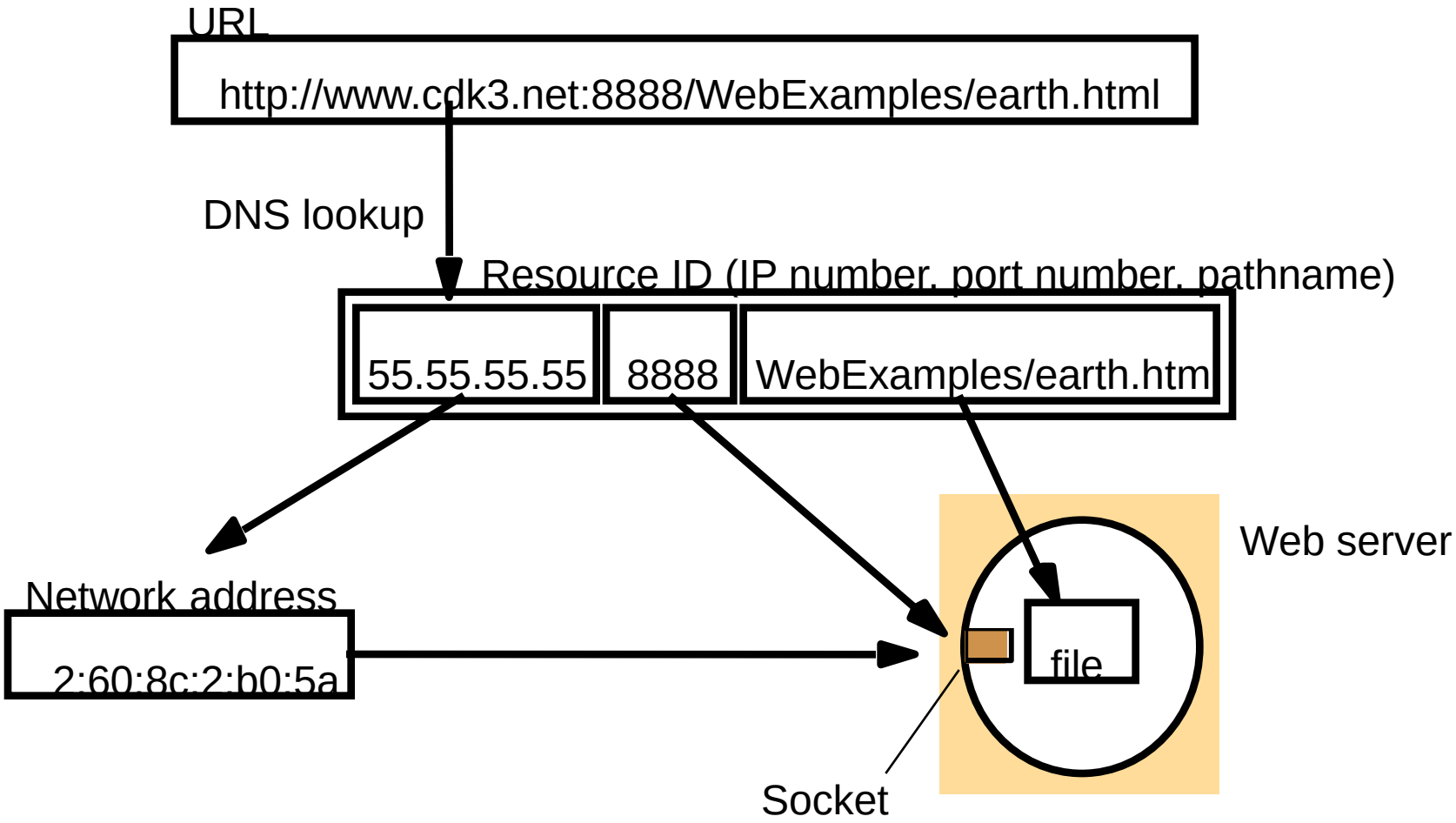
Module 4 – Distributed Naming

Names, Addresses, etc.

- Names
  - Identification of objects
    - → Resource sharing: Internet domain names
    - → Communication: domain name part of email address
  - How much information about an object is in a name?
    - → Pure names: uninterpreted bit patterns
    - → Non-pure names: contain information about the object, e. g., its location
- Name Services
  - Entries of the form <name, attributes>, where attributes are typically network addresses
  - Type of lookup queries
    - → name → attribute values
    - → also called name resolution
- Directory Services
  - <name, attributes> entries
  - Types of lookup queries
    - → name → attribute values
    - → attribute values → names

Composed Naming Domains



From Coulouris, Dollimore and Kindberg, Distributed Systems: Concepts and Design, 3rd ed.

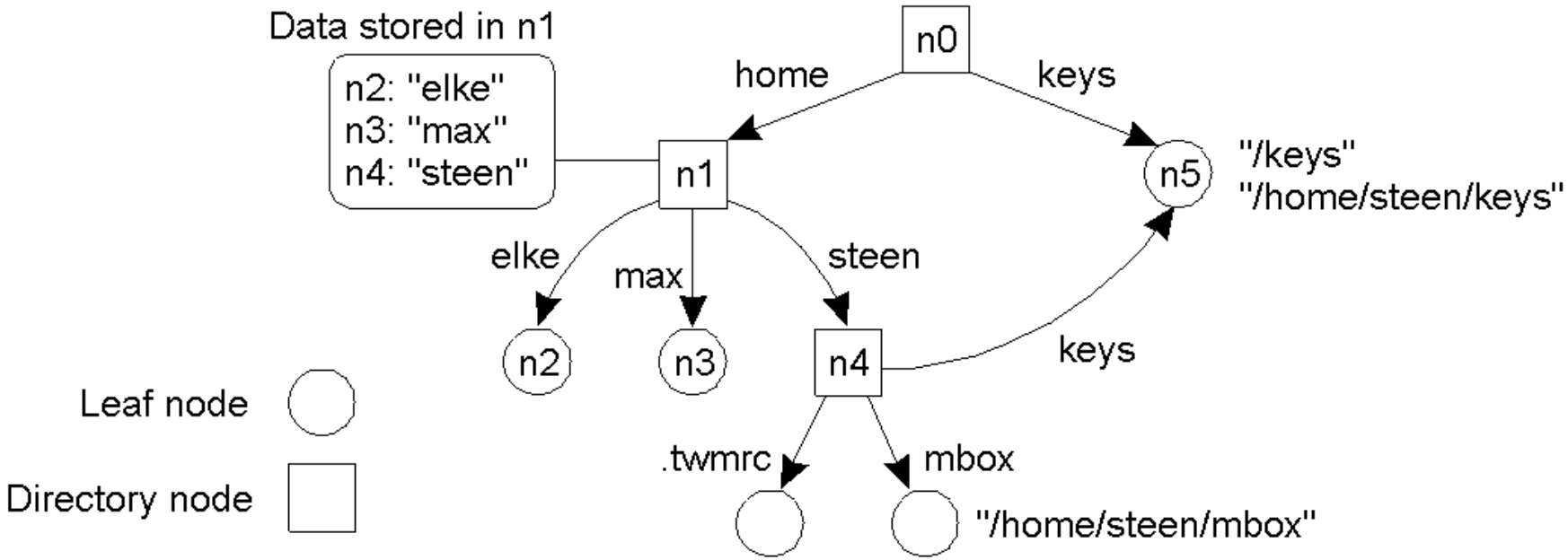
Requirements on Name Services

- Usage of a convention for unique global naming
  - Enables sharing
  - It is often not easily predictable which services will eventually share resources
- Scalability
  - Naming directories tend to grow very fast, in particular in Internet
- Consistency
  - Short and mid-term inconsistencies tolerable
  - In the long term, system should converge towards a consistent state
- Performance and availability
  - Speed and availability of lookup operations
  - Name services are at the heart of many distributed applications
- Adaptability to change
  - Organizations frequently change structure during lifetime
- Fault isolation
  - System should tolerate failure of some of its servers

