# Name Services

Bina Ramamurthy

Chapter 9



## Introduction

- You need to name an entity in order to use it.
- If you don't have a name or don't know a name you should be able to describe its characteristics in order to identify it.
- According to these two requirements we have two services:
  - Naming service
  - Directory service



## Naming Service

- Given the name of a resource returns the information about the resource.
- For example consider the **white pages**: given the name of a person you get the address/telephone number of that person.
- Other examples: LDAP (Lightweight Directory Access Protocol) a person on UB computers gives you information about the person's email, campus address, phone number, position held etc.



## Directory Service

- Given a description, find a service or resource that matches the description.
- For example consider the **yellow pages**: when you want to rent a car, it may give a list of car rental agencies.



## Names, Attributes and Addresses

- Names: human readable names such as /etc/passwd, URLs such as http://www.cd3.net/
- An address is an attribute of a name. Ex: *castor* is the name, its address is 128.205.34.1
- Identifiers: refer to names that are interpreted by programs. Examples: remote object reference, NFS file handles.
- Name Resolution: a name is resolved when it is translated into data about the name's resource or object.



## Names, Attributes and Addresses

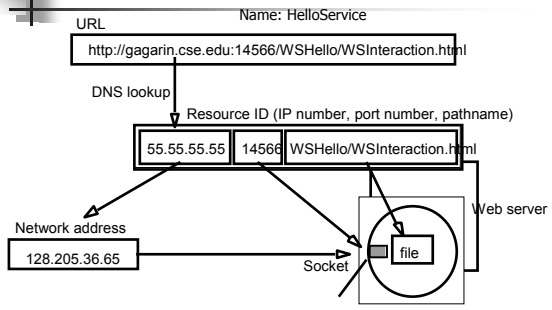
- Binding: association between a name and an object. In general, names are bound to the attributes of the object rather than an implementation of an object.
- Attribute: Value or property of an object.
- Names and services: many of the names are specific to some services.
- URL: principle means of identifying web resources

## Some Familiar Naming Services

- DNS: maps domain names to the attributes of a host computer.
- X500 maps person's name to personal information.
- CORBA naming service maps a name to remote object reference.



## Example






## Name Service

- A name service stores a collection of one or more naming contexts – set of bindings between names and attributes for objects such as users, computers, services and remote objects.
- Name Management is separated from other services because of the openness of the distributed system.
- Requirements:
  - Unification (EX: URLs, uniform names, UUID)
  - Integration: Share resources for resolving names.
  - Handle arbitrary number of names and domains
  - Long lifetime, High Availability, Fault isolation, Tolerance of mistrust



## NameSpaces

- Namespace: is a collection of all valid names recognized by a particular service (context). Requires syntactic definition.
- Can be flat or hierarchical: Hierarchical is scalable and reusable and can be managed separately.
- May provide aliases for names.
- Can be broken down into domains.

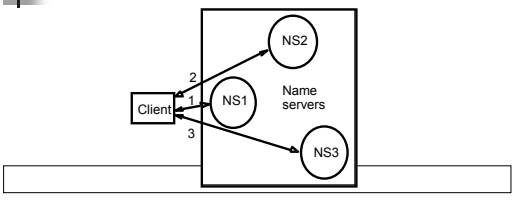


## Name Resolution

- Is a repetitive process in which a name is presented successively to naming contexts until its context is located or not locatable.
- When a context contains the name its attributes are returned.
- Navigation among the contexts can be iterative or recursive as shown in the next slides.

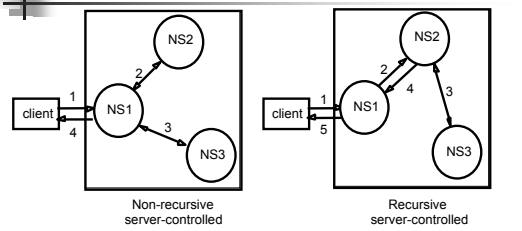


## Figure 9.2 Name Resolution: Iterative navigation




A client iteratively contacts name servers NS1–NS3 in order to resolve a name

## Figure 9.3 Non-recursive and recursive server-controlled navigation




A name server NS1 communicates with other name servers on behalf of a client

Navigation: Process of locating naming data from among more than one Name server in order to resolve a name. (iterative or multicast navigation)



## Directory Services

- A more powerful service than naming where you look up for names using the attributes than the other way.
- Clients can Lookup for services by providing their attributes rather the name.
- A discovery service provides registry and lookup for spontaneous networking.
- Registry is used by server to publish a service and lookup is used by a client to locate a service.



## Jini: A case study

- Jini (Jini Is Not Initials) is Java's solution to providing connectivity to services and devices.
- It is network-centric computing model as opposed to network-transparent model offered by CORBA and other earlier distributed system models. Software infrastructure that includes devices must be incredibly robust.
- The devices have to support true "plug and play".
- Devices and services should form spontaneous communities.



## Jini and Name Servers

- Jini does serves the functionality of a name server. But it is much more than that.
- Jini differs from names servers such as LDAP (Light Weight Directory Access Protocol) or DNS (Domain Name Service) in two aspects:
  - Services can appear and disappear without much overhead. Interested parties can be notified when a service changes.
  - Jini is self-healing. It accepts partial failures and has mechanisms for taking care of this.



## Five Key Concepts

**1. Discovery**
**2. Lookup**
**3. Leasing**
4. **Remote Events**
5. Transactions



## Jini Services/Devices

- Service providers in Jini can be:

1. **Pure software component**
2. Pure hardware device
3. Combination of the two

For obvious reasons we will consider only software services.

## Discovery and Lookup

- Discovery is Jini service that finds communities on the network to join to form a spontaneous "federation".
- It basically searches and locates lookup services of communities.
- Lookup service has the details of services, their location, code, attributes etc.

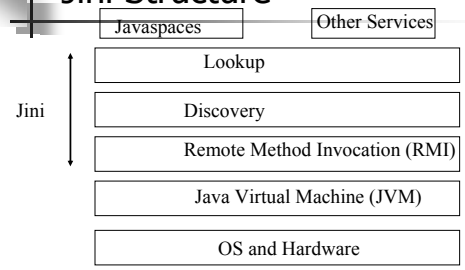


## Leasing

- Leasing is the technique that provides the self-healing characteristic of Jini.
- Every service provider keeps renewing its lease with the holder of the services (probably a lookup service) periodically. If it fails to update lease the service will be deleted from the community.
- This automatically removes failed or crashed server from the network thus carrying out the self-healing.



## Jini Structure






## Discovery

- To find and join a group of Jini services
- Sends out multicast packet and unicast packet
- Receives RMI reference to a lookup service where the requested service may be found.



## Lookup

- Repository of available services.
- Stores proxy of object and its attributes.
- Proxy can be thin or fat.
- Lookup interface: Registration, Access, Search, Removal
- Note: Best way to study a service is through its interfaces.



## Summary

- We studies the essential features of a Name Service.
- We also looked at some existing name servers.
- Jini extends the concepts of a simple name service to build a spontaneous networking distributed system model.
- Think about: How will you build a sophisticated name service using the common name service and the web services infrastructure?