Virtual Organizations, Pervasive Computing, and an Infrastructure for Networking at the Edge

White Paper

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
1 (800) 786.7638
1.512.434.1511

Contents

CHAPTER 1

Introduction

For at least the past two decades, networks have been a part of computing. Local area networks have allowed users to share peripherals and files. The Internet has enabled communication with electronic mail, remote access via telnet, and management at a distance. More recently, the Web has become a feature of everyday life, allowing the publishing and finding of information in ways that had been dreamed of, but never realized.

However, the Internet is now more than a way of connecting computers (and the people who use them). Rather than being an infrastructure for computers and their users, the Internet is an infrastructure for everyone. The way in which people connect to the Internet is a secondary consideration. Instead, the fact that the connections can be made has taken on a life of its own, leading to new ways of thinking about how we work, how we are educated, and how we can use the Internet itself.

The vision of the future of this kind of networking has been widely publicized. Access to the Web will be through a multitude of devices, from the ubiquitous personal computer to personal digital assistants (PDAs), cell phones, and pagers. In addition, many of the devices we use in everyday life, including our automobiles, televisions, most appliances, and homes will be able to access the Internet. In fact, this access will be so common and pervasive that we will be surprised only when we don’t have it, much like we are now surprised when we find ourselves without access to the phone system or electrical grid.

Some have questioned if such pervasive computing and networking will really be a benefit to the users of the network. Given the examples that are often provided, such skepticism is understandable. These include connecting refrigerators to the Internet so we can find out, via cell phone or PDA, what is in our own refrigerator while we are at the store. Likewise, our stove will conspire with the refrigerator to decide what recipe makes the best use of the available ingredients, then guide us through preparation of the recipe with the aid of a network-connected food processor and blender. My car will use the Internet to find an open parking space or the nearest vegetarian restaurant.

While easy to understand, such examples have an air of the ridiculous about them, and for good reason. Using the Internet to replace a shopping list hardly seems like a good reason to increase the cost of an already costly appliance, or the risk of others gaining access to the contents of my home. Likewise, letting our appliances decide what we are going to eat reminds us of the *The Hitchhiker's Guide to the Galaxy*[1], where the computerized food machines can never quite figure out the tastes of their organic friends. And it is clear that the examples of "location-aware" uses of the Internet, like those for the car, are really ways to target advertising, in ways that the advertising's targets are not going to be willing to pay for.

1. Adams, Douglas, *The Original Hitchhiker Radio Scripts*, 1985, Harmony Books, New York.

CHAPTER 2

An Internet of Things

In fact, it is both highly likely that our appliances will be connected to the Internet, and highly unlikely that those connections will be used in the ways that have been mentioned. Rather than connecting the refrigerator to the Internet so I can find out what I need to buy when I'm at the store, the connection will enable the manufacturer to monitor the appliance to ensure that it is working correctly and inform me when it is not. Rather than my stove and refrigerator conspiring in an effort to determine what I will eat, they will conspire to optimize the energy usage in my house (neighborhood, region). My car will connect to the Internet to allow the manufacturer to diagnose problems before they happen, and either inform me of the needed service or automatically install the necessary (software) repair.

These connections are fundamentally unlike those we associate with networks. Rather than using the network to connect computers that are being used directly by people, the coming age of Internet-connected appliances will usher in a time when most of the communication occurring over networks is between machines and programs that are *not* directly monitored by people. The refrigerator will not send information to a human who is monitoring its performance at the manufacturer, but rather to another computer that is running a program monitoring all of the refrigerators made by the manufacturer. The appliances in our homes will not communicate to someone working for the electrical utilities, but rather to a computer (or set of computers) running programs that regulate the flow of energy to all of the utility's customers.

In addition, the majority of these communications will occur in an end-to-end structure that does not include a human at any point. This can be seen from the simple mathematics of the Internet. The number of machines connected to it has been increasing at an exponential rate for some time. It will continue to grow at this rate as the existing networks of embedded computers, including those that already exist within our automobiles, are connected to the larger, global network, and as new networks of embedded devices are constructed in our homes and offices.

Soon, there will be more processors connected to the Internet than there are people in the world.