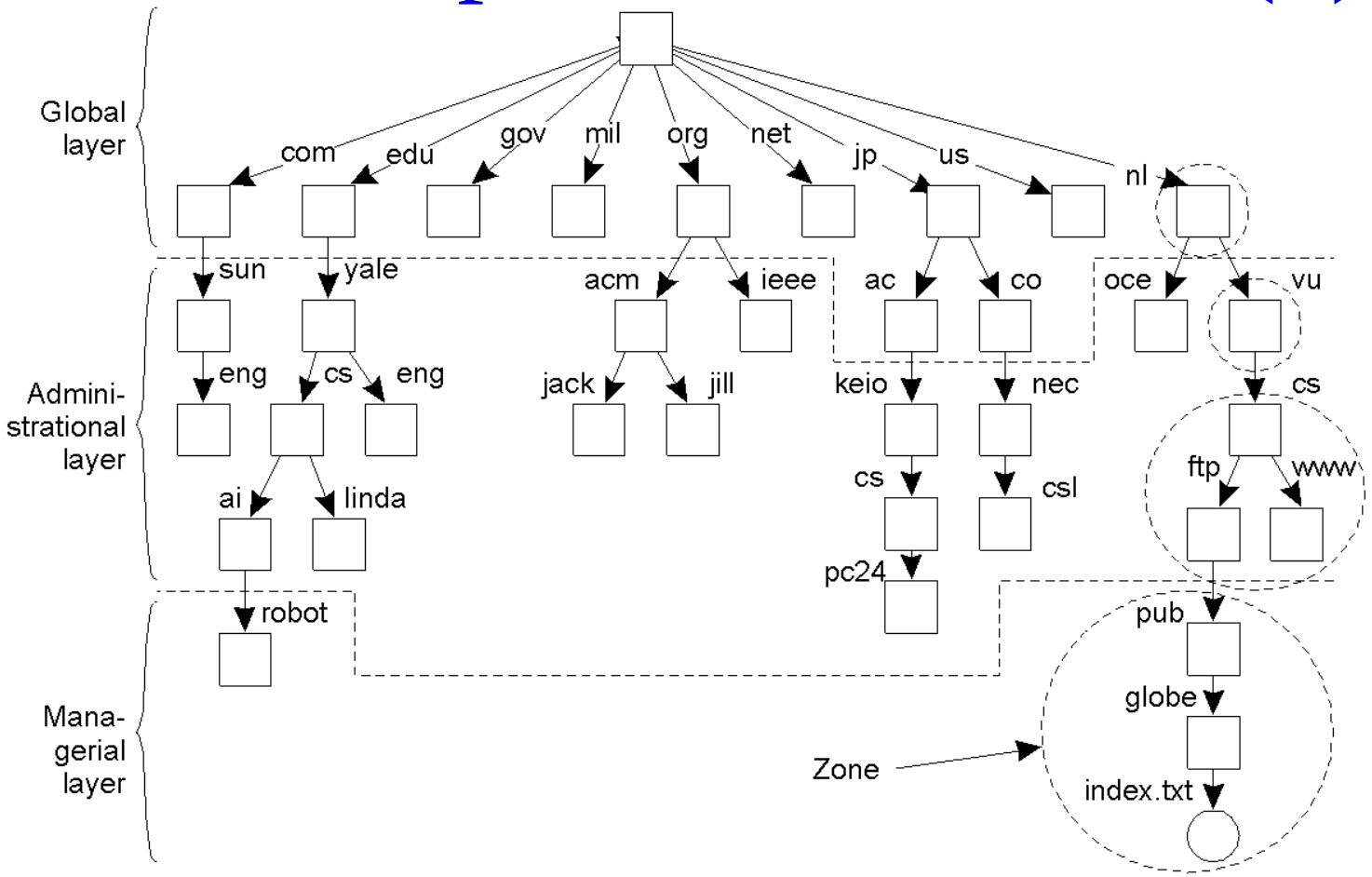
Name Spaces

- Set of all valid names to be used in a certain context, e. g., all valid URLs in WWW
- Can be described using a generative grammar (e. g., BNF for URLs)
- Internal structure
  - Flat set of numeric or symbolic identifiers
  - Hierarchy representing position (e. g., UNIX file system)
  - Hierarchy representing organizational structure (e. g., Internet domains)
- Potentially infinite
  - Holds only for hierarchic name spaces
  - Flat name spaces finite size induced by max. name length
- Aliases
  - In general, allows a convenient name to be substituted for a more complicated one
- Naming domain
  - Name space for which there exist a single administrative authority for assigning names within it

Naming Graph with Single Root



Name Space Distribution (1)



An example partitioning of the DNS name space, including Internet-accessible files, into three layers.

Name Space Distribution (2)

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

A comparison between name servers for implementing nodes from a large-scale name space partitioned into a global layer, as an administrational layer, and a managerial layer.

Issues in Name & Directory Services

- Partitioning
  - No one name server can hold all name and attribute entries for entire network, in particular in Internet
  - Name server data partitioned according to domain
- Replication
  - A domain has usually more than one name server
  - Availability and performance are enhanced
- Caching
  - Servers may cache name resolutions performed on other servers
    - → Avoids repeatedly contacting the same name server to look up identical names
  - Client lookup software may equally cache results of previous requests

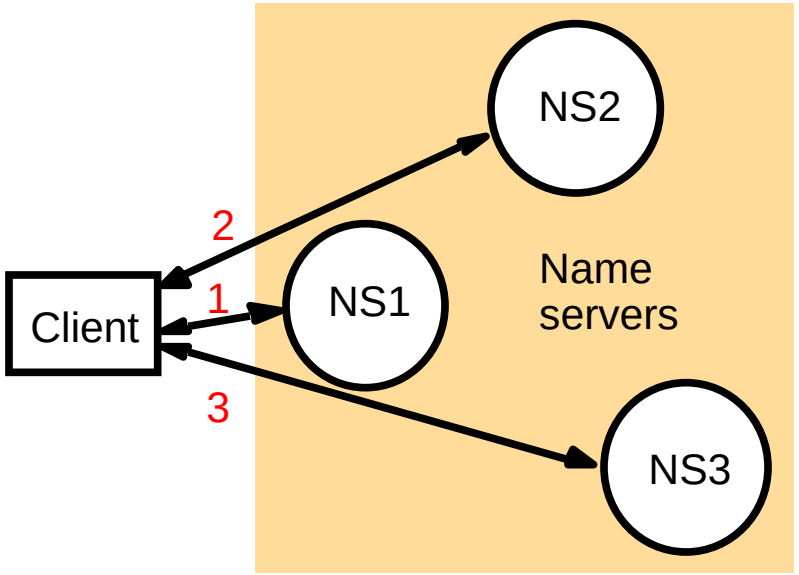
Name Resolution

- Translation of a name into the related primitive attribute
- Often, an iterative process
  - Name service returns attributes if the resolution can be performed in it's naming context
  - Name service refers query to another context if name can't be resolved in own context
- Deal with cyclic alias references, if present
  - Abort resolution after a predefined number of attempts, if no result obtained

Navigation

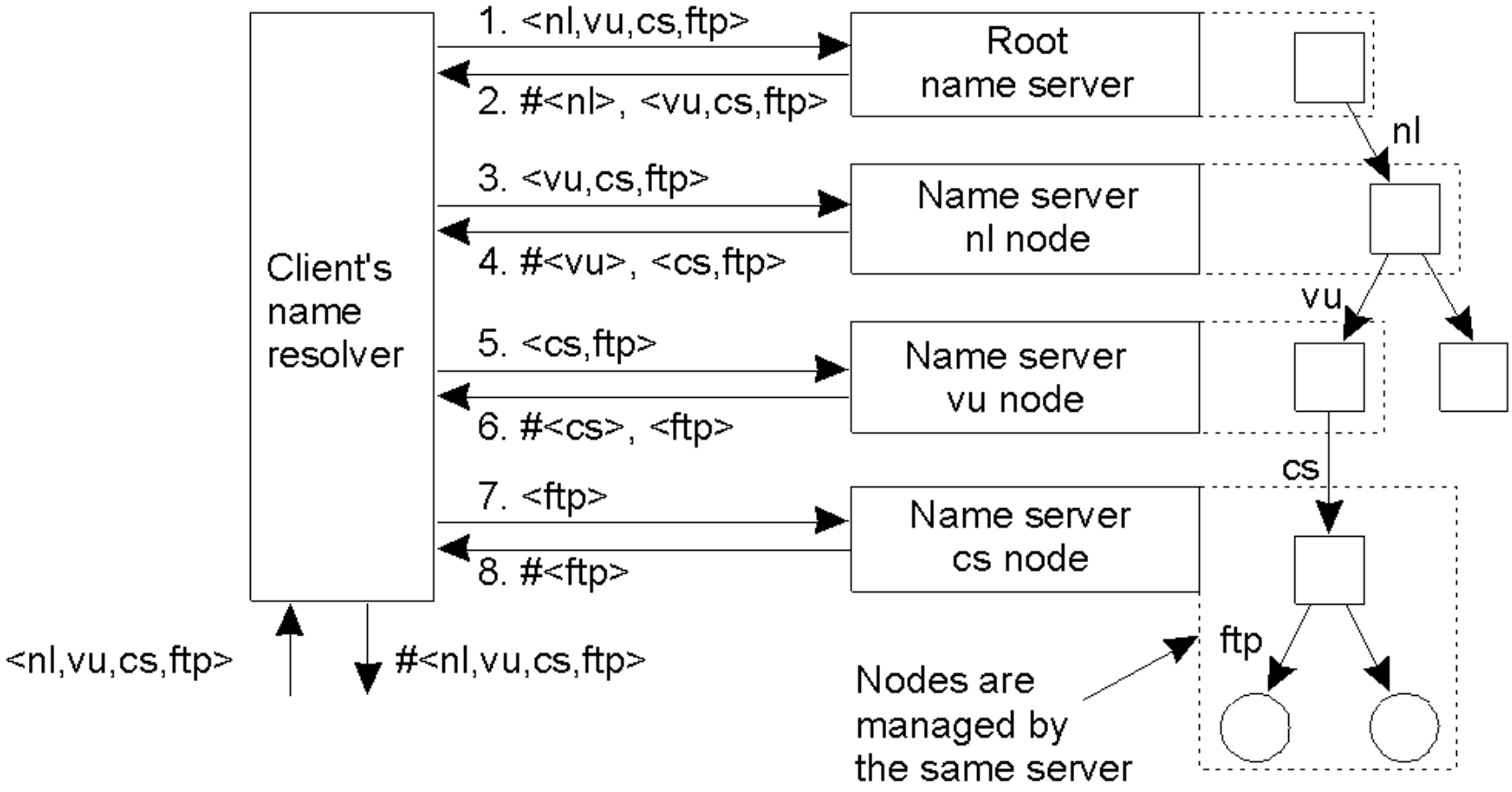
Accessing naming data from more than one name server in order to resolve a name

