Straightforward Jini™ Network Technology: Jini Services for the Uninitiated

James Morgan and Clark Richey

Principal Consultants
SRA // RABA Center
http://rport.raba.com/StraightforwardJini/

TS-1161

Goal

Learn how to use Jini™ network technology for distributed services

Demonstrate how to create and configure a Jini network technology service, advertise it, discover, and use it.

Help identify what in the Jini network technology distribution is important, necessary, or just nice to have.

Agenda

Quick Introduction

A Service and Its Proxy

Configuration and Administration

Packaging and Launching Services

Discovering and Using Services

User Interfaces

Agenda

Quick Introduction

A Service and Its Proxy

Configuration and Administration

Packaging and Launching Services

Discovering and Using Services

User Interfaces

Jini Network Technology… What It Is and What It's For

- An architecture for secure, distributed services
- Built on and extends Java™ technology
  - Requires Java platform 1.4.x or newer
- Dynamic discovery of services on the network
  - No knowledge of service location is necessary
- Designed to work with a dynamic network
  - Services can be added and removed any time
- Expects the network to be fallible

Deutsch's Fallacies of Networking

http://today.java.net/jag/Fallacies.html

- **The network is reliable**
- **Latency is zero**
- **Bandwidth is infinite**
- **The network is secure**

$\Big\}$ Bill Joy and Tom Lyon

- Topology doesn't change
- There is one administrator
- Transport cost is zero
- **The network is homogeneous** $\}$ James Gosling

So How Do I Start Using Jini Network Technology?

- No clear place to start
  - Where are the tutorials?
  - Where are the books written since 1.2?
- Jini.org is not easy to navigate and is being reorganized into a wiki
- http://jini.dev.java.net/ is relatively new and still forming content and subprojects
- Mailing list JINI-USERS appears to be the place everything gets answered but is never codified

What Jini Network Technology Has to Offer

EventMailbox PreferredList Exporter

Entry Security Lookup Registry

Lease

Service Discovery Smart Proxy HTTPMD

Activatable

Proxy LeaseManagement

JRMP

TransactionManager JavaSpace

JERI

RemoteException

RemoteEvents Class Server

Lease Landlord

Proxy Trust

Configuration DynamicPolicy

How the Jini Technology Starter Kit Offers It

ComputeDigest

Norm

JarWrapper

Outrigger

Fiddler

ClassServer

Mercury

Reggie

PreferredListGen

ClassDepAndJar

Browser

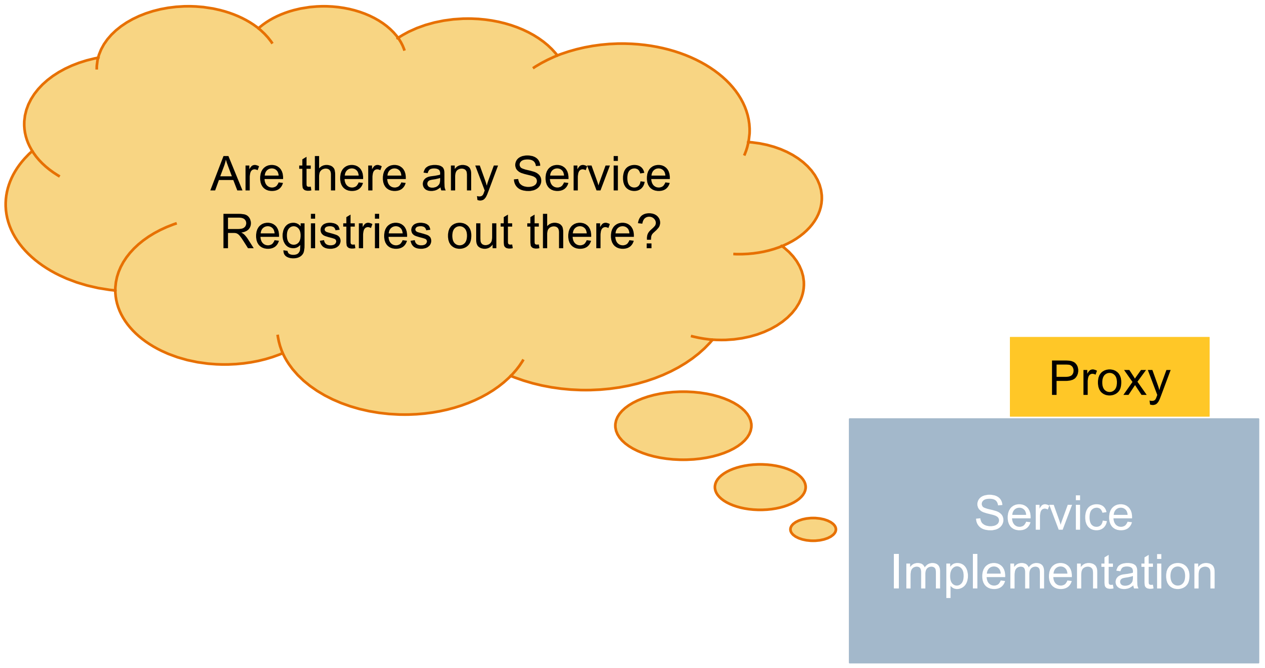
Mahalo

CheckConfigurationFile

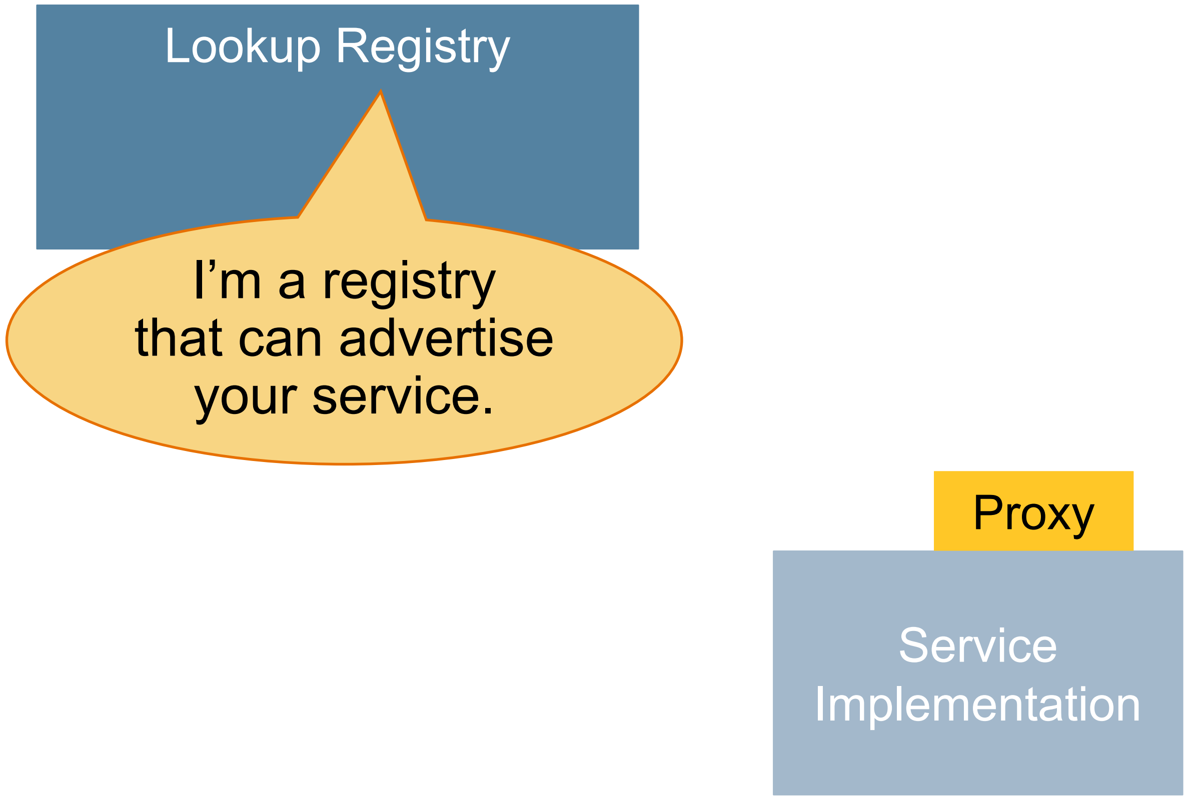
Jini Network Technology View of the World

- Service (Interface and Implementation)
- Lookup Registry (Service Registrar)
  - Place to let the world know about the service
- Proxy (Implements Interface)
  - What is stored in the registry
  - What the client uses to talk to the service
- Client or Service User
  - Discovers and uses the service