And the rate of increase of computers will be faster than the rate of increase of the human population, so the disparity will continue to grow. It is just a matter of time before most of the traffic over our networks is sent from, received by, and interpreted by programs that do not forward that information to a human recipient.

Not only is the network we call the Internet becoming unimaginably large, it is also becoming much more heterogeneous than networks have been in the past. The technology we developed for computer networks assumes that the machines connected by the network are desktops or servers. While there may be significant differences between such machines, they are all reasonably competent with regards to the amount of computation they can do, reasonably unlimited in terms of the amount of memory and electrical power available to them, and connected by networks that are generally static, and in which bandwidth is high and growing fast.

However, all of these characteristics are about to change. The kinds of devices that will be used to access the Internet are no longer confined to desktops and servers, but include small devices with limited user interface facilities (such as cell phones and PDAs); wireless devices with limited bandwidth, computing power, and electrical power; and embedded processors with severe limitations on the amount of memory and computing power available to them. Many of these devices are mobile, changing not only geographic position, but also their place in the topology of the network.

Such a network requires that we rethink how we approach the organizing and constructing of networks. In a network that is in constant flux, with new kinds of devices coming into the network and old devices leaving, our methods for administering the network must change dramatically. If they don't, we are doomed to spending all of our time doing network administration. In a network where the differences in available computing power, memory, and stable storage differ by multiple orders of magnitude, we must control where computation happens, as well as where information is located. And in a network where most of the communication is done by computing elements (hardware or software) rather than people, we need new approaches to the ways in which information and services are discovered and described, ways that are appropriate for the skills of the computing elements involved.

CHAPTER 3

Networking at the Edge

One way to envision what this new environment will be like is to imagine what life will be like at the edge of the Internet. Today, the edge is that part of the network mostly filled with clients. It is predominantly made up of desktop-class machines accessing the Web through browsers. Whether the edge is thought of as the machine, browser software, or user is irrelevant. The edge is made up of components that have these characteristics in common:

- Access to high levels of computing power, reliable energy sources, and reasonable (and rapidly growing) network bandwidth.
- Used by people (or are people) who are looking for read-mostly information.
- Change slowly, are not used as a source of information, and stay in a constant location (at least as far as the network is concerned).

However, as we move to a world where the Internet is used as an infrastructure for embedded computing, all this will change. As we connect our appliances, automobiles, and homes to the Internet; as we begin to access the Internet with mobile, hand-held devices; and as we use the Internet for activities such as environmental monitoring and automated remote diagnostics, the edge will be made up of components with these characteristics:

- Many will have small, inexpensive processors with limited memory and little or no persistent storage.
- They will connect to other computing elements without the direct intervention of users.
- Often, they will be connected by wireless networks.
- They will change rapidly, sometimes by being mobile, sometimes by going on and offline at widely varying rates. Over time, they will be replaced (or fail) far more rapidly than is now common.
- They will be used as a source of information, often sending that information into the center of the network to which they are attached.

This edge of the network requires a new kind of infrastructure than what we have used in the past. In particular, the heterogeneity and transient nature of the participants on the edge require different approaches than previous networks, which had relatively homogeneous participants and a somewhat stable infrastructure. Rather than requiring explicit installation and removal from the network, this kind of edge calls for components that are able to join and leave the network in a far more ad hoc fashion. Since the services on the network will need to be found by programs rather than humans, a new way of describing and locating those services must be designed that is centered around nonhuman participants.

A more straightforward characterization is to note that the edge is subject to radical and nearly constant change. This includes the kinds of processors and capabilities that are available to devices on the edge, the kinds of networks that are used for connection on the edge, and the kinds of information that are sent from the edge. Indeed, even the greater number of faults and movement of components on the edge can be thought of as system-wide change in the participants and their locations.

At any given point in time, this change is a defining characteristic of the edge of the network, and is even more pronounced when we look at the edge over time. Moore's law tells us that, over any given 18-month period, the basic hardware components used in devices at the edge will double in power or halve in price. Either of these developments will allow new kinds of devices and services to be added to the edge of the network. And this constant change must be absorbed without the network itself ever being taken down. All of the components that exist on the edge of the network should be designed for constant change; those that are not will be obsolete before they can be produced. Further, the infrastructure to which those devices are connected must allow for this constant change, without requiring alternations in the infrastructure itself.

