

# Naming, Trading, Directory and Discovery Services (CDK – chap.9)

**“The *name* of a resource indicates *what* we seek, an *address* indicates *where* it is and a *route* tells us *how* to get there.” - J. F. Shock 1978**

- A name is a symbol (human readable string) identifying some resource or object. Highest level of identifier.
- The purpose of names in computing systems is to give independence from implementation structures.
- To be useful there will be some mechanism to map names into addresses. This mapping may change over time and the mapping can be done at runtime.

Key distributed systems issue is transparency. Users of a distributed system shouldn't need to know (ideally) anything about the underlying infrastructure.

Naming:

- Locating components by external names
- Similar to telephone directory

Trading & Directory services:

- Locating components by service characteristics
- Similar to yellow pages

Discovery services:

- Similar to Trading/Directory services
- “Find” the desired service

# Example Name & Directory Services

- **X.500 Directory Service**
- **Internet Domain Name Service (DNS)**
- **CORBA Naming Service**
- **Sun NIS+**
- **Java Registry**
- **Novell Directory Services (NDS)**

# Example Discovery Protocols

- **Multicast udp**
- **UPNP SSDP**
- **Jini**

# Name Spaces

## ***Global (Absolute)***

A single name space where there is a one to one mapping between names and entities so that one entity is known by one name to all other entities. *A white page directory for the whole world* . Simple but unmanageable for large distributed systems as the name space becomes too large.

## ***Local (Relative)***

A particular name may identify different entities in a different context. Names could identify the same entity or different entities depending on the context in which it is taken.

# Naming Services

- Name Assignment

Performed by the creators of the object where a name is bound to an object.

- Name Distribution

Apportions the responsibility for managing parts of the name space among naming authorities

- Name Resolution

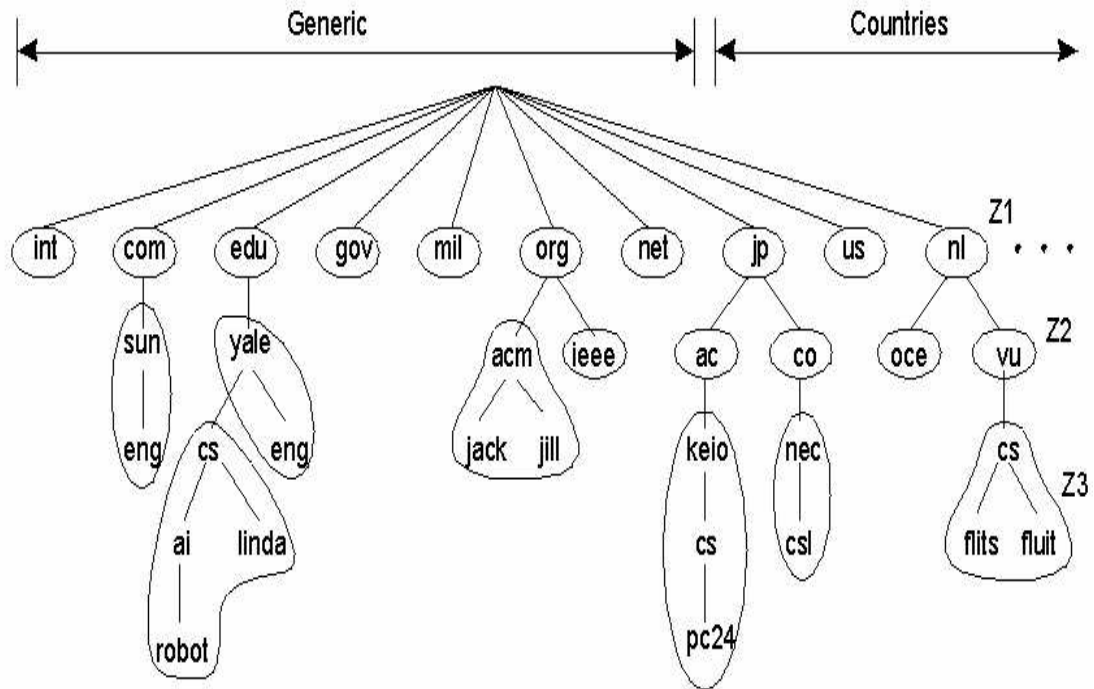
Denotes the process of determining the naming authorities for an object given only its name. Once the authorities are discovered operations can be invoked to read or update information about the object such as the location.

## Issues

### Qualities of service!

- Distribution of name spaces
- Performance profile
- Caching
- Replication
- Transaction properties of naming operations
- Naming servers are distributed systems

