

Distributed systems

Week 7, Lecture 1

Logistics

- ▶ Book assignment #2 posted.
- ▶ Less problems per chapter than first assignment, but a few more chapters.
 - ▶ Should be somewhat shorter than last time.
- ▶ Fifth research paper eliminated.
 - ▶ I said “four or five”. It’s four.
- ▶ PA#2 grading.
 - ▶ I will try to have the grades back by next week. My time this week and last week was largely swamped by this upcoming conference.



Change to chapter schedule

- ▶ Web-services will be replaced with peer-to-peer due to my absence next Tuesday.



Projects

- ▶ Optional project status update this Thursday.
- ▶ Intended to just let me know what the status of your project is. Very informal.
- ▶ Take this as an opportunity to see if your team is on track and making progress.
 - ▶ ~4 weeks to go, this is a good time to check up on yourselves.
 - ▶ Good time to ask me for any help – better than last minute. 😊



Today

- ▶ Naming services.
- ▶ Naming services play a very important role in most distributed systems.
- ▶ How do entities find each other?
 - ▶ Typically done using names.
 - ▶ Concrete addresses may change. Names are intended to stay consistent.



Naming

- ▶ Does anyone recognize the address |28.223.6.4|?
- ▶ Probably not. That's because we all call that address "ix.cs.uoregon.edu".
- ▶ One of the most common naming services we use is DNS.
 - ▶ Maps hostnames to IP addresses on the internet.



Naming

- ▶ Naming decouples the identifier of an object from its concrete address.
- ▶ A name is *bound* to an address by the naming service.
- ▶ The act of looking up a name and obtaining the address is known as *resolution*.

- ▶ Resolution turns a name into some attribute that describes an object in a way that makes it accessible. It does not return the object itself.
 - ▶ E.g.: Looking up an object name in an RMI system allows the object instance to be obtained, but the lookup itself does not necessarily provide this instance reference.



Examples

- ▶ **DNS:** Maps domain names onto attributes about the server.
 - ▶ IP address
 - ▶ Mail host information
 - ▶ Length of time that host name entry is valid.
- ▶ **X500:** Maps personal names onto useful attributes.
 - ▶ Phone number
 - ▶ Address
- ▶ **CORBA/JavaRMI:** Maps abstract names onto concrete remote object instances that provide a service or present some expected API.
- ▶ **Filesystem:** Maps filenames onto UFIDs that are used to identify actual data on some storage medium.
 - ▶ May combine DNS with local filesystem naming in the case of a distributed filesystem.



Combined naming

- ▶ Some names actually contain more information than just a name/address resolution.
- ▶ E.g.: <http://www.cs.uoregon.edu/index.html>
- ▶ This name specifies that:
 - ▶ We are looking for a file called index.html in the root directory of ...
 - ▶ A host on the internet called www.cs.uoregon.edu ...
 - ▶ Which is accessible using the HTTP protocol.
- ▶ This notation is quite powerful for having a uniform method for locating arbitrary resources on the Internet.
 - ▶ Hence their name, Uniform Resource Locators (or URLs).



URI, URL, and URN

- ▶ *Uniform Resource Identifiers* are used to identify resources. These can range from web pages, e-mail addresses (<mailto:matt@cs.uoregon.edu>), to telephone numbers (tel:+1-555-555-1212).
- ▶ *Uniform Resource Locators* are more specific, and are used to find resources instead of just identify them.
- ▶ *Uniform Resource Names* are used to identify pure resource names instead of locations. Examples are DOI identifiers used to identify documents.
 - ▶ E.g.: doi:10.1145/1131322.1131331



Name services

- ▶ Name management is typically separated from specific services. This facilitates openness.
- ▶ **Unification:** Convenient to use a consistent naming scheme for different services.
- ▶ **Integration:** It is hard to predict how things will be used in a distributed system. If all resources use a similar naming scheme, then it is easier to integrate them in creative ways later.
 - ▶ Otherwise, integration would require building tools aware of different naming services. This would be inconvenient and time consuming.