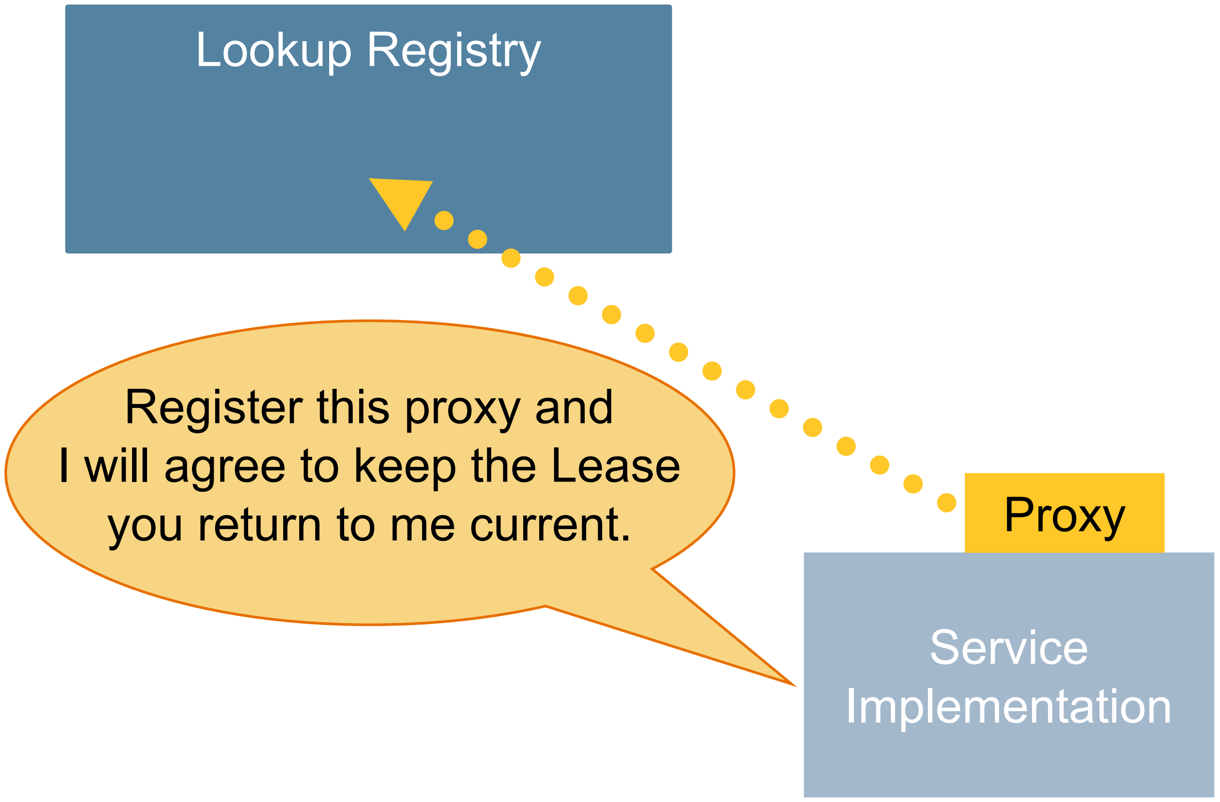
Service Registry Discovery



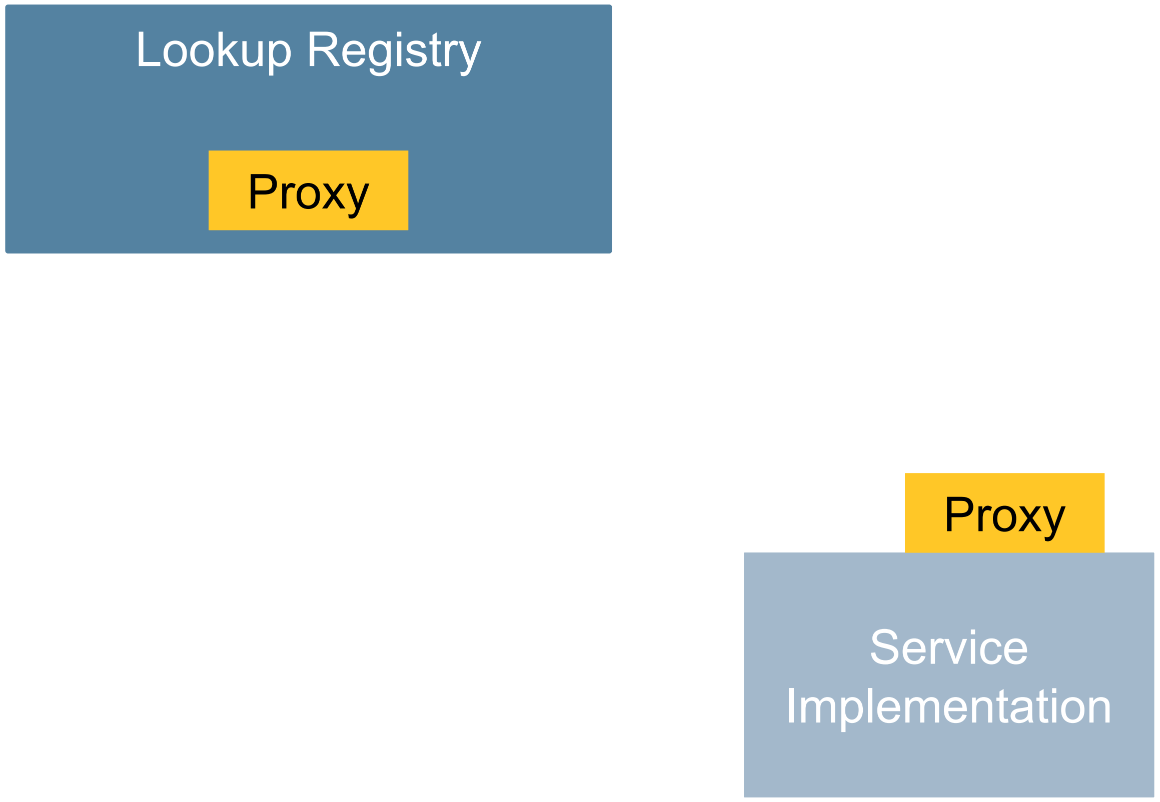
Service Registry Discovery



Service Registration


Lookup Registry
Register this proxy and I will agree to keep the Lease you return to me current.
Proxy
Service Implementation