- Iterative Navigation
  - Used in DNS
  - Client contacts one NS
  - Name Server either resolves name, or suggests other name server to contact
  - Resolution continues until name resolved or name found to be unbound



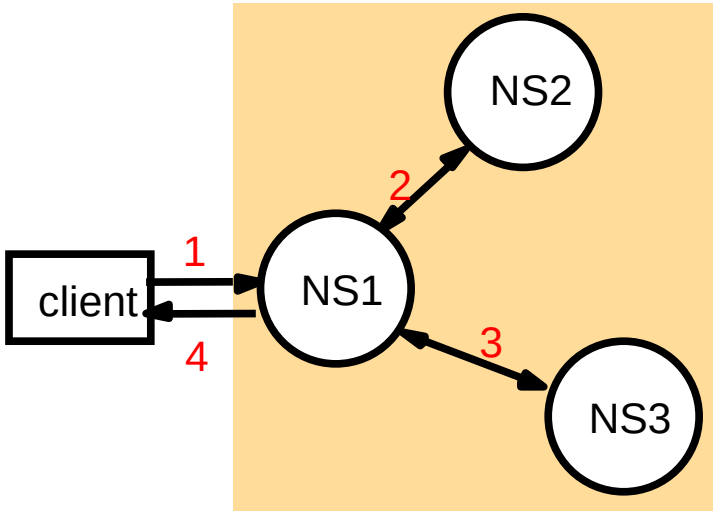
From Coulouris, Dollimore and Kindberg, Distributed Systems: Concepts and Design, 3rd ed.

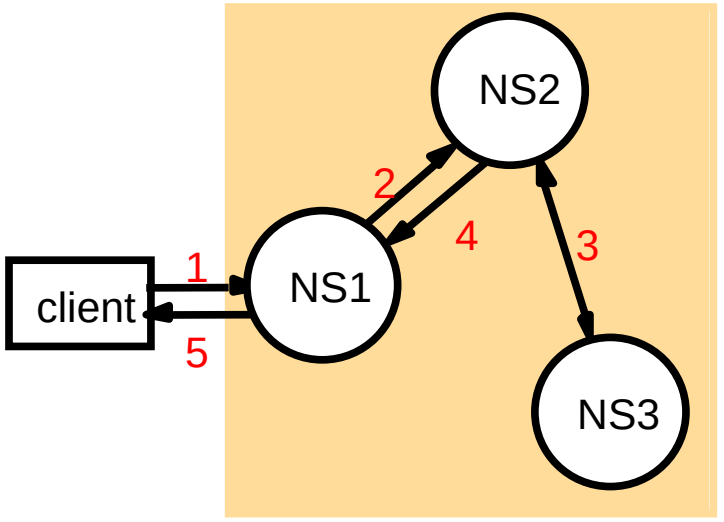
Iterative Navigation Example



Navigation (2)

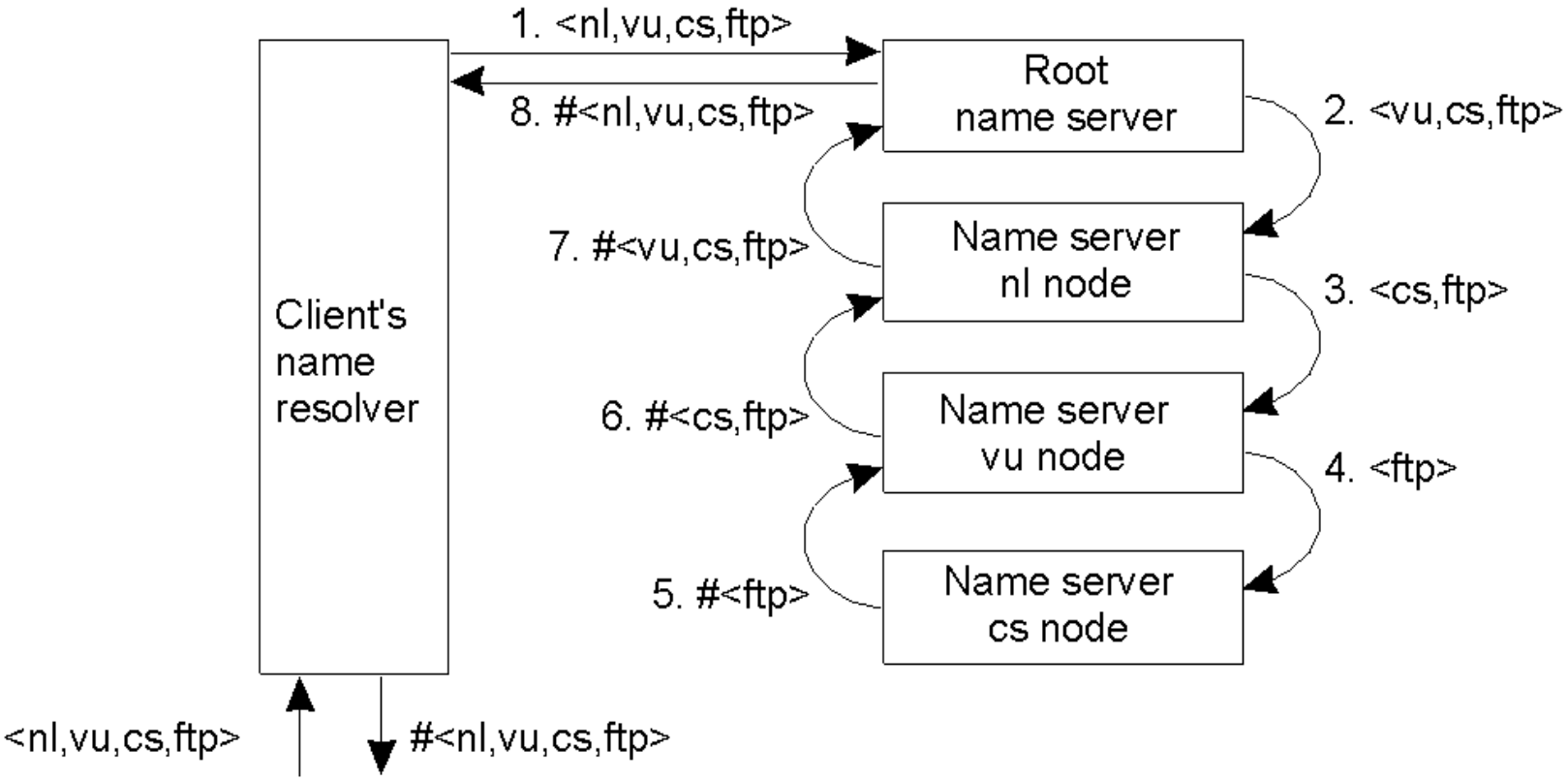
- Non-recursive, server-controlled
  - Server contacts peers if it cannot resolve name itself
    - → by multicast or iteratively by direct contact
- Recursive, server-controlled
  - If name cannot be resolved, server contacts superior server responsible for a larger prefix of the name space
    - → recursively applied until name resolved
    - → can be used when clients and low-level servers are not entitled to directly contact high-level servers





From Coulouris, Dollimore and Kindberg, Distributed Systems: Concepts and Design, 3rd ed.