# The DNS name space

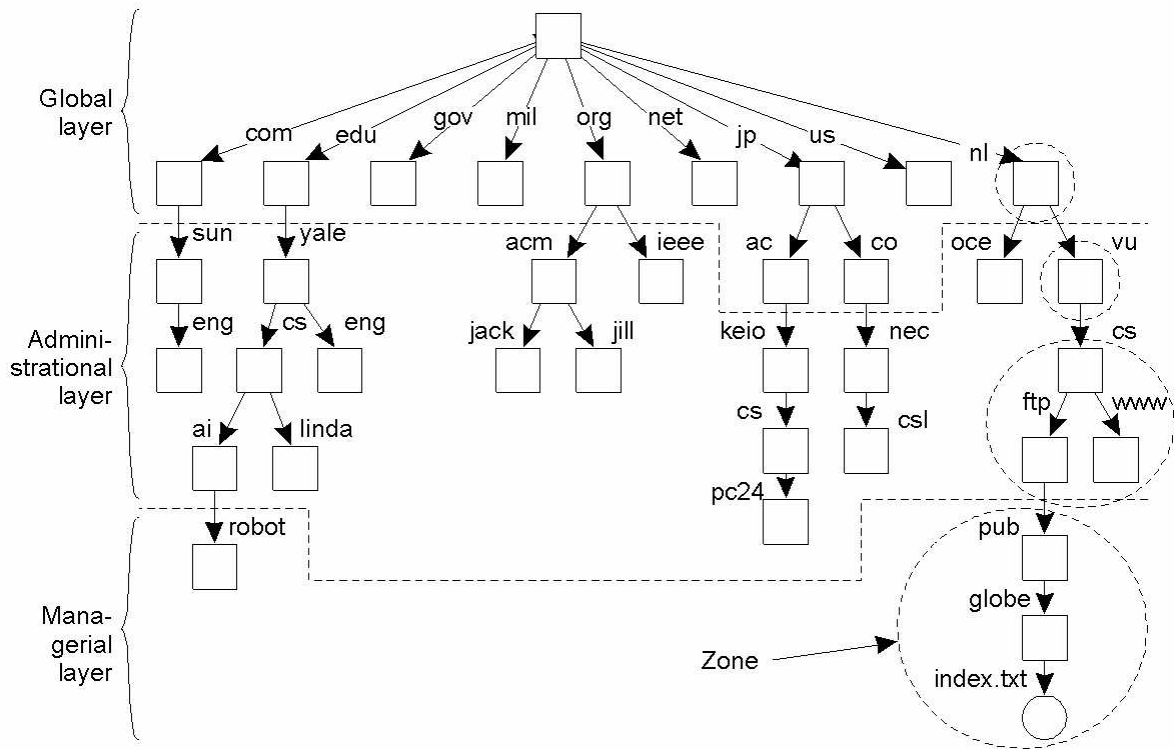


Divided into zones.

Zones contain one (or more) domains.

Name server(s) allocated to zones.

# DNS Name space partitioning

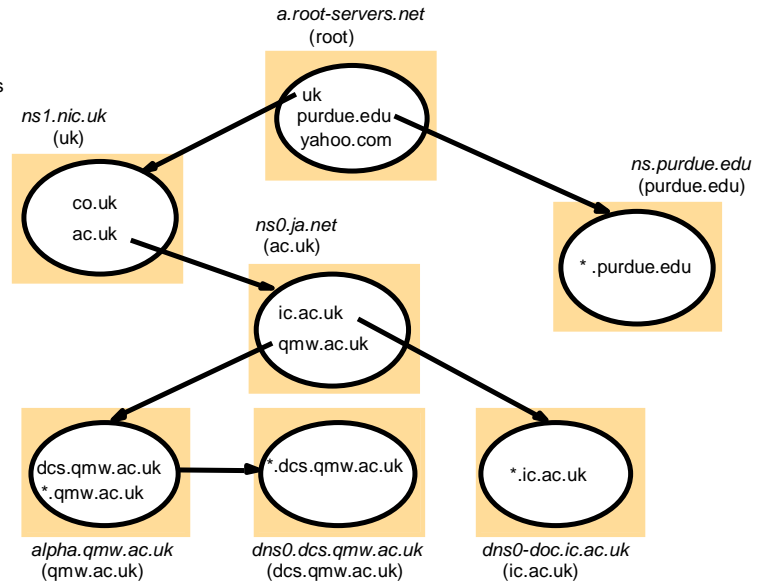


# DNS Name Space Distribution

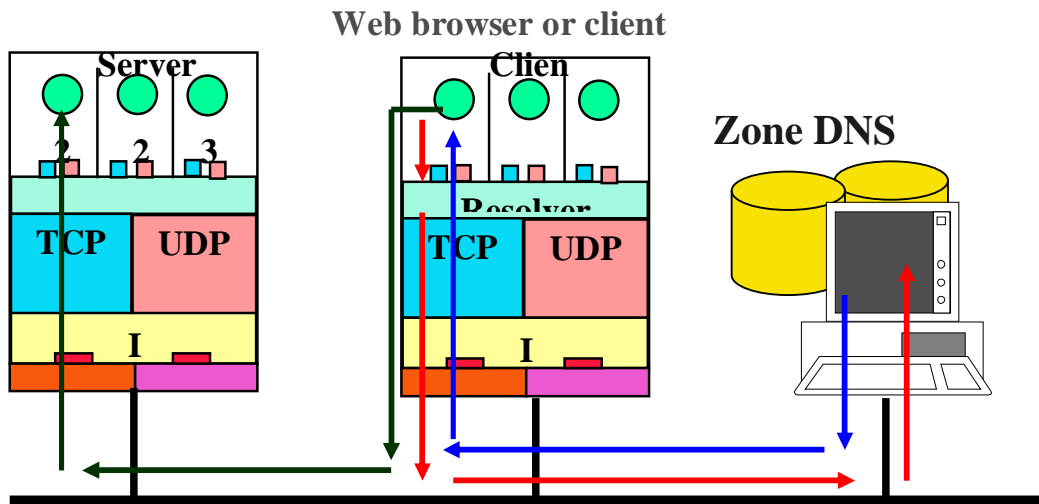
Item	Global	Administrational	Managerial
Geographical scale of network	Worldwide	Organization	Department
Total number of nodes	Few	Many	Vast numbers
Responsiveness to lookups	Seconds	Milliseconds	Immediate
Update propagation	Lazy	Immediate	Immediate
Number of replicas	Many	None or few	None
Is client-side caching applied?	Yes	Yes	Sometimes