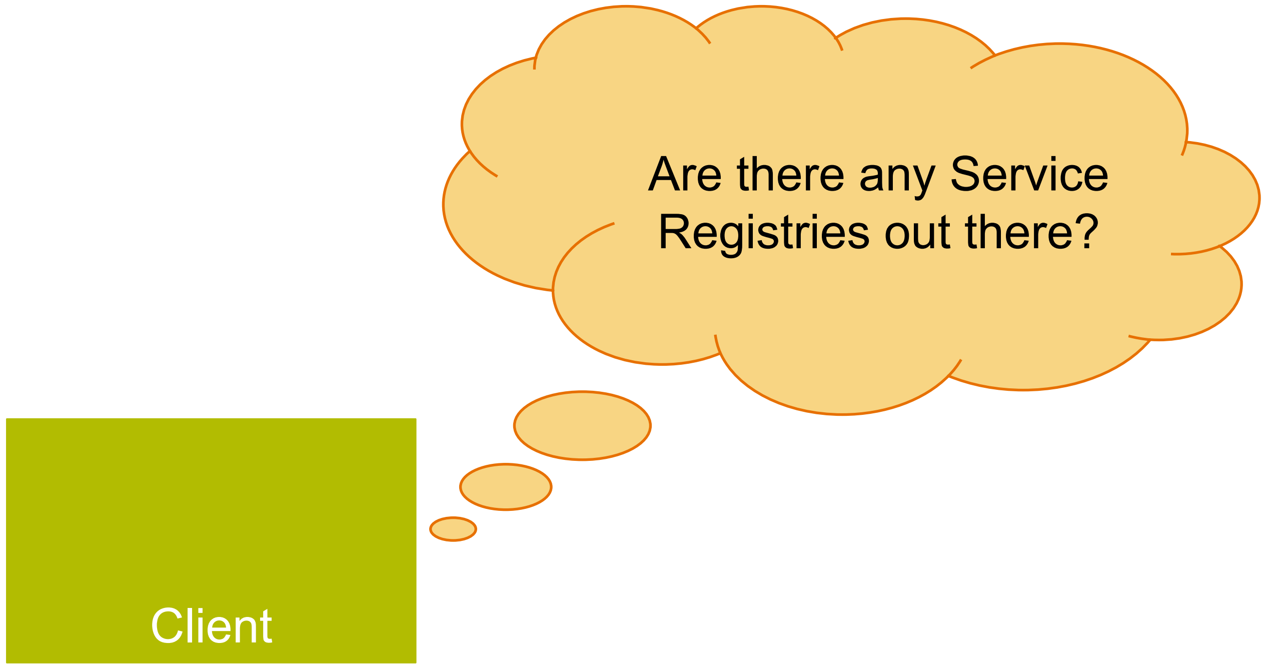
Service Registration



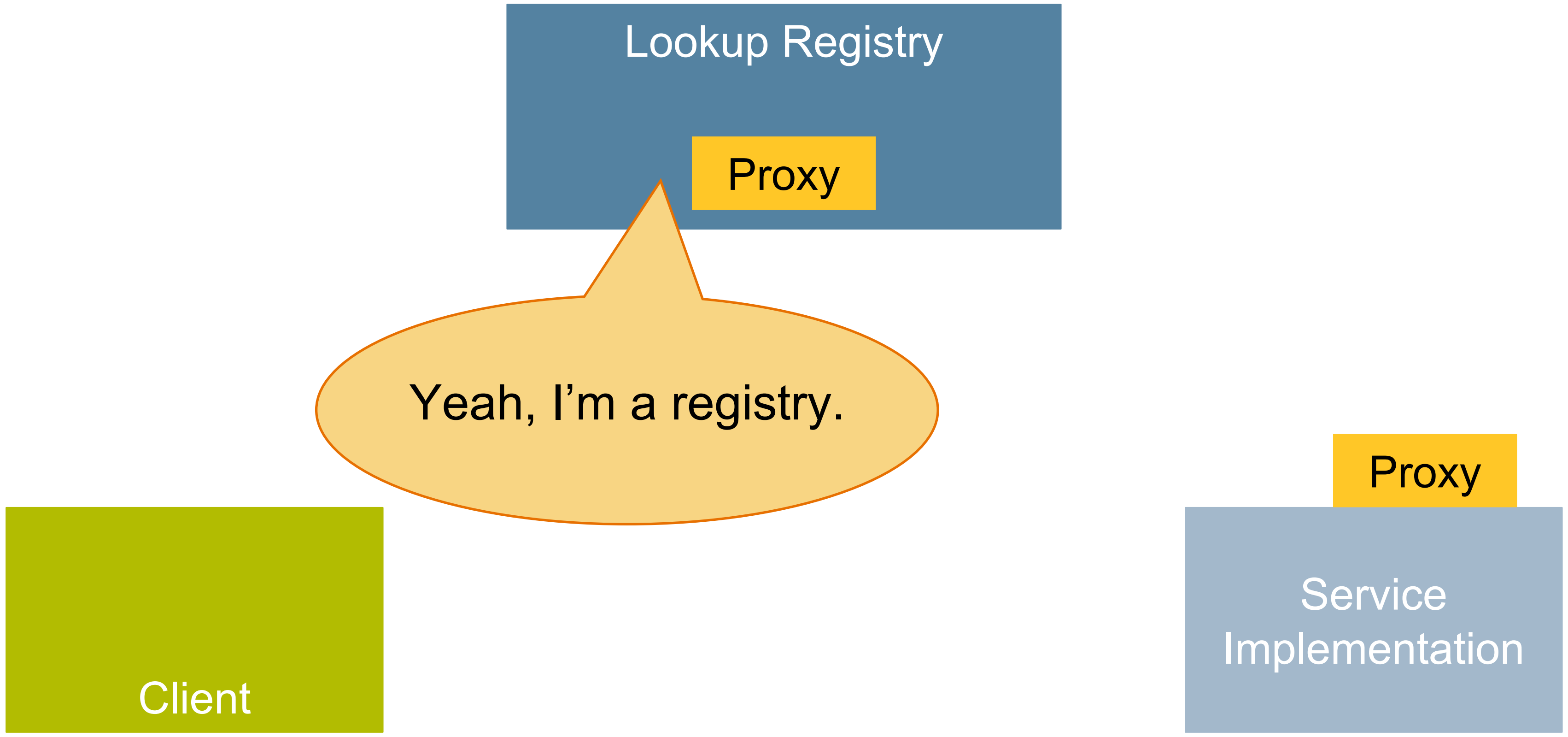
Server-Side Summary

- Define the service interface
- Create a service implementation
- Create a service proxy
- Find Lookup Registries
- Register the service proxy
- Manage the returned Lease
- Continue to register as Registries appear

