

Jini and Mobile Agent Security

Agent Technology '98



Jeff Williams
Arca Systems, Inc.

williams@arca.com
(703) 734-5611

Disclaimers:

The agenda mentions a “Jini Security Model” -- that is not the subject of this presentation. I look forward to reviewing the Jini security provisions in the next beta release.

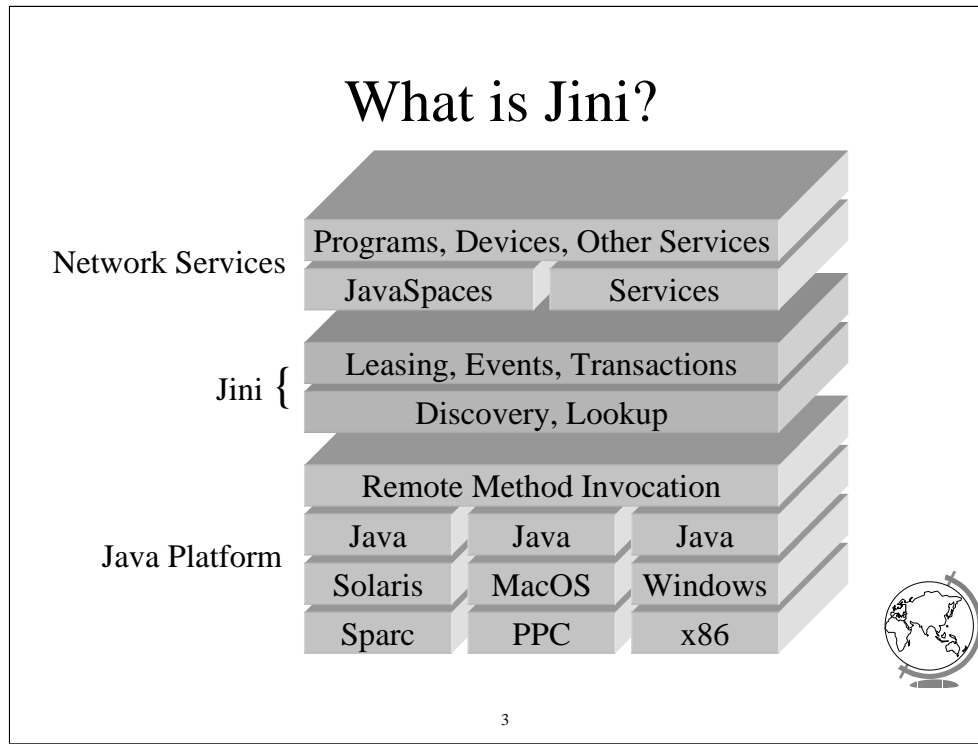
I don't work for Sun, and all the information in this presentation is based on the specifications available at <<http://java.sun.com>>

Jini is NOT an agent framework. As we will discuss, Jini provides several services that make distributed computing in an unreliable environment possible. I will argue that these services make an attractive foundation for an agent environment.

Overview

- ◆ What is Jini?
- ◆ Agents in Jini environment
- ◆ Security for agents in Jini
- ◆ Conclusions





Basically, Jini is a set of services designed to address distributed computing on an unreliable network, including

- failures
- latency
- language incompatibility
- multiple owners

Java platform

- Cross platform, sandbox, strong typing, designed to reduce complexity

Remote Method Invocation

- Full object transfer between Java platforms, both data and code

Discovery and Lookup

- Discovery allows services to “join a federation” by publishing their existence
- Lookup allows services to be found

Jini Services

- ◆ Address unreliability of distributed systems
 - Transaction Manager
 - Distributed Events
 - Distributed Leasing
 - JavaSpaces

- ◆ Not “kitchen sink” approach
 - least common denominator platform
 - can be extended and enhanced



Transaction Manager

- ◆ Allows two phase commit of transactions
- ◆ ACID properties



5

Transactions

- wraps a set of operations into a single step

ACID Properties

- **A**tomicity - all operations grouped into a transaction either all occur or none do. Voting and rollback.
- **C**onsistency - enables programming so that completion of transaction leaves system in consistent state
- **I**solation - Ongoing transactions should not affect each other
- **D**urability - Transaction results should be as persistent as the entity on which the transaction commits

Distributed Events

- ◆ Enable variety of notification schemes
- ◆ Based on AWT events and JavaBeans



6

Distributed Events

- Allow notification of events from one JVM to another
- Can use third parties (strung together) to enable a variety of delivery guarantees, filtering, off-loading, and storing
- Extend AWT and JavaBeans event models to work in an unreliable network environment, instead of on a single platform

Distributed Leasing

- ◆ Enables time limited resource usage
- ◆ Uniform response to abandoned or lost resources



7

Leasing

- notion of granting resource use for a limited period of time
- can be renewed or cancelled
- allows resources which are no longer in use (failure, forgotten, or ignored) to be cleaned up