# DNS Name Server Organisation

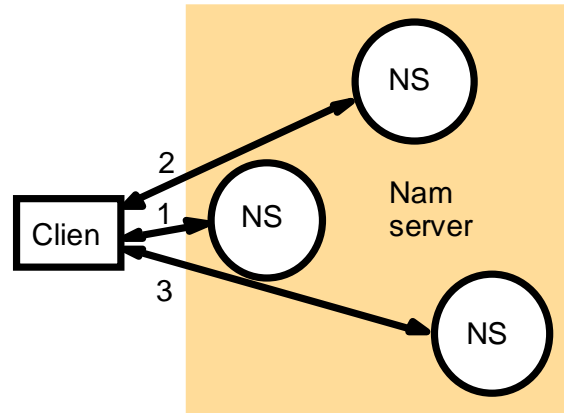
*Note:* Name server names are in italics, and the corresponding domains are in parentheses.  
Arrows denote name server entries



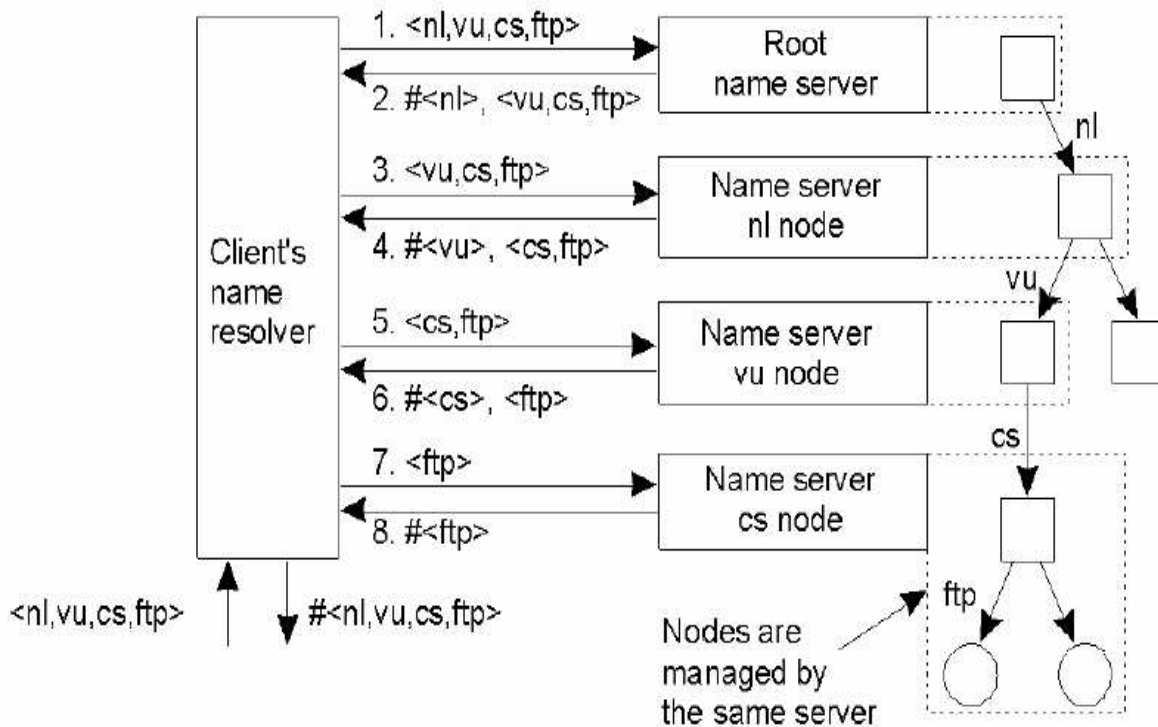
