Jini Technology: Impromptu Networking and its Impact on Telecommunications

Lai Olstad, Javier Ramirez, Clint Brady, Bruce McHollan
olstad@colorado.edu ramirejj@colorado.edu cdbrady@usa.net bmchollan@category5.net

1. Introduction

Impromptu Networking, as brought by Sun Microsystem's Jini technology, is poised to radically change telecommunications. While it is too soon to tell if Jini technology will be the dominant means of implementing impromptu networking, the benefits of this new paradigm are unmistakable. Providing the ability to automatically interconnect devices into impromptu networks will make the lines between today's numerous disparate networks disappear.
This paper examines Jini technology and its ramifications on telecommunications, including both voice and data networks.

2. The Vision of Jini

According to Bill Joy, one of the founders of Sun Microsystems and the original creator of Berkeley UNIX: "*The complexity of the systems today has exceeded the professional systems of twenty years ago. A better way of constructing new systems is needed, to reduce the frustration and better match the computing devices to human uses* [1]". The main goal of Jini is to provide simpler, more flexible and robust networks to the telecommunications industry.

2.1. Jini History

A synopsis of Jini's history will help explain how the vision of Jini evolved. The history of Jini is largely the history of Java. Jini fulfills the original Java vision of consumer-oriented groups of electronic devices interchanging data. Java evolved from a language called Oak. Oak was designed by Sun Microsystems in 1990 to serve as a portable way to write programs for embedded processors. In 1994 engineers Patrick Naughton and Jonathan Payne from Sun wrote a Web browser using Oak. This browser, named WebRunner, later became the foundation for the HotJava browser. HotJava had the unique ability to download executable programs from Web servers and execute them in a browser. These programs are called applets. Although Java got its start in consumer electronics, it was the ability to build applets that propelled it into the computing industry. Realizing that the original Java concept was still compelling, a group of Sun engineers recognized the need for continuing development. Although Java enables moving code from machine to machine, problems exist that hamper implementing constellations of easily administered devices. This requires mechanisms normally not associated with desktop computing. These mechanisms for such devices are as follows:

- A robust software infrastructure.
- The ability to dynamically configure additional devices and peripherals.
- The ability to share components without reconfiguration.
- The ability to form impromptu communities.

Prior to 1998, Jini was being developed secretly at Sun until New York Times technology reporter John Markoff published a story on Jini. Jini was first introduced to the public on January 25, 1999. Accompanying the introduction was a host of licensed supporting

manufacturers. In May 1999, the first Jini Community Summit was held in Aspen, Colorado. During the summit, Bill Joy and Jim Waldo, Sun's chief Jini architect, explained further how Jini fundamentally changes the architecture of computing systems [2].

2.2. Two Emerging Trends

Two emerging trends, the proliferation of small embedded devices and high speed networks will point to a future in which more devices will be enhanced with embedded microprocessors and interconnected via networks. Embedded devices are appliances or devices with an embedded microprocessor that has a relatively focused functionality. Examples of embedded devices include cell phones, microwave ovens, VCRs, pagers, clocks, and printers. It is important to note that although a PC contains an embedded microprocessor, it is not usually considered an embedded device because it does not have a focused functionality. Rather, a PC is a general platform suitable for many different software applications. The proliferation of embedded devices is being driven by the decreasing costs of adding processing power to devices. This is a result of increasing the functionality of the microprocessor chips. As both the cost and size of computers continue to decrease, it becomes cost-effective to include processors in a wide variety of products. Manufacturers are adding embedded processors to their products to increase the functionality and value. This trend will yield an explosion of embedded devices in the coming years.

Another trend underway is the proliferation of high-speed networks. Companies like Qwest, Level 3 and Global Crossing, to name a few, are rushing to build high-speed backbones. High-bandwidth pipes are being sent into homes. Wireless networks are being developed. The network will be increasingly fast and accessible by all. Jini will take advantages of these two trends and make networking simple and flexible.

2.3. Disappearing Computer

"*Forget the desktop*" bellowed Joy at the Summit. With that, he predicted the demise of desktop computers and the emerging network-centric environment. Joy went on to describe a flat-panel display currently being manufactured that can be folded or rolled like paper. He said that in the future when he needed to do some work, he would set down his "handset" and roll out his flat-panel display. The handset would have a wireless connection to the Internet and a wireless connection to the display. The flat-panel display or "smart paper" is an example of how computers will disappear into application specific devices we will use everyday [3].

2.4. Jini Scenarios

Jini technology gives users a unique, more convenient option for accessing services in their neighborhood and the world at large. Examples of possibilities that Jini technology enables include: You walk into a conference room or a colleague's office and instantly your laptop computer is connected to a full array of services. Rather than drive to your local instant-copy store, you might simply access their services application from your home computer. You tell them what you want copied, where to deliver the job, and they take care of the rest. Or, imagine appliances that maintain themselves through remote diagnostics and repair procedures; Water heaters that adjust temperature and water pressure to compensate for someone turning on the dishwasher while the shower is running. All these are possible due to the vision of Jini technology within the impromptu community.

