The *Jini* System Architecture

Alexander V. Konstantinou  
akonstan@cs.columbia.edu

CUCS Graduate Student Seminar  
*December 8, 1998*

Copyright (c) 1998, Alexander V. Konstantinou (akonstan@cs.columbia.edu)

---

**Presentation Outline**

- Introduction/Overview
- Architecture
- Infrastructure
- Programmatic Interface
- JavaSpaces
- Summary

Copyright (c) 1998, Alexander V. Konstantinou (akonstan@cs.columbia.edu)
What is Jini?

• New distributed system architecture
• Sun R&D project inspired by Bill Joy
• Goal to simplify interaction with networks
• Built around model of clients looking for services
• Plug-and-participate (spontaneous networking)
• New class of network services

Why Change?

What has changed in the picture?

“Windows NT is 16.5 million lines of code that will never be debugged”
-- Bill Joy
Why Jini?

- You are the new system administrator,
- Computers are nowhere,
- The one computer is everywhere!
- Networks are becoming ubiquitous
- Consumer electronics devices are getting smarter
- Current networks rely on static configuration and are hard to administer

Jini thoughts “Services”

- Jini members federate to share access to services
- A service is an entity that may be used by a person, a program or another service
- A service may be a computation, storage, a communication channel, a software filter, a hardware device, or another user
- Services are composed for performance of a particular task
Key Concepts

• Federating groups of devices and software components into single distributed system
• Single Jini system targeted to workgroup
• Members of federation agree on basic notions of trust, administration, identification and policy
• It is possible to federate Jini systems for larger organizations

Jini Environment

• Entertainment devices will continue to increase their “digital component”
• Existence of network of reasonable speed (≥10mbps)
• Devices have some memory & processing power.
  – Proxies provided for “dumb” devices
• Java-technology centered : bytecode everywhere
Base Java Features

• Java **byte-code** is platform independent
  – Other languages may be compiled to byte-code (Ada)

• Java objects may be **serialized** into stream of bytes
  – Lack of pointers & strong typing allows for deep-copy
  – Mobile objects: code + data saved as a byte-stream

• Java **RMI** (Remote Method Invocation)

• Java **security** model - beyond the sandbox
  – Signed applications + domain-based access control

Architecture

• Jini **infrastructure** :
  – Discovery and Join
  – Lookup

• Jini **services** :
  – JavaSpaces

• **Programming model** :
  – Leasing, Transactions, Distributed Events

Extremely light-weight: Jini classes take 48Kb
Jini Infrastructure

Discovery

- Host requirements:
  - functioning JVM
  - configured network stack
- IP stack requirements:
  - IP address
  - unicast TCP, multicast UDP
  - RMI stub export mechanism (e.g. HTTP)
- Group: an arbitrary string that acts as a name (recommend DNS names eng.sun.com)
  - initially empty (new service)

Copyright (c) 1998, Alexander V. Konstantinou (akonstan@cs.columbia.edu)
Join

- Register with each lookup service discovered
- Persistent state:
  - Service ID
  - Lookup entry
  - Groups
  - Lookup services

Joining entity

Lookup entry

lookup service

Retrieve attributes from persistent storage. If this is a "new" service, generate appropriate default values

Write com.hp.PrintService service proxy object and an entry with attributes indicating manufacturer, model, speed, resolution, duplex printing support, etc.

Copyright (c) 1998, Alexander V. Konstaninou (akonstan@cs.columbia.edu)

Lookup

- Lookup service maintains flat collection of service items
- Items contain the RMI stub or proxy object used to access the service + an extensible collection of descriptive attributes
- Entries may be manipulated graphically by administrator (JavaBeans)

Lookup printer based on service type and attributes

Return matching services (up to max match)

Invoke print method on local proxy
Proxy uses internal protocol to transmit print request

Copyright (c) 1998, Alexander V. Konstaninou (akonstan@cs.columbia.edu)
Entry

• **An entry** is a class containing a number of public fields of object type
• Service may provide **multiple** entries (set of sets)
• **Exact** lookup semantics

• **Design issues:**
  – matching cannot always be *automated* (reduce)
  – attributes are mostly *static* (order of minutes)
  – *humans* need to understand most attributes
  – attributes can be changed by services or humans, but not both
  – attributes must interoperate with *JavaBeans* components

Programmatic Interface

Copyright (c) 1998, Alexander V. Konstantinou (akonstan@cs.columbia.edu)
Distributed Leasing

- **Failures** in distributed systems result in unbounded resource consumption growth
- **Services in Jini are leased** based on time
- Jini provides a simple **interface** for requesting, renewing, and canceling a lease
- Lease time may be **absolute** or **durational**

Distributed Events

- Remote events contain: *source, event ID, sequence number,* and *hand-back object*
- Notification of remote event may **fail** (Exception)
- Remote event registration is **leased**
- **Third party objects** may act as filters/proxies:
  - Store-and-forward agents (offload generator object)
  - Notification Filters (offload receivers)
  - Notification Mailboxes (offline receipt)
Transactions

- Provides **coordination mechanism** (API) for performing a distributed two-phase commit
- **No monitors:** objects responsible for correct implementation
- Transaction created and overseen by *manager*
- Semantics represented by *semantic objects*
- Default action semantics preserve *atomicity, consistency, isolation, and durability*

Transactions (cont.)

```
Participant A
  \- op A(Transaction, …)
      \- join
          \- Manager
              \- join
                  \- Client
                      \- Transaction
                          \- long id
                              \- create
                                  \- Transaction.create
                                      \- Transaction.createElement
                                          \- semantic object
                                              \- active
                                                  \- prepare
                                                      \- voting
                                                          \- abort
                                                              \- abort
                                                                  \- commit
                                                                      \- committed

Participant B
  \- op B(Transaction, …)
      \- join
          \- returns
              \- active
                  \- prepare
                      \- voting
                          \- abort
                              \- abort
                                  \- prepared
                                      \- notchanged
                                          \- cleanup
```

Copyright (c) 1998, Alexander V. Konstantinou (akonstan@cs.columbia.edu)
JavaSpaces

• Distributed algorithms as flows of objects
• JavaSpaces implementations provide reliable distributed storage for objects
• JavaSpaces store Entries with public fields
• Field lookup is exact match or don’t care
• Operations: write, read, take, notify
• Entries written into JavaSpace are leased
• Operations may be part of distributed transaction

Open Issues

• Network infrastructure: DHCP, Multicast
• Schema standardization/evolution
• Embedded Java future
• Adoption by device manufacturers
• Jini vs. directory-based approaches
Summary

• Jini challenges the predominant (PC) network and computer architecture.
• Killer App?
  – Home network
  – Mobile computing: navigating both local & home-base environment
  – Robust computing: service (object) redundancy
• Programming model may survive independently
• Mirror worlds?

Companies & References


http://java.sun.com/products/jini