Teaching material based on Distributed Systems: Concepts and Design, Edition 3, Addison-Wesley 2001.




Viewing: These slides must be viewed in slide show mode.

# Distributed Systems Course
# Name Services

## CDK3 - Chapter 9:

**9.1 Introduction**
**9.2 Name services and the DNS**
**9.3 Discovery services**
**9.6 Summary**

# Learning objectives

- To understand the need for naming systems in distributed systems
- To be familiar with the design requirements for distributed name services
- To understand the operation of the Internet naming service - DNS
- To be familiar with the role of discovery services in mobile and ubiquitous computer systems



*

# The role of names and name services

- Resources are accessed using *identifier* or *reference*
  - An identifier can be stored in variables and retrieved from tables quickly
  - Identifier includes or can be transformed to an address for an object
    - *E.g. NFS file handle, Corba remote object reference*
  - A *name* is human-readable value (usually a string) that can be *resolved* to an identifier or address
    - *Internet domain name, file pathname, process number*
    - *E.g ./etc/passwd, http://www.cdk3.net/*
- For many purposes, names are preferable to identifiers
  - because the binding of the named resource to a physical location is deferred and can be changed
  - because they are more meaningful to users
- Resource names are *resolved* by name services
  - to give identifiers and other useful attributes



*

# Requirements for name spaces

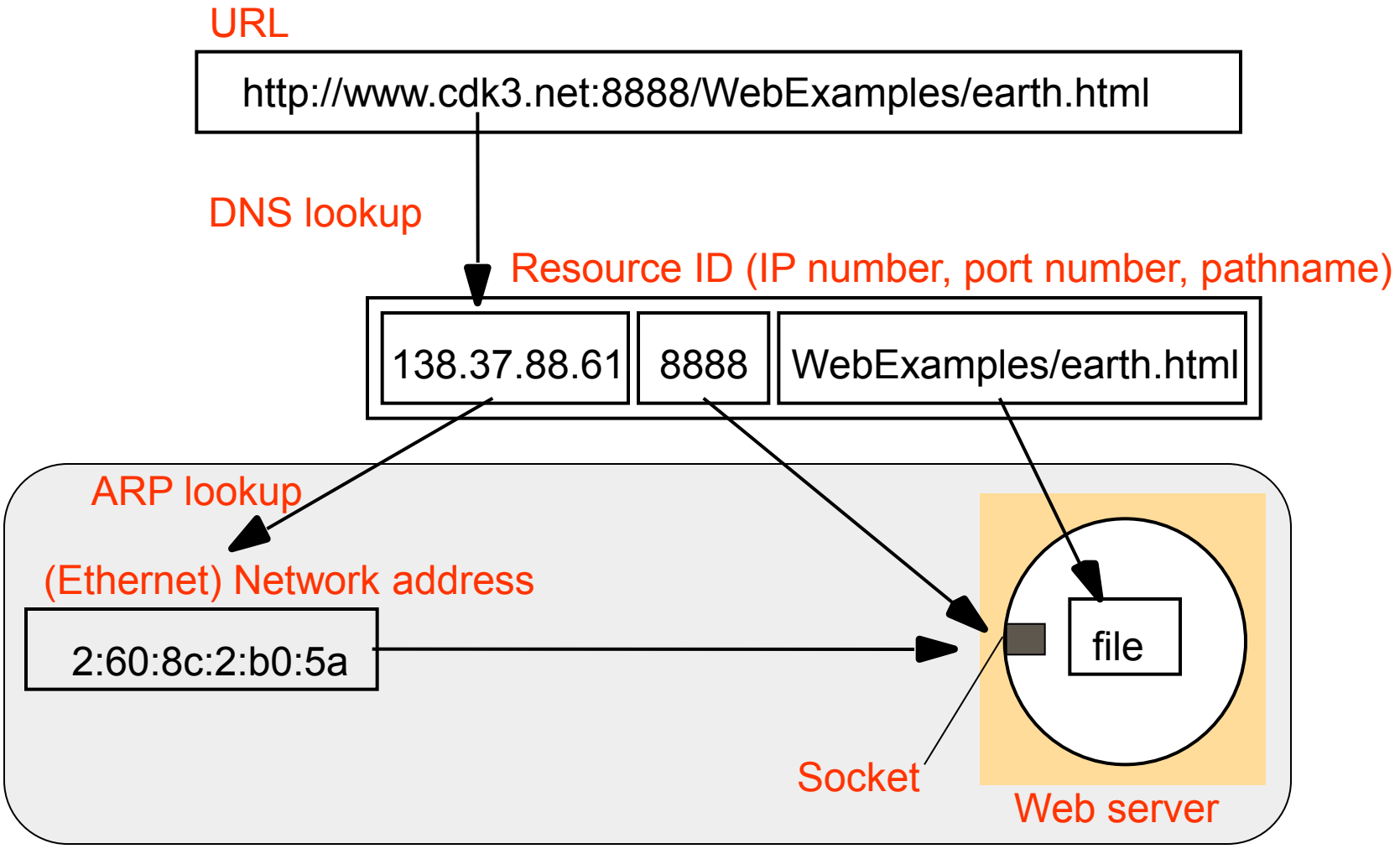
- Allow simple but meaningful names to be used
- Potentially infinite number of names
- Structured
  - to allow similar subnames without clashes
  - to group related names
- Allow re-structuring of name trees
  - for some types of change, old programs should continue to work
- Management of trust



