Global Name Services (GNS)

CAT3053/N Distributed Computing

Week 10

Global Name Service

- Designed and implemented by Lampson and colleagues at the DEC Systems Research Center (1986)
- Provide facilities for
  - resource location, email addressing, authentication
- When the naming database grows from small to large scale, the structure of name space may change.
- The service should accommodate:
  - Any changes in the names of the individuals, organizations and groups that it holds
  - Any changes in the naming structure such as those that occur when one company is taken over by another

Global Name Service (Cont.)

- The potentially large naming database and the scale of the distributed environment in GNS
  - Make the use of caching essential
  - Difficult to maintain complete consistency between all copies of a database entry.
- Cache consistency strategy assume
  - updates to the database will be infrequent
  - slow dissemination of updates is acceptable
  - clients can detect and recover from out-of-date naming data

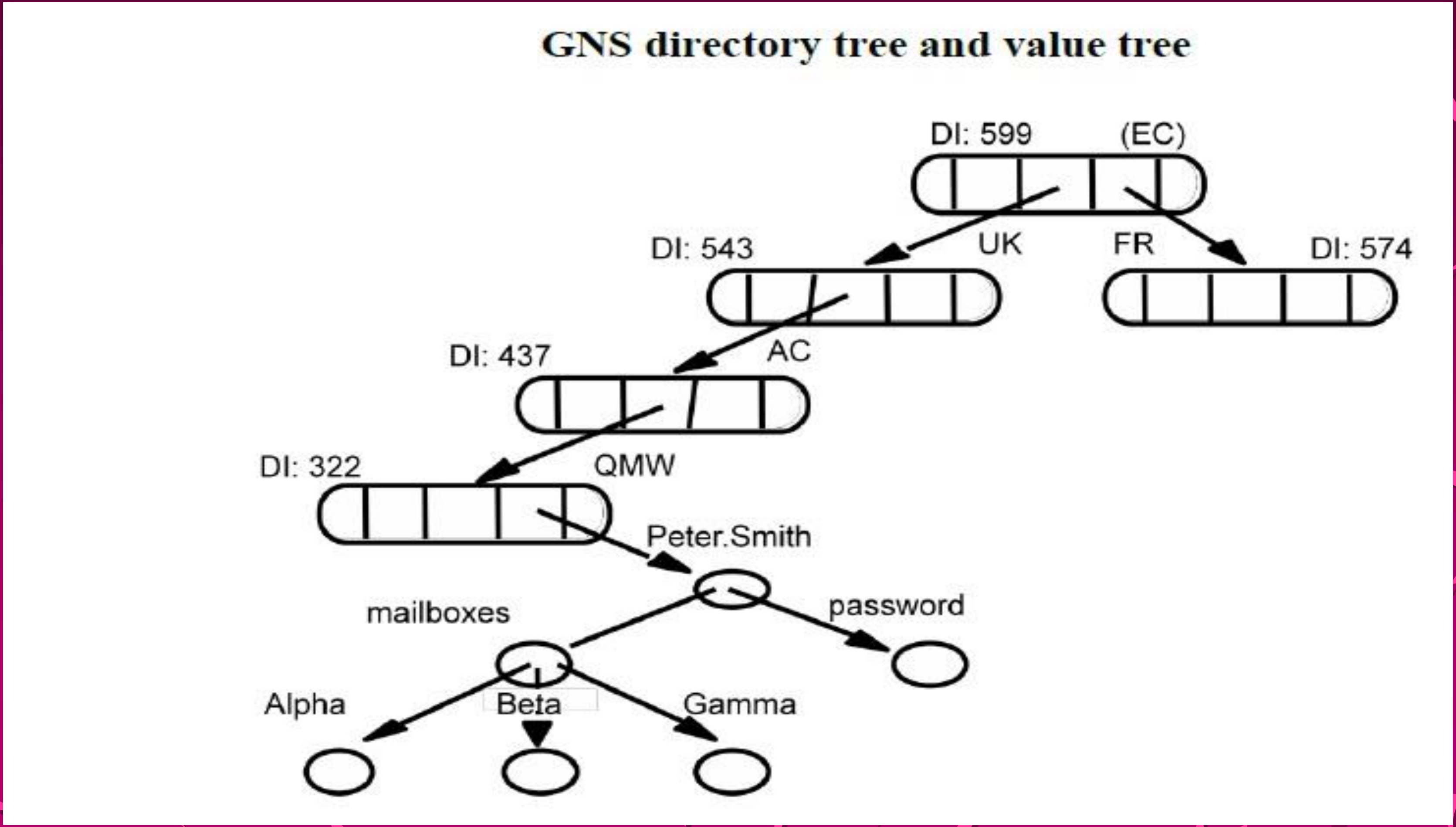
Global Name Service (Cont.)

