Jini™ Network Technology

An Executive Overview
Introduction

The Internet is changing the way we do many things. But most importantly, it is changing the concept of networks from multiple connected computers to connected devices — including computers — that deliver smart Web services through the network. Although Jini™ technology is not a Web service, it can communicate with, or be used to build, Web services. Jini technology was developed to address an important set of long-term problems that must be solved before generic Web services can be transformed into the highly personalized, cross-network, intelligent services that future systems will provide.

As the next level beyond network connectivity, Jini technology provides developers with tools to construct systems from distributed objects over networks. It offers a simple infrastructure for delivering services over the network and creating spontaneous interaction between programs that use these services, regardless of their hardware or software implementations.

Any type of network made up of services and clients of those services can be easily assembled, disassembled, and maintained on the network using Jini technology. Examples of services that can be utilized in this way are:

- Applications
- Operating systems
- Servers
- Devices
- Information systems
- Mobile appliances
- Storage
- Printers
Jini technology delivers access to services over any network for any platform, any operating system, any application — regardless of the network complexity, distance, or host device. This means it provides an easy, simple, and fast way to interact with services simply by locating them on a network, with no further action required by the user. Jini technology can be applied to any industry serving commercial, government, community, and consumer markets.
The Analogy of Generational Systems

The evolution of network technology does not necessarily happen in a linear fashion. Just like a family, technologies exist in generations. One generation is born, a few years later the next generation is born, and so on. The earlier generations are still around for quite awhile, and several generations are in existence at the same time. The newest generations may be more advanced in certain ways, but that doesn’t mean all the earlier generations are superseded overnight.

As applied to network technology, this means that rather than being time-based, the different generations are really problem-based. The first two generations discussed in the context of this paper, client/server and n-tier, were brought about by the set of problems that had to be solved when building client/server and n-tier networks within an enterprise. A different set of problems, part of what can be called the network-to-network generation, arose when companies began bringing applications into the Internet and Web framework so the networks could offer services to each other. The next set of problems, called participant-to-participant, will emerge when the individual entities in the networks (rather than just the networks themselves) start offering services to each other. It is this last group, the participant-to-participant generation of problems, that Jini technology addresses.
FIGURE 1    Generational Systems: Each generation is characterized by a set of problems and the technologies that address those problems.

The Web-enabled services offered by Sun’s most recent group of technologies belong, by and large, to the network-to-network generation, and address problems that are ubiquitous today. In the future, Web-enabled services will cause enterprises to run into the problems that are addressed by Jini technology. In fact, a few of Sun’s customers have already been encountering some of these problems, which include:

■ Finding and connecting services on a network, much like the component problem in standard, single-address space, object-oriented programming
■ Creating reliable sets of services out of unreliable parts, including an unreliable network
■ Dealing with networks that are very large or long-lasting
■ Evolving parts of the service set without halting the service set itself at any time

Seeing Jini technology as an attempt to solve these problems allows it to be placed in the generational framework. Each generation can be characterized by a set of problems that dominate it, along with the technologies that have been used to address those problems. These technologies lead to a set of products that solve application-level problems. The technology itself deals with the generational problems.
Today’s N-tier Generation

One could reasonably claim that the current generation is dominated by technologies that solve the problem of constructing an n-tier network of services. In contrast to the client/server model, the n-tier architecture is characterized by distributing applications and services across systems. An n-tier network is designed in such a way as to enable the services to interact with each other within the network. The technologies used to solve this problem are CORBA, DCOM, and other RPC-based systems. These systems address the problems of building the components that will live in the network. The components must:

- Know about each other while they are being built, at least to the extent of sharing IDL definitions
- Be evolved in lock step, because changing the sub or skeleton files requires changing all of the corresponding sub or skeleton files
- Require fairly tight levels of administration

Seven Fallacies of Networking

Because today’s systems are deployed on tightly controlled networks, the network is assumed to be reliable. The underlying technologies can afford to ignore Deutsche’s Seven Fallacies of Networking, since these underlying fictions are not part of the generational problem set. The Seven Fallacies — which are based on problems that happen in real networks over time — are:

1. The network is reliable
2. Latency is zero
3. Bandwidth is secure
4. The network is secure
5. Topology doesn’t change

6. There is one administrator

7. Transport cost is zero

Because the Jini architecture is all about simplifying interactions on a network, it can help eliminate the reasons for some of these problems, such as network outages and configuration changes. This will become increasingly important as networks continue to evolve.
The Network-to-Network Generation

Explosive growth in bandwidth, networks, and digital devices is changing the world and yielding massive opportunity for innovation. This phenomenon is called the Net Effect. The service-driven network is the first example of the Net Effect, where Web services will become smart because they are context-aware and can be reliably delivered over multiple networks.

