



A tutorial report for SENG 609.22

Agent Based Software Engineering

Course Instructor: Dr. Behrouz H. Far

A Tutorial on The Jini Technology

Lian Chen

Introduction

Jini network technology provides a simple infrastructure for delivering services in a network and for creating spontaneous interaction between programs that use these services regardless of their hardware/software implementation. Any kind of network made up of services (applications, databases, servers, devices, information systems, mobile appliances, storage, printers, etc.) and clients (requesters of services) of those services can be assembled, disassembled, and maintained on the network using Jini Technology. Services can be added or removed from the network, and new clients can find existing services.

Jini technology is architecture for the construction of systems from objects and networks. The Jini architecture lets programs use services in a network without knowing anything about the wire protocol that the service uses. One implementation of a service might be XML-based, and another RMI-based, and a third CORBA-based. The client is, in effect, taught by each service how to talk to it. A service is defined by its programming API, declared as a Java programming language interface.

When a service is plugged into a network of Jini technology-enabled services and/or devices, it advertises itself by publishing a Java programming language object that implements the service API. This object's implementation can work in any way the service chooses. The client finds services by looking for an object that supports the API. When it gets the service's published object, it will download any code it needs in order to talk to the service, thereby learning how to talk to the particular service implementation via the API.

In other words, the Jini architecture uses objects that move around the network to make each service, as well as the entire network of services.

Goal of the System

A Jini system is a distributed system based on the idea of federating groups of users and the resources required by those users. The overall goal is to turn the network into a flexible, easily administered tool on which human and computational clients can find resources. Resources can be implemented as either hardware devices, software programs, or a combination of the two. The focus of the system is to make the network a more dynamic entity that better reflects the dynamic nature of the workgroup by enabling the ability to add and delete services flexibly.

A Jini system consists of the following parts:

- A set of components that provides an infrastructure for federating services in a distributed system
- A programming model that supports and encourages the production of reliable distributed services
- Services that can be made part of a federated Jini system and which offer functionality to any other member of the federation

Infrastructure

The infrastructure is a set of components that provides for federating services in a distributed system. It provides mechanisms for devices, services, and users to join and detach from a network. Joining into and leaving a Jini system is an easy and natural, often automatic, occurrence. The infrastructure of a Jini system includes the following:

- A distributed security system, integrated into RMI (Java Remote Method Invocation), which extends the Java platform's security model to the world of distributed systems
- The discovery/join protocol, a service protocol that allows services (both hardware and software) to discover, become part of, and advertise supplied services to the other members of the federation
- The lookup service, which serves as a repository of services. Entries in the lookup service are objects that can be downloaded as part of a lookup operation and act as local proxies to the service that placed the code into the lookup service

Jini technology uses a lookup service with which devices and services register. When a device plugs in, it goes through an add-in protocol, called discovery and join-in. The device first locates the lookup service (discovery) and then uploads an object that implements all of its services' interfaces (join). The lookup service acts as an intermediary to connect a client looking for a service with that service. Once the connection is made, the lookup service is not involved in any of the resulting interactions between that client and that service.

Programming model

The programming model supports and encourages the production of reliable distributed services. It is a set of interfaces that enable the construction of reliable services, including those that are part of the infrastructure and those that join into the federation. The Jini programming model is the following:

- The leasing interface, which defines a way of allocating and freeing resources using a renewable, duration-based model
- The event and notification interface, which is an extension of the event model used by JavaBeans components to the distributed environment that enables event-based communication between Jini services
- The transaction interfaces, which enable entities to cooperate in such a way that either all of the changes made to the group occur atomically or none of them occur

Services

A service is an entity that can be used by a person, a program, or another service. A service may be a computation, storage, a communication channel to another user, a software filter, a hardware

device, or another user. In a word, services can be made part of a federated Jini system, which offer functionality to any other member of the federation. Example Jini services include the following:

- A printing service, which can print from Java applications and legacy applications
- A JavaSpaces service, which can be used for simple communication and for storage of related groups of objects written in the Java programming language
- A transaction manager, which enables groups of objects to participate in the Jini Transaction protocol defined by the programming model

Jini technology also defines a leasing and transaction mechanism to provide resilience in a dynamic networked environment. The underlying technology and services architecture is powerful enough to build a fully distributed system on a network of workstations.