*

# Composed naming domains used to access a resource from a URL

Figure 9.1






*

# Names and resources

Currently, different name systems are used for each type of resource:

| *resource* | *name* | *identifies* |
|---|---|---|

More on URNs

*format: urn:<nameSpace>:<name-within-namespace>*

*examples:*

*a) urn:ISBN:021-61918-0*

*b) urn:dcs.qmul.ac.uk:TR2000-56*

*resolution:*

*a) send a request to nearest ISBN-lookup service - it would return whatever attributes of a book are required by the requester*

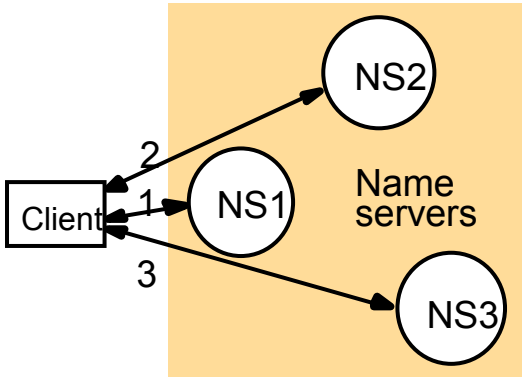
*b) send a request to the urn lookup service at dcs.qmul.ac.uk - it would return a url for the relevant document*