Therefore, this generation of technologies is characterized by the need for separately developed networks to talk to each other, in the safest way possible over the emerging global network, so they can share services more easily. Each separately developed network wants to use services that are exported and supplied by another separately developed network. Because this involves communication between two or more networks, it is called the network-to-network generation.

All of these networks want to interact only in very controlled, well-specified ways, since the level of trust between the networks is not very high. At most, they interact by stating what information needs to be exchanged for one network to provide the advertised service to the other network. To each network, the other networks appear as single entities that provide services.

In this environment, the problems are answered by technologies such as:

- **XML**, which allows the exchange of information in a well-defined, stylized fashion. The textual nature of the XML exchange is well-suited to this sort of problem, where safety is more important than spontaneity or flexibility.

- **UDDI**, which allows the discovery of what services are being offered by a network.
At this stage, the interaction of the networks is similar to the interaction of world superpowers when they are first establishing diplomatic relations. Since the consequences of a misunderstanding at this stage can be catastrophic, flexibility is secondary to maintaining a measure of stylized safety. This is a good stage for standards-based work, since the standards process ensures safety over flexibility. In addition, the open nature of the standards process assures each party that its interests are not being undermined.
The Participant-to-Participant Generation

The types of problems addressed by Jini technology are not apparent in the network-to-network stage, which is characterized by an entire network offering a service to another (client) network. Jini technology comes into its own in the participant-to-participant generation, when participants within a network start looking for services not just from other networks, but from the participants in the other networks as well. The participant-to-participant (or participant-to-client) generation is marked by the merging of networks, enabling any client in any of the networks to find services offered by any participant in any other network.

While this change may seem simple, it is in fact rather profound and complex for several reasons:

- The scale of possible interactions increases dramatically if the environment enables every participant to either provide or consume a service to or from every other participant.
- The rate of exchange increases rapidly if services can be offered and consumed by all of the participants of the networks, either due to change in the network or failure in the infrastructures.
- The complexity of interactions increases together with the options to provide or consume services from entire networks, individual participants, or both.

To return to the metaphor of superpowers establishing relations, the participant-to-participant generation of networking is much like the citizens of two opposing superpowers talking, trading, and interacting directly. The number and kinds of interactions are far greater and more complex, but the importance of each individual interaction is far less than the interaction of the superpowers themselves.

In the participant-to-participant generation, a solution such as Jini technology can:

- Deal with the inherent unreliability of the network (see Fallacy #1)
- Keep up with constantly changing topology (see Fallacy #5)
- Allow multiple administrators (see Fallacy #6)
- Evolve components
- Take failure seriously in the construction of interactions

The ability to form relationships in an ad-hoc fashion will be extremely important, because this sort of networking will begin at the periphery of the network. That is where the freedom to interact in unexpected ways will first be tolerated, and it will only be allowed on the backs of networks as an isolated organizing principal for the networks, which will appear from the outside as single entities.
Standards Versus Flexibility

The participant-to-participant generation is still benefiting from experimentation, which is one reason why Jini technology is run by a community, rather than a standards body. Because Jini technology is evolving, it requires a far more flexible and rapidly changing environment than that needed by the more mature, network-to-network generation of products. Those technologies are ready for the benefits of a standards body, which can ensure safety and stability. The experimental state of the network-to-network generation has passed, so there is enough experience to start thinking about and developing standards for it. Not so for the participant-to-participant generation, which includes Jini technology.

This doesn’t mean there is no use for Jini technology now — a number of Sun’s customers are already hitting the problems of scale, component integration, and ad-hoc networking that Jini technology can help solve. And software written to the Jini specification gives traditional service providers the ability to dynamically add and manage network services. However, for most enterprises that are just worried about establishing the safest possible, business-to-business interchange, Jini technology may not be the answer. For them, Sun offers proven solutions such as Enterprise JavaBeans® components, the Java™ 2 Platform, Enterprise Edition, UDDI, and XML technologies.

However, when the problems of the network-to-network generations have been addressed, those enterprises will want to enable participants on one network to find individual services on another network. At that point, they will encounter the participant-to-participant generation of problems and be ready for Jini technology, which is freely available (with only the investment of time involved in joining the community investigating those problems). Sun makes Jini source code open to the developer community under the Sun Community Source License, offering access to shared source code for all uses at no cost. This enables developers to “look under the hood” to make informed decisions, avoid vendor lock-in, leverage existing knowledge and skills, as well as develop and deploy implementations.
Already, companies are using Jini technology to build communities of services out of simple pieces of hardware and software that have been built according to the Jini specification. Examples range from business enterprises and home networking to battlefield implementations, and even a Jini technology-enabled car.
How It Works

Jini technology consists of an infrastructure and a programming model that address the fundamental issue of how clients connect with each other to form an impromptu community. Based on the Java language, Jini technology uses the methods pioneered by Java Remote Method Invocation (RMI) protocols to move objects, including their behavior, around the network. Network services run on top of the Jini software architecture.

