A

Seminar report

On

Jini Technology

Submitted in partial fulfillment of the requirement for the award of degree
Of Computer Science

SUBMITTED TO:
www.studymafia.org

SUBMITTED BY:
www.studymafia.org

Preface

I have made this report file on the topic **Jini Technology**; I have tried my best to elucidate all the relevant detail to the topic to be included in the report. While in the beginning I have tried to give a general view about this topic.

My efforts and wholehearted co-corporation of each and everyone has ended on a successful note. I express my sincere gratitude to ……..who assisting me throughout the preparation of this topic. I thank him for providing me the reinforcement, confidence and most importantly the track for the topic whenever I needed it.

Acknowledgement

I would like to thank respected Mr…….. and Mr. ……..for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a seminar report. It helped me a lot to realize of what we study for.

Secondly, I would like to thank my parents who patiently helped me as i went through my work and helped to modify and eliminate some of the irrelevant or un-necessary stuffs.
Thirdly, I would like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Next, I would thank Microsoft for developing such a wonderful tool like MS Word. It helped my work a lot to remain error-free.

Last but clearly not the least, I would thank The Almighty for giving me strength to complete my report on time.

ABSTRACT

Jini 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, database, servers, devices, printers, storage, etc.) and clients(request 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 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 line implementation of a service might be XML-based and another RMI-based, 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.

INTRODUCTION ABOUT JINI

Jini is a simple set of Java classes and services that has the potential to create its own revolution because it allows technology to be exploited in new ways.

Created by Sun Microsystems as software for networking in all sorts of electronic devices, services, and applications, Jini lets them join up easily, seamlessly and gracefully - it is a sort of plug-and-play capability for spontaneously forming networks of heterogeneous equipment to share code and configurations transparently.

And Jini has the potential to radically alter our use of computer service networks, since it allows and encourages new types of services and new uses of existing networks.

HISTORY OF JINI

The idea of the Jini system was invented by Sun cofounder Bill Joy at Sun Aspen Small works R&D lab in 1994.

Sun introduced Jini in July 1998. In November of 1998, Sun announced that there were some firms supporting Jini.

The word 'jini' means "the devil" which is the origin of the English word 'genie'. On January 25, 1999, Jini was officially launched and the technology is available for download. In September 1999, 18,000 click-thru agreement /downloads of Jini release from Sun's web site.

WHAT IS JINI

Jini is a set of specifications that enables services to discover each other on a network and that provides a framework that allows those services to participate in certain types of operations. For an instance, take a Jini-enabled laptop into a Jini-enabled conference room and the laptop automatically be able to find and use the services of the conference room such as the laptop will automatically find the printer inside the room, seamlessly download any drivers required by the printer and will send its output to the printer. But Jini is not about hardware and devices. Jini is all about services.

A Jini-enabled printer provides a print service to the network; that the particular service is provided by a piece of hardware is irrelevant. There may be a software service on the network that also provides the same print service by accepting print requests, rasterizing them and emailing the rasterized images to a distant recipient. There are many services that are only software.

Thus Jini not only allows hardware and applications to interact but also allows this interaction to happen in a dynamic, robust way. Jini software also gives network devices self-configuration and self-management capabilities. It lets devices communicate immediately on a network without human intervention. And Jini has the potential to radically alter our use of computer service networks, since it allows and encourages new types of services and new uses of existing networks.

These networks are self-healing in that devices that leave the network for any reason, such as machine crashes or power surges, do not affect the remaining devices' operation. A Jini client that loses contact with a server can recover and continue processing. It is precisely these features that make Jini technology ideal for embedded systems in a dynamic environment. But network plug-and-play capabilities and self-configuration are also attractive for enterprise systems.