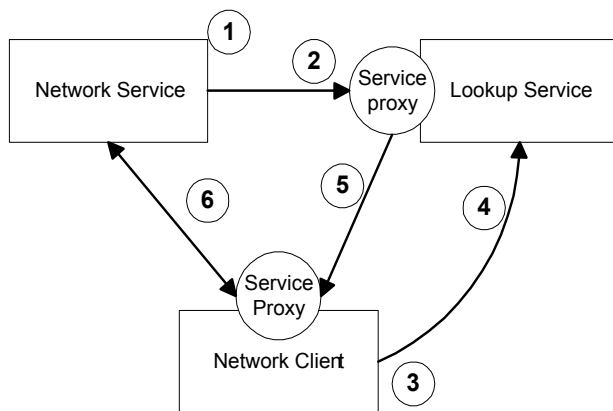
Jini is a new distributed systems technology from Sun Microsystems. The goal of Jini network technology is to federate groups of devices and software components together to form a single, dynamic distributed network system, which is that any of these devices can come or go without bringing down the whole system. The system can respond to changes in its environment and its constituency in reliable and predictable ways.

How Jini Technology Works

Jini network technology consists of an infrastructure and programming model that addresses the fundamental issues of how clients and services discover and connect with each other to form an impromptu community. Written entirely in the Java language and utilizing its object-oriented features, Jini technology uses the mechanisms pioneered by the Java Remote Method Invocation API to move objects around the network.

Services employ a proxy (an object with service attributes and communication instructions) to move around the network. Through the processes of discovery and join, services are found and registered on a network. Registering means that the service has sent a service proxy to all lookup services on the network, or a selected subset. A lookup service is equivalent to a directory or index of available services, where proxies to these services and their code are stored. When a service is requested, its proxy is sent to the requesting client so that the service can be used. After that, the proxy conducts all communication between the client and the service.

To preserve the networks flexibility and resilience, Jini technology introduces the concept of leasing. When a service joins the network, it registers its availability for a certain leased time. Before the time expires, the service may renegotiate the lease. If the service is removed from the network, its entry in the lookup is removed automatically when the lease expires.



1. **Discover**
Network service discovers available lookup services (LUS)
2. **Join**
Network service sends Service proxy to LUS
3. **Discover**
Network client discovers available LUS
4. **Lookup**
Network client sends a request to LUS to find desired services
5. **Receive**
LUS sends registered service proxy to network client
6. **Use**
Network client interacts directly with network service via service proxy

Ronin Agent Framework and JPES

Introduction

The Ronin Agent Framework is a Jini-based distributed agent development framework. Ronin is designed to aid in the development of highly demanded "intelligent" distributed applications.

Ronin introduces hybrid architecture, a composition of agent-oriented and service-oriented architecture, for deploying dynamic distributed systems. The framework contains a number of features that distinguish it from comparable toolkits and frameworks, including an Agent Communication Language (ACL) and network protocol independent communication infrastructure. It also includes an agent proxy architecture that provides mobile agent behavior, a simple but powerful agent description facility that allows agents to find each other and an infrastructure that encourages the reuse of the existing non-Java AI applications.

Jini Prolog Engine Service (JPES)

Jini Prolog Engine Service (JPES) is a Jini service that provides remote Prolog engine services to Jini-enabled components in the network. JPES is based on the Ronin Agent Framework that is introduced above.

Taking the advantage of the Jini architecture, JPES provides a flexible infrastructure for distributed components to gain access to the powerful Prolog engine. JPES makes sophisticated AI techniques like rule-based programming, constrained problem solving, knowledge sharing etc. to be available to the development of the next generation "intelligent" distributed system.

JPES can be extended to interoperate with various Prolog implementations. A set of Jini service management functions is provided by the JPES to ease the service administration.

JIMA

JIMA is Jini-based Infrastructure for Active Documents and Mobile Agents. Jini technology enables Java-based clients to dynamically discover and employ services in a local area network without prior knowledge of URLs, hosts, or port numbers. In a Jini framework agents may act as both services and clients.

In the JIMA, mobile agents implement active documents that actively seek out their recipients. A recipient may be locally or remotely connected to the application network. The infrastructure therefore also includes personal user proxies, a localization service, terminal services and subsystem resources. Jini is proposed as a convenient technology for connecting the various components.

In this application, they have focused on the ability to cast document flows into a framework which allows the document themselves to be intelligent, proactive and social participants rather than inanimate frames of stored data. Instead of people reaching for documents, the documents try to reach the user.

Agents use Jini to discover and negotiate with other agents. In the Jini framework, agents thus appear as both clients and services. Some services (like proxies) are persistent within the local network, while others (like terminal services) may appear and leave at random.

Reference

<http://gentoo.cs.umbc.edu/ronin/>
<http://gentoo.cs.umbc.edu/jpes/>
<http://agents.umbc.edu/Topics/index.shtml>
<http://www.sun.com/jini/>