*

# Iterative navigation

Figure 9.2




**Reason for NFS iterative name resolution**

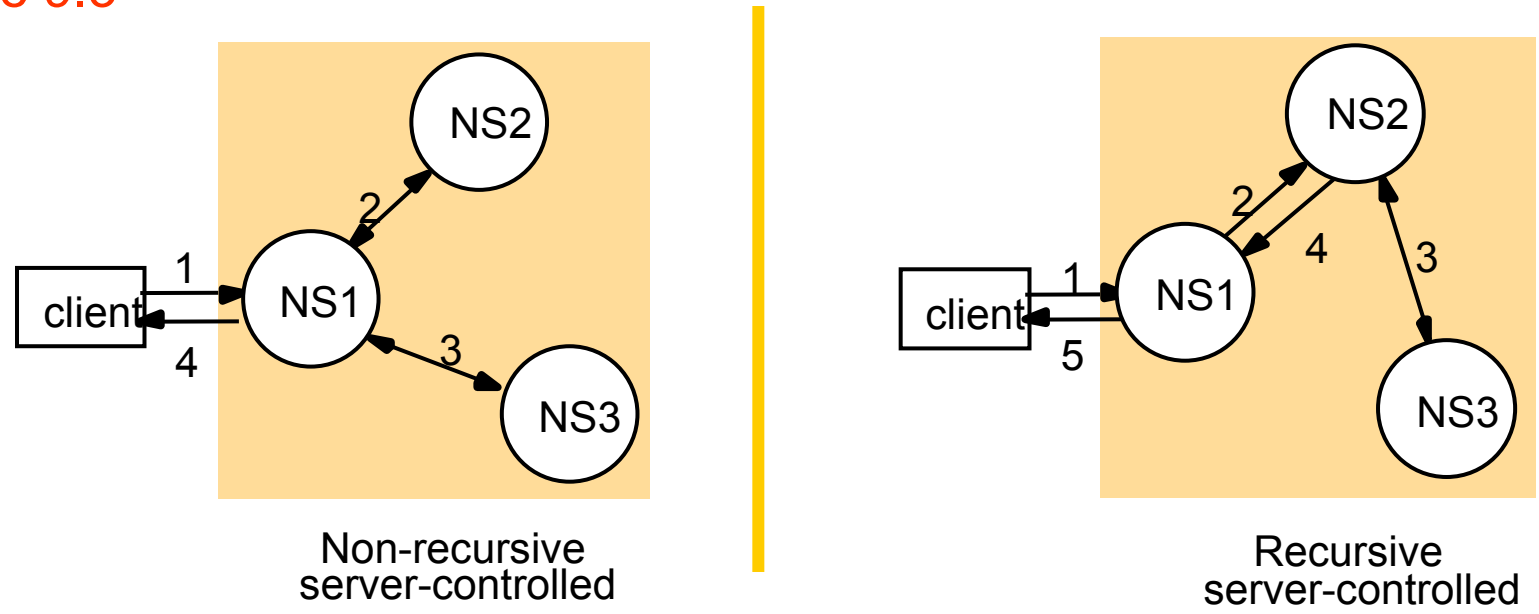
This is because the file service may encounter a symbolic link (i.e. an *alias*) when resolving a name. A symbolic link must be interpreted in the client's file system name space because it may point to a file in a directory stored at another server. The client computer must determine which server this is, because only the client knows its mount points. (p.362.)



*

# Non-recursive and recursive server-controlled navigation

Figure 9.3




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

DNS offers recursive navigation as an option, but iterative is the standard technique. Recursive navigation must be used in domains that limit client access to their DNS information for security reasons.



*

# DNS - The Internet Domain Name System

- A distributed naming database
- Name structure reflects administrative structure of the Internet
- Rapidly resolves domain names to IP addresses
  - exploits caching heavily
  - typical query time ~100 milliseconds

**Basic DNS algorithm for name resolution (domain name -> IP number)**

- Look for the name in the local cache
- Try a superior DNS server, which responds with:
  - another recommended DNS server
  - the IP address (which may not be entirely up to date)