Recursive Navigation Example

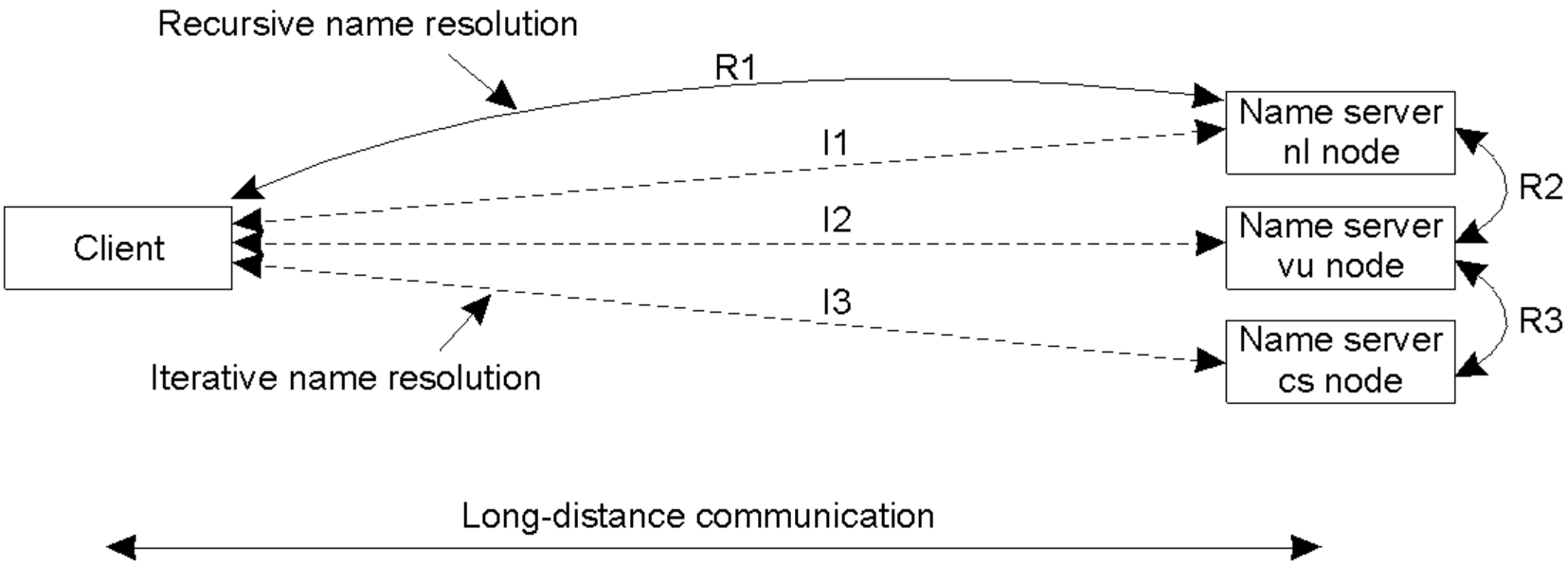


Recursive Name Resolution

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>

- Recursive name resolution of *<nl, vu, cs, ftp>*. Name servers cache intermediate results for subsequent lookups.

Communication Cost Comparison of Iterative vs Recursive



Name & Directory Services

- Domain Name System (DNS)
  - Name service used across the Internet
- Global Name Service (GNS)
  - Developed at DEC
- X. 500
  - ITU - standardized directory service
- LDAP
  - Directory service
  - Lightweight implementation of X.500
  - Often used in intranets
- Jini
  - Discovery service used in spontaneous networking
  - Contains directory service component

Domain Name System (DNS)

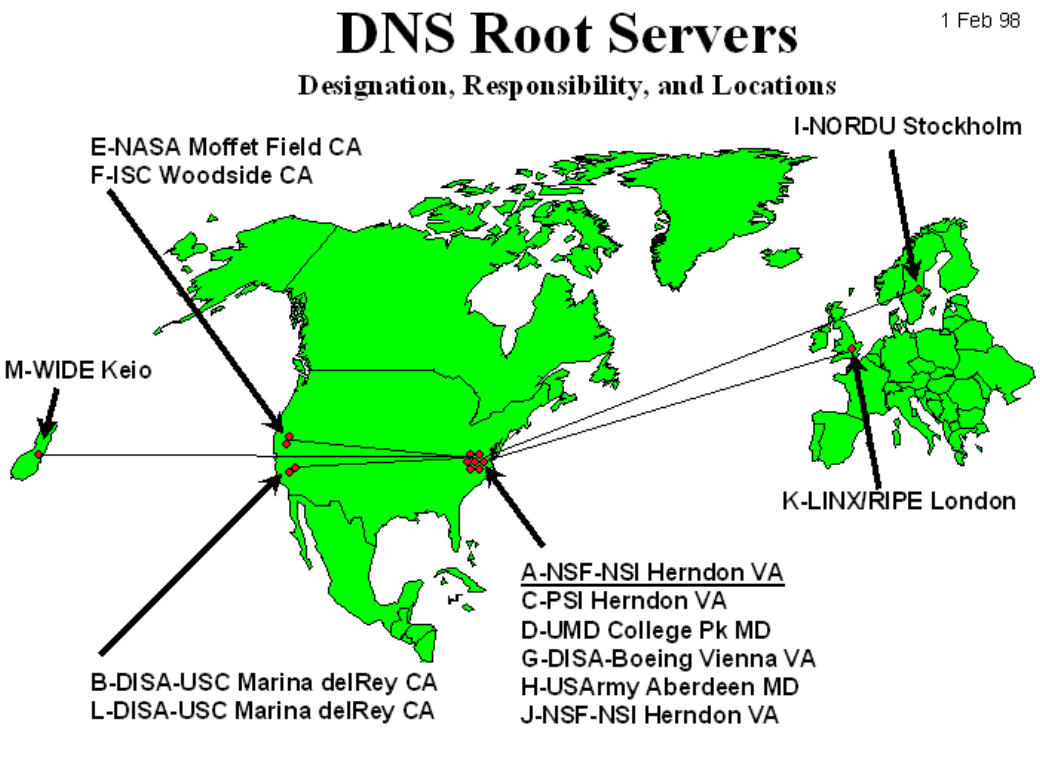
- Performs name-to-IP mapping
- *Distributed database*
  - implemented in hierarchy of many name servers
- *Application-layer protocol*
  - host, routers, name servers to communicate to resolve names (address/name translation)
- Why not centralize DNS?
  - single point of failure
  - traffic volume
  - distant centralized database
  - maintenance
  - ☒ doesn't scale!

DNS Types

- No server has all name-to-IP address mappings
- Local name servers
  - Each ISP, company has local (default) name server
  - Host DNS query first goes to local name server
- Authoritative name server
  - For a host: stores that host's IP address, name
  - Can perform name-to-address translation for that host's name
- Root name server
  - Establishes a rooted tree of DNSs

DNS: Root Name Servers

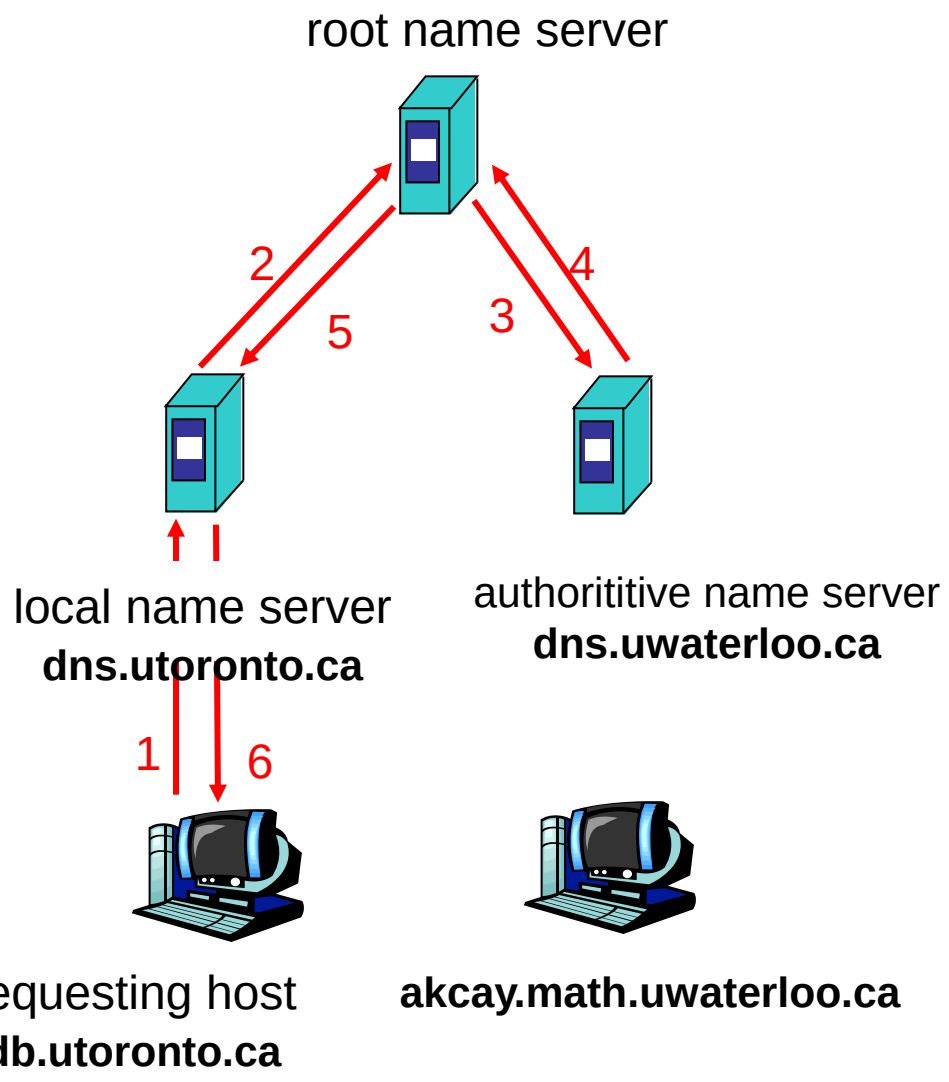
- Contacted by local name server that can not resolve name
- Root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server
- Approximately dozen root name servers worldwide