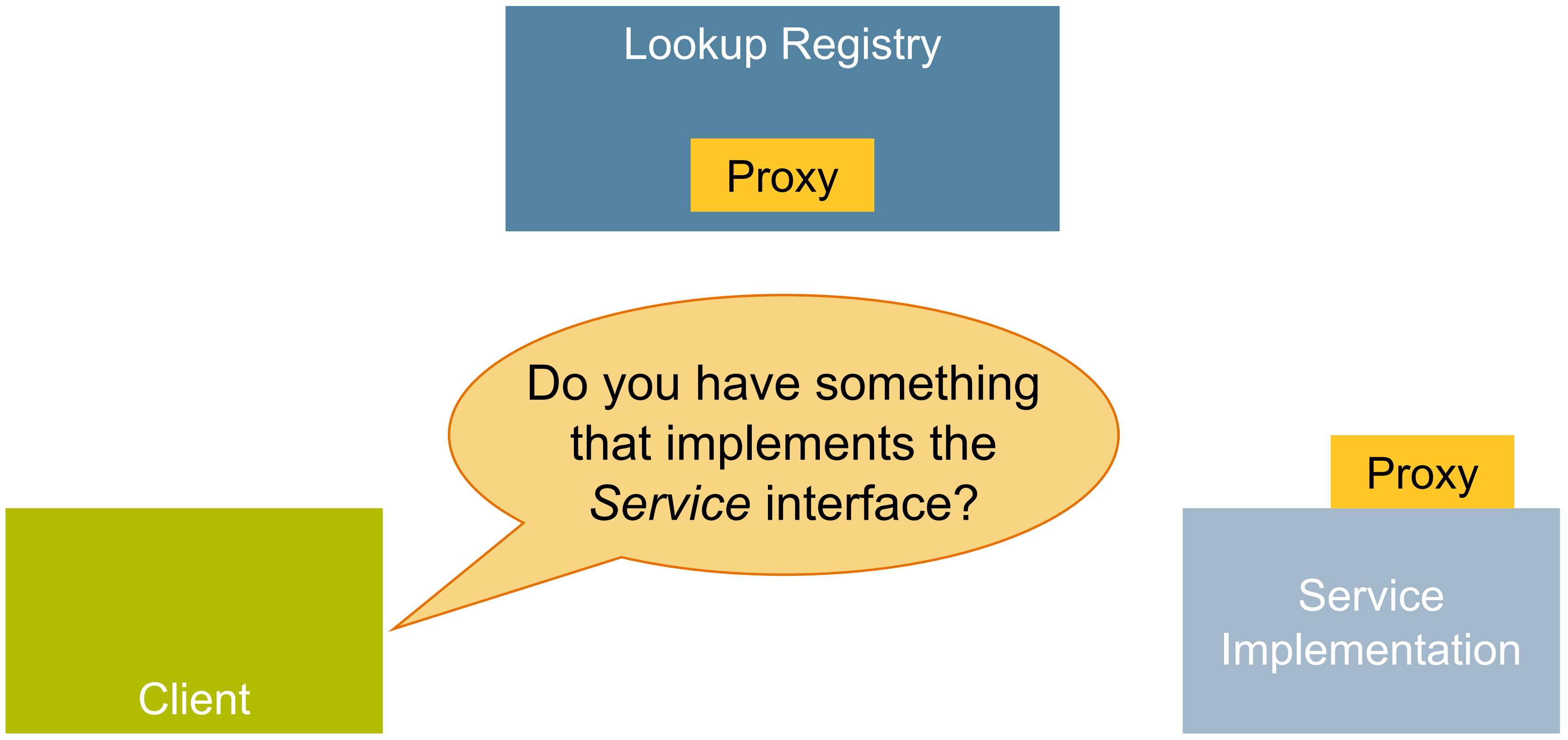
Service Discovery



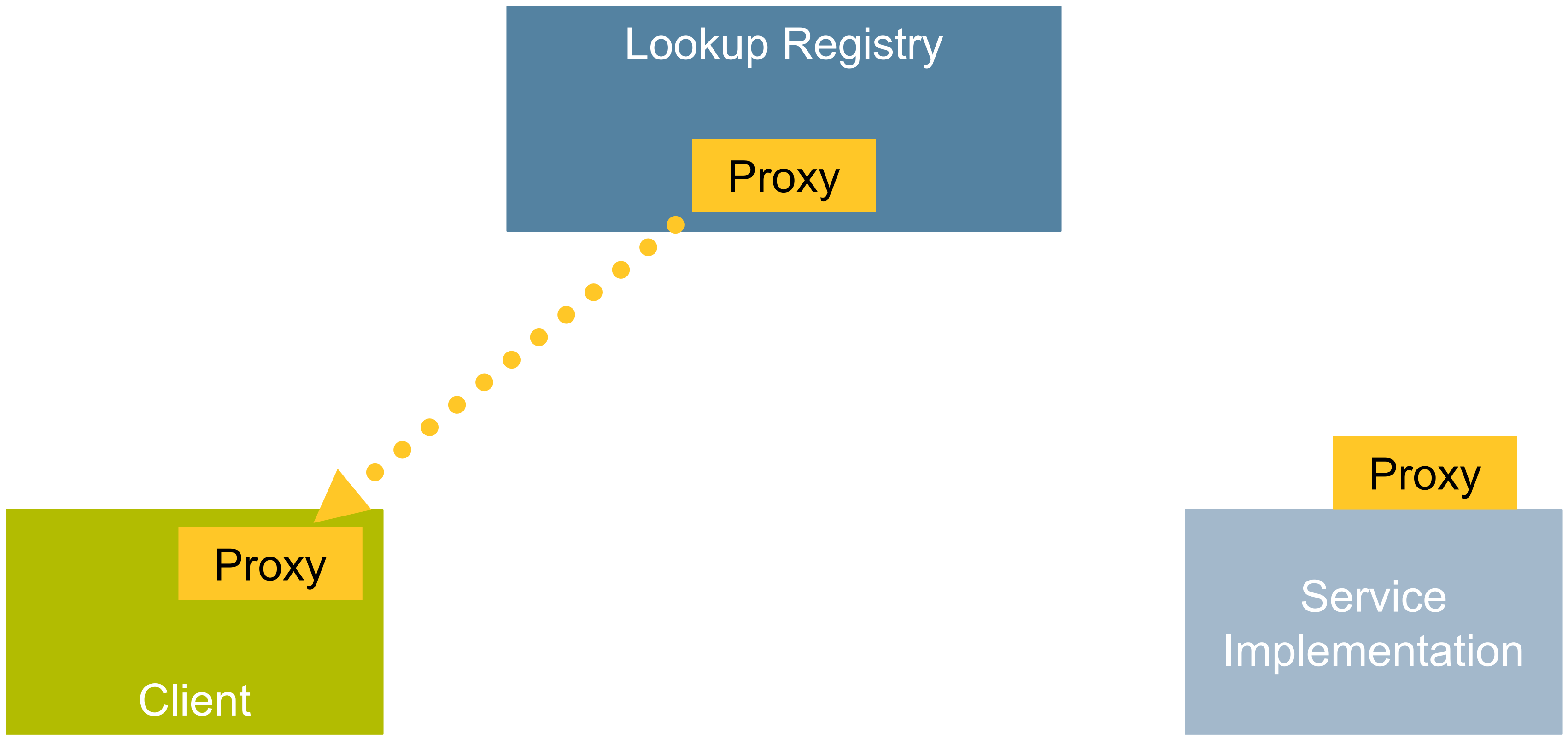
Service Discovery



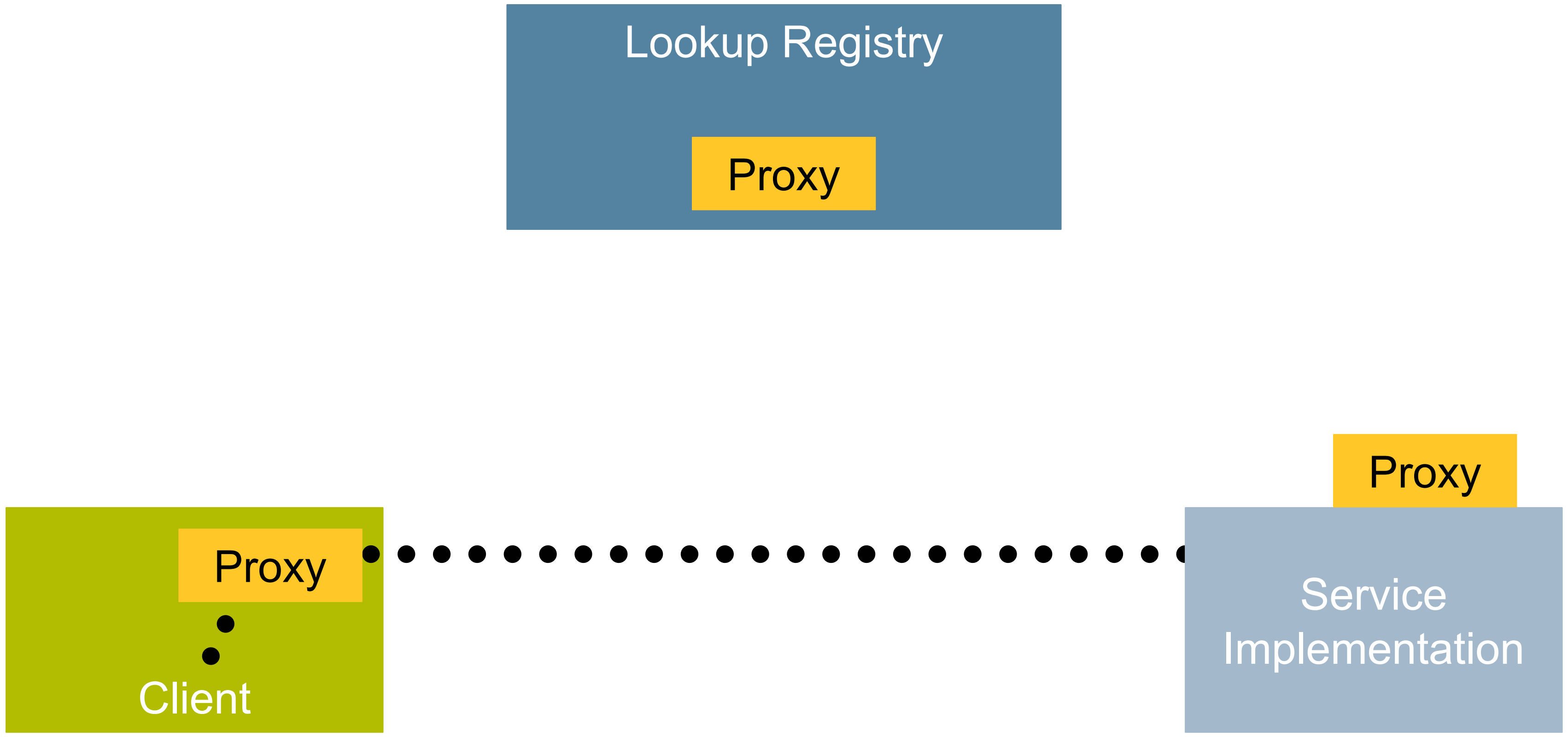
Service Discovery


Lookup Registry
Proxy
Do you have something that implements the *Service* interface?
Proxy
Service Implementation
Client

Service Discovery



Service Use Through the Proxy



Client-Side Summary

- Find Lookup Registries
- Ask discovered registries for a service that implements the service interface
  - Additional constraints can be used to filter matches
- Use the service like any other implementation of the interface
- On RemoteException:
  - Drop the service reference
  - Rediscover service

Agenda

Quick Introduction

A Service and Its Proxy

Configuration and Administration

Packaging and Launching Services

Discovering and Using Services

User Interfaces

A Thermostat Service

- Access to fan and power settings
- Access to current temperature
- Access to high and low temperature setting
  - If temperature is greater (less) than the high (low) setting and the power setting is cool (heat), the thermostat will try to drive the temperature down (up) to this setting's value

Remote Version of the Same Service

- Each exposed method must declare it throws java.rmi.RemoteException
- All arguments, return types, and exceptions must be either primitive, Remote, or Serializable
- Ensures the interface will be able to be implemented regardless of transport

Service Interface

```
public interface Thermostat {
    FanSetting getFanSetting()
        throws RemoteException;
    boolean setFanSetting(FanSetting setting)
        throws RemoteException;
    PowerSetting getPowerSetting()
        throws RemoteException;
    boolean setPowerSetting(PowerSetting setting)
        throws RemoteException;

// Interface continued...
```

Service Interface (Cont.)

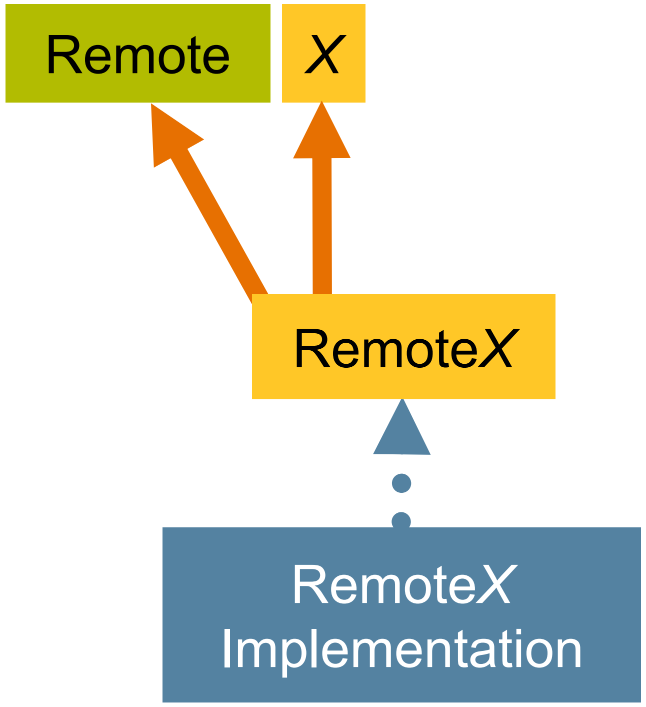
```
    Temperature getLowSetting()
        throws RemoteException;
    boolean setLowSetting(Temperature temperature)
        throws RemoteException;
    Temperature getHighSetting()
        throws RemoteException;
    boolean setHighSetting(Temperature temperature)
        throws RemoteException;
    Temperature getTemperature()
        throws RemoteException;
}
```

Remote Thermostat Service Interface

- Extends Thermostat Interface
- Extends java.rmi.Remote

```
public interface RemoteThermostat
       extends Remote, Thermostat {
}
```

Remote *X* Service Classes



Stub, Proxy, and Smart Proxy

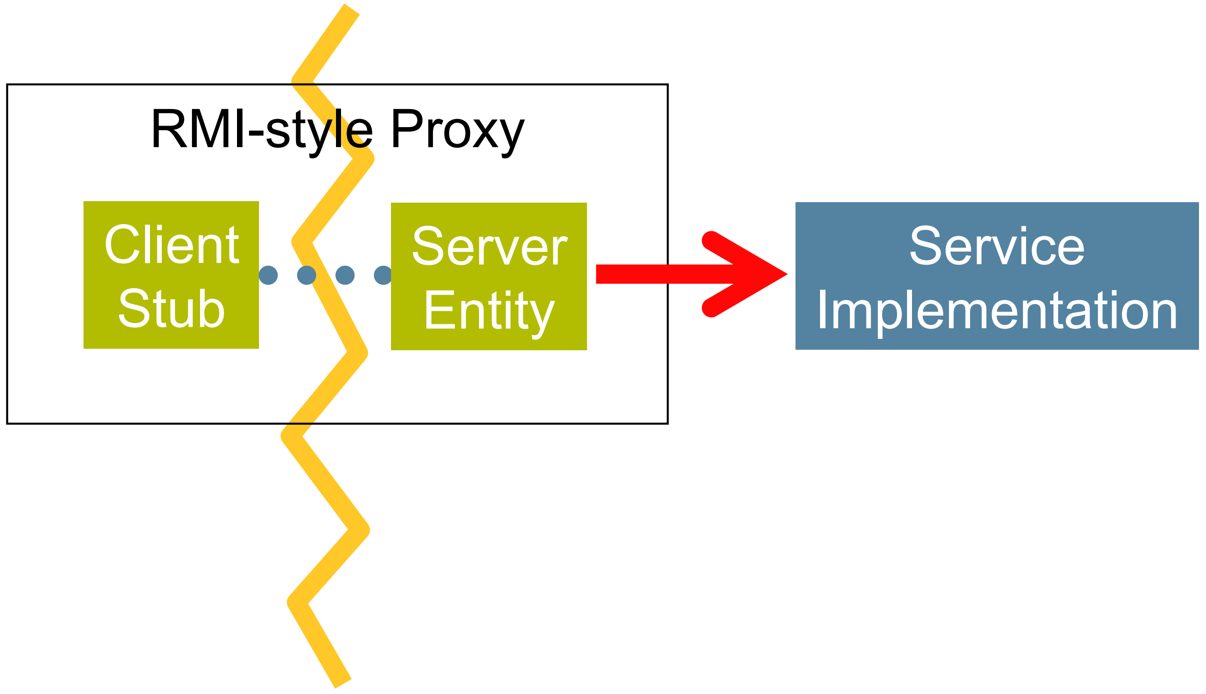
From Dan Creswell’s blog

- Stub
  - Client-side code that handles remote communication
  - Generated either by rmic or, lately, auto-generated
- Proxy
  - Registered with lookup service
  - May be either stub or Serializable object
- Smart Proxy
  - Proxy that is not a stub; just a Serializable object
  - Might contain a reference to a stub