# Name Resolution



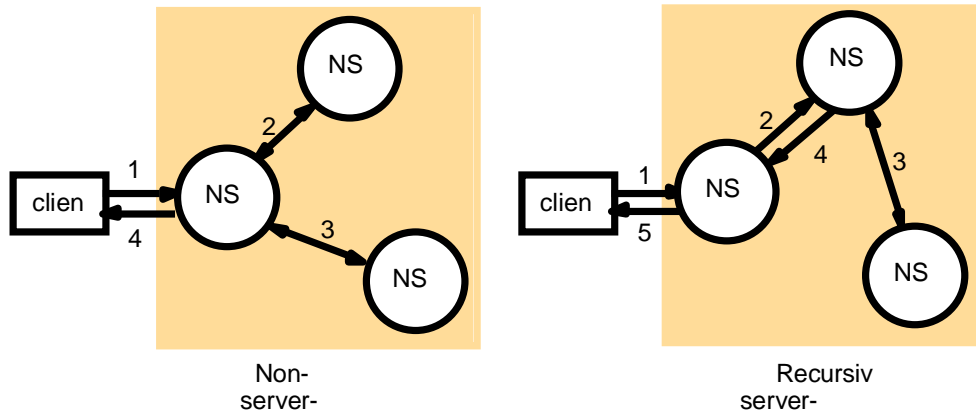
# Iterative Name Lookup



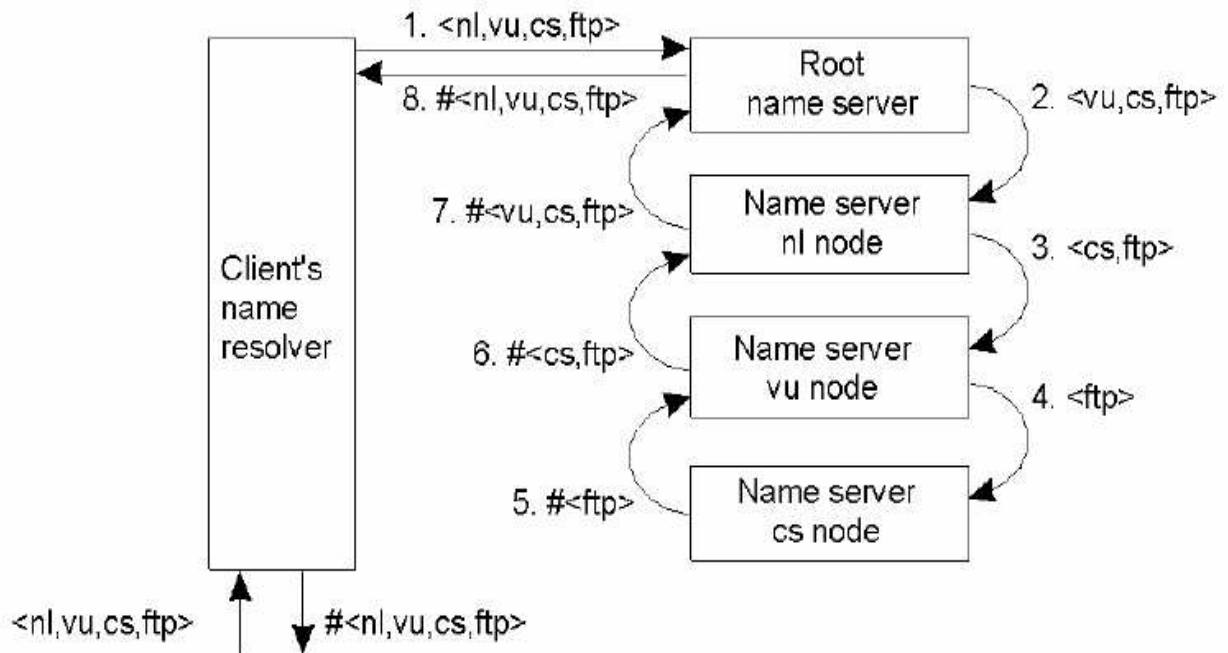
A client iteratively contacts name servers NS1–NS3 in order to resolve a name