Simple DNS Example

Host **db.utoronto.ca** wants IP address of **akcay.math.uwaterloo.ca**

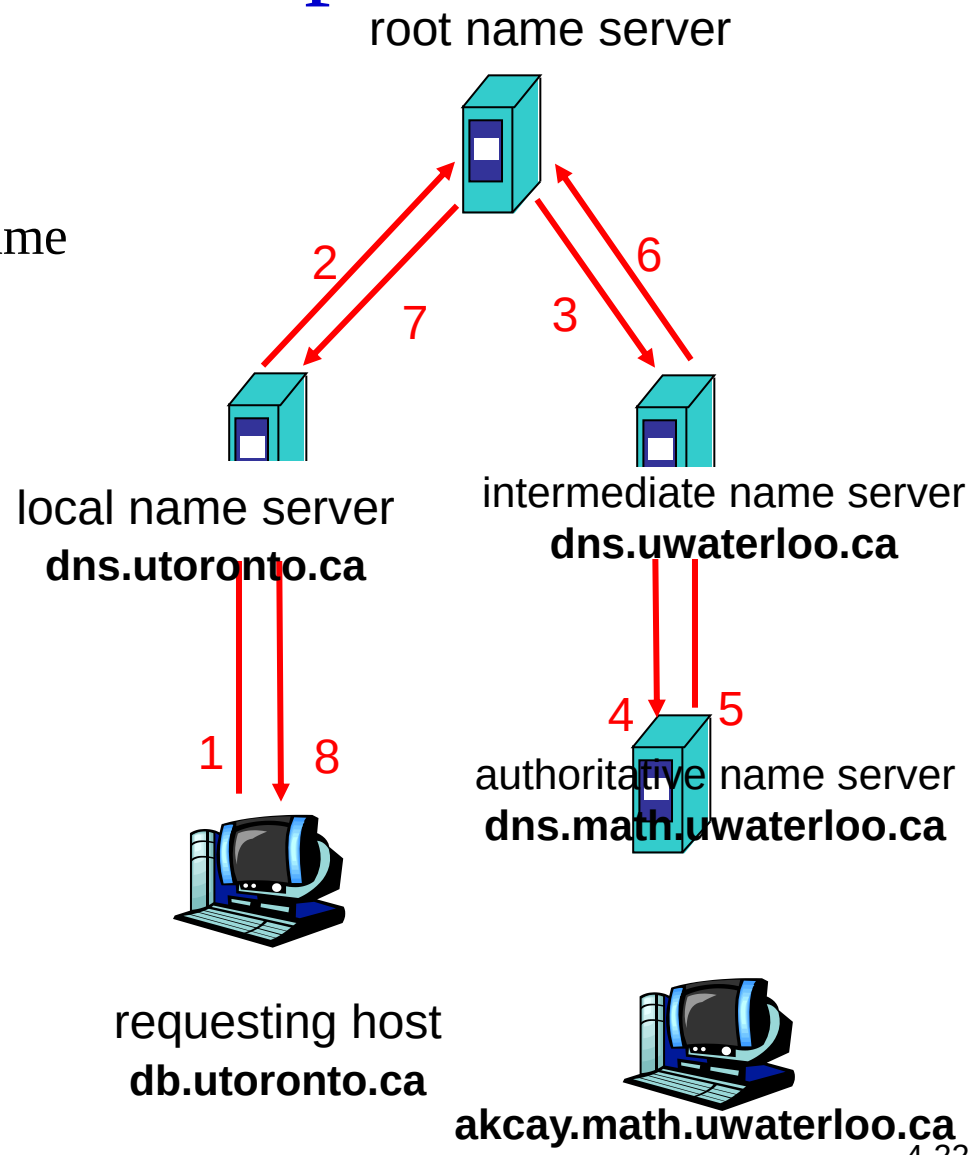
1. Contacts its local DNS server, **dns.utoronto.ca**
2. **dns.utoronto.ca** contacts root name server, if necessary
3. root name server contacts authoritative name server, **dns.uwaterloo.ca,** if necessary



DNS Example

Root name server:

- may not know authoritative name server
- may know *intermediate name server:* who to contact to find authoritative name server



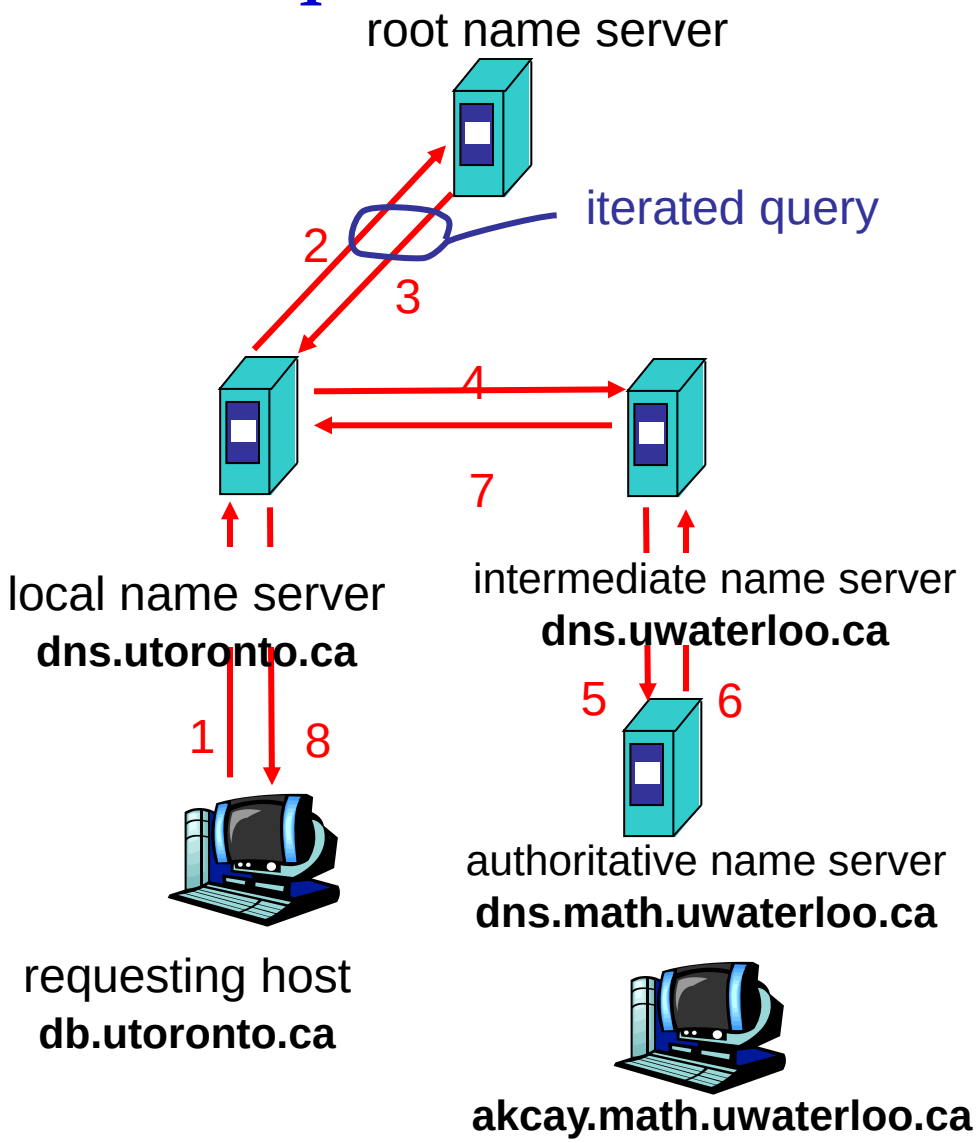
DNS: iterated queries

Recursive query:

- Puts burden of name resolution on contacted name server
- Heavy load?