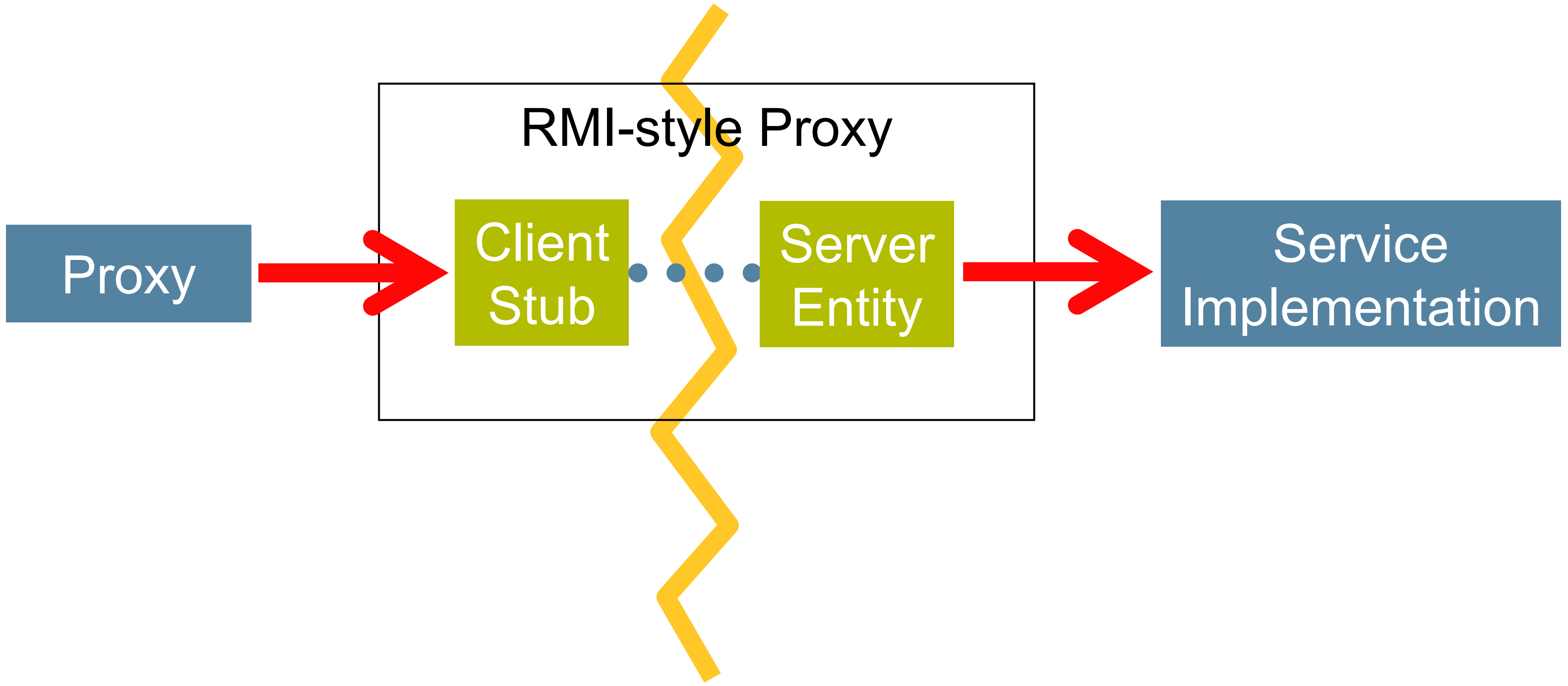
Advantages of a Smart Proxy

- Can be completely standalone—no remote calls
- Can wrap a stub and provide additional client-side functionality like caching, security, etc.
- Hide the communication protocol
  - Can use RMI, SOAP, REST, Java Message Service (JMS), FTP, etc.
- Provide an API to the client with a different level of granularity than the server
- Act as adaptor to provide a single interface that communicates with multiple backend servers

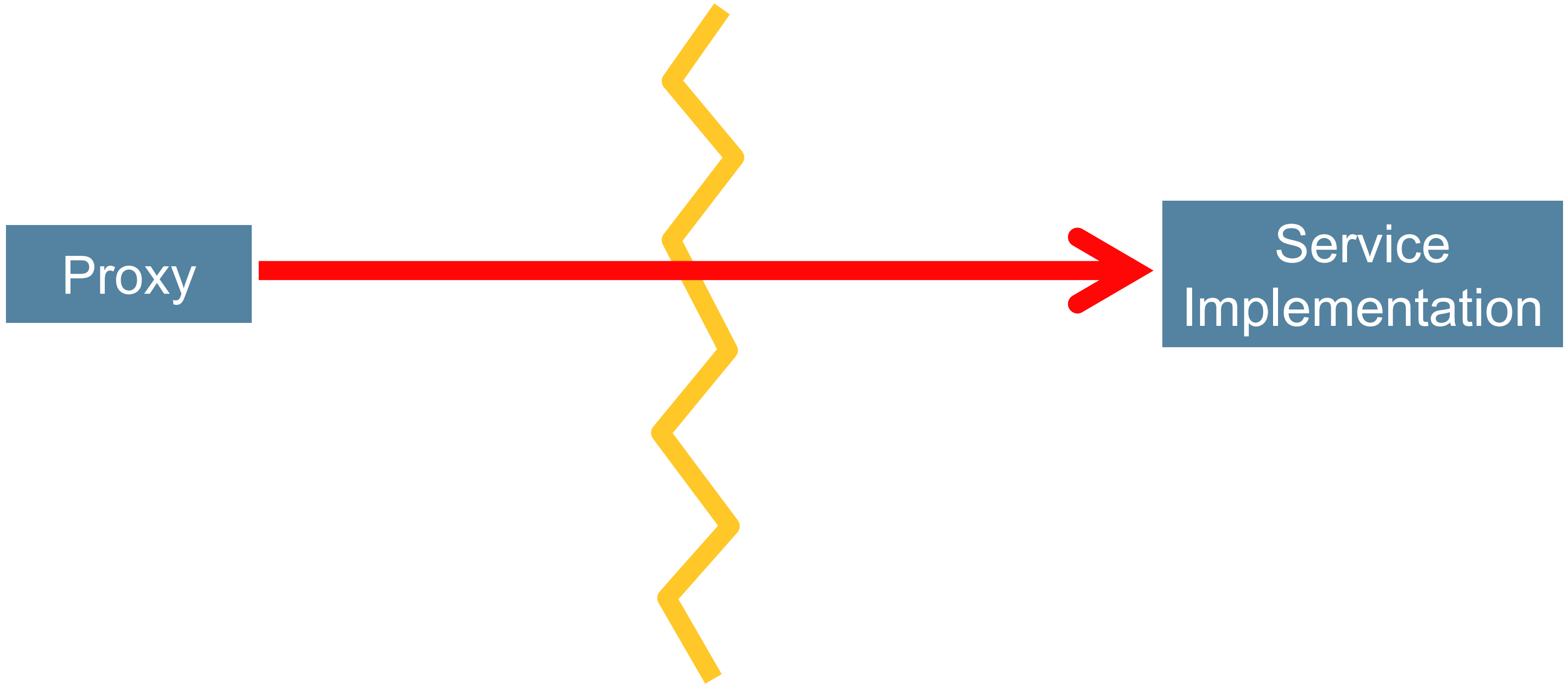
Stub


RMI-style Proxy
Client Stub
Server Entity
Service Implementation

Smart Proxy With a Stub Reference



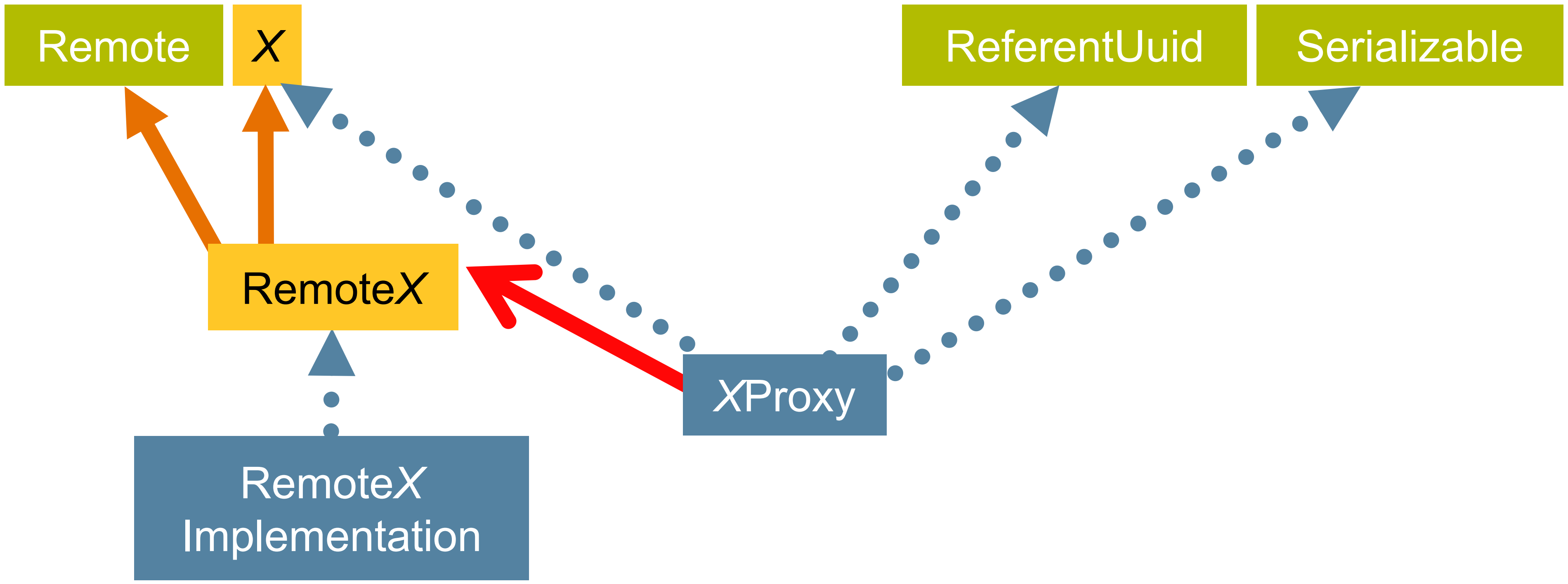
Smart Proxy Without a Stub Reference



Create a Service Proxy

- Implements Service interface
  - Not the Remote Service interface!
- Implements java.io.Serializable
  - Not Remote since it doesn’t make remote calls
- May implement net.jini.id.ReferentUuid
  - Allows proxy to express identity of its service
- May have reference to the Remote Service