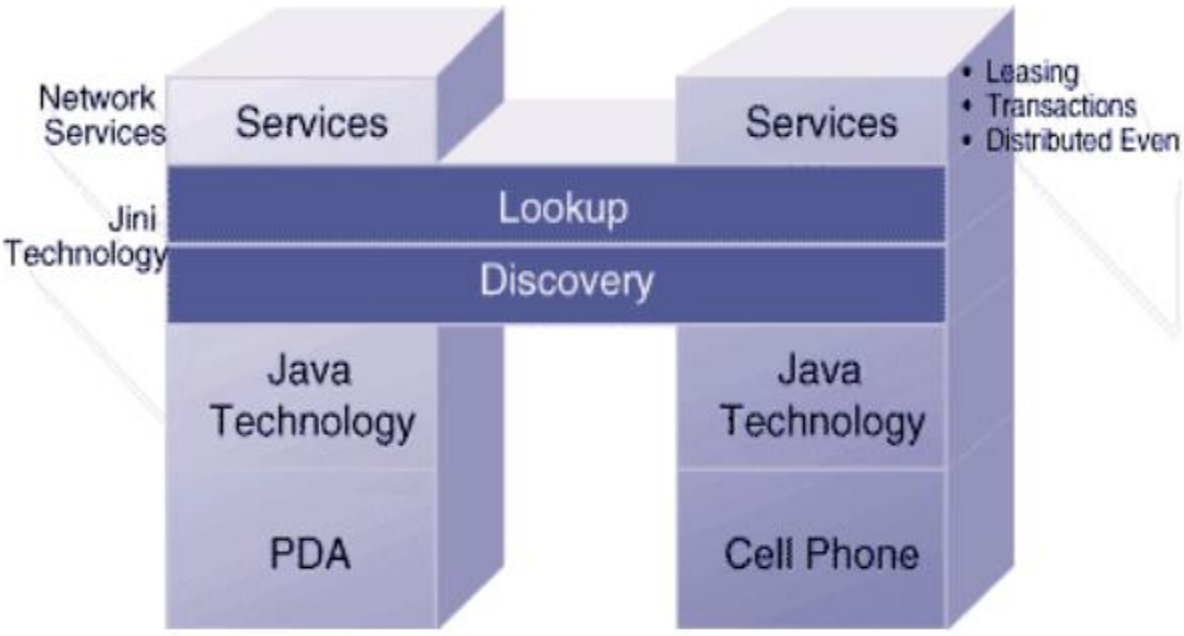
GOAL OF JINI TECHNOLOGY

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 flexible.

A JINI system of consist 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.

ARCHITECTURE



When you plug a new Jini-enabled device into a network, it broadcasts a message to any lookup service on the network saying, in effect, "Here I am. Is anyone else out there?" The lookup service registers the new machine, keeps a record of its attributes and sends a message back to the Jini device, letting it know where to reach the lookup service if it needs help. So when it comes time to print, for example, the device calls the lookup service, finds what it needs and sends the job to the appropriate machine. Jini actually consists of a very small piece of Java code that runs on your computer or device.

Jini is a set of APIs and network protocols that can help you build and deploy distributed systems that are organized as federations of services. A service can be anything that sits on the network and is ready to perform a useful function. Hardware devices, software, communications channels -- even human users themselves can be services. A Jini-enabled disk drive, for example, could offer a "storage" service. A Jini-enabled printer could offer a "printing" service. A federation of services, then, is a set of services, currently available on the network, that a client (meaning a program, service, or user) can bring together to help it accomplish some goal.

Jini defines a runtime infrastructure that resides on the network and provides mechanisms that enable you to add, remove, locate, and access services. The runtime infrastructure resides on the network in three places: in lookup services that sit on the