Name services

- ▶ With popular naming systems, service is a primary goal.
- ▶ For example DNS:
 - ▶ Replication
 - ▶ Caching
- ▶ Why? The Internet is heavily reliant on operations based on names. The naming service is quite critical.
- ▶ Consistency is a secondary concern. It is more important that clients receive an answer to a query, not necessarily the most up to date.
 - ▶ Service providers must take this into account when updating naming entries.



Namespaces

- ▶ The idea of a namespace is that one can organize entities in a naming scheme such that large numbers of objects are manageable.
- ▶ A namespace also defines the syntax of valid names.
 - ▶ E.g.: “...” is not a legal DNS name.
- ▶ Hierarchic namespaces are common.
 - ▶ Very good to support unanticipated growth.
- ▶ Flat namespaces occasionally used.
 - ▶ Typically finite, bounded by a maximum length identifier.



Aliases

- ▶ Aliases provide mechanisms to map complex names onto simpler ones.
 - ▶ E.g.: `/usr/lib/libfoo.so` instead of `/usr/lib/libfoo.2.0.10.so`
- ▶ Also useful to allow updates in the actual entity that the aliased name points at, transparently to the user.
 - ▶ DNS aliases hide the fact that the actual host that the name is intended to point at may change.
 - ▶ Library aliases in the file system can hide actual version numbers of libraries from the user.
 - ▶ Assuming the API provided by the library hasn't changed.
 - ▶ Library aliasing sometimes uses many aliases for version-level aliasing:
 - `libfoo.2.0.10.so`, `libfoo.2.0.so`, `libfoo.2.so`, `libfoo.so`



Domains

- ▶ Typically domains correspond to administrative authorities.
 - ▶ Internet authorities control who can obtain names in .edu.
 - ▶ The University determines who can obtain names within the .uoregon.edu domain.
 - ▶ CIS admins control who can obtain names within the cs.uoregon.edu domain.
- ▶ And so on...



Combining namespaces

- ▶ Sometimes it is useful to take multiple disjoint namespaces and combine them into a single unified namespace.
 - ▶ Need to make sure merger doesn't cause conflicts.
- ▶ Merging is a simple method. Assuming a hierarchical namespace, add higher level names into which the roots of the disjoint namespaces are bound.
 - ▶ E.g.: Mount points in a filesystem.
- ▶ Some systems allow customized namespaces to be created.
 - ▶ E.g.: Plan9 allows you to bring multiple directories together into a single directory.
 - ▶ Useful for treating a set of directories in a search path as a single unit.
 - ▶ Makes searches consistent: looking at a directory is the same as a search path, instead of having to treat search path variables separately.



Name server structuring

- ▶ In a simple distributed system, naming can be performed by a single server.
 - ▶ E.g.: RMI Registry
- ▶ In scalable systems though, the naming service itself must be distributed.
 - ▶ DNS would not work if there was one single server.
 - ▶ Single point of failure.
 - ▶ Performance bottleneck.
- ▶ What have we learned so far that would help with this?



Replication

- ▶ Replication can be used on the server side to increase performance and be robust to failure.
- ▶ Clients can have a set of possible name servers to contact.
 - ▶ See `/etc/resolv.conf` on many Unix machines.
 - ▶ Multiple “nameserver” lines are valid.



Iterative lookup

- ▶ Rarely does any server in a naming service know the entire namespace, but, servers can find other servers that can help resolve names.
- ▶ Name servers will be associated with each administrative domain (eg: uoregon.edu). These name servers will have details about their local domains, but may not have detailed name information for other domains.
- ▶ Lookups to servers will sometimes result in an incomplete response that indicates a server that should be queried that may have further information.
 - ▶ E.g.: Query to uoregon.edu for host.dept.pdx.edu returns a response that indicates the pdx.edu or dept.pdx.edu DNS server should be contacted to resolve the host.