Since the Internet is an extremely dynamic network, composed of fiber, wire, cable, wireless, satellite, cellular and other communications media an adaptive technology like Jini makes sense. Being able to seamlessly access resources provided by the network based on proximity and / or authority is becoming much more valuable. A printer that is unavailable today, should not appear as an option. Yet today's network operating systems implementations are quite static in this regard and this is exactly how and un-useable printer is handled. If a new, previously unknown resource becomes available, no one should need to tell the user about it so they can configure their computer to take advantage of the resource. With Jini, utilizing this new resource may not even involve a conscious (human) decision. For example, the user may want to print color photos when a color printer becomes available at some unknown future time.

Any technology that experiences exponential growth, will encounter limiting factors that have to be addressed through fundamental shifts in paradigms or the explosive growth self-limits. For example, it is estimated that had it not been for the development of automated switching in the telephone network, the number of operators that would be required today would exceed the population of the planet. Automated switching removed a barrier that would have impeded the further advancement of technology.

3. The Jini System Components

Jini is a small set of rules that enables clients and services to interact; Jini is an extension of Java, Sun's cross-platform programming language. Jini's purpose is to enable the creation of simpler, more flexible networks. Jini will provide a way for entities (machines, applications, or devices) to automatically "discover" and share resources called "services" which can be of any type (hardware, software, communication channels, users) [4]. Jini is based on the Java programming language and it requires a Java Virtual Machine (JVM) definition. Java is an interpreted language and is independent from the platform on which it is run. This is in contrast to the JVM, which is platform dependent and must be implemented accordingly [5]. (Figure 1)

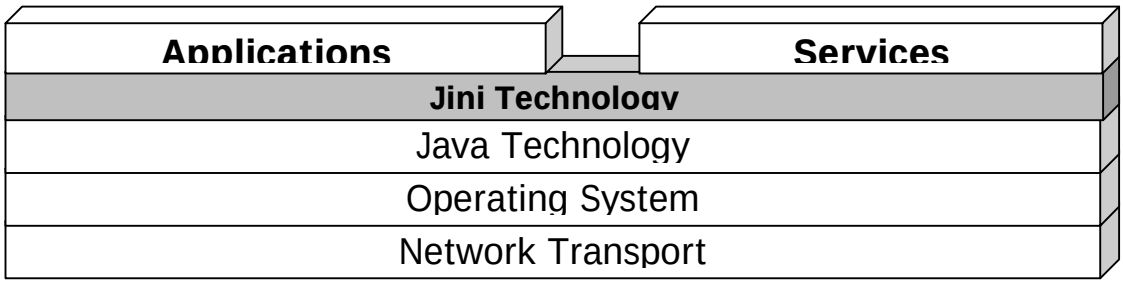

Figure 1. Jini Stack

Jini technology provides simple mechanisms, which enables devices to plug together to form an impromptu community. Each device provides services that other entities in the community may use. Devices provide their own interfaces, which ensures reliability and compatibility. In Jini a group of services is called a community; all services in a community are aware of and able to use each other. Federation is the ability for Jini communities to be linked together into large groups making it very scalable. Ideally the size for a single Jini community is about the size of a workgroup (10 to 100 people) [1].

3.1. Important Considerations

Jini addresses how services connect to one another, not what those services are, what they do or how they work. Jini is not a Name Server even though it has similar functionality. Jini is not RMI (Remote Method Invocation). While Jini uses RMI facilities for mobile code, Jini is a set of services and conventions built on top of RMI. Jini is not a Distributed Operating System; Jini is only aware of services and has the facilities for finding those services. Jini is not middleware. Consequently, it does not create communication links between common commercial databases. In the current Jini implementation and specifications there is no authentication performed on either the discovery requests or responses [1].

All of the components for Jini are implemented as a set of software libraries and conventions that are used by the code that participates in Jini communities. Jini has five key components: Discovery, Lookup, Leasing, Remote Events, and Transactions [6].

3.2. Discovery

The Discovery component is the process used to find/join communities on the network. Discovery is responsible for the spontaneous community-building properties of the system. Once a community is found, the Jini-aware entity can step through a series of conventions to become part of the community and publish services to the community. The **Multicast Request Protocol** is used when an application or service first becomes active and needs to find a nearby lookup service. The **Multicast Announcement Protocol** is used by lookup services to announce their presence. The **Unicast Discovery Protocol** is used when an application or service already knows the particular lookup service required. The process of publishing a service to make it available to others in the community is called joining [6].