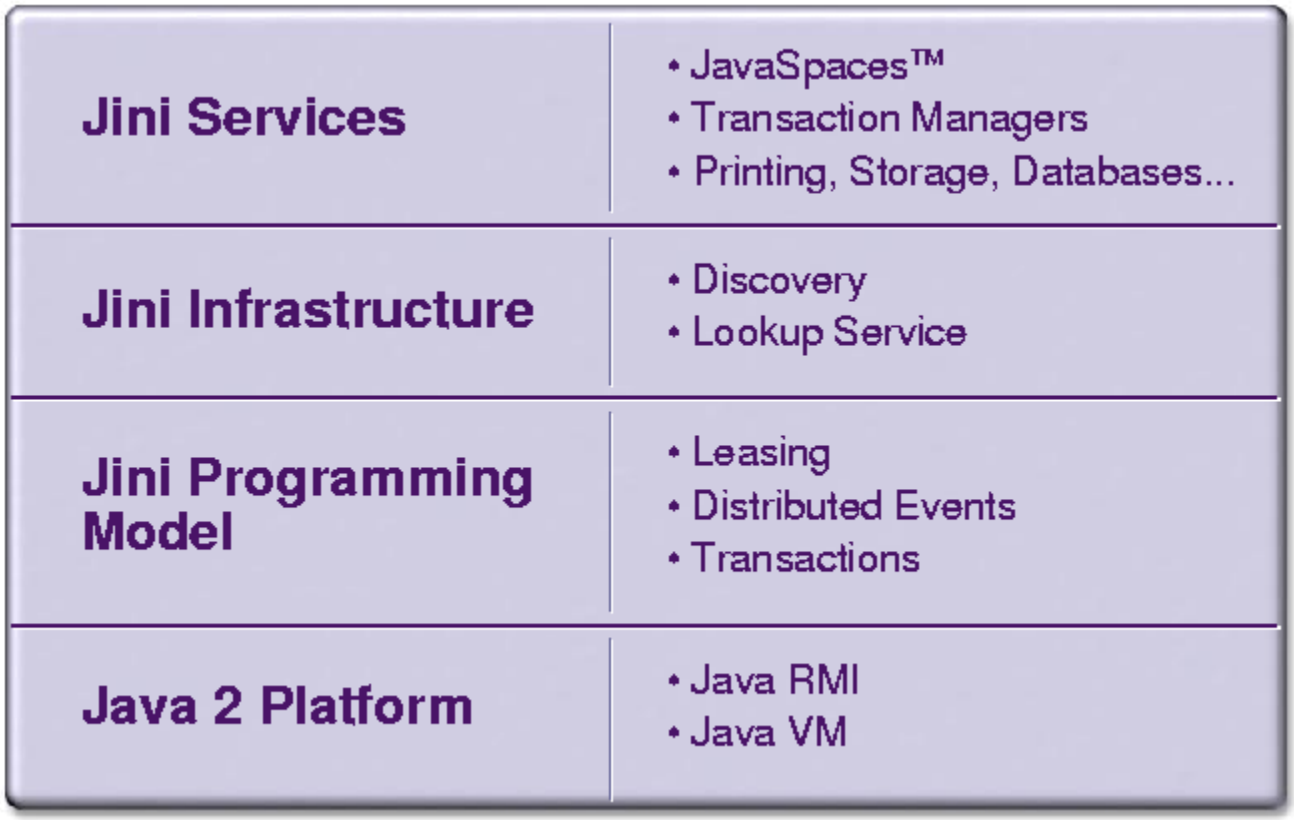
network; in the service providers (such as Jini-enabled devices); and in clients. Lookup services are the central organizing mechanism for Jini-based systems. When new services become available on the network, they register themselves with a lookup service. When clients wish to locate a service to assist with some task, they consult a lookup service.
The runtime infrastructure uses one network-level protocol, called discovery, and two object-level protocols, called join and lookup .Discovery, illustrated in the middle of Figure 1, enables clients and services to locate lookup services. Join enables a service to register itself in a lookup service. Lookup enables a client to query a lookup service for services that can help the client accomplish its goals.

Discovery works like this: Imagine you have a Jini-enabled disk drive that offers a persistent storage service. As soon as you connect the drive to the network, it broadcasts a presence announcement by dropping a multicast packet onto a well-known port. Included in the presence announcement is an IP address and port number where the disk drives can be contacted by a lookup service.