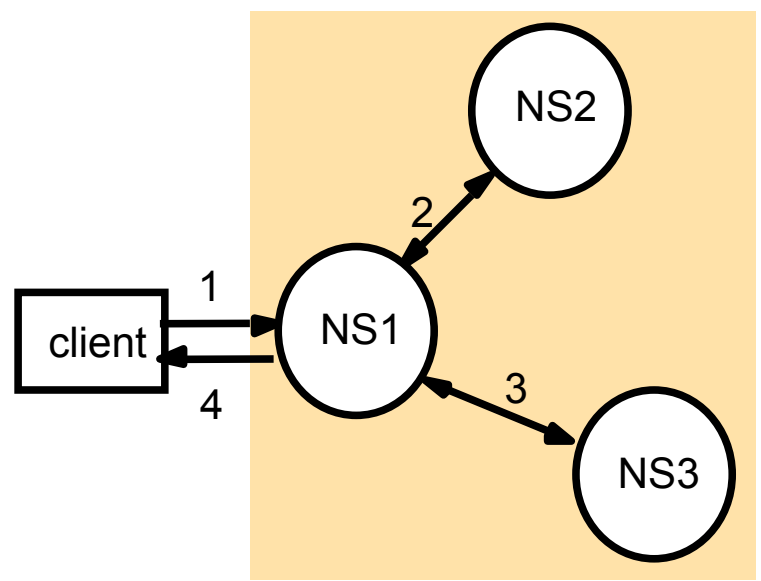


Controlled lookups

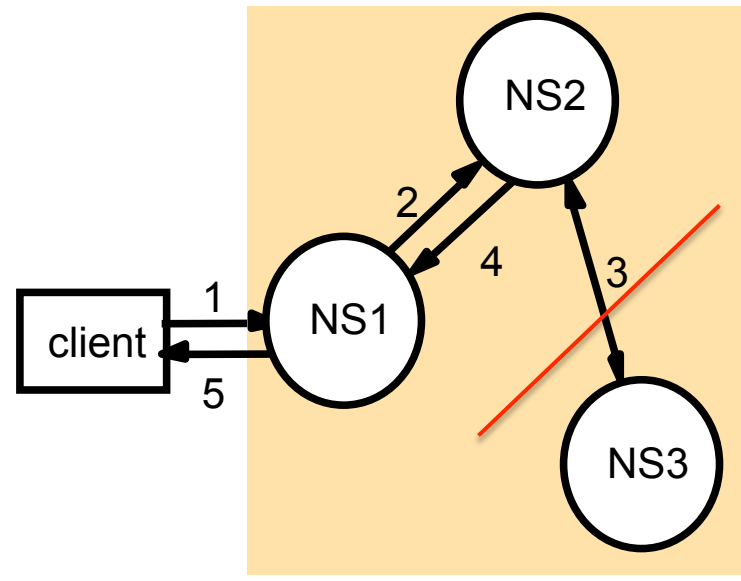
- ▶ A name server can also perform this iterated resolution of a name on behalf of the client, instead of asking the client to do it.
- ▶ Both of these schemes require that the name servers containing the detailed information be visible to the client or server executing the lookup.
- ▶ This may not be a legal assumption, as the servers with the information may be hidden behind some administrative boundary (e.g.: firewall).
- ▶ Recursive searches can deal with this. A server asks another server for the information, and that server asks others if it cannot find it, and so on.
 - ▶ Names are returned but as they return, the information about who resolved them is not passed back down the chain.



Iterative vs. recursive controlled lookups



Non-recursive server-controlled



Recursive server-controlled

Admin. boundary. NS2 trusted, NS1 and client not.

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

Caching and iterative lookups

- ▶ Say I add a new host today and give it a name “matt.cs.uoregon.edu”.
- ▶ That name will be registered in the UO DNS servers, but likely not cached elsewhere very soon.
- ▶ Someone, somewhere may wish to find that host in the future. Since it is not cached on their side, their DNS server will have to seek out the information from the UO server, or some other DNS server that has it cached.