Iterated query:

- Contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



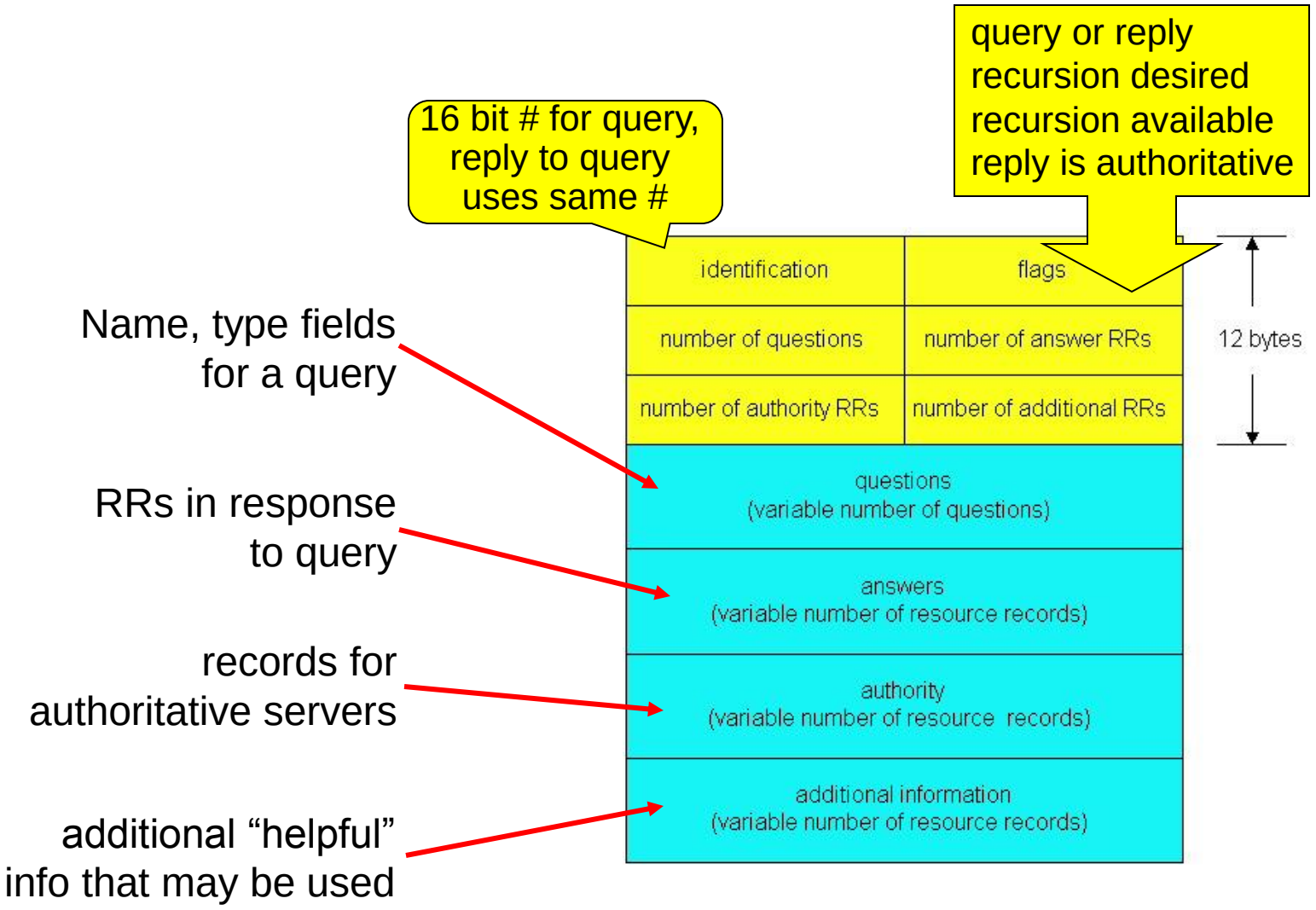
DNS Records

- DNS holds resource records (RR)
  - RR format: **(name, value, type,ttl)**
- Example records:
  - Type=A
    - → **name** is hostname
    - → **value** is IP address
  - Type=NS
    - → **name** is domain (e.g. foo.com)
    - → **value** is hostname of authoritative name server for this domain
  - Type=CNAME
    - → **name** is an alias name for some "canonical" (the real) name
    - → **value** is canonical name
  - Type=MX
    - → **value** is hostname of mail server associated with **name**

DNS Record Types

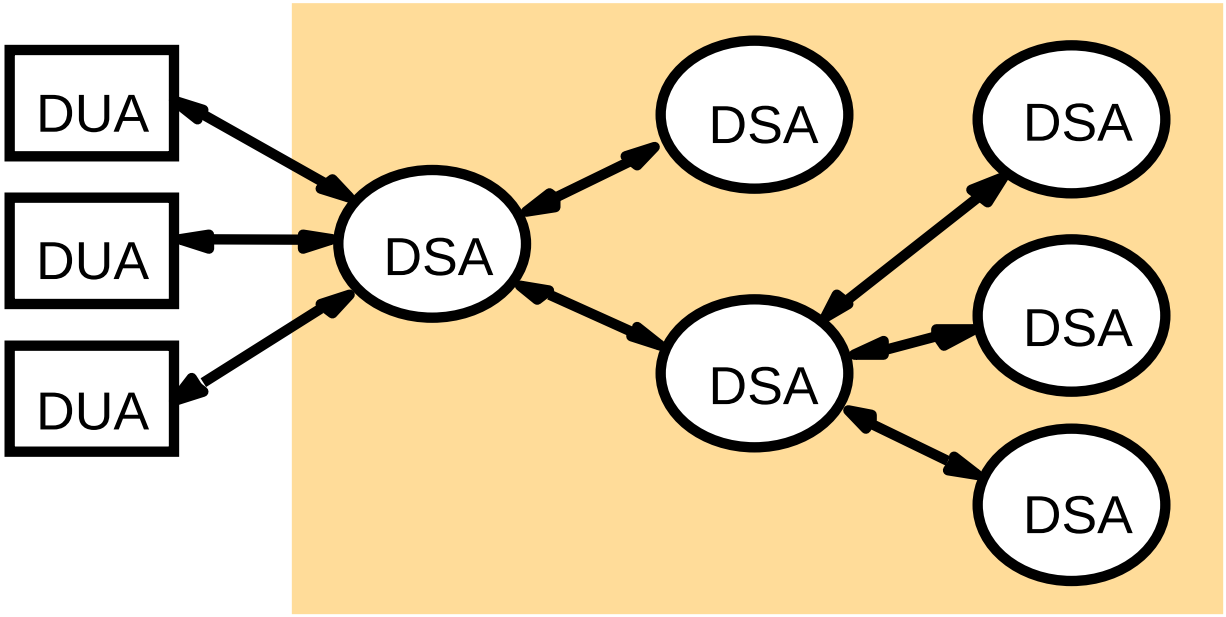
Record type	*Meaning*	*Main contents*
A	A computer address	IP number
NS	An authoritative name server	Domain name for server
CNAME	The canonical name for an alias	Domain name for alias
SOA	Marks the start of data for a zone	Parameters governing the zone
WKS	A well-known service description	List of service 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 Protocol & Messages



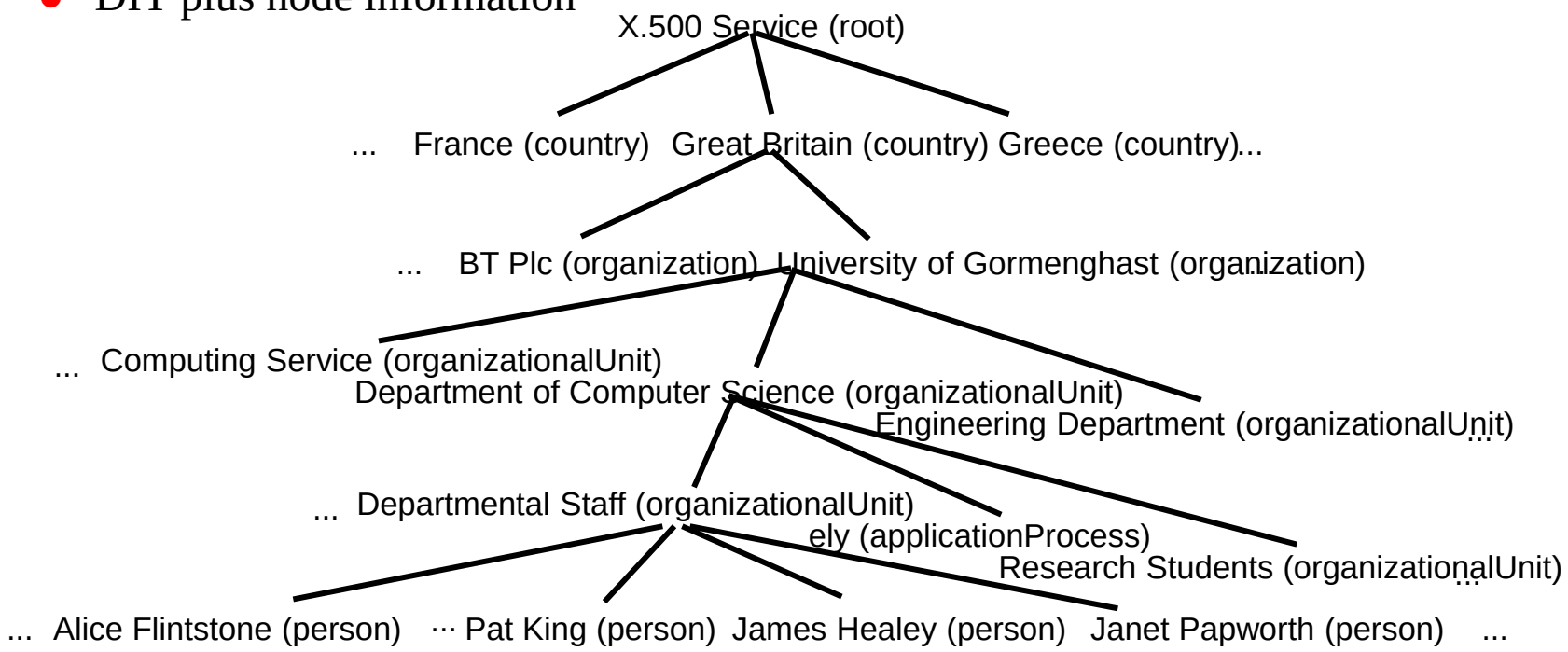
X.500 Directory Service

- Geared towards satisfying descriptive queries
  - Provide attributes of other users and system resources
- Architecture
  - Directory user agents (DUA)
  - Directory service agents (DSA)