X Service Classes and Service Proxy



Agenda

Quick Introduction

A Service and Its Proxy

Configuration and Administration

Packaging and Launching Services

Discovering and Using Services

User Interfaces

Thermostat Service Implementation

Business Logic

- Method to inject local service instance
- Delegate all service interface calls to local service instance

Framework Logic

- Create a service proxy
- Discover lookup services
- Register the service proxy with the lookup services
- Maintain Leases

Jini Network Technology Framework—Create Proxy

- Create a standalone smart proxy or use an instance of net.jini.export.Exporter to prepare the Remote Service
- Exporter creates a proxy from the Remote service
  - `Remote proxy = exporter.export(remoteImpl);`
  - Proxy implements Remote and Service interface
- Exporter defines how proxy and service instance communicate
- Can wrap this proxy within a smart proxy

Jini Network Technology Framework— Supplied Exporters

- net.jini.jrmp.JrmpExporter
  - Adaptor to java.rmi.server.UnicastRemoteObject
- net.jini.jeri.BasicJeriExporter
  - Enables very sophisticated security constraints
  - Provides greater transport customization
  - Generates reflective stubs even with Java Development Kit (JDK™) 1.4
- net.jini.iiop.IiopExporter
  - For use with Java RMI-IIOP (CORBA)

Java RMI-IIOP = Java Remote Method Invocation over Internet Inter-ORB Protocol Technology (Java RMI-IIOP Technology)

Jini Network Technology Configuration

- Jini Network Technology Configuration separates logic from settings
- Enables runtime decision of service configuration
- Need code to get configured items into service
- Configuration files look like pseudo Java code
  - Items must be initialized through constructors or static method factories
- *Not the same thing as Spring Inversion of Control*

Thermostat Service Configuration

```
String component = getClass().getName();

// Throws ConfigurationException if not found
setThermostat((Thermostat)configuration.getEntry(
  component, “thermostat”, Thermostat.class));

// Provides a suitable default if not found
Exporter exporter = (Exporter)
  configuration.getEntry(component, "exporter",
  Exporter.class, new JrmpExporter());
```

Thermostat Configuration File

```
import net.jini.discovery.LookupDiscovery;
import net.jini.jrmp.JrmpExporter;
import org.jini.thermostat.PseudoThermostatDriver;

org.jini.thermostat.DefaultThermostatService {
    // Local service implementation
    thermostat = PseudoThermostatDriver.create();

    initialLookupGroups = LookupDiscovery.ALL_GROUPS;
    exporter = new JrmpExporter();
}
```

Jini Network Technology Configuration File Checking

- Configuration file errors are easy to make
- IDEs don't yet provide real-time checking
- Starter Kit provides a file checking application
  - Checks for missing import statements
  - Checks for invalid formats
- `java -jar ${jini_lib}/checkconfigurationfile.jar \`
  `-cp ${classpath} ${path_to_config}/file.config`

Jini Network Technology Framework—Discover Registries

- Use dynamic discovery to find Lookup Registries
  - May be constrained by group name
  - May be constrained by location *(host-port* pair*)*
- Continue to discover other Registries
- Manage set of Registries as they come and go
- Can be handled by creating an instance of net.jini.discovery.LookupDiscoveryManager

Jini Network Technology Framework— Register Proxy

- Register the service proxy
  - Qualify it with Entry attributes
- Manage the lease of the registered proxy
  - Required to renew the lease regularly
  - Must do this for each registry
- Can all be handled by creating an instance of net.jini.lookup.JoinManager

Motivation to Make a Service Administrator Interface

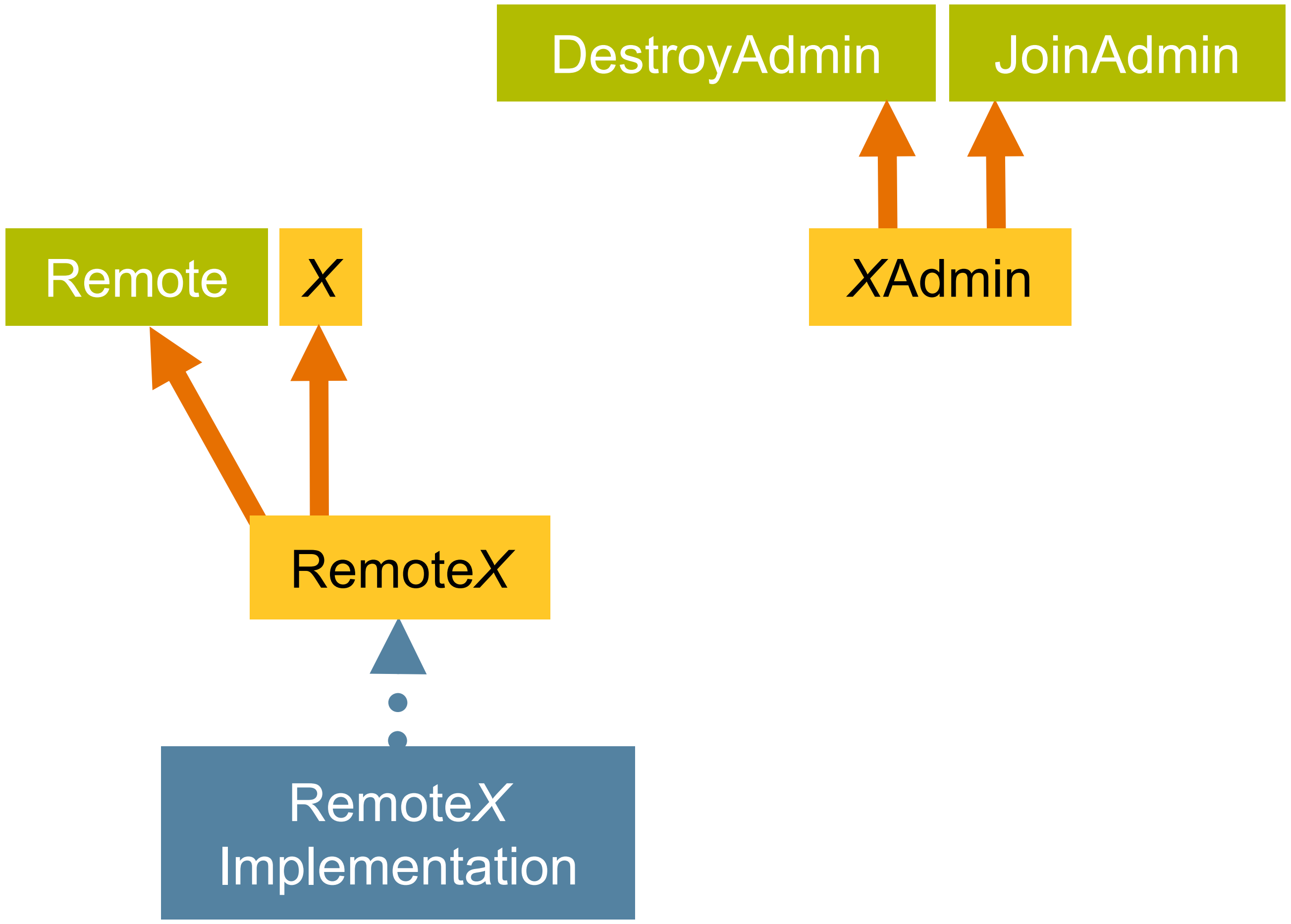
- Admin interface distinct from service
- Can constrain use to permitted service administrators only
- Permits remote control over service
  - Modify join groups and lookup locators
  - Modify service attributes
  - Destroy a running service

Create a Service Admin Interface

- Extend net.jini.admin.JoinAdmin
- Extend com.sun.jini.admin.DestroyAdmin
- Define service-specific administration methods

```
public interface ThermostatAdmin
       extends JoinAdmin, DestroyAdmin {
}
```

X Service Administrator Classes



Exposing Service Admin Interface

- Remote Service interface extends
  - net.jini.admin.Administrable
  - Service Admin interface
- Remote Service implementation handles these additional methods