Use Any Protocol

A major advantage of the Jini architecture is that it is the only technology available today that lets programs use services in a network without knowing anything about the wire protocol used by the service. One implementation of a service might be XML-based, another might be RMI-based, and a third might be CORBA-based. In effect, each service provides its clients with the mechanism to talk to it. A service is defined by its programming API, declared as a Java language interface. Jini technology is the only solution that makes the details of how a service uses the network into an implementation detail that can differentiate between implementations of the same service, without changing the client code.

When a service establishes itself on a network of Jini technology-enabled services or devices, it finds a place where it can advertise itself (the process is called discovery) by publishing a Java object that implements the service API. This object’s implementation may work in any way the service chooses. The client finds services by looking for an object that supports the API. When it receives 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 service implementation via the API. The API that defines a service is simply a list of what the service does — it does not tell it how to do its job. (Compare this to the way that other solutions work: they must agree, not only on what remote objects will do, but also how requests are transmitted on the wire.)
This means that the programmer who implements the service can choose how to translate an API request into bits on the wire using XML, RMI, CORBA, or a private protocol. Jini technology is often compared to universal plug and play (UPnP), which is a discovery protocol for devices. UPnP is developing XML standards for home networking by operating primarily at the connectivity level of a network stack focused on devices. Once devices are connected to a network, they need an advanced service delivery architecture, such as Jini, that can deliver services and a higher order of interoperability to networked devices. Jini technology and UPnP (or any connectivity scheme) can interoperate because the Jini architecture is wire-protocol and transport-protocol neutral. Through Jini technology’s wire/transport protocol neutrality, UPnP devices can work in the same network as devices using all the other protocols.

In this way, the Jini architecture uses objects that move around the network to make each service, as well as the entire network of services, adaptable to changes in the network over time. With countless numbers of services participating in this impromptu community, network software needs to ensure that communications and transactions are completed in an orderly manner. Jini software provides the Java APIs necessary to manage distributed events, and makes sure that all events in a transaction occur before the entire transaction is committed.

Jini technology has considerably more functionality than competitive solutions because it allows services to be removed from the network, new services to be added, and new clients to find existing services — all without administrative intervention. This is an inherent strength of object-oriented programming, allowing the code and data to be transported over the network to perform tasks.

**Simplifies Network Design and Management**

Another significant differentiator is Jini’s potential to eliminate the need to design to each layer in a network, simplifying network design and management.

---

**FIGURE 1** ISO Seven-Layer Model of Networks

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.0</td>
<td>Applications (NFS, FTP, HTTP)</td>
</tr>
<tr>
<td>6.0</td>
<td>Presentation (XDE, XML, ASCII, Java serialization)</td>
</tr>
<tr>
<td>5.0</td>
<td>Session (Sun RPC, DCE RPC, IIOP, RMI)</td>
</tr>
<tr>
<td>4.0</td>
<td>Transport (TCP, UDP)</td>
</tr>
<tr>
<td>3.0</td>
<td>Network (IP)</td>
</tr>
<tr>
<td>2.0</td>
<td>Data Link (wire formats for messages)</td>
</tr>
<tr>
<td>1.0</td>
<td>Physical (wires, signaling)</td>
</tr>
</tbody>
</table>
Because Jini technology is protocol-independent, developers can design to the interface, not to each protocol. Network application design doesn’t have to adjust to changes in the protocols, because the proxies take care of that. Performance is a factor of how efficiently the network is designed; Jini technology has no impact on it.

With Jini technology, developers can realize enormous benefits:

- Design to the interface
- Create applications and services that interact without preinstalled drivers
- Take advantage of Jini technology’s self-healing nature; no need to reconfigure upon failure
- Clean up is automatic, so garbage collection is eliminated
- Pay only for services that are used
- Ensure that applications are always current, without downtime or upgrading
- Change vendors at any time
- Modify equipment configurations easily, including legacy hardware and software
Conclusion

Sun’s Jini technology provides open, end-to-end solutions for creating dynamically networked products, services, and applications that scale from devices to the enterprise. This technology enables developers, service providers, and content creators to gain a competitive business advantage and capitalize on new revenue streams, by rapidly and cost-effectively developing and deploying compelling new applications and services for their customers worldwide. All of Sun’s technologies, including Jini, are developed with industry collaboration, allowing more widespread innovation and providing a standards-based platform.

Because Jini technology addresses problems that only some companies are experiencing, the requirement for this technology is not always readily apparent. For the increasing number of companies that are already hitting the problems of scale, component integration, and ad-hoc networking, especially in the financial, automotive, and telecommunications industries, Jini technology is the premier solution available today.

Resources

Additional information about Jini technology is available at www.sun.com/jini.