# Recursive Name lookup



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

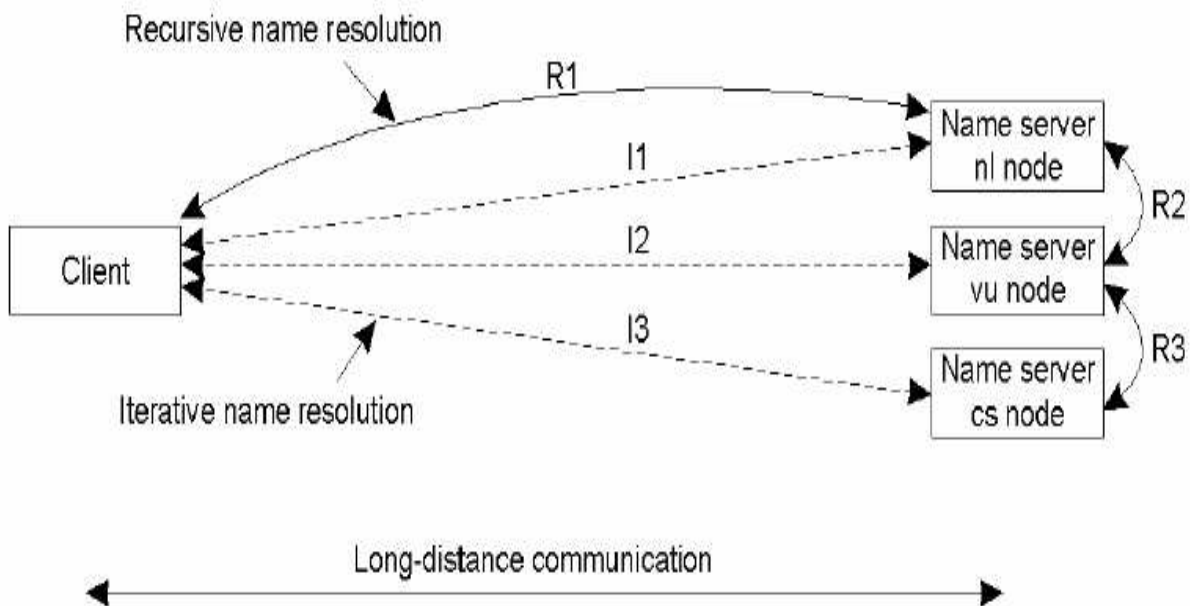


DNS permits both forms and includes caching

## Recursive Name Resolution

Server for node	Should resolve	Looks up	Passes to child	Receives and caches	Returns to requester
cs	<ftp>	#<ftp>	--	--	#<ftp>
vu	<cs,ftp>	#<cs>	<ftp>	#<ftp>	#<cs> #<cs, ftp>
ni	<vu,cs,ftp>	#<vu>	<cs,ftp>	#<cs> #<cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>
root	<ni,vu,cs,ftp>	#<nl>	<vu,cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>	#<nl> #<nl,vu> #<nl,vu,cs> #<nl,vu,cs,ftp>

## Comparison of Recursive & Iterative Resolution



# Types of DNS Records

<i>Record type</i>	<i>Meaning</i>	<i>Main contents</i>
A	A computer address	IP number
N	An authoritative name server	Domain name for server
ƆNAME	The canonical name for an alias	Domain name for alias
SO	Marks the start of data for a zone	Parameters governing the zone
WKS	A well-known service description	List of service names and protocols
PTR	Domain name pointer (reverse lookups)	Domain name
HINFO	Host information	Machine architecture and operating system
MX	Mail exchange	List of <preference, host> pairs
TXT	Text string	Arbitrary text

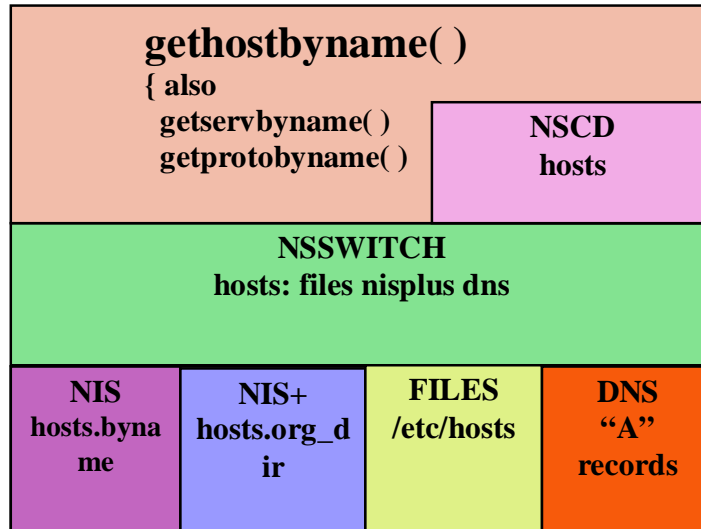
## DNS API

There exist a number of routines which will allow DNS searching and mapping of:

- **network names to network address numbers**  
*gethostname( ), gethostbyname( ), gethostbyaddress( )*
- **protocol names to protocol numbers**  
*getprotobyname( ), getprotobynumber( ),*
- **service names to port numbers**  
*getservbyname( ), getservbyport( ),*

# Organisation of name services (assuming Sun Solaris)

`gethostbyname()`  
call on Sun  
Solaris is able to  
leverage a whole  
number of  
directory and  
name services in  
a seamless  
manner.



NSCD = Name Server Caching Daemon – caches all below plus LDAP

# Limitations of DNS

Devised as the only Internet name service

Does not coexist (cf. NSswitch) with local name services such as

- Suns Network Information services NIS+
- Microsoft Active Directory Services
- 

Stores <name, address> pairs with name lookup rather than attribute lookup

Stores limited naming data

- computers
- name servers
- mail hosts

However being upgraded for IPv6

# Directory Services

Attribute lookup, much more useful both for local and Internet use.

Most often object oriented based information systems

Examples:

- X500
- Microsoft Active Directory Service

# Global Directory Services

**Large number of different network directories:-**

**Internet**

**DNS, SNMP MIBS**

**GSM Wireless**

**HLR, VLR, TMN MIBS**

HLR (Home Location Register) contains user information such as account information, account status, user preferences, features subscribed to by the user, user's current location, etc. The data stored in HLRs for the different types of networks is similar but does differ in some details.

VLR (Visitor Location Register) – when roaming

TMN = Telecommunications Management Network

**Network Operating Systems**

**Sun NIS++, Novel NDS, Microsoft ADS**

**Motivation to create more “intelligent” global network by making directory services more interoperable**

- **IETF DEN Directory Enabled Networking (CISCO, MS)**
- **LDAP a common directory access protocol**

# Naming & Directory Services

## Summary

Constructing a world-wide scaleable naming service using the ideas presented has been possible e.g., DNS

Directory services have enhanced basic naming servers by allowing object location search on class and attributes. Very useful when you don't remember the full name of what you seek.

However Directory Services are generally platform dependent with different object models and schemas so full universal search is not always possible

LDAP (Lightweight Directory Access Protocol) has emerged as a way of accessing many types of directories using one protocol. Most directories are LDAP enabled.

Mobile objects where name to address bindings change frequently presents a problem. Can lead to long chains of forward referencing in directories and name servers.