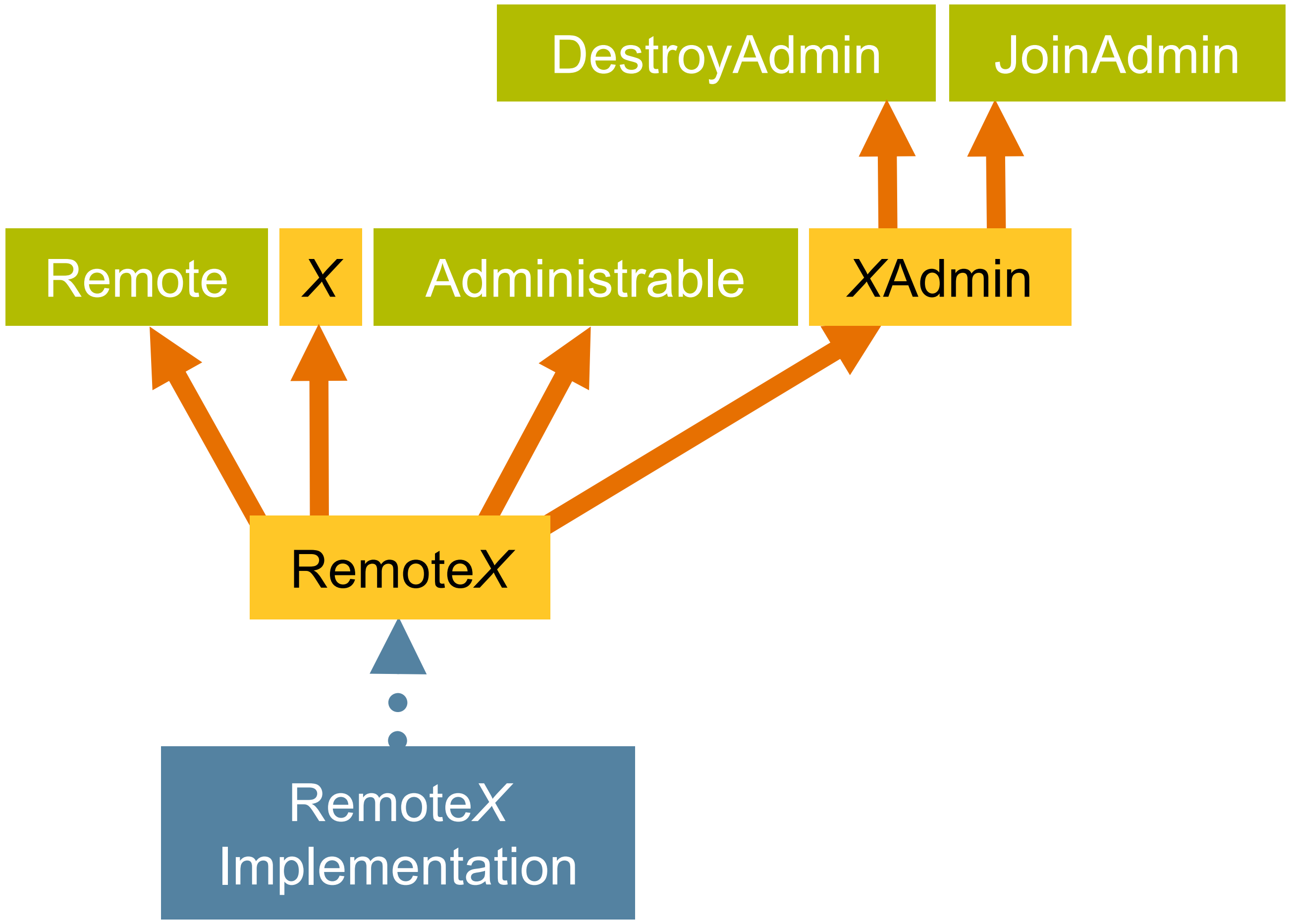
Administrable Remote Thermostat Service

- **Extends Thermostat Interface**
- **Extends java.rmi.Remote**
- Extends net.jini.admin.Administrable
- Extends ThermostatAdmin Interface
  - Handles Thermostat administration methods

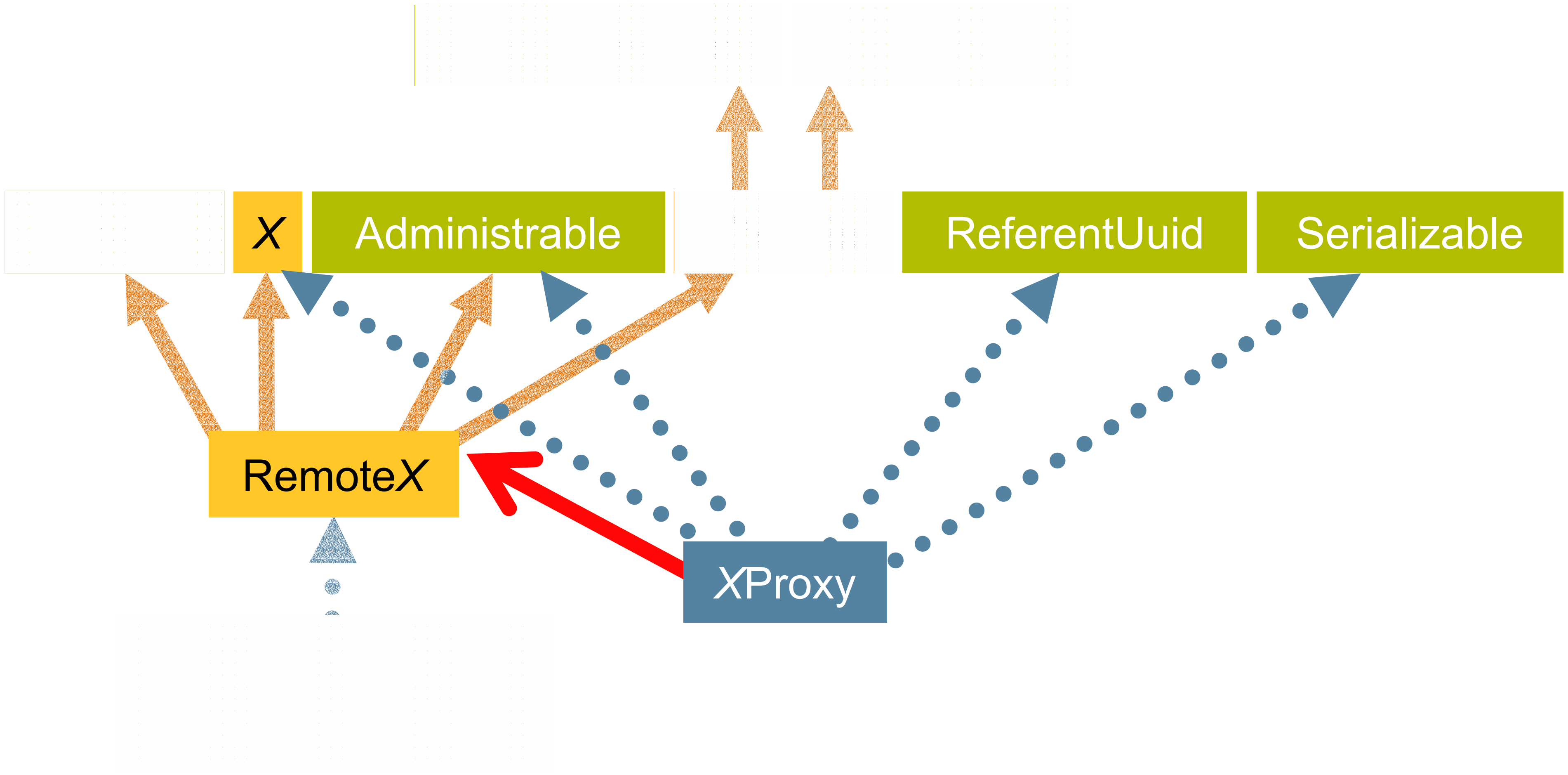
```
public interface RemoteThermostat extends Remote,
   Thermostat, Administrable, ThermostatAdmin {

}
```

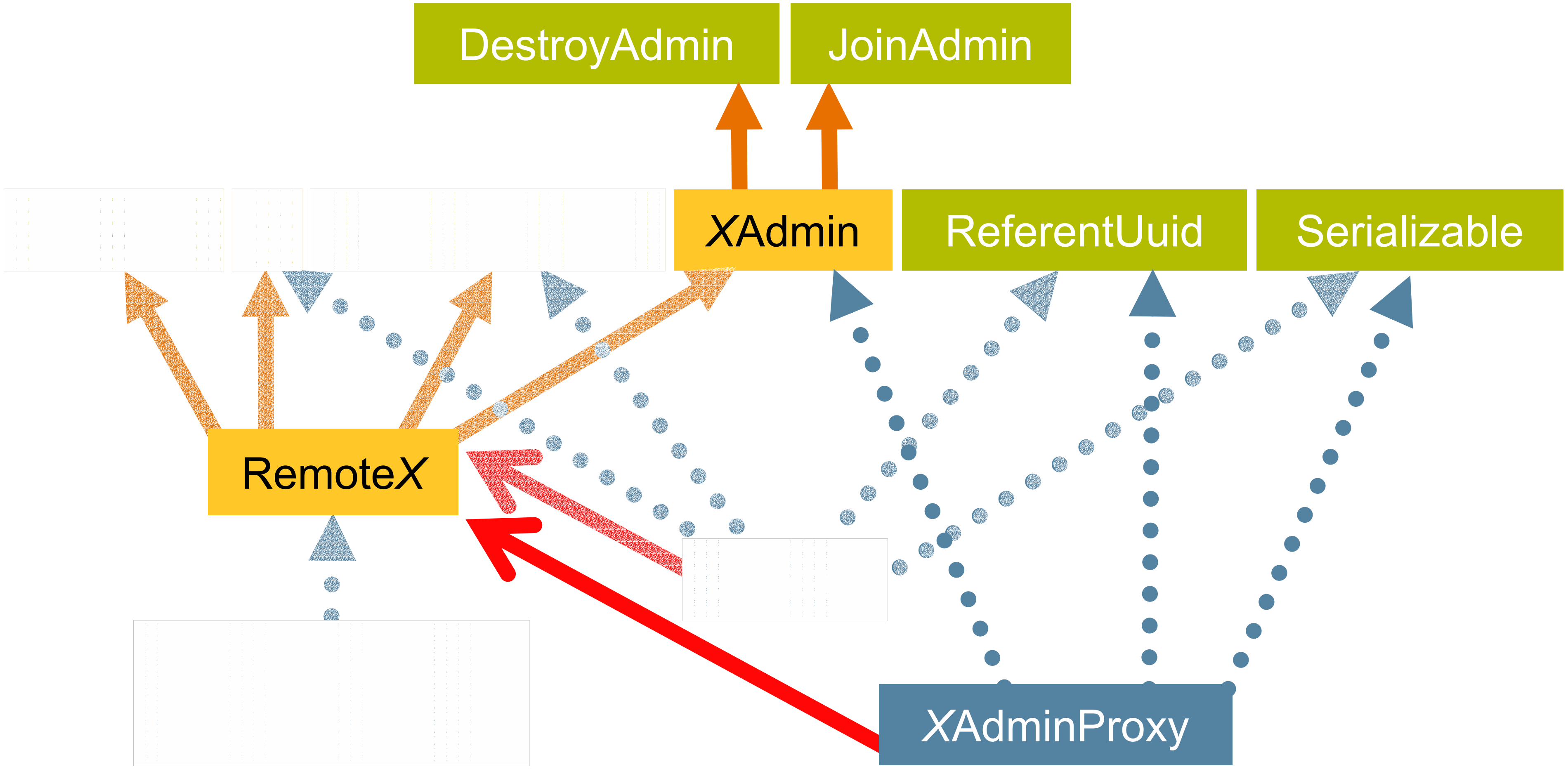
Administrable *X* Service Classes



Administrable *X* Service Classes and Service Proxy



Administrable Service X Classes and Service Proxies



Agenda

Quick Introduction

A Service and Its Proxy

Configuration and Administration

Packaging and Launching Services

Discovering and Using Services

User Interfaces

Packaging Up the Service

- Need to compile the code
  - Include jsk-platform.jar in the classpath
    - May also need jsk-lib.jar
  - jini-core.jar and jini-ext.jar now deprecated
- Create a Java Archive (JAR) file using the service name
  - Thermostat.jar   *// CONVENTION*!
  - Use JAR file to launch the service