Lookup services monitor the well-known port for presence announcement packets. When a lookup service receives a presence announcement, it opens and inspects the packet. The packet contains information that enables the lookup service to determine whether or not it should contact the sender of the packet. If so, it contacts the sender directly by making a TCP connection to the IP address and port number extracted from the packet. Using RMI, the lookup service sends an object, called a service registrar, across the network to the originator of the packet.

The purpose of the service registrar object is to facilitate further communication with the lookup service. By invoking methods on this object, the sender of the announcement packet can perform join and lookup on the lookup service. In the case of the disk drive, the lookup service would make a TCP connection to the disk drive and would send it a service registrar object, through which the disk drive would then register its persistent storage service via the join process.

COMPONENT OF JINI SYSTEM



Infrastructure is the set of components that enables building a federated Jini system:

- **The lookup service,** which serves as a repository of services. It reflects the current members of the federation and acts as the central marketplace for offering and finding services by members of the federation
- **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
- **A distributed security system,** integrated into RMI, defines how entities are identified and how they get the rights to perform actions on their own behalf and on the behalf of others

Programming Model:

A set of interfaces that enables the construction of reliable services, including those that are part of the infrastructure and those that join into the federation

- **Leasing interface**: defines a way of allocating and freeing resources using a renewable, duration-based model
- **Event and notification interface**: a notification mechanism between services and resources that stretches beyond machine boundaries
- **Transaction interface**: wrapping of multiple operations on multiple services into a single unit of work that either completely succeeds or is rolled back

Java2 Platform:

- **Java RMI:** It is an API that provides a mechanism to create distributed application in java. It allows an object to invoke methods on an object running in another JVM. The RMI provides a remote communication between the application using object stub and skeleton.

- **Java VM:** JVM provides runtime environment in which java byte code can be executed. JVM are available for many hardware & software platforms. It performs four tasks.
  - ✓ Load codes
  - ✓ Verify codes
  - ✓ Execute codes
  - ✓ Provides runtime environment

How Jini Works

The first action of a Jini-enabled device after connecting to the network is locating the Lookup service, which resides on a server in a network. It accomplishes this by using the discovery protocol, which is multicast-based.

For this, a device first sends a message to every other Jini device on the network that is within a certain number of network hops. When the Lookup server receives this broadcast request, its returns its address to the Jini client, which stores the address as its link to the Lookup server.

The client thereafter sends requests to the Lookup server directly to this address. Similarly Jini servers offering services uses the discovery protocol, which is multicast-based, to find a Lookup Service and then uses the join protocol to register and thus become available to client applications. A Jini client asks the Lookup service for a list of Jini servers that match the requested attributes. It is then up to the Jini client to select the specific Jini server from the returned list.

Similarly Jini servers offering services uses the discovery protocol, which is multicast-based, to find a Lookup Service and uses the join protocol to register and thus become available to client applications. Services may join multiple lookup servers for increased reliability. Once clients and services are paired up, they no longer need the Lookup Service's assistance and can work together directly.

The primary job of a Lookup service is to act as a central market for Jini services scattered around the network, grouping similar ones together and making them accessible to client application programs. The Lookup service maintains the maps between each Jini service and its attributes

Server-side Processing

A unique feature of Jini technology is its ability to transfer executable code between Jini devices. This capability is similar to Java applets, which execute on Web browsers. With applets, Web servers download an entire applet executable code to the client, which executes the code.

In contrast, Jini software sends only the code for the interface that the client uses to communicate with the server; the rest of the program remains in the server and actually executes there.

Thus, in a way, Jini server teaches the Jini client how to communicate with it. This strategy mitigates one of the biggest problems with Java applets: their slow speed. The system downloads less because a Jini network usually only transfers the results of whatever code the server executes.

