ORBInitialport 1050`
- Run Server
- Run Client