Packaging Up More

- Identify classes that download to the client
  - Not all of the classes in Thermostat.jar
- Same name but with –dl suffix
  - Thermostat-dl.jar   *// CONVENTION!*
- Make download jar accessible
  - Can use a web server if available
  - Alternative to a web server exists
- Location of the download JAR file needs to be attached to the registered proxy

ClassDepAndJar

- Ant task that identifies the classes that should go into the download JAR file
  - Matching Maven plugin exists
- Needs some help to identify the top classes to include and what can be safely excluded
  - Especially true with classes identified only in the configuration files
  - Standard packages like java and javax can be excluded since they will exist locally

Launching Jini Network Technology Services

- Start services using ${jini_lib}/start.jar
- Argument is a launch configuration file
    - Configuration file contains an array of com.sun.jini.start.ServiceDescriptor instances
- Security policy file must be specified
    - Required for dynamic code download
    - Start security policy not the same as service policy

Launch Configuration File

```
com.sun.jini.start {
  private static jiniLib = "${jini_home}${/}lib${/}";
  private static codebaseRoot = "http://" +
    ConfigUtil.getHostName() + "/";
  private static policy = jiniLib +
    "installverify${/}support${/}jsk-all.policy";
  private static dlJarPath = // Classpath to dl jars

  serviceDescriptors = new ServiceDescriptor[] {
    // Service Descriptor definitions...
  };
}
```

Descriptors for Support Services

```
new NonActivatableServiceDescriptor(
    "", policy, jiniLib + "classserver.jar",
    "com.sun.jini.tool.ClassServer",
    new String[] { "-dirs", dlJarPath }),

new NonActivatableServiceDescriptor(
    codebaseRoot + "reggie-dl.jar" + " " +
    codebaseRoot + "jsk-dl.jar",
    policy, jiniLib + "reggie.jar",
    "com.sun.jini.reggie.TransientRegistrarImpl",
    new String[] { }),
```

Descriptor for Thermostat Service

```
new NonActivatableServiceDescriptor(
    codebaseRoot + "Thermostat-dl.jar" + " " +
    codebaseRoot + "jsk-dl.jar",
    policy,
    ThermostatClasspath,
    "org.jini.thermostat.DefaultThermostatService",
    new String[] { thermostatConfig } // Args
)
// Calls constructor of form:
// DefaultThermostatService(String[], LifeCycle);
```

Service Prerequisites

- Registered with at least one registry
- Download jar remotely accessible
- Starter Kit includes:
  - ClassServer—Service to offer up JAR files and classes from specified directories
  - Reggie—Reference lookup registry implementation
  - Browser—Monitors registered services

Agenda

Quick Introduction

A Service and Its Proxy

Configuration and Administration

Packaging and Launching Services

Discovering and Using Services

User Interfaces

Discovering Jini Network Technology Services

- Use dynamic discovery to find Lookup Registries
  - May be constrained by group name
  - May be constrained by location (*host-port* pair)
- Discover service by interface type
  - Entry attributes can help filter discovery results
- Can be handled by creating an instance of net.jini.lookup.ServiceDiscoveryManager

Simple Thermostat Example

```
Configuration c =
   ConfigurationProvider.getInstance(args, ...);
ServiceFinder serviceFinder = new ServiceFinder(c);
Thermostat thermostat = (Thermostat)
   serviceFinder.getService(Thermostat.class);
// Finder uses ServiceDiscoveryManager to get service


log("Power Setting " + thermostat.getPowerSetting());
log("Fan Setting " + thermostat.getFanSetting());
log("Temperature " + thermostat.getTemperature());
```

Lookup Caches for Jini Network Technology Services

- What to do when RemoteException is thrown?
  - Likely due to some network trouble
  - Drop the remote service reference
  - Rediscover a new reference
- Rather than pay the cost to lookup each remote Registry, ServiceDiscoveryManager provides a way to dynamically cache current remote state
  - `LookupCache cache = sdm.createLookupCache(...);`
  - Provides a way to remove unusable services
    - `public void discard(Object serviceReference);`

Agenda

Quick Introduction

A Service and Its Proxy

Configuration and Administration

Packaging and Launching Services

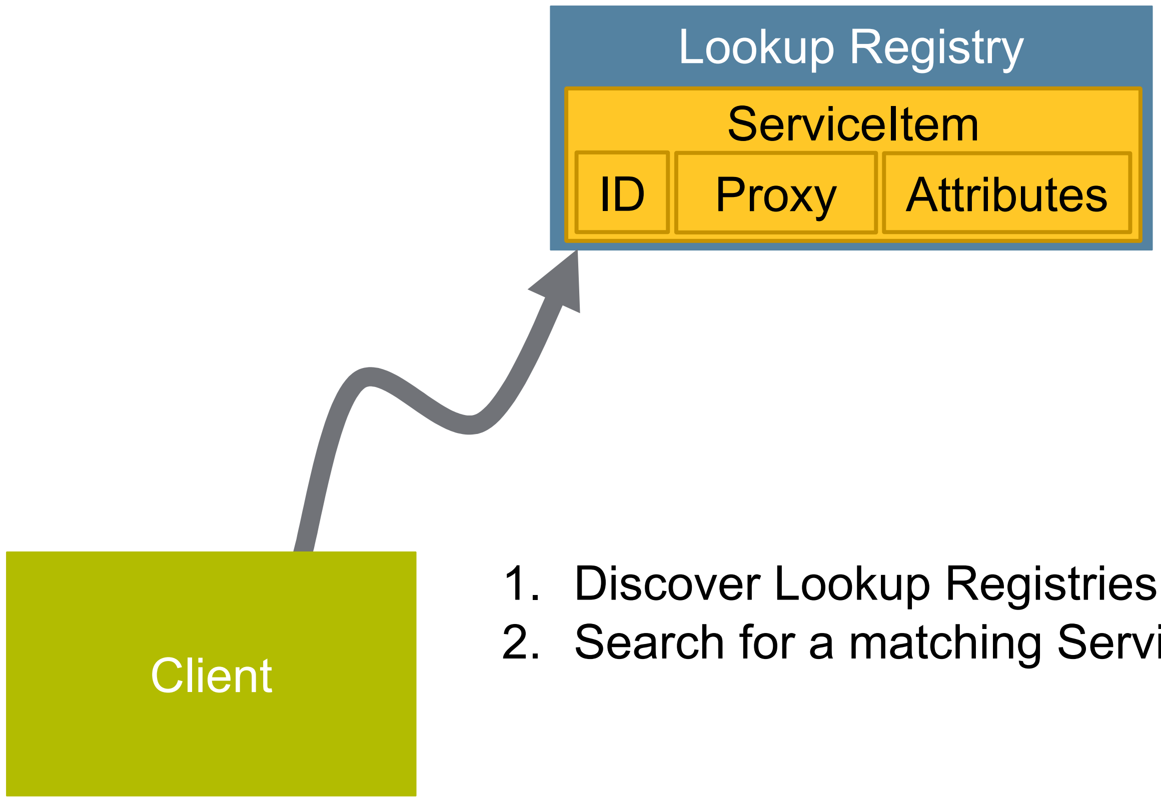
Discovering and Using Services

User Interfaces

ServiceUI Project

- Project started to provide a way for UIs to be attached to Jini network technology services
- Not currently bundled with Jini Technology Starter Kit
  - `http://www.artima.com/jini/serviceui/`
- Don’t want all the UI code to be directly attached to the service
  - Entry attribute indicates UI Factory
- Requires creation of additional download JAR file(s)
  - Also need to add to codebase in starter configuration

Jini Network Technology Client With ServiceUI



1. Discover Lookup Registries
2. Search for a matching ServiceItem

Finding a Service With a UI

- Get discovered service’s attributes
- Find instance of UIDescriptor attribute
- Match Toolkit and Role
- Unmarshall UIFactory
- Is factory an instance of JFrameFactory?
- Get frame with service item as argument
- Show frame

ServiceUI Caveats

- No guarantee to exist at all
- Created with a reference to ServiceItem
  - No knowledge of a specific implementation; just a reference to the service interface
- UI only lasts as long as service
  - If service dies, UI dies
- Any authentication must happen with the service
  - The UI cannot handle this since it is created with a reference to the service

DEMO

A ServiceUI Client

Summary

- Jini network technology acknowledges the fallacies of distributed computing rather than trying to ignore/hide them
- Jini network technology framework provides the resources to create robust and secure distributed services
- High-level concepts were introduced including service class structure, lookup discovery and registration, leasing, configuration, security, service launching, client-side discovery and use of services, and user interfaces

Summary (Cont.)

- Many of Jini network technology’s more advanced features can be added incrementally and sometimes with only a change to the service’s configuration file
- While not easy to learn, once the path through the framework is understood, building Jini network technology services is straightforward

Resources

- http://www.jini.org/
- http://jini.dev.java.net/
- http://www.jini.org/wiki/Mailing_Lists
- Code: http://rport.raba.com/StraightforwardJini/

Q&A
Straightforward Jini

James Morgan

Clark Richey

Straightforward Jini™ Network Technology: Jini Services for the Uninitiated

James Morgan and Clark Richey

Principal Consultants
SRA // RABA Center
http://rport.raba.com/StraightforwardJini/

TS-1161