Caching

- ▶ Caching is used to increase performance, reduce network overhead.
- ▶ Local servers lookup unknown names from authoritative servers when not found in their local store.
 - ▶ When response received, request is answered and data is kept in cache.
- ▶ Cached data has a “time to live” and the time when the data is retrieved is recorded.
 - ▶ Cached data is reused if within TTL limit.
- ▶ Clients are given cached data with an annotation that indicates it is a non-authoritative answer.



DNS

- ▶ Designed in the mid-1980s.
- ▶ Replaced a crude system based on a centralized repository of name/address mappings that was downloaded via FTP onto computers that required tem.
 - ▶ Clearly this old scheme could not scale.
- ▶ Also made it difficult to locally manage domain naming within an organization.
- ▶ Internet designers wanted a naming scheme that had more than host/address mappings.
 - ▶ For example, DNS allows mail-related information to be looked up.



DNS

- ▶ Based on hierarchical naming, replicated data, and caching.
- ▶ Interestingly, DNS was designed for more general purpose naming than we see today.
 - ▶ Today it's pretty much exclusively used for identifying hosts on the Internet.
- ▶ The hierarchy of names has organizational (.com, .net, .org) or geographical (.us, .cx, .de, .uk, etc...) domains for very coarse grained organization.



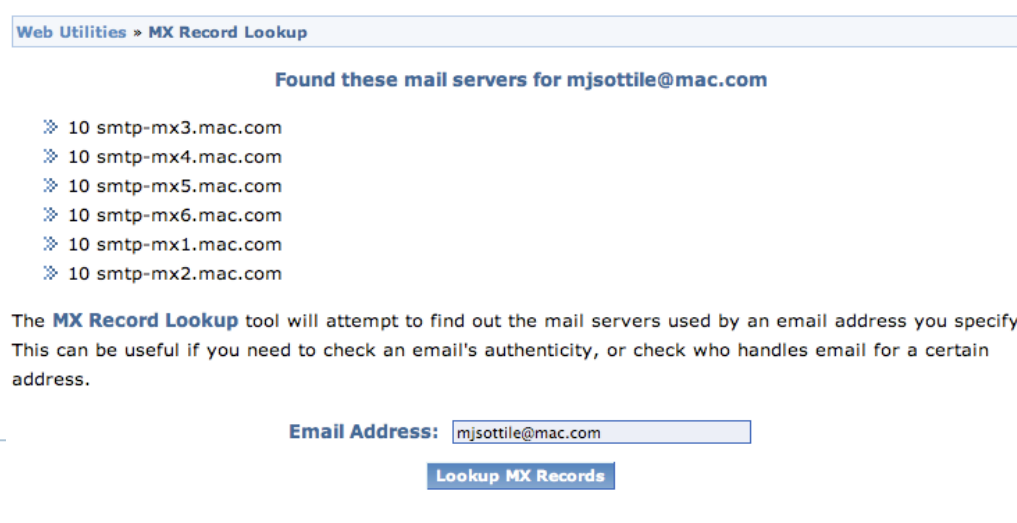
Domain names

- ▶ Com: commercial
 - ▶ Edu: educational
 - ▶ Gov: US Government
 - ▶ Mil: US Military
 - ▶ Net: Network support centers
 - ▶ Org: Organizations not in the above.
 - ▶ Int: International organizations.
- ▶ .us: United States
 - ▶ .uk: United Kingdom
 - ▶ .dk: Denmark
 - ▶ .cx: Christmas Island
 - ▶ .tv: Tuvalu



Name resolution

- ▶ **Host names:**
 - ▶ Simple mapping of hostnames to IP addresses.
- ▶ **Mail hosts:**
 - ▶ Used to see what host handles SMTP mail for a domain.
 - ▶ Look up the MX record for cs.uoregon.edu, and you will see that “vitalstatistix.cs.uoregon.edu” is the mail host.
 - ▶ Multiple servers can be listed in case any are unreachable.



