Jini™ Connection Technology Architecture Overview

Bill Day
bill.day@sun.com
www.biliday.com

Technology Evangelist
Sun Microsystems

Agenda

- Why Jini?
- What is Jini?
- Jini Architecture Overview
- The Details
- Example Service Models
- Subtopics on:
  - JavaSpaces™
  - Jini Surrogate Architecture
  - Related Technologies
- Availability and Resources
Distributed computing is more difficult than local computing because of:

- Network latency
- Concurrency issues
- Memory management
- Inevitable partial failure
Evolution of distributed computing

Key insight: Remote increasingly looks local

What is Jini?
What is Jini™?

- Jini addresses the problems of distributed computing using a set of simple interfaces and protocols
- Jini enables spontaneous networks of software services (and devices) to assemble into working groups of objects, or Federations
- Jini enables self-healing when one or more devices are removed from the Federation
- Jini builds on the Java™ 2 Platform

What is Jini™?

- Jini is usable for hardware as well as software services
  - Everything is represented by Java Objects.
  - Everything is located and accessed through Java Interfaces.
The Value of Jini™

- Consumers
  - Plugging in networked devices and application software is as simple as plugging in a phone today
- Service providers
  - Simplifies management of application service delivery

The Value of Jini™

- Product manufacturers
  - Opens entirely new markets (diagnostics, post-purchase upgrades and software enhancements, customer care)
- Programmers
  - Simplifies the task of writing distributed applications by freeing them from network topology
Jini Architecture Overview

Jini™ Architecture Overview

Jini Services

• JavaSpaces™
• Lookup
• Discovery/Join
• RMI

• Java
• Java
• Java
• Solaris
• Mac
• Windows
• SPARC™
• PPC
• x86

Jini Infrastructure

Other Services

•••
Java Technology Features that Benefit Jini™

<table>
<thead>
<tr>
<th>Feature</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Virtual Machine</td>
<td>Homogeneous network</td>
</tr>
<tr>
<td>Portable object code</td>
<td>Architecture independence</td>
</tr>
<tr>
<td>Downloadable code</td>
<td>Dynamic environment</td>
</tr>
<tr>
<td>Unified type system</td>
<td>No impedance mismatch</td>
</tr>
</tbody>
</table>

Jini Extends the Java Platform

<table>
<thead>
<tr>
<th>Infrastructure</th>
<th>Programming Model</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jini</td>
<td>Discovery</td>
<td>Lease</td>
</tr>
<tr>
<td></td>
<td>Lookup</td>
<td>Event</td>
</tr>
<tr>
<td></td>
<td>Extended Security</td>
<td>Transaction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td>Java VM</td>
<td>Java technology-based APIs</td>
</tr>
<tr>
<td></td>
<td>RMI</td>
<td>Beans</td>
</tr>
<tr>
<td></td>
<td>Security</td>
<td>Swing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Details: Jini Infrastructure

Jini Lookup Service

- Repository of available services
- Stores each service as Java objects
- Clients download services on demand

Diagram:
- Lookup Service (LUS)
- Service Proxies
Using a Lookup service

- May be federated with other lookup services
- Lookup service interface provides:
  - Registration, Access, Search, Removal
Discovery and Join Protocols

- Used to find and join a group of services.
- Based on UDP multicast.
- Services advertise capabilities.
- Provide required software drivers.
- Establish reference with lookup service.
- Unicast discovery also supported.

Discovery Protocol
Join Protocol

Steps in Discovery and Join

- Jini™ enabled Object (representing a hardware and/or software service) multicasts a packet with a reference to itself
- The service receives RMI reference to Lookup service (one or more)
- The service joins a Federation by placing an object representing its capabilities into the Lookup service for other clients and services to use
Discovery and Join: Traditional versus Jini™ Technology

Traditional Technology
- Device is added to computer
- Device definition is edited in node configuration table
- Computer is restarted
- Updated configuration table is pushed to network device map

Jini Technology
- Device is plugged in and powered up
  - Lookup service receives multicast
  - Lookup service authenticates source
  - Lookup service sends advertisement
  - Device joins Lookup service

Finding a Service
- To find a service, a Jini client locates service by querying a Lookup Service by type (Java Interface).
- Code moves from service to client via Lookup Service.
- Code needed to use service is dynamically loaded on demand.
Finding a Service

Using a Service

- Communication is between service and its proxy
- Independent of wire protocol
- Protocol can change without affecting client
- RMI semantics core to functionality
Using a Service

Why Jini™ uses RMI

- RMI can pass entire Java objects and their code
- Works in any compliant JVM
- Provides automatic serialization, transport, and de-serialization of objects
- Robust and configurable security options (RMI over SSL, Authenticated Principal, etc.)
The Details:
Programming using Leases, Events, and Transactions

Why Distributed Leasing?