# Trading & Discovery

Locating objects in location transparent way

Naming simple but may not be suitable when

- clients do not know server
- there are multiple servers to choose from

Inappropriate if client wants to use a service of a certain quality but does not know /care who from

- Video on demand,
- Electronic commerce.

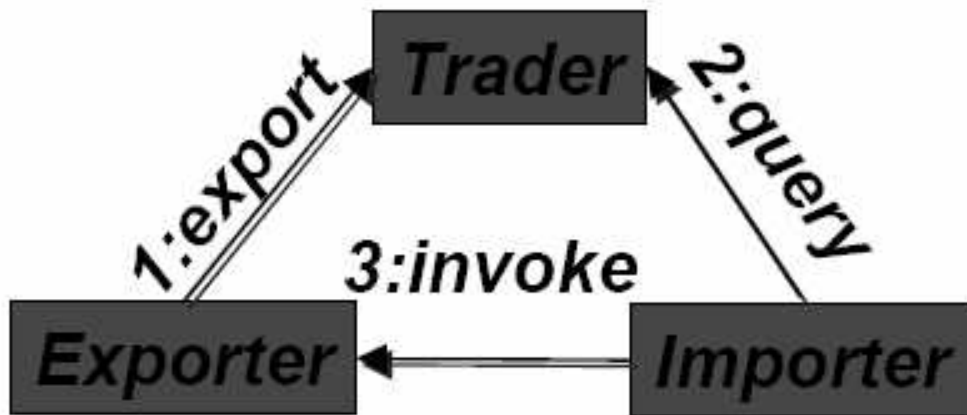
Trading supports locating servers based on service functionality and quality.

Naming = White pages

Trading = Yellow Pages

Trader operates as broker between client and server.

Enables client to change perspective from 'who?' to 'what?'



# Characteristics

**Common language between client and server:**

- **Service types**
- **Qualities of service**

**Server registers service with trader.**

**Server defines assured quality of service:**

- **Static QoS definition**
- **Dynamic QoS definition.**

**Clients ask trader for**

- **a service of a certain type**
- **at a certain level of quality**

**Trader supports**

- **service matching**
- **service shopping**

# Trader Policies

Depending on constraint and available services, a large set of offer might be returned by a query.

Trading policies are used to restrict the size of the matched offers

- Specification of an upper limit
- Restriction on service replacements
- Restriction on modifiable properties (these might change between match making and service requests)

# Federated Traders

Scalability demands federation of traders

A trader participating in a federation

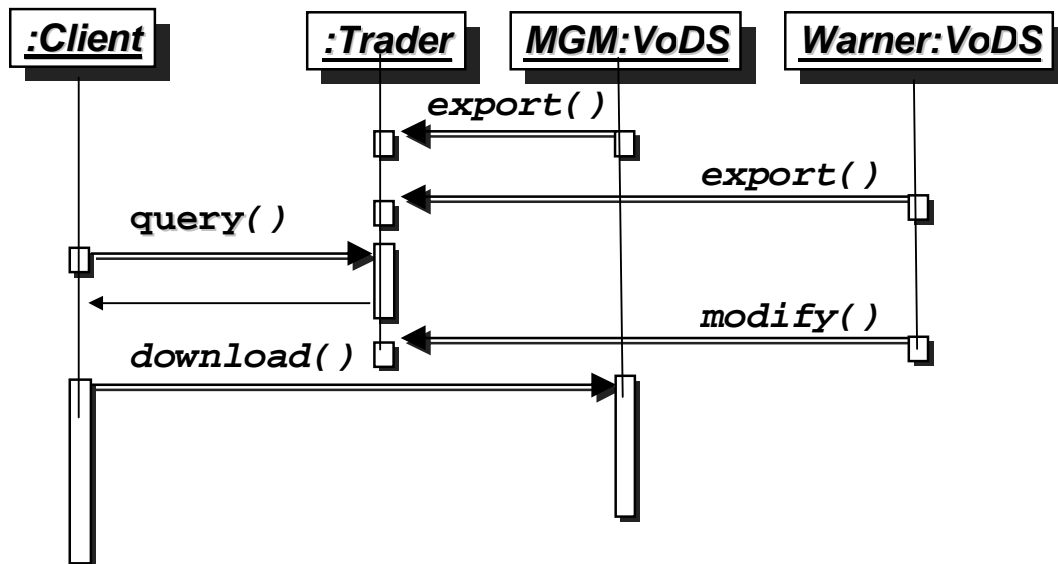
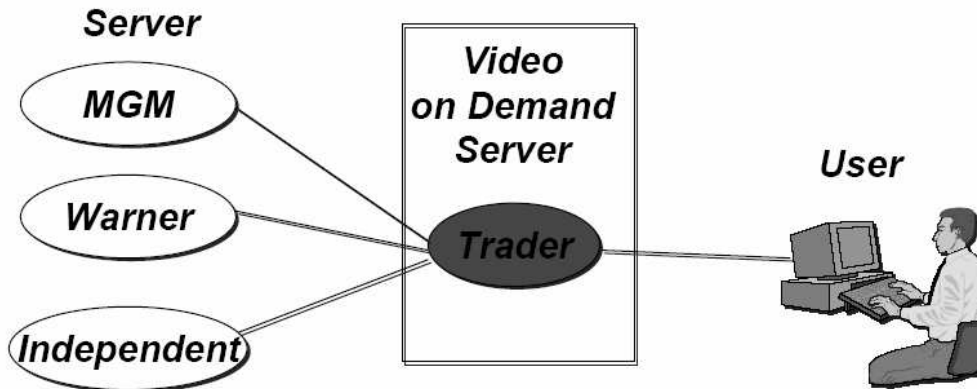
- offers the services it knows about to other traders
- forwards queries it cannot satisfy to other traders