Leasing of Service

When the Jini client receives the lookup service reply that responds to its request for a service, it initiates a connection with the server. The Jini specification does not stipulate how an application should implement a service. The only requirement is that a Jini service must implement a specific interface, which the server supplies to the client. The client has no prior knowledge of any server and only knows how to communicate with the server through the given interface.

When a client accesses a resource through a service, it leases that service. A lease is a guarantee that the client may access that resource for a specific length of time. The client has the option of renewing the lease before its expiry.

If suppose the service does not receive an extension request within the alloted period of time, the resources may be allocated for other clients. This prevents a network or client failure from keeping an indefinite lock on the resource. A nonexclusive lease permits other clients also to access the service at the same time. A lease could be exclusive, allowing only one client to access a resource at any time.

BENEFITS OF JINI

- It is easy to add and remove services.
- Services can be relocated on the network without affecting users.
- Jini is open-source, meaning that the program code is freely available on the Internet and there are no fees for using it.
- The Jini architecture is scalable.
- Services are available immediately and are found automatically.
- Everyone can access to same information & resources.

LIMITATIONS OF JINI

- Depends on java/RMI or external mechanism for security.
- Does not scale well to very large systems because jini use lookup service as a broker between client & services.

JINI EXAMPLE

For say, a CD-ROM drive, registering itself with a Jini network is essentially a three-step process. When plugged into the network, the drive first looks for a Lookup Service by sending a data packet asking to be registered. Once the Lookup Service accepts it, the drive sends a file with the byte code, a client application will need to establish communication with the drive. The file is stored in a table that contains data on all the services the Lookup service can make available on the Jini network.

For this system to work, the marketplace must first be set up by placing a Jini Lookup Service, a simple piece of application software, on one or more hosts in the network. There can be more than one Jini Lookup Service in a system. Services seeking a market for their wares send discovery requests to multicast request servers for Jini Lookup Services. Upon hearing such a request, an interested Jini Lookup Service initiates a private conversation with the service to establish its joining in the market.

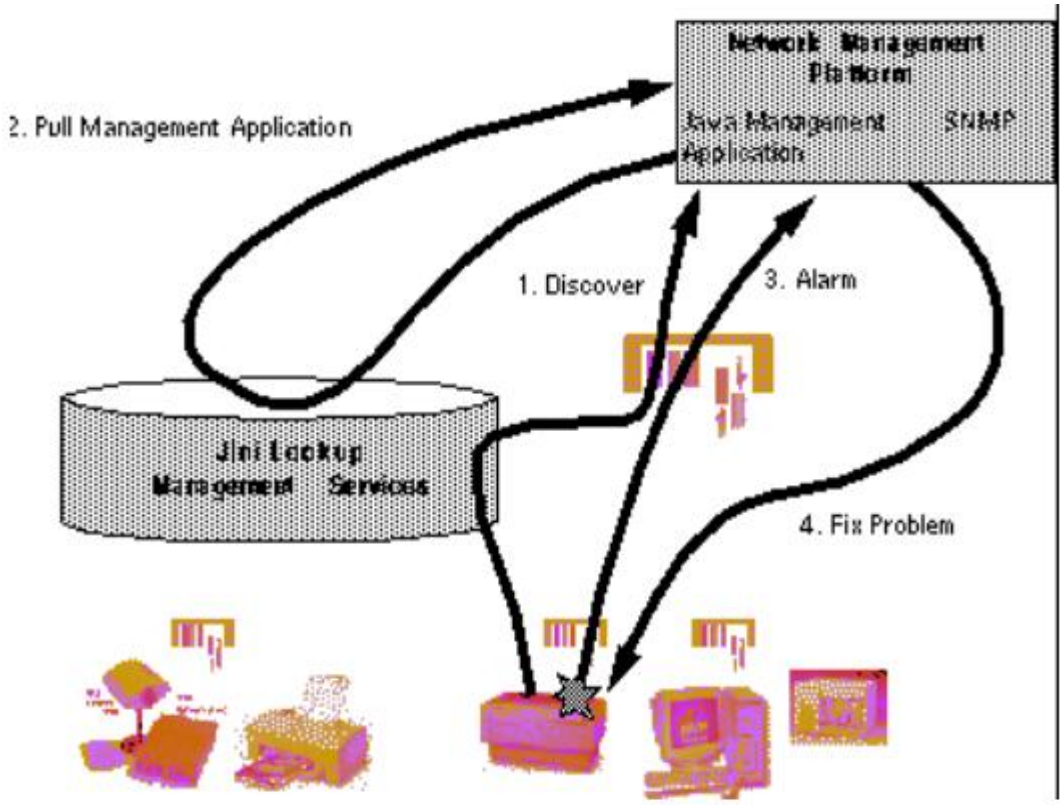
When the service and the Lookup Service first converse as part of this joining process, the Lookup Service promises the newcomer space in its market. In return, the service gives the Lookup Service basic information with which client applications can establish a connection to it. The information here is a software interface through which a Jini client can interact with a Jini service.