CHAPTER 4

Designing for the Edge

It was just such a network that formed the design center of the Jini™ networking technology. A full discussion of the technology is beyond the scope of this paper. However, a characterization of some of its major features will help explain what follows.

Jini Networking System

The Jini networking system is a distributed infrastructure built around the Java™ programming language and environment. The basic communication model is based on the semantic model of the Java Remote Method Invocation (RMI) system, in which objects in one Java virtual machine communicate with objects in another Java virtual machine by receiving a proxy object, which implements the same interface as the remote object. This proxy object deals with all communication details between the two processes, and it may introduce new code into the process to which it is moved. This is possible because Java bytecodes are portable, and is safe because of the built-in verification and security of the Java environment. It should be emphasized that the Jini system uses the RMI semantic model; it does not require that the communication over the wire be done using the RMI implementation contained in the Java platform. Since that communication takes place between the remote implementation and the proxy object that came from that implementation, it can be done using RMI or other communication mechanisms such as CORBA, XML-based messaging, or custom protocols.

To this underlying communication model, the Jini system adds some basic infrastructure and parts of a programming model. The infrastructure provides a mechanism by which clients and services can join into the Jini network, while the programming offers a mechanism to build reliable, distributed systems.

The infrastructure centers around the Jini lookup service. This establishes a place for providers of a service to advertise their availability, and for clients desiring a service to find out what is available. Bootstrapping is accomplished using the Jini discovery protocol, by which an entity (either hardware or software) can find a Jini lookup service when it first connects to the network. The discovery protocol is the specification of a multicast packet which, when received by a Jini lookup service, causes the lookup service to send back a Java object that implements the lookup service interface. This object is loaded into the sender's process. Then, the sender registers itself with the lookup service if it wishes to provide services, or queries the lookup service for other service providers. A Jini participant can be both a service and a client of other services.

Services describe themselves by the Java language interfaces that they implement. So when a client wants to find a service, the client requests an object that implements a particular Java interface. Any object that is an instance of the requested type can be returned to the requester. This includes objects that implement a subtype of the requested type, or a set of types that includes the requested type.

The interfaces that encapsulate the programming model center around three kinds of object interactions:

- A set of interfaces defines the notion of an event notification, allowing an object to give other objects the abilityto register for notification when a conceptual event occurs in the first object.
- A set of transaction interfaces defines a two-phase commit protocol that can be used by objects to coordinate activities within the objects.
- A leasing interface defines a model of time-based resource allocation, in which a resource is granted for a renewable period of time, determined by a single-round negotiation between the resource requester and the resource granter.

Renewal of a lease can be requested by the holder of the lease, but if the lease is not renewed before it expires, both the grantor and the holder of the lease can assume that the resource is now available to others.

The infrastructure within the Jini system uses the Jini programming model. This enables the lookup service to register interest in various kinds of events, allowing participants in a Jini network to track changes in the set of services available via the lookup service. Registration of a service within a lookup service is also leased, so if a service does not renew the lease, it is deleted from the set of available services. This ensures that the lookup service does not advertise services that no longer exist for any period of time longer than the maximum lease period, as such services will not renew their leases.

From this brief description, it should be clear that the Jini system is tied to the Java programming language and platform. Java provides a mechanism that allows mobile objects, including their code, to be safely and efficiently moved from a service to a client of that service. This system is used to identify services. Its polymorphic nature allows requests for a service to be treated as requests for something that implements a certain type, and it may offer more. However, it should be noted that the requirement for the Java language and platform is only at the network level; objects that do not utilize the Java language in any way may be used to implement a Java network object that lives within the Jini world.

Ad Hoc Networking

What is often less clear is that the Jini networking system is built around requirements that will be presented by the edge of the Internet in the near future. This is because Jini technology is designed to allow ad hoc networking, in which the network participants can join and leave the network with minimal human intervention. So a Jini system allows participants to join the network by finding one or more lookup services, which can then be used to either register a service being offered or find a service to use. Once a participant receives a connection to a physical network (using, for example, a mechanism like DHCP), the finding of a lookup service can be entirely automated. Leaving the network is equally automated. Clients may leave the network at any time, because any references the client has to services are leased. When those leases expire, the referenced resources can be reclaimed. Services may leave in an equally abrupt manner, since their registrations with the lookup service are also leased. After a lease expires, no client will attempt to use that service, because it is no longer registered and therefore will not be considered part of the network federation.