X.500 Name Tree

- Directory Information Tree (DIT)
  - Every node of the tree stores extensive information
- Directory Information Base (DIB)
  - DIT plus node information



From Coulouris, Dollimore and Kindberg, Distributed Systems: Concepts and Design, 3rd ed.

Example X.500 DIB Entry

info
Alice Flintstone, Departmental Staff, Department of Computer Science,
University of Gormenghast, GB

commonName
Alice.L.Flintstone
Alice.Flintstone
Alice Flintstone
A. Flintstone

surname
Flintstone

telephoneNumber
+44 986 33 4604

uid
alf

mail
alf@dcs.gormenghast.ac.uk
Alice.Flintstone@dcs.gormenghast.ac.uk

roomNumber
Z42

userClass
Research Fellow

X.500 Directory Accesses

- Read
  - Absolute or relative name provided
  - DSA navigates tree and returns requested attributes
- Search
  - Input
    - → base name: starting point for search in tree
    - → filter expression: boolean condition on directory attributes
  - Returned
    - → list of DIT node names for which filter evaluates to true
- Lightweight Directory Access Protocol (LDAP)
  - Lightweight version of X. 500; access of DSA through TCP/ IP; simpler API; textual encoding in place of ASN. 1 encoding
- Practical Usage of X. 500/LDAP
  - LDAP currently widely used for intranets
  - adoption in Internet

Discovery Services

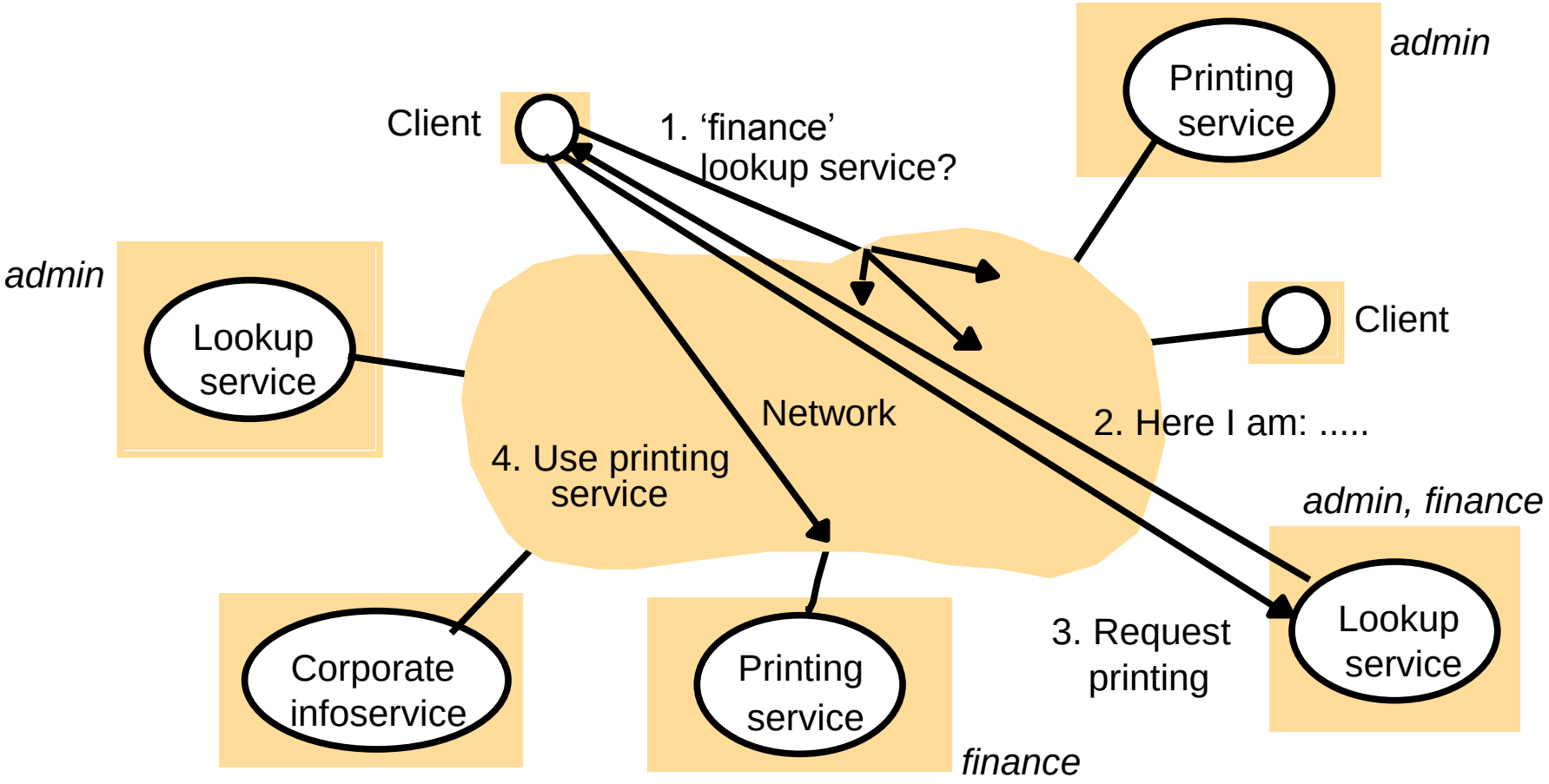
- Directory services that allow clients to query available services in a spontaneous networking environment
  - e. g., which are the available color printers
  - services enter their data using registration interfaces
  - structure usually rather flat, since scope limited to (wireless) LAN
- JINI (http://www.sun.com/jini/)
  - JAVA-based discovery service
    - → clients and servers run JVMs
      - communication via Java RMI
      - dynamic loading of code

JINI Discovery Related Services

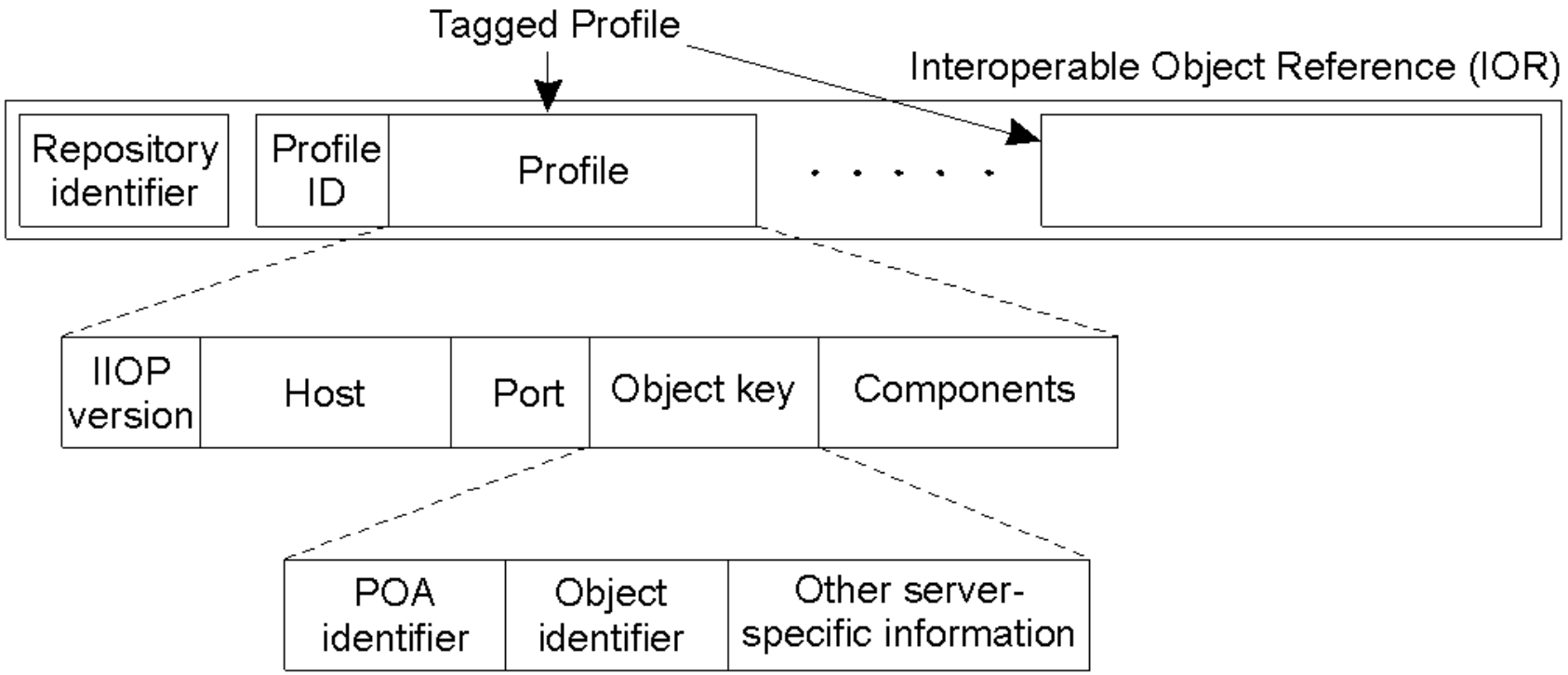
- Lookup service
  - Holds information regarding available services
- Query of lookup service by Jini client
  - Match request
  - Download object providing service from lookup service
- Registration of a Jini client or Jini service with lookup service
  - Send message to well-known IP multicast address, identical to all Jini instances
  - Limit multicast to LAN using time-to-live attribute
  - Use of leases that need to be renewed periodically for registering Jini services