Currently Jini has been implemented only on IP networks. Based on the Discovery/Join specification from Sun, IP addresses may be statically assigned to some hosts and other components may be dynamically assigned. Any Jini-aware entity must have a properly configured network stack capable of UDP multicast and TCP unicast messages. (Figure 2)

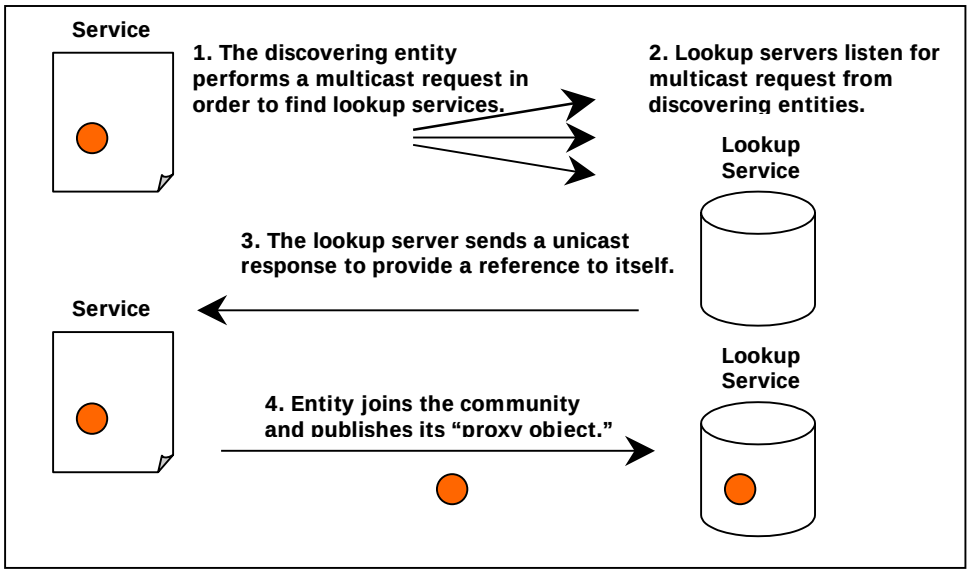

Figure 2. Discovery Component

3.3. Lookup

The Lookup component is responsible for distributing code to participants requesting a service. Lookup fulfills the role of a directory service within each Jini community. It provides facilities

for searching and finding services that are known within a community. Lookup tracks all active services and listens for join requests. Lookup also provides detail information about the services running.

Each active service contains a Java object called "proxy object[1] ". Other participants in the community can download the proxy object to use the service. When an entity needs another service, it will query the Lookup Service by a unique identifier or attribute. Once the service is found, the proxy object is downloaded to their JVM and the entity will make calls on it to use the service. This idea of downloadable proxy objects gives Jini its ability to use services and devices without doing any explicit driver or software installation [6]. (Figure 3)

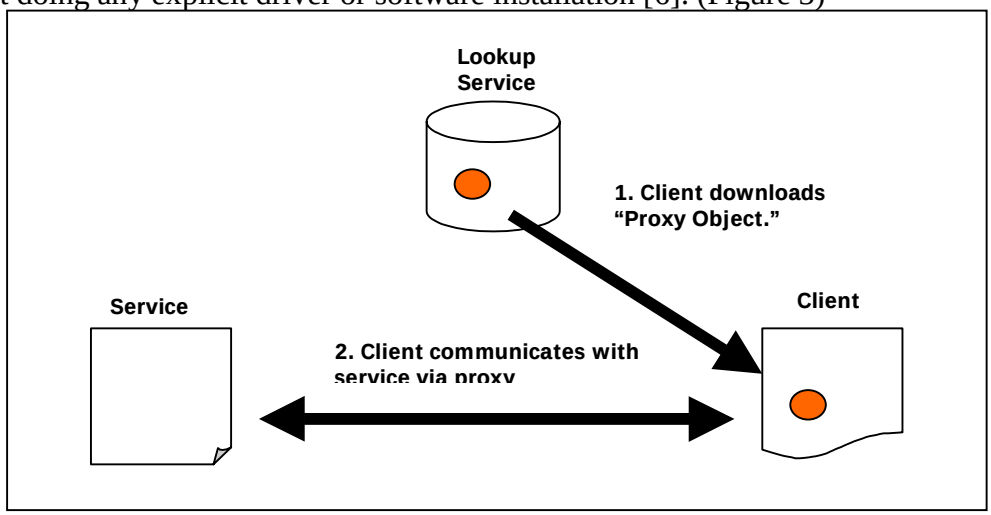

Figure 3. Lookup Component

3.4. Leasing

The Leasing component is a technique that provides Jini's self-healing nature. Leasing is based on the idea of leasing resources for a fixed period of time rather than granting access to a resource for an unlimited amount of time. Leases expire after a predetermined amount of time unless they are renewed. Leases are always defined in terms of time duration instead of a specific time in the future [6]. If a service becomes unavailable, either intentionally or unintentionally without cleaning its components, its lease will eventually expire.

3.5. Remote Events

Remote Events allow services to notify each other of changes in their state. Jini services, like many software components in a system, need to be notified when some interesting change occurs. Asynchronous notification messages are sent directly to a software component instead of polling for changes. Asynchronous notifications can often simplify the programming of components. Jini, like most of Java, uses events to do asynchronous notifications. An event is an object that contains information about some external state change that a software component may be interested in. RMI is an event notification built into Java, which allows objects to move across the network to be executed. The Lookup service can use remote events to notify interested parties when the set of services, available to a community, changes [6].

[1] A "Proxy Object" is a list of attributes that are used to describe the service. This object contains all the information to use and configure the service.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.