Client applications ask Jini Lookup Services for certain type of services. The client does this by sending a template file - a generic description of the desired service to the Jini Lookup Service. The Jini Lookup Service responds with a list of candidates that match the template from which the client may select a particular service. Thus Jini supplies the infrastructure for marketing a set of services, as well as provides the programming model by which resources may form impromptu groups called Jini federations and provide their services to users and other resources.

When a Jini-capable program that is a client application needs a service such as, say printing, it requests that the Jini Lookup Service send it a list of potential printer services. From the lists provided, the client selects one or more randomly, algorithmically or even manually through a user interface. For the selected service provider, the Lookup Service delivers the corresponding piece of executable Java byte code. While most typically this byte code would simply implement communications between the service and the client program, it could be the actual service itself.

APPLICATION JINI TECHNOLOGY

- Managing a Printer:

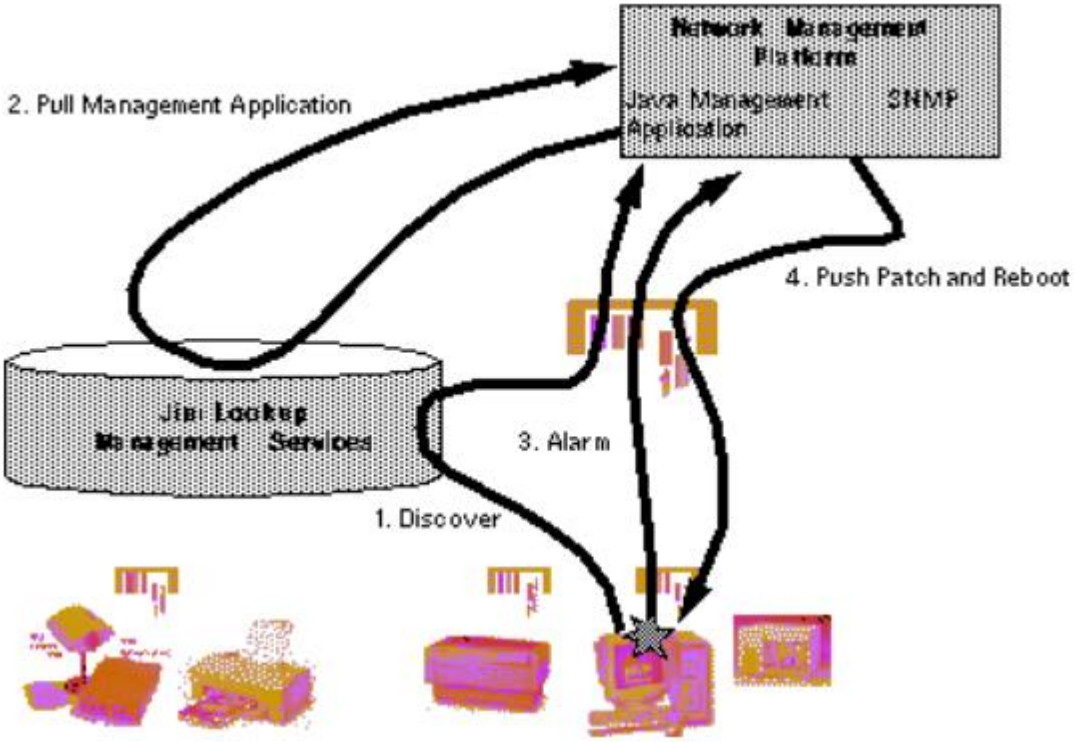


The particular example shows how alarms are forwarded from the Jini technology-enabled resource to the network management platform, and what actions are taken. First, the printer is connected to the network.

Through the Jini technology and the Java Dynamic Management kit, the printer is discovered and automatically appears on the map on the network management platform. Second, the system administrator, via a pop-up window on the network management platform, accesses the Jini technology-based lookup server directory to recover the management application from the printer.

Finally, when a problem arises, and a corresponding alarm is generated. One example might be "job too big." When this occurs, the job is cancelled from the network management platform, via the pop-up printer management application, and the owner warned of the cancellation. In addition, statistics on printer usage.

❖ Managing NT server:



A second example often sited is managing an NT server. As shown in Figure 4, using the Jini technology-based lookup service, the system administrator discovers the NT server BIOS as the NT server is booted, and adds it to the map on the network management platform. When the inevitable server crash occurs, a corresponding alarm is generated to the network management platform.

Then the system administrator again uses the lookup service directory, this time to recover the management application from the NT server. Finally, a patch is downloaded to the NT server, using the management application. The NT server is rebooted, using the management application. When the NT server is finally up and running, a cancellation alarm appears on the network management platform acknowledging things have returned to normal.