Web Utilities » MX Record Lookup

Found these mail servers for mjsottile@mac.com

- ❖ 10 smtp-mx3.mac.com
- ❖ 10 smtp-mx4.mac.com
- ❖ 10 smtp-mx5.mac.com
- ❖ 10 smtp-mx6.mac.com
- ❖ 10 smtp-mx1.mac.com
- ❖ 10 smtp-mx2.mac.com

The **MX Record Lookup** tool will attempt to find out the mail servers used by an email address you specify. This can be useful if you need to check an email's authenticity, or check who handles email for a certain address.

Email Address:

[Lookup MX Records](#)

Other DNS services

- ▶ Some servers implement more services.
 - ▶ Reverse lookup:
 - ▶ Translate an IP into a name. If the IP is in the server's domain it will respond.
 - ▶ Host information:
 - ▶ This is discouraged, as it can compromise sensitive security information.
 - E.g.: Query for a known version of an OS w/ a vulnerability.
 - ▶ Well-known services:
 - ▶ Discover what services a host provides (e.g.: web, FTP) and the protocol to contact them (UDP,TCP).



General naming

- ▶ Queries to DNS are made up of:
 - ▶ Domain to resolve.
 - ▶ Class of query.
 - ▶ Type of query.
- ▶ Domain is just the name.
- ▶ Class of query. “IN” is the class for Internet domains names. Others can exist (e.g.: Experimental DNS naming databases).
- ▶ Type of query defines what information about the domain name is being sought. The types are related to the class.



Types

- ▶ Here are some examples for the IN class.
 - ▶ **A**:Address (IP address)
 - ▶ **NS**:Authoritative name server.
 - ▶ **CNAME**: Canonical name of an alias.
 - ▶ **SOA**: Marker for start of data for a zone.
 - ▶ **WKS**:Well known service description.
 - ▶ **PTR**: Domain name pointer (for reverse lookup)
 - ▶ **HINFO**: Host info.
 - ▶ **MX**: Mail exchange
 - ▶ **TXT**:Arbitrary text.



Example: PDX.EDU

Answer records

name	class	type	data	time to live
pdx.edu	IN	MX	preference: 5 exchange: smtp-incoming.pdx.edu	14400s (4h)
pdx.edu	IN	A	131.252.120.50	14400s (4h)
pdx.edu	IN	SOA	server: dns0.pdx.edu email: hostmaster@pdx.edu serial: 2008110501 refresh: 900 retry: 300 expire: 3600000 minimum ttl: 1800	14400s (4h)
pdx.edu	IN	NS	dns0.pdx.edu	14400s (4h)
pdx.edu	IN	NS	dns1.pdx.edu	14400s (4h)
pdx.edu	IN	NS	walt.ee.pdx.edu	14400s (4h)
pdx.edu	IN	NS	phloem.uoregon.edu	14400s (4h)



Scalability

- ▶ Replication and caching.
- ▶ Local DNS servers hold complete local information.
 - ▶ Most queries are local, so fast, local servers perform well.
- ▶ DNS is split into zones.
 - ▶ A zone is:
 - ▶ Attribute data for a domain minus that which is managed by a subdomain. (E.g.: uoregon.edu could defer to cs.uoregon.edu for CIS dept. info)
 - ▶ Names and addresses of at least two authoritative servers. Authoritative servers are most likely to be up to date.
 - ▶ Names and addresses of subdomain servers.
 - ▶ Zone management data.