- Problem: Partial failure in distributed systems can lead to unchecked resource consumption
- Traditional solution: system administration
  - Error-prone
  - Costly
  - Only happens when it’s too late
Distributed Leasing in Jini™

- Protocol for managing resources using a renewable, duration-based model
- Contract between objects
- Resources can be shared or private
- Provides a method of managing resources in an environment where network failures can, and do, occur

What are Leases?

- Time-based grants of resources or services
- Loose contracts between grantor and holder
- Negotiated for a set period of time
- Can be shared or exclusive
Lease Operations

- Leases may be:
  - Cancelled (explicit cleanup)
  - Renewed (explicit extension)
  - Allowed to expire (implicit cleanup)
  - Obtained and manipulated by third parties

Distributed Leasing

- Resources are allocated only as long as continued interest is shown
- Leasee is responsible to renew lease before expiry
- The network is *self-healing*
**Distributed Events**

- Extend Java™ event model to work in a distributed network
- Register interest, receive notification
- Allow for use of event managers
- Support various delivery models
  - Push, pull, filter
- Use the Distributed Leasing protocol

**Transaction model in Jini™**

- Designed for distributed object coordination
- Light weight, object-oriented
- Supports two-phase commits
- Uses Distributed Leasing protocol
- Implemented in Transaction Manager service
The Details: Jini Services

Services

- Represented by Java Interfaces
  - Separation of “how” from “what”
  - Implement in hardware or software
  - Implementations can change over time
  - Multiple implementations
- Set of services open ended
  - Devices
  - Software services
  - Business objects
Services Enable Evolution

- Base interfaces can be extended to:
  - Add new functionality
  - Combine functionality
- Old clients can use base interface
- New clients can use extended interface
- Allows evolution of implementation without changing clients or infrastructure

Services Are Defined by Java Interfaces

I'm a Printer

I'm a ColorPrinter, therefore a Printer too
Services Defined by Interfaces

I’m a PhotoCopier and a Printer

I’m a FaxMachine, a PhotoCopier, and a Printer

Client Finds Service by Interface

Give me a PhotoCopier

LUS
Client Finds Service by Interface

Network Client

Give me a PhotoCopier

LUS

Interface Printer

Interface ColorPrinter

Client Finds Service by Interface

Network Client

Give me a PhotoCopier

LUS

Interface Printer
Enabling Services

- Transaction manager
  - Default service implementation.
- Jini 1.1 Utility Services
  - Lookup Discovery service, manages discovery on behalf of its clients.
  - Lease Renewal service, receives and stores remote event notifications on behalf of its clients.
  - Event Mailbox service, manages lease renewal activities on behalf of its clients.

Service Interfaces

- Community driven
  - Printing
  - Service UI
- Don’t have to be done by Sun
- Only base types need to be common
- Interfaces can evolve
  - New apps can use new functionalities
  - Old apps still work
Legacy Services

- Jini requires Java at the network
  - Services identified by Java type
  - Proxies may need code downloaded
- Implementation can be non-Java
  - Wrap legacy language in Java
  - Make calls to legacy with JNI

Example Service Models
Example service model #1: Local device concentrator

Exposing a disk cluster as a single networked file store

Example service model #2: Remote device concentrator

Home network (AV equipment, appliances, etc.) communicate via Java Embedded Server™
Example service model #3: EJB services

- Service Client
- Service Proxy
- EJB Server

Accessing enterprise resources

Detailed Example: Jini-Enabled Printer

- GUI
- Drivers
- Jini Infrastructure
- Java VM
Discovery

- Multicast query for lookup service

Discovery

- Lookup service discovered
Join

- Printer registers service object with LUS

Locating Service

- Client asks LUS for Printer
Locating Service

- LUS returns printer service object to client

Using the Service

- Client communicates to printer through service object
JavaSpaces™

What Is JavaSpaces?

- A Jini™ Service
- A Persistent Shared Object Store
- Transactionally Secure
- Astonishingly Simple
- Based on “tuple-spaces”
  (see David Gelernter’s Linda system)
JavaSpaces Design

- Parallel programming
- Distributed systems
- Cooperating software ensembles

JavaSpaces Benefits

- Anonymity between applications
- Uncoupled communication
- Programs can communicate through time or space
- Vast savings in design and development time
JavaSpaces Concepts

- High-level communications between Java program
- SIMPLE: only a few operations to learn
- EXPRESSIVE: a large number of computing problems can be solved

JavaSpaces Programs

- Processes do not directly interact
- Processes interact indirectly through one or more spaces
What Is a Space?