Problems

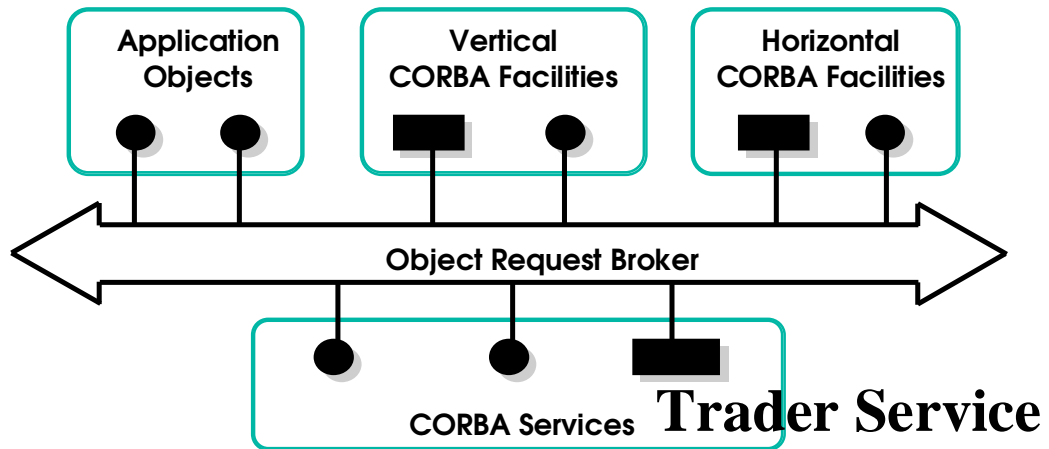
- Non-termination of import
- Duplication of matched offers

# Consider a Video on Demand Service.

"I want to watch *the matrix*"



# Example CORBA



Within CORBA, a naming service is available as well as a trading service

The ORB mediates between clients & servers

# Service Discovery

Service discovery is an integral part of building ad hoc, self-configuring networks.

Mobile computing requires a time-dependent resolution of many network relationships where name to address bindings change frequently. (The whole network can reconfigure - *Spontaneous ad hoc network*)

Services may advertise their existence in a dynamic way.

## ***Discovery protocols***

These provide the mechanisms for dynamically discovering services in a network, providing the necessary information to:

- Search and browse for services
- Choose the right service
- Use the service

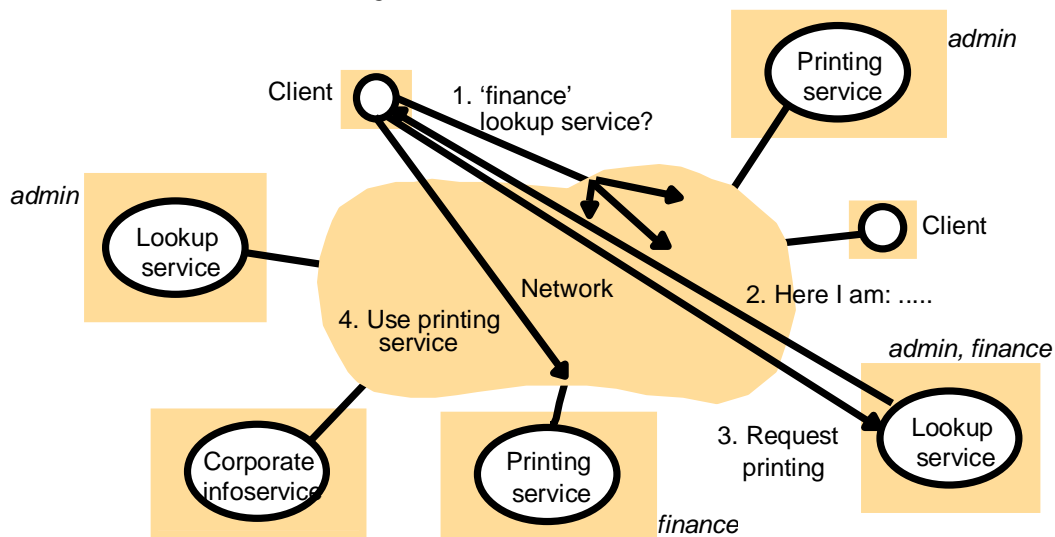
Current Examples:

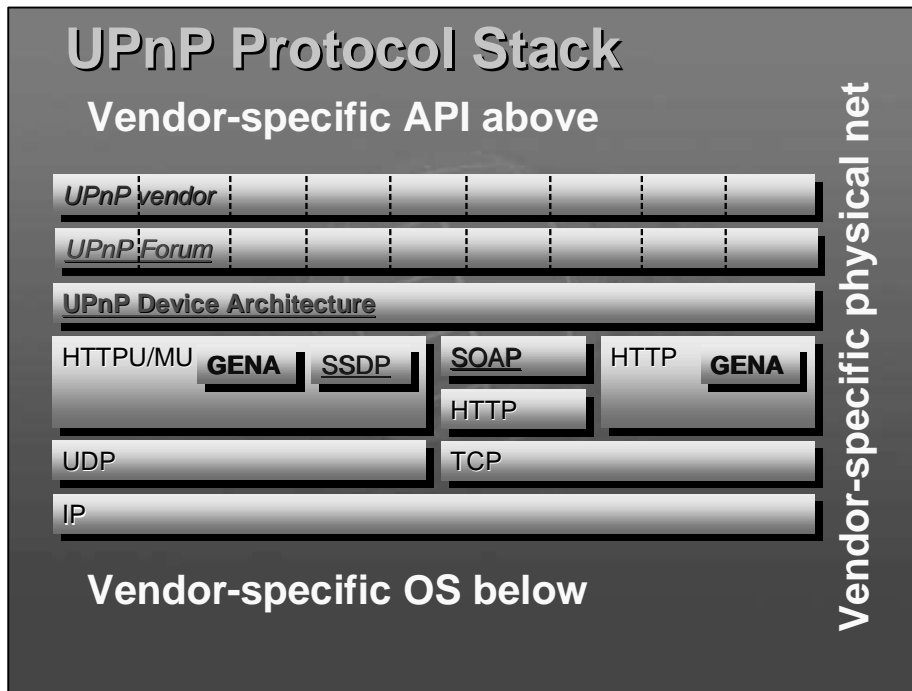
- Service Location Protocol (SLP)
- Universal Plug and Play (UPnP) (more correctly SSDP)
- Jini
- Salutation
- Secure Service Discovery Service (SSDS)
- Bluetooth Service Discovery Protocol

## Broad Comparison:

Feature	SLP	Jini	Salutation	UPnP
Developer	IETF	Sun Microsystems	Salutation Consortium	Microsoft
License	open source	open license, but fee for commercial use	open source	open (only for members)
Version	2	1.0	2.1	0.91
Network transport	TCP/IP	independent	independent	TCP/IP
Programming language	independent	Java	independent	independent
OS and platform	dependent	independent	dependent	dependent
Code mobility	no	yes (Java RMI)	no	no
Srv attributes searchable	yes	yes	yes	no
Central cache repository	yes (optional)	optional using SLP	yes (optional)	no
Operation w/o directory	yes	Lookup Table required	yes	-
Leasing concept	yes	yes	no	yes
Security	IP dependent	Java based	authentication	IP dependent

## Service Discovery in JINI





## 1 Discovery

- ◆ Control point finds interesting device
  - ◆ 0 get address
  - ◆ 1 discover device
- ◆ Advertise / find typed devices (services)
  - ◆ Guarantee of minimal capabilities
  - ◆ Simple
- ◆ Devices
  - ◆ Advertise when added
  - ◆ Refresh advertisements (cf. lease)
  - ◆ Cancel advertisements when removed
  - ◆ Control points search as needed
    - ◆ Devices respond
    - ◆ Control points filter

- Discovery is done primarily on the basis of the type of the device and service
  - Domain-specific device and service types are defined by UPnP Forum working committees.
  - Types guarantee a minimum set of capabilities. Vendors may add other capabilities to differentiate. Description (upcoming) indicates what actually got implemented.
- Discovery also on the basis of unique ID. Allows apps to reconnect to specific devices.
- Discovery uses very simple matching. Helps keep bandwidth requirements and device costs down. Control points must filter discovery.
- Devices advertise and cancel advertisements to join and leave the network. Because devices might leave the network w/o warning (e.g., user pulls power cord to plug in vacuum cleaner), advertisements expire on their own. This is like a lease where the device must renew the advertisement (lease) to keep it active.
- Two-party discovery covers both types of nodes being added to the network
  - Devices advertise when they are added to the network.
  - Control points search when they are added to the network.

Discovery uses a multicast variant of HTTP (HTTPMU) for device advertisements and control point searches. Discovery uses a unicast variant of HTTP (HTTPU) for responses to searches.

# 1 Discovery: SSDP Sidebar

- ◆ What is SSDP?
  - ◆ IETF Draft *Simple Service Discovery Protocol*
- ◆ Key design principles
  - ◆ Administratively-scoped multicast
  - ◆ Unicast responses
  - ◆ UDP
  - ◆ Very simple advertisements
  - ◆ Very simple search

Description starts after getting an IP address and after discovering a device.

Descriptions are in two parts: one for the device, and one per nested service. The location of the device description is in the discovery messages (LOCATION header). The location of the service description is in the device description.

Descriptions explicitly declare capabilities whereas device and service type implicitly declare them.

Simple protocol stack using only HTTP -> TCP -> IP.

Description expressed in XML

# Revision:

## 3 Key Topics

- Security & Cryptography
- Naming, Trading & Discovery
- Time & Global state

## CDK Chapters

- 1, 2, = intro
- 4, 5, 17 = lab exercises
- 7, 9, 10 = key chapters

## Assumptions:

- You've read CDK
- Attended the lectures
- You've done the lab exercises & assignments!
- You won't need to write any code