The Mathematics of Probability

What this means is that the participants in the network can change over time, whether by design or by accident. Just because a particular service was available on the network at some time in the past is not a guarantee that the service will continue to be available. When failures occur, services may be unavailable. As we know, one of the most common causes of accidental change is failure of a component or network link. No matter how reliable a particular component or network link is, there is no way to make them perfect. This is why the reliability of a component is always stated as a probability that the component will be available. The best we can do is add additional nines to the percentage of uptime (five nines is the current norm, meaning the probability that the component is up is 99.999 percent, and some components claim a sixth nine). The probability of all the components of the network being available is the product of the individual probabilities. Given that the reliability of a component is never going to be 100 percent, this means that the larger the set of components, the less likely it is that all components will be available.

Thus the basic mathematics of probability tell us that as the network gets larger and larger, the chance of some component failing increases. Assuming a network in which all networking links are completely reliable and all components joined by the network have a reliability of 99.999 percent, a network of 10 components will have a reliability of 99.99 percent. Such a network with 100 components will have a reliability of 99.9 percent, still acceptable. But the kind of networks we are talking about on an Internet scale have millions of machines. A network with one million machines will have a total reliability of .0045 percent. In other words, something is failing all the time (FIGURE 1).

1 component = 99.999% reliable

100 components = 99.99% reliable

1 million components = .0045% reliable

FIGURE 1 Overall Reliability of Networks

The Importance of Location

However failure, while important, is not the only change going on in such networks. As the clients at the edge of the network become more and more mobile, the act of moving from one location to another on the network will create change at the edge. Where a client is located may well determine at least some of the services that are visible to it, as well as alter preferences for which services are used. If I am using a service only to perform some transformation of information, I don't care where on the network that service exists. If the service produces something physical (like a printed page), I may care very much where the service resides. A compute engine in France may well serve my purposes no matter where I am. But if I am in Boston, I don't want to use a printer in London. Location changes must be reflected in the network, and the Jini notion of a lookup service (which reflects the local services that are registered) deals with this kind of change.

The Effect of Heterogeneity

Even when clients don't move, they will change over time. As new kinds of client devices are brought to market, the heterogeneity of the edge will increase. New processors and operating systems will change the environment of the edge of the Internet at a faster and faster pace, especially as innovations in hardware and the embedding of computing into devices that are not generally thought of as computers (such as appliances and automobiles) becomes more common. Jini technology copes with such change by adding a layer of abstraction to the processors and operating systems in the form of the Java virtual machine. While the set of devices on the edge will change rapidly, the homogeneity needed to allow interactions between those devices is introduced at a higher level, through the Java environment.

From Human Understanding to Program Interface

The switch from human-centered interactions to program-centered interactions is supported in Jini technology by how it identifies services in the system. Rather than using their names (as in naming systems) or descriptions (as in directory systems), the Jini lookup service identifies services by their Java language type. It does this because, while humans are very good at understanding names and textual descriptions, programs that can do this are not yet generally available. But today's programs are very good at understanding type systems. They must be told that something named "printer" will produce a hardcopy when requested, but they already understand that the function will be performed by something that supports the printing interface.

Network Evolution

While the use of the Java language type system for service identification was incorporated in Jini technology to support identification by programs rather than people, it had the unintended (but welcome) side effect of also supporting the kinds of change created by network evolution. Since Java is a polymorphic language, a particular service can have many types. This can happen when a type evolves through subtyping, or when a service supports multiple interfaces. So, for example, a service may be both a printing service and a color printing service (assuming that the latter is a subtype of the former), or a printing service and a faxing service (by supporting both the printing interface and the independent faxing interface). A client looking for a printer can get either a simple printer (it supports only the base printer interface), a color printer (it supports the subtype interface), or a multifunction device (which supports multiple, independent interfaces). Since Java is a reflective language, the client can find out exactly what kind of service was returned (if it wishes), or use it as a simple printing service without knowing what else it can do. This allows a simple but effective ontology to be integrated into the Jini lookup service, one that is supported directly by the programming language used for implementation, and which gives a rich, descriptive framework for the services being offered.