JINI ADVANTAGES

Jini developers intended Jini technology as a sophisticated platform on which to develop network-aware applications.

Jini technology provides users access to resources located anywhere on the network. Both user and resource locations can change without affecting the application. Users, devices, and resources can join and leave the network without manual reconfiguration.

Jini developers used the Internet as a model for developing their product and sought to take advantage of the Internet's advantages in terms of reliability, scalability, maintenance and administration, and security.

Jini is freed from having to deal with specific operating system and hardware requirements by Java technology, while Jini itself frees the client and service to interact without having to concern themselves withe the particulars of the network.

CONCLUSION

Jini's promise is not limited to the domain of network devices. It can be expanded to scanners, printers, phones, radios etc. Jini provides an extensive framework for developing flexible and robust distributed systems. It also deals with network failures better than the traditional object-based distribution solutions, because of its powerful and careful interface design.

A number of different distributed system frameworks exist as introduced in addition to the ones that where shortly mentioned in this thesis. The interoperability between these techniques is a key question, when considering the global development work in the future.

It seems that Jini is quite adaptable to these other solutions because of its flexible nature, but there is a lot of competition going on between vendors like, for instance, Sun Microsystems and Microsoft Corporation. The worst scenario is that the techniques of these competing vendors try to drop each other out from the market instead of trying to complement each other. In this kind of competition, the best technology cannot always survive.

More discussion about this rigid competition can be found in. Some research is already ongoing related to the complementing issues, and the results seem to be very promising , especially with Jini and SLP .In addition to the solutions for the challenges of distributed computing, Jini offers capabilities, which are required to fulfil the needs of modern end users of the network services.

Flexibility and spontaneity are the most important issues, which are possible to deploy, if a Jini based system is designed in a sophisticated way. As the degree of distribution of the services grows all the time, security issues are also very important. In this area, the developers of Jini still have a lot of work to do, because Jini does not provide any security mechanisms in addition to the basic Java environment properties.

REFERENCES

- www.silicon-press.com/briefs/brief.jini/
- www.jini.org
- www.devx.com/assets/download/4508.pdf
- Java.sun.com/developer/technical Articles/jini/.../Javatanks.html
- en.wikipedia.org/wiki/Jini
- www.studymafia.org