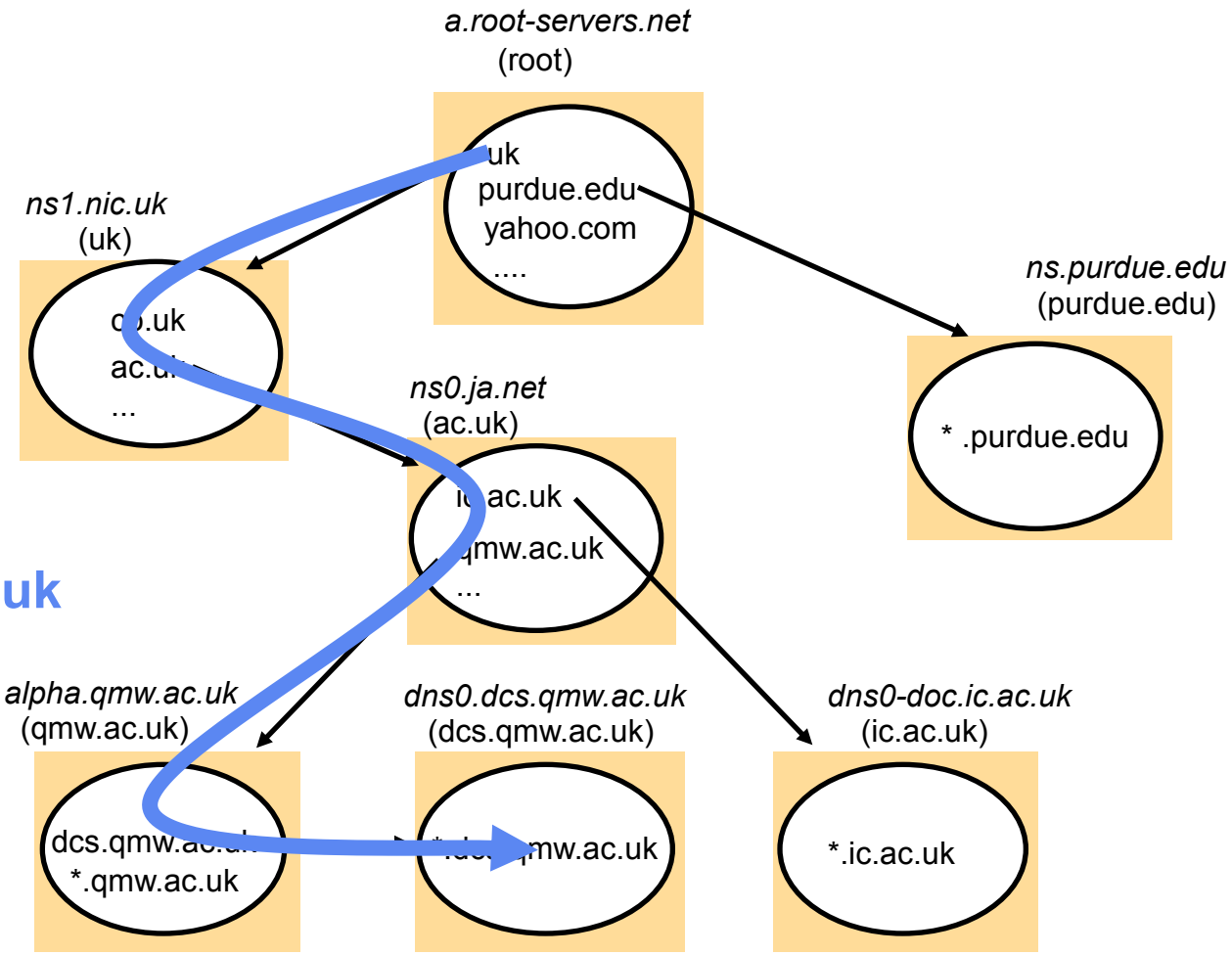


*

# DNS name servers

Figure 9.4

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




authoritative path to lookup:

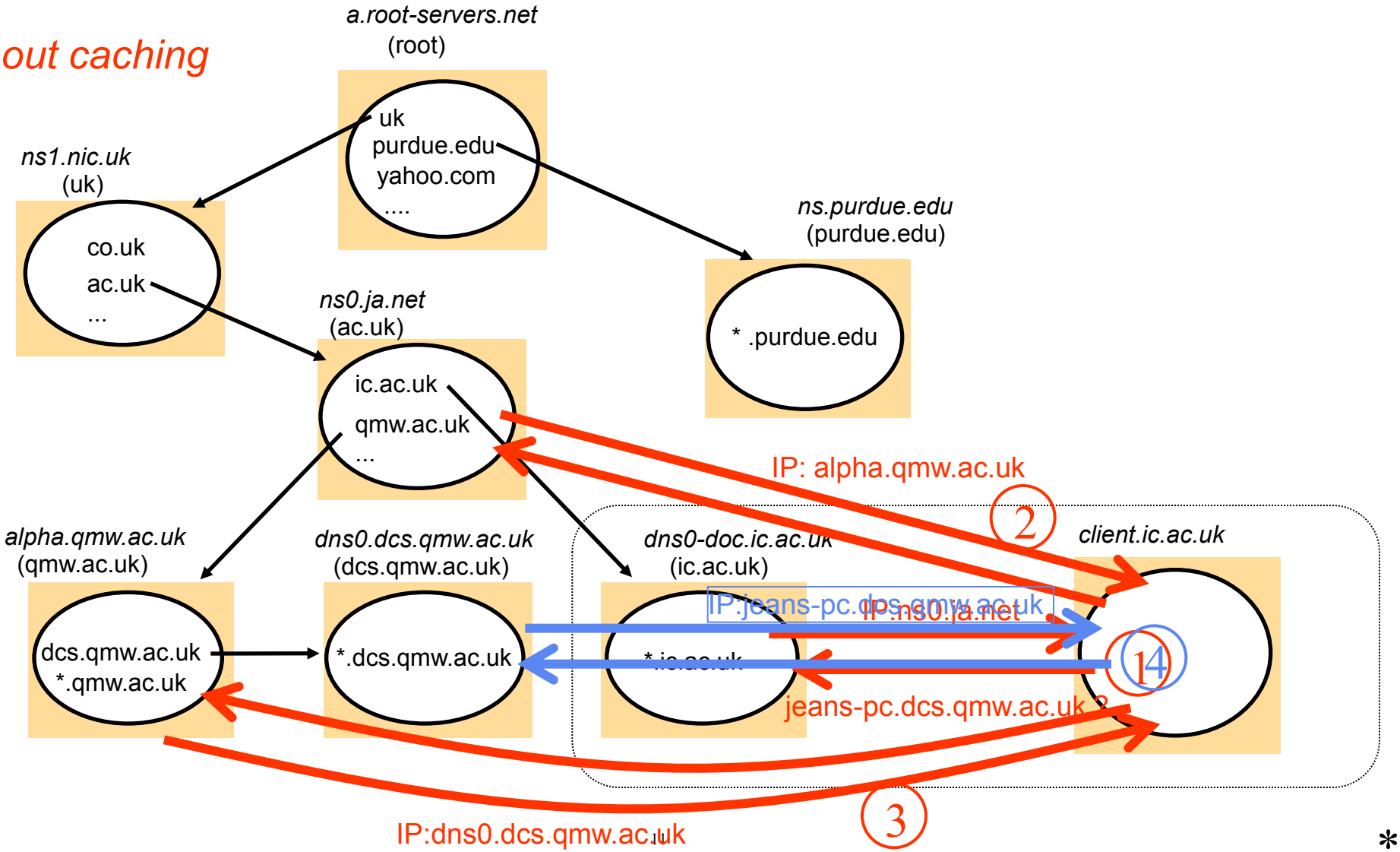
**jeans-pc.dcs.qmw.ac.uk**



*

# DNS in typical operation

*Without caching*



# DNS server functions and configuration

- Main function is to resolve domain names for computers, i.e. to get their IP addresses
  - caches the results of previous searches until they pass their 'time to live'
- Other functions:
  - get *mail host* for a domain
  - reverse resolution - get domain name from IP address
  - Host information - type of hardware and OS
  - Well-known services - a list of well-known services offered by a host
  - Other attributes can be included (optional)



*

# DNS resource records

Figure 9.5

| *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 issues

- Name tables change infrequently, but when they do, caching can result in the delivery of stale data.
  - Clients are responsible for detecting this and recovering
- Its design makes changes to the structure of the name space difficult. For example:
  - merging previously separate domain trees under a new root
  - moving subtrees to a different part of the structure (e.g. if Scotland became a separate country, its domains should all be moved to a new country-level domain.

See Section 9.4 on GNS, a research system that solves the above issues.