This framework also allows services to evolve over time without disrupting their clients. If a service that once offered only printing now wishes to also offer color printing (or faxing) of documents, it reregisters with the lookup service under the new (extended) type. Any client that previously used the service can still do so, since the service continues to support the original printing interface. But now, services that need the new color printing or faxing interfaces can use them as well. In this way, additional functionality can be introduced, without disrupting current clients or requiring they be updated in a coordinated fashion with the new service. This evolution is also a characteristic of the large-scale networks that we are beginning to see, and is another requirement for change in such networks.

A Wider Range of Protocols

A final form of change will be seen in the network itself. Unlike the networks we are used to, which are carried on cables and tend to speak a single network protocol, the edge of the coming Internet will have a variety of networks, both wired and wireless, using a variety of protocols. The kind of network being used for communication will change from place to place and over time, as various protocols are enhanced, invented, or optimized for the new situations and traffic that will characterize the network. This type of change has not occurred in our previous experience with networked systems.

The Jini Technology Approach

Jini technology allows the objects in the network to be genuinely distributed, that is, the objects that offer services can be thought of as existing on both the machine that is offering the service and the client that is requesting the service. A client accesses a service by downloading the proxy object for it. This proxy may bring its own implementation into the client address space when the object is downloaded, which is possible because of the portable and secure nature of Java bytecodes. So the client interacts with the service through the programmatic interface that defines that service, but the implementation of the interface comes from the service itself (via the lookup service). In turn, this means that the communication mechanism used by the client address and service address spaces is a private matter between the service and its proxy (which was supplied by the service). This allows the service to change the communication mechanism at any time, without having to assume coordinated change in the client. New protocols, new networks, and new ways of passing information from the client to the server can be introduced unilaterally by the service. The communication mechanisms can change as needed, since those mechanisms have become an internal implementation detail of the service.

The Jini technology approach is not the only way to solve the problems of change at the edge of the network. Other systems, such as Salutation (www.salutation.org), Universal Plug and Play (www.upnup.org), and the Service Location Protocol address some of these aspects of change. If the Jini technology is unique, it is due to the range of changes that it addresses in a unified fashion.

CHAPTER 5

The Fractal Network

When Jini technology was first introduced, it was seen as a way of attaching devices to the edge of the network. Even though it had been designed as a general infrastructure for the construction of distributed systems that were resilient in the face of the kinds of change that have been discussed, the perception was that the Jini technology would allow small devices to be attached to the network without explicit administration.

The problem with this perception is that the infrastructure on the small devices was insufficient to allow those devices to be full participants in a Jini network. Most of the devices cannot run a Java environment, and those that can often run a truncated version that does not allow the safe downloading of code into the device. More troubling is the fact that many of these devices have a very limited amount of memory available for programs. Moving more objects (such as the proxies for services) into such devices only postpones the point at which those devices will fail from lack of memory to some indeterminable time in the future, until an object that exceeds the device's memory capacity is downloaded.

Two technological answers are proceeding in parallel to answer these problems. The first is the effect of Moore's law on small devices, which while remaining small in comparison to desktop and server machines, are nonetheless becoming more powerful. Combined with the shrinking size of the Java environment, this means even small devices will soon be able to participate fully in a Jini network. The second technical answer is surrogate architecture, a project being undertaken in the Jini community, in which a service on the network offers itself as a home for objects representing small devices that cannot host a full Java environment themselves. Surrogates will be loaded and run in the host's address space. Because the surrogates originate with the small devices, they know intimate details about their hosts. By moving an object away from the memory-limited device onto a host with a full Java environment, the limited device ensures its integrity. The surrogate can join the Jini network from the host machine. Services are connected to the limited device through the surrogate.