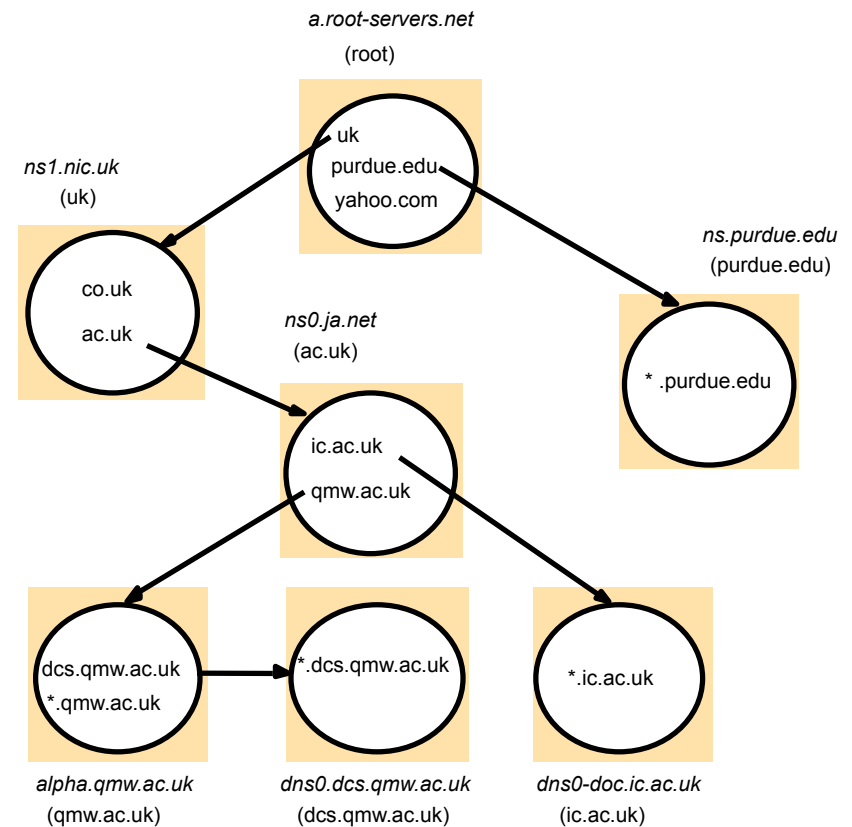
Server relationships

- ▶ **Authoritative servers can be trusted to have up-to-date data.**
 - ▶ Primary servers: These servers store the naming data in a file.
 - ▶ Secondary servers: These servers retrieve data from primary servers and cache it locally.
- ▶ Secondary servers update periodically (~1 day resolution) with primary servers.
- ▶ This delay in information propagation is not uncommon to encounter.
- ▶ Commonly seen when signing up for a new domain or moving a domain from one host to another.
 - ▶ This is acceptable, as once set up, changes are infrequent.



Hierarchy

- ▶ Root-level servers have entries for authoritative servers for top-level domains (TLDs), and generic TLDs like .com and .edu.
- ▶ These other authoritative servers resolve names further and pass down the hierarchy until the names are resolved.



Discovery services

- ▶ Name/attribute mappings are not restricted to name-to-attribute lookups only.
- ▶ Discovery services allow attribute-oriented queries.
 - ▶ “Who has phone number 555-1212?”
 - ▶ “What are the second floor printers?”
- ▶ *Examples:* LDAP, X.500, Microsoft Active Directory Services, etc.
- ▶ Note that this is different than discovery in the sense of spontaneous or ad hoc networks.
- ▶ In this case, we have a well known service with pre-defined entries. Spontaneous situations attempt to infer similar information via multicast message/response protocols.



X.500

- ▶ ISO/ITU standard
- ▶ Basis for LDAP (Lightweight Directory Access Protocol)
- ▶ Tree-based data organization