*

# Directory and discovery services

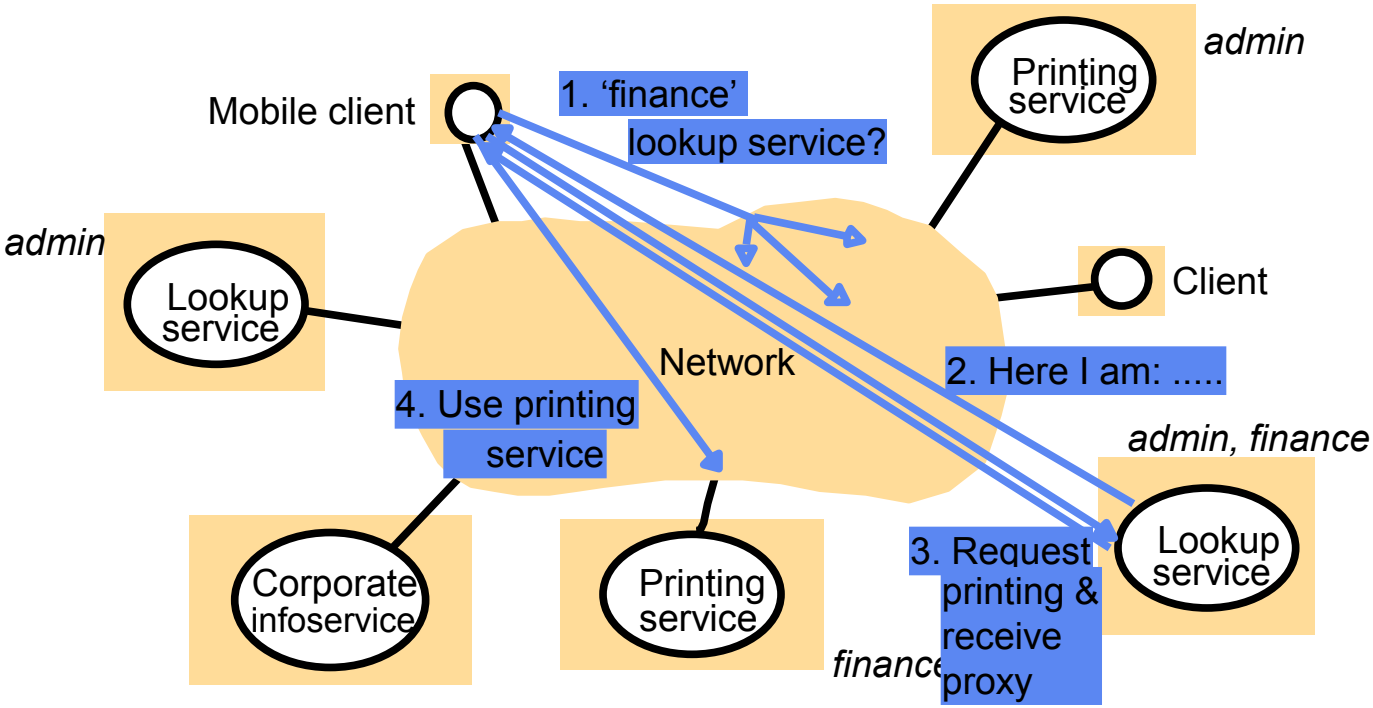
- Directory service:- 'yellow pages' for the resources in a network
  - Retrieves the set of names that satisfy a given description
  - e.g. X.500, LDAP, MS Active Directory Services
    - (*DNS holds some descriptive data, but:*
      - the data is very incomplete
      - DNS isn't organised to search it)
- Discovery service:- a directory service that also:
  - is automatically updated as the network configuration changes
  - meets the needs of clients in spontaneous networks (Section 2.2.3)
  - discovers services required by a client (who may be mobile) within the current *scope*, for example, to find the most suitable printing service for image files after arriving at a hotel.
  - *Examples of discovery services*: Jini discovery service, the 'service location protocol', the 'simple service discovery protocol' (part of UPnP), the 'secure discovery service'.



*

# Service discovery in Jini

Figure 9.6




- Jini services register their interfaces and descriptions with the Jini *lookup* services in their scope
- Clients find the Jini lookup services in their scope by IP multicast
- Jini *lookup* service searches by attribute or by *interface type*
  - The designers of Jini argue convincingly that this the only reliable way to do discovery



*

# Topics not covered

- GNS case study (*Section 9.4*)
  - an early research project (1985) that developed solutions for the problems of:
    - *large name spaces*
    - *restructuring the name space*
- X.500 and LDAP (*Section 9.5*)
  - a hierarchically-structured standard directory service designed for world-wide use
  - accommodates resource descriptions in a standard form and their retrieval for any resource (online or offline)
  - never fully deployed, but the standard forms the basis for LDAP, the Lightweight Directory Access Protocol, which is widely used
- Trading services (*see Section 17.3*)
  - Directories of services with retrieval by attribute searching
  - Brokers negotiate the contract for the use of a service, including negotiation of attribute such as quality and quantity of service



*

# Summary

## Name services:

- defer the binding of resource names to addresses (and other attributes)
- Names are resolved to give addresses and other attributes
- Goals :
  - *Scalability (size of database, access traffic (hits/second), update traffic)*
  - *Reliability*
  - *Trust management (authority of servers)*
- Issues
  - *exploitation of replication and caching to achieve scalability without compromising the distribution of updates*
  - *navigation methods*

## Directory and discovery services:

- 'yellow pages' retrieval by attributes
- dynamic resource registration and discovery



*