JINI Discovery Scenario

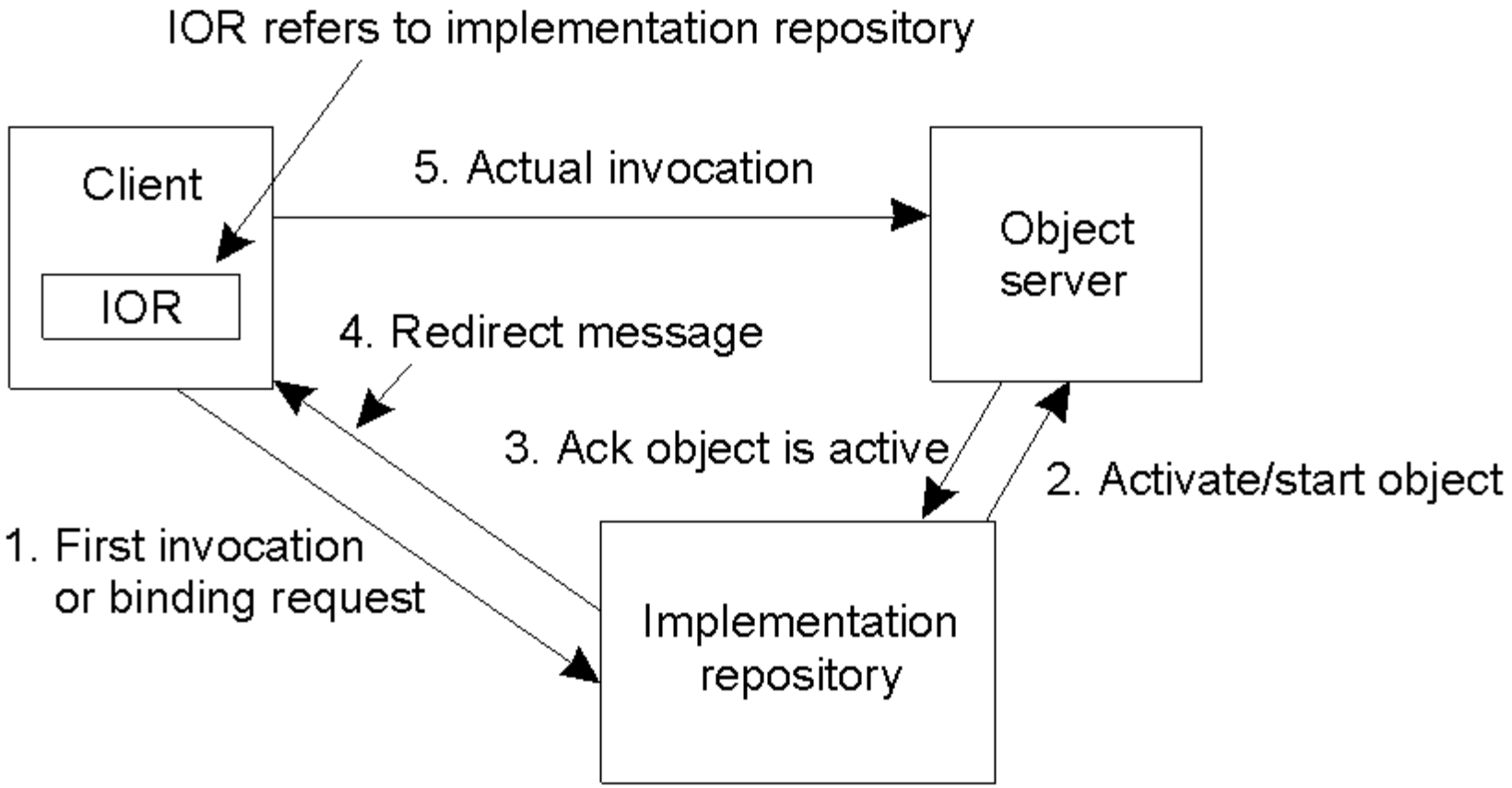
- new client wishes to print on a printer belonging to the finance group



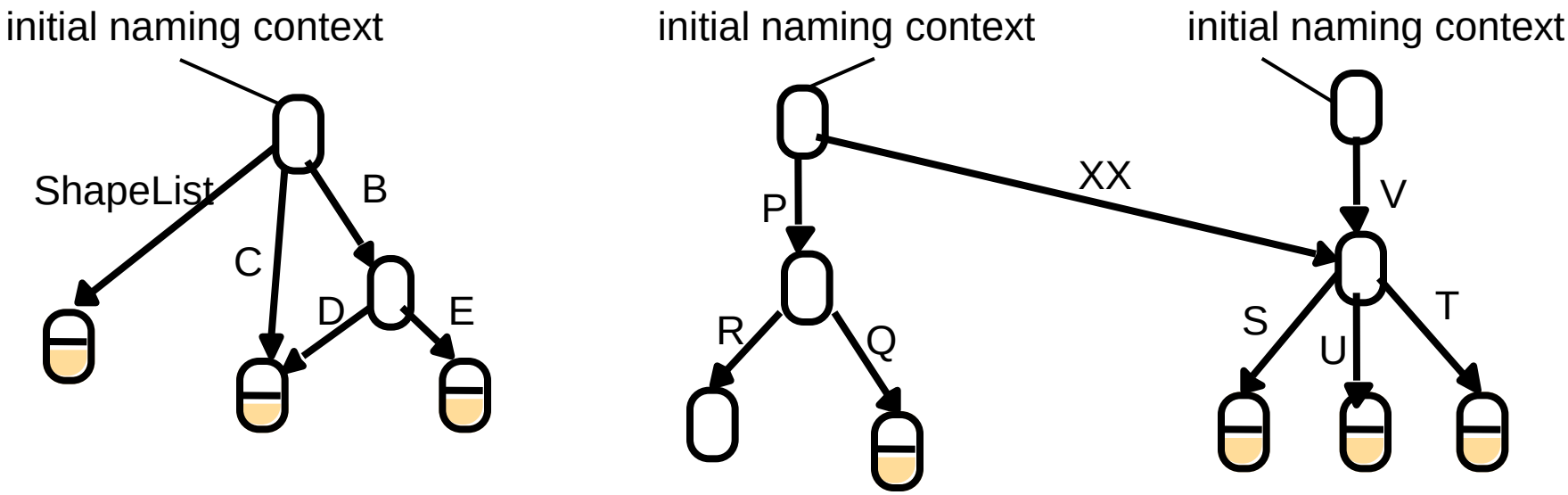
CORBA Object References (1)



CORBA Object References (2)



Naming graph in CORBA Naming Service



CORBA NamingContext Interface (Partial)

struct NameComponent { string id; string kind; };

typedef sequence <NameComponent> Name;

interface NamingContext {

void bind (in Name n, in Object obj);

binds the given name and remote object reference in my context.

void unbind (in Name n);

removes an existing binding with the given name.

void bind_new_context(in Name n);

creates a new naming context and binds it to a given name in my context.

Object resolve (in Name n);

looks up the name in my context and returns its remote object reference.

void list (in unsigned long how_many, out BindingList bl, out BindingIterator bi);

returns the names in the bindings in my context.

};

From Coulouris, Dollimore and Kindberg, Distributed Systems: Concepts and Design, 3rd ed.