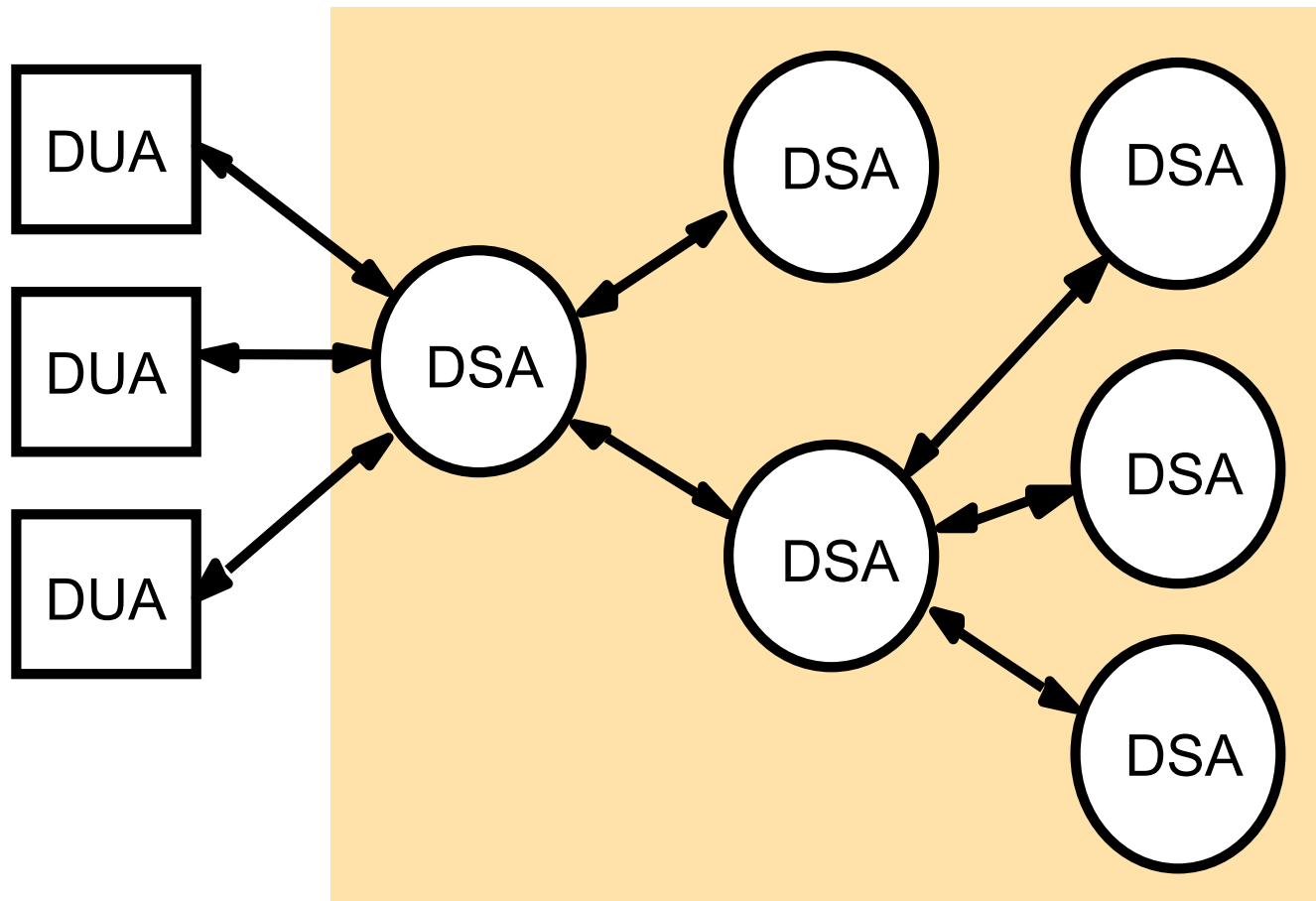


X.500 organization

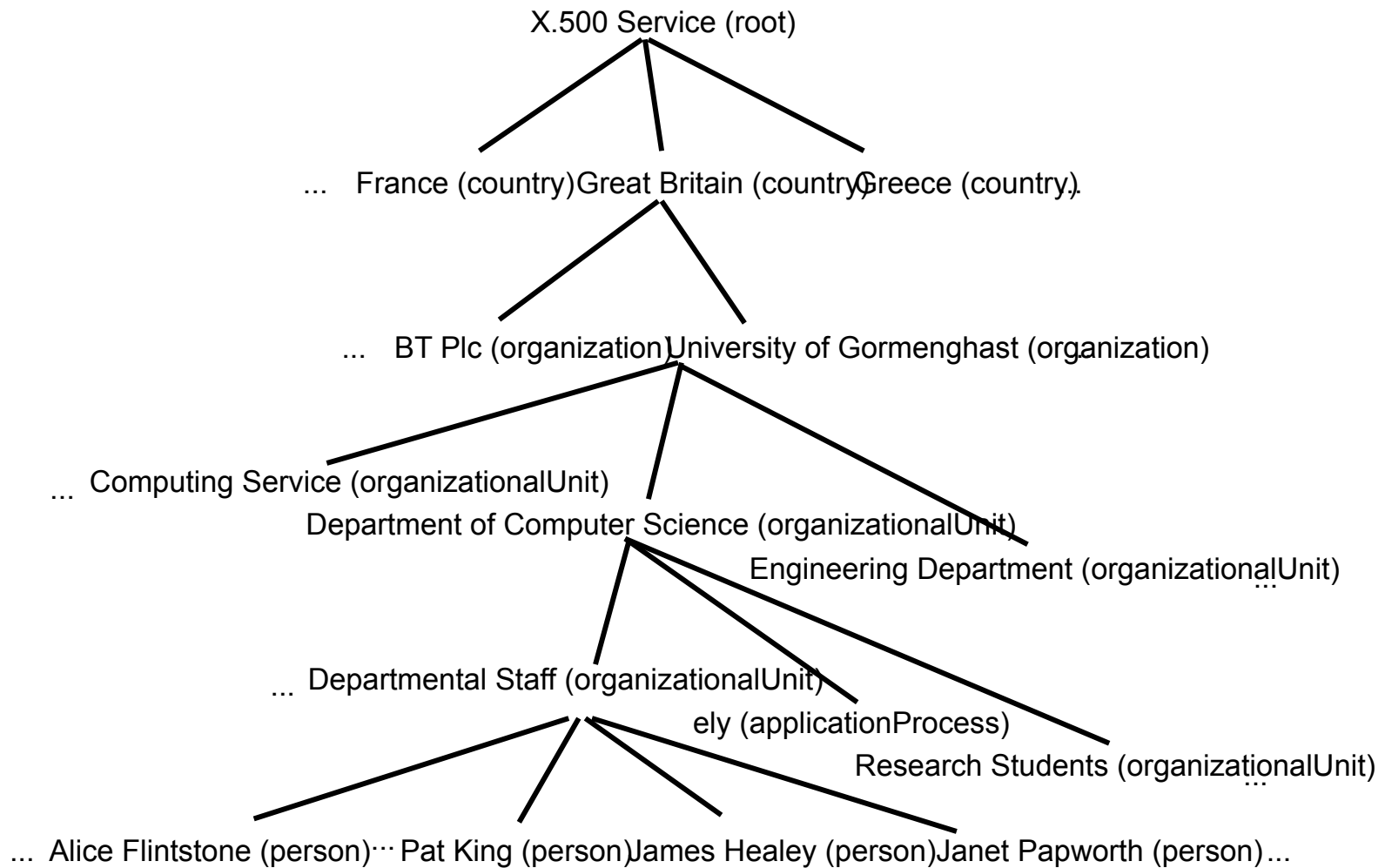
- ▶ **Directory information tree (DIT).**
 - ▶ Name tree.
- ▶ **Directory information base (DIB).**
 - ▶ Data associated with nodes.
- ▶ **Directory service agent (DSA).**
 - ▶ Servers. These hold DIBs, answer queries from clients.
 - ▶ If a server DIB doesn't have the data, it can ask another server or redirect the client to another server.
- ▶ **Directory user agent (DUA).**
 - ▶ Clients.



X.500



DIB structure



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

uid

alf

mail

alf@dcs.gormenghast.ac.uk

surname

Flintstone

Alice.Flintstone@dcs.gormenghast.ac.uk

roomNumber

Z42

telephoneNumber

+44 986 33 4604

userClass

Research Fellow

Queries

- ▶ **Read:**

- ▶ Given a path to an entity and an attribute set (possibly all), read the information for the entry.

- ▶ **Filter:**

- ▶ Given a base-path name (e.g.: CIS dept.), and an attribute to match (e.g.: room number), return the set of entry names that match the attribute.
- ▶ This list can then be read using the “read” operation.



Remaining lectures

- ▶ Peer-to-peer systems
- ▶ Distributed shared memory