- Persistent object store
- Network accessible
- Shared
- Transactionally secure
- Can store executable content

Modification of Entries

- An Entry in a space cannot be modified
- Must be taken, modified, written

process → take
process → modify
process → write
Matching Entries

- A template is used to match entries in a space
- The template is simply an entry of the type to be matched
- Fields of the entry are used for matching
- Non-assigned fields are wildcards

Entries

- Entries are collections of typed objects
- Must implement net.jini.core.entry.Entry
- Fields can be any serializable Java™ object
- Fields must be public to be matchable
- Entry needs no-arg constructor
Example Entry

- This shows a minimal entry:

```java
public class SpaceShip implements Entry {
    public String name;
    public Integer score;
    public SpaceShip() {
    }
}
```

Accessing a Space

- Jini is used to locate a space
- Sun’s implementation identifies spaces by name
- Convenience method from JavaSpaces book:

```java
SpaceAccessor.getSpace("name");
```
Space Operations

- Very small number of operations are defined on a space
- The three fundamental operations are:
  - Write
  - Read
  - Take

Write Operation

- Writes an entry to a space
Write Operation

- Instantiate an Entry
- Set its fields as necessary
- Write the entry to the space

```java
SpaceShip s = new SpaceShip();
s.name = "USS Enterprise";
space.write(s, null, Lease.FOREVER);
```

Read Operation

- Reads an entry from a space
- Copy of object is returned
- Original remains in space
Read Operation

- Build a template
- Read a matching entry from the space

```java
SpaceShip template = new SpaceShip();
template.name = "USS Enterprise";
SpaceShip s = space.read(template, null, Long.MAX_VALUE);
```

Take Operation

- Takes an entry from a space
- Matched entry is removed from space
Take Operation

- Build a template
- Take a matching entry from the space

```java
SpaceShip template = new SpaceShip();
    template.name = "USS Enterprise";
SpaceShip s = space.read(template,
    null, Long.MAX_VALUE);
```

Other Space Operations

- ReadIfExists()
  - Non-blocking read
- takeIfExists()
  - Non-blocking take
- Notify()
  - Inform me when a matching entry is written to the space
JavaSpaces Patterns

- Distributed Data Structures
  - Multiple entries stored in one or more spaces
- Protocols
  - Application-defined way to manipulate/access data structures

Example: Distributed Array

- Each array element is an entry in a space
- Entry stores content and position in array
- Protocol is defined to add/remove array elements
Example: Distributed Array

- Multiple processes concurrently access the array through the space

JavaSpaces Leases

- Based on Jini lease model
- Resources allocated for fixed time period
- Lease is specified when entries are written to space
- When the lease expires, the space removes the entry from its store
JavaSpaces Leases

- Lease time is parameter to write operation
- Specified in milliseconds in Sun’s implementation
- Lease object is returned by write()

```java
Long duration = 1000*60*5; // 5 minutes
Lease 1 = space.write(ship, null, duration);
```

JavaSpaces Transactions

- JavaSpaces operations are transactionally safe
- All operations are atomic
- Multiple operations may be grouped using a transaction object
- Based on Jini transaction model
- Transactions may span multiple spaces
- Spec supports nested transactions
Example Application

- Image processing
- Goal: perform operation on every pixel in large image
- Problem: image ops are computationally expensive
- Solution: JavaSpaces

Example Application

- Perform edge detection on this image
Example Application

- Step 1: Divide image into titles

Example Application

- Step 2: Build Entry object for each tile
- TileEntry contains code to perform operation
Example Application

- Step 3: Write tile entries to space

Example Application

- Step 4: Server process concurrently takes tile entries from space (blocking if not available)
Example Application

- Step 5: Server process executes code (contained in TileEntry) to process the tile

Example Application

- Step 6: When finished processing, server writes completed tile back to space, after setting its FINISHED flag
Example Application

- Step 7: Concurrently, client takes FINISHED entries from space (blocking if not available) and reconstructs image

Pure Scalability

- Want to process more images?
- Just add more clients/server
Advantages

- Servers can be any platform that supports Java
- Can mix and match servers
- Server needs no prior knowledge of image operation
- Highly scalable
- Highly flexible
- Highly reliable
- Simple design

JavaSpaces™ at JavaOne 1998
Not every device supports a full Java™ 2 Standard Edition (J2SE) virtual machine.

Even if a device does support Java technology (say through J2ME), it may not have enough memory or meet other requirements to support a full Jini implementation.