JavaSpaces Service

- ◆ Shared “dynamic memory” for Jini
 - based on Gerlenter’s Linda tuple spaces

- ◆ Operations
 - read, write, take, notify



8

Description

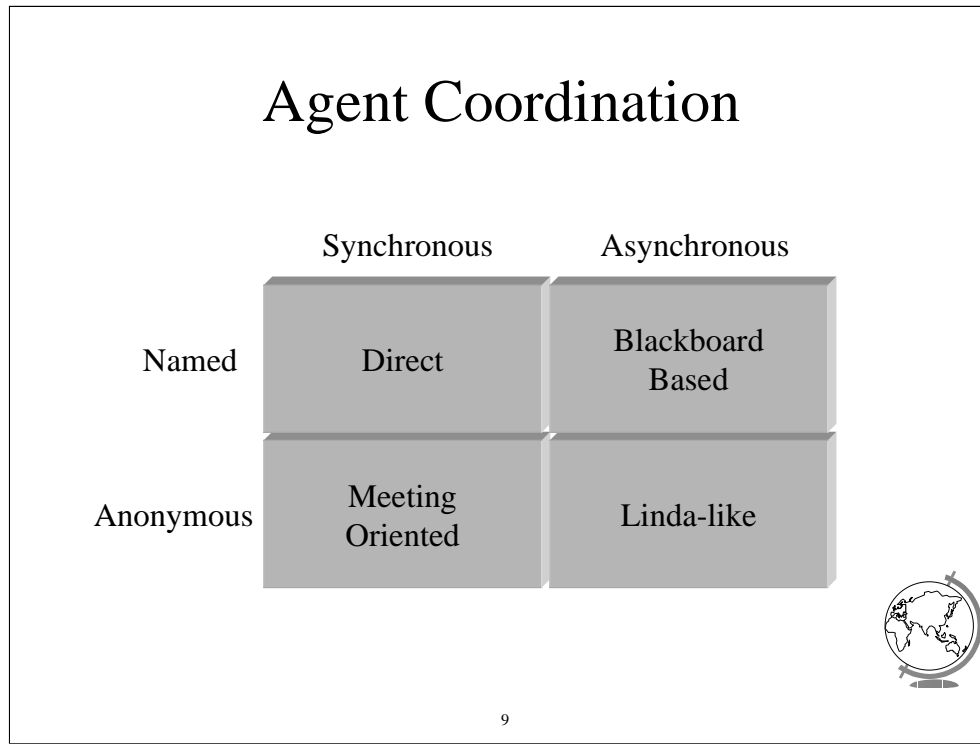
- helps with two problems :
 - distributed persistence
 - design of distributed algorithms
- something between a database and a file system
- provides tools for building distributed protocols
- Java strong typing
- Persistence is possible

For example

Book buyers post their requests to a JavaSpace
Interested sellers detect requests and send responses to buyers.

Operations

- write - write the entry into this JavaSpace
- read - read an entry from this JavaSpace that matches the given template
- take - read and remove an entry from this JavaSpace that matches the template
- notify - notify a specified object when matching entries are written



Direct Coordination:

- two agents must agree on time and manner of communication
- requires stable network, complex routing protocols
- directly adverse to dynamic asynchronous agents

Meeting Oriented Coordination:

- two agents interact anonymously at a known site
- problem of missed meetings

Blackboard Based Coordination:

- two agents use a common repository to store messages
- no agreement on time of reading or writing, but based on stable repository

Linda-like Coordination:

- requires neither temporal agreement or mutual knowledge for coordination

JavaSpaces provides a Linda-like coordination mechanism for Agents

Jini Security

- ◆ Jini extends the Java and RMI security
 - A Jini “federation” trusts its members
 - Twin notions of principal and access control list
 - Security features not in current release



10

Extensions

- services are accessed on behalf of a principle
- access is controlled by ACLs

Java security mechanisms

- code verification
- sandboxing
- code signing
- secure channels
- strong typing
- etc...

Jini federations target a workgroup that agrees on basic notions of

- trust, administration, identification, policy

Also, Jini services provide integrity protections for agent systems

AgentSpace Service

- ◆ Extending JavaSpace to support Agents
- ◆ Example:
 - Agent is created in an AgentSpace
 - Agent can access services
 - Or find and move to another AgentSpace
- ◆ All benefits of JavaSpace and Jini services



11

AgentSpace is a hypothetical agent framework layered on Jini. AgentSpace is a simple extension to JavaSpace which runs Agents stored there.

Benefits

- can use distributed storage in JavaSpaces
- can communicate with other agents using distributed events
- can find other services and agents
- can use two-phase commit manager
- can use cryptographic channels to communicate

Malicious Agents

◆ Mechanisms

- Bytecode checking
- Sandbox
- Code signing
- Transactions
- Cryptography
- ACLs*
- Verified principals*

◆ Problems

- Distributed configuration management
- Implementation flaws
- Key management

* not in current beta



12

This is a discussion slide for the “malicious agent” problem.

Remember that Jini is NOT an agent platform. So to a certain extent we are hypothesizing an agent platform layered on top of the Jini platform. This is not such a huge leap, since Voyager and others have promised Jini support.

Jini provides many conventional sorts of mechanisms for dealing with this problem.

However, there are still some difficult problems left to work on, that Jini does nothing to alleviate.

Malicious Hosts

◆ Mechanisms

- support for “agent gangs”?

◆ Problems

- tampering
- spying out code, secrets



13

This is a discussion slide for the “malicious host” problem.

As in the previous slide, remember that Jini is NOT an agent platform.

Jini provides no explicit help for dealing with this problem (no time limited blackboxing, partial results, or encrypted functions).

Nevertheless, the services offered in the Jini platform may provide important building blocks for future solutions to this problem.

Summary

- ◆ Jini provides
 - fully asynchronous and anonymous environment
 - interesting communication features
 - no explicit agent support

- ◆ Still waiting for
 - security specifications