Defining the Edge

As the technology evolves to allow the fulfillment of the original purpose of the Jini platform, early adopters have realized that it is applicable to a much wider set of problems than just attaching a device to the network. Jini technology was defined to solve the problems that arise when attempting to deal with changes found at the edge of a network. Developers soon realized that any node in a network, when looked at from the point of view of that node, is an edge in the network. Each edge must face the challenges of change, which may manifest themselves in slightly different ways, but are basically the same. This reveals the fractal nature of the network; every point in the network is an edge, confronted with a set of problems revolving around change that, while not identical, are similar.

Building Software Networks

Consider, for example, the problems faced by the designer of a software service. The service can be decomposed into a set of constituent services, each defined by an interface. Some of these may be used by other services, and some may be used only by this service. Composing the larger service requires finding services that implement its constituent parts. Making sure that the larger service is robust is, in part, ensuring that the loss of any of the component services will not stop the large service from responding to requests. This can be done by making the components independent of each other, and providing alternate implementations for each constituent service. At times, the constituent services may need to be replaced by updated implementations. But these updates should not require that the overall service be halted, and new implementations should not require changes in the other, cooperating components that make up the service.

All of these requirements can be supported by Jini technology. Instead of registering devices in the lookup service, software components get registered. Each component is found by the interfaces that it supports, allowing components to be found by other pieces of software. Alternate implementations of the same interface may be registered. This allows an implementation to be removed in an ad hoc fashion from the network, and replaced by an updated implementation. Since all interaction is via proxy objects registered by various components, such changes are transparent to other components. If a component must be changed to support additional or enhanced (extended) interfaces, existing components can use the enhanced component just as they used the original. New components may be introduced at will to exploit new functionality.

In this case, Jini technology is being used to build ad hoc networks of software components rather than devices. These software services live, not on what we generally think of as the edge of the network (that part of the network that mediates between services and end users), but on the edge that is the membrane between the network and back end services offered on the network. Both edges are subject to radical change over the lifetime of network components, requiring that the overall network continue to function in the face of change. The particular kinds of change seen on the two edges are different, but the need to react and adapt to those changes is the same.

Capacity Planning

Another example may be seen in the case of a system administrator who needs to plan capacity for a Web server. The demand profile for the services being offered can vary by as much as two orders of magnitude, and peak demand may last as long as an hour at a time. Having enough capacity to deal with the peak means that way too much capacity is being wasted the rest of the time. But not having sufficient capacity at the peak means losing customers, which is equally unacceptable.

By using the Jini networking approach, servers can offer different services at certain times of day. During peak periods, servers may change from supporting nonvital services to high-demand services. When the peak has passed, some servers may return to supporting services that are not time critical, but which are nonetheless necessary. The ability to quickly change the services being offered without bringing down the network service is another kind of change seen at this edge of the network. While the particular problems are different from connecting a device or building a software service out of components, the solution is similar because, at root, the problem is one of dealing with change.

These are not hypothetical cases, they are examples of real companies solving real problems with Jini technology. Other projects within the Jini community differ in detail, but are also centered around the problem of dealing with various kinds of change in the network.

CHAPTER 6

Conclusion

Networks are expanding and changing in ways that would have been hard to imagine only a few years ago. Part of this change is due to the rate of change of the networks themselves. All aspects of the network — the devices that are connected, the way in which services are offered, the kinds of services offered, and even the connecting tissues of the protocols and physical transports used by the network itself — are changing.

This demands a new kind of networking infrastructure, one that is built around the notion of change. Originally devised to connect devices to the network, Jini technology is now seen as a way of dealing with the problems posed by a constantly changing network to the development and deployment of services. Jini technology is not the final step in the infrastructure's evolution, but it does point in the right direction. As our networks become larger, more heterogeneous, and as nonstop access to services becomes more essential, Jini technology offers a solution. Rather than being an end point, Jini technology enables an infrastructure built around a fractal notion of the network, where change is a constant.

Resources

Additional information about Jini network technology is available at www.sun.com/jini. In addition, please see:

- Waldo, Jim and the Jini Technology Team, *The Jini Specifications (Second Edition)*, 2001, Addison Wesley, Reading, MA. Available through www.sun.com/jini/books.
- The Community Resource for Jini Technology, www.jini.org.

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303

1 (800) 786.7638
1.512.434.1511

http://www.sun.com/consumer-embedded/

May 2001