However, any networkable device can proxy via a surrogate that supports Java and Jini technologies.
Jini Surrogate Architecture

- Allows almost any device to participate in Jini federations, including small J2ME devices or even non-Java devices.
- Independent of device type.
- Independent of network type.
- Preserves Jini Plug and Work functionality.
- Developed as community project at developer.jini.org/exchange/projects/surrogate/index.shtml
Related Technologies

**Jini**
- Defines a set of protocols for dynamic federations of services
- Designed for dynamic and distributed environment

**EJB™**
- Defines server-side component model
- Defines hosting environment of components
- Designed for relatively "static" server-centric environment
## Jini and CORBA/DCOM

- **Jini**
  - Java™ technology centric
  - Code movement during runtime
  - Subtype support
- **CORBA/DCOM**
  - Cross-language support
  - No code movement
- **Integration**
  - CORBA/DCOM components can be wrapped as Jini services

## Jini and UPnP

- **Jini**
  - Code movement during runtime
  - Service-centric
- **UPnP**
  - No code movement
  - Device-centric
- **Integration**
  - UPnP devices can be wrapped as Jini services
Jini and OSGi

- OSGi
  - Defines how smart consumer and small business appliances can be connected with commercial Internet services
- Jini and OSGi are complementary
  - Jini can provide unified service framework for OSGi network

Jini and Bluetooth

- Bluetooth
  - Open standard for wireless proximity networking communication
  - Based on low-cost short-range radio link
  - Ad-hoc connection among mobile devices
- Jini and Bluetooth
  - For IP-based Bluetooth devices, Jini should work with no modifications
  - For non-IP-based Bluetooth devices, proxy-based Jini services can be built
Jini and HAVi

- HAVi
  - Stands for Home Audio Video Interoperability
  - Network interoperability of audio and video devices from multiple vendors
    - Transfer of digital contents between HAVi devices
  - Java technology based
- Jini and HAVi integration
  - HAVi software modules can be built as Jini services

Availability and Resources
Where are we today?

- 47,000 FCS downloads
- Average 2,000 downloads/month
- 29,000 SCSL Licensees
- 23 Commercial Licenses
- Estimated 50,000 active Jini developers
- Four Jini Community meetings to date
- 103 Jini Community projects

Jini Partners Include...

Adaptive Networks
America Online
Axis Communications
BEA Systems
Bosch Siemens
Cisco Systems, Inc.
Computer Associates
Creative Design Solutions
Dallas Semiconductor
Phoenix Technologies
Sony Corporation
Palm Computing
Inprise Corporation

Bull
Echelon
Encanto
Funai
Kinkos Corporation
Kodak
Metrowerks
Motorola
Novell
Nokia
Oki Electric
Philips

Quantum
Samsung
Seagate
Seiko
Epson
Canon
Sharp
Symbian
3Com
Tatung
Toshiba
Xerox Corp.
Jini Technology
Starter Kit v1.1

- No changes to the Core.
- Focus of release is to add “helper” utilities and services to aid developers in building Jini clients and services.
- JavaSpaces Technology rolled into the extended platform (JXP) in this release.

More Information

- Main Sun Jini page
  - www.sun.com/jini
- Jini Community
  - www.jini.org
- Mailing lists
  - archives.java.sun.com/archives/jini-users.html
  - archives.java.sun.com/archives/javaspaces-users.html
More Information

- Jini™ Technology Webcasts
  - java.sun.com/jdc/onlineTraining/webcasts
- 3rd-party Jini websites
  - www.artima.com/jini
  - www.cdegroot.com/cgi-bin/jini
  - www.kedwards.com/jini/index.html
  - www.litefaden.com/sv/jd/

More Information

- Online tutorials
  - pandonia.canberra.edu.au/java/jini/tutorial/Jini.xml
  - www.oreilly.com/catalog/jininut/chapter/ch04.html
  - www.eli.sdsu.edu/courses/spring99/cs696/notes/index.html
Books

For more, please visit:
www.sun.com/developers

© Copyright 2000 Sun Microsystems, Inc., All rights reserved.

Sun, Sun Microsystems, the Sun logo, Solaris Operating Environment, Java, Jini, JavaOS, PersonalJava, JavaPhone, JavaTV, EmbeddedJava, Java Card, JDBC, the Java Coffee Cup logo, JDK, Java virtual machine, JavaCheck, JavaScript, Java Filter, JavaBeans, Java Blend, JavaOS for Consumers, JavaOS for Business, Java Embedded Server, Java Dynamic Management Kit, JavaEngine, 100% Pure Java, HotJava Views, HotJava and The Network Is The Computer are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the United States and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.