- The GNS manages a naming database that is composed of a tree of directories holding names and values.
- Directories are named by multi-part pathnames referred to a root, or relative to a working directory, much like file names in a UNIX file system.
- Each directory is also assigned an integer, which serves as a **unique *directory identifier*** *(DI).*
- A directory contains a list of names and references.
- The values stored at the leaves of the directory tree are organized into *value trees, so that the attributes associated with names can be structured* values.

Global Name Service (Cont.)

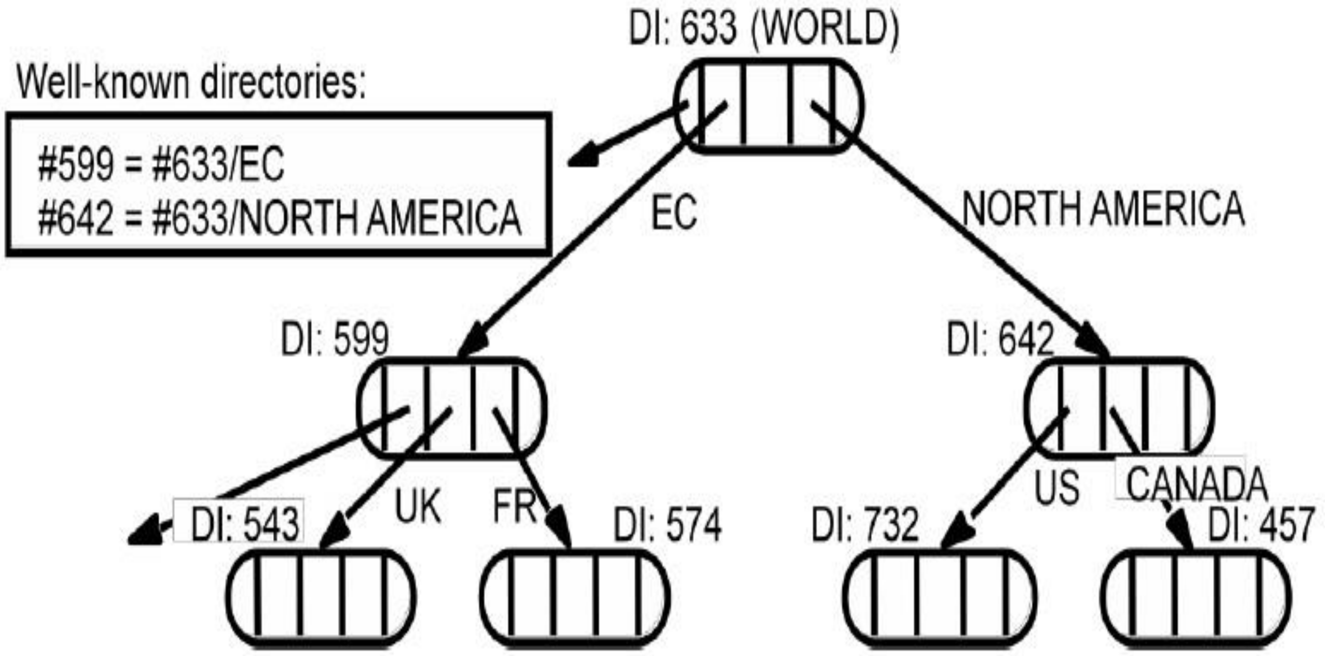
- Names in the GNS have two parts:
  - *<directory name, value name>*.
- *First part identifies a* directory;
- Second refers to a value tree, or some portion of a value tree.

Global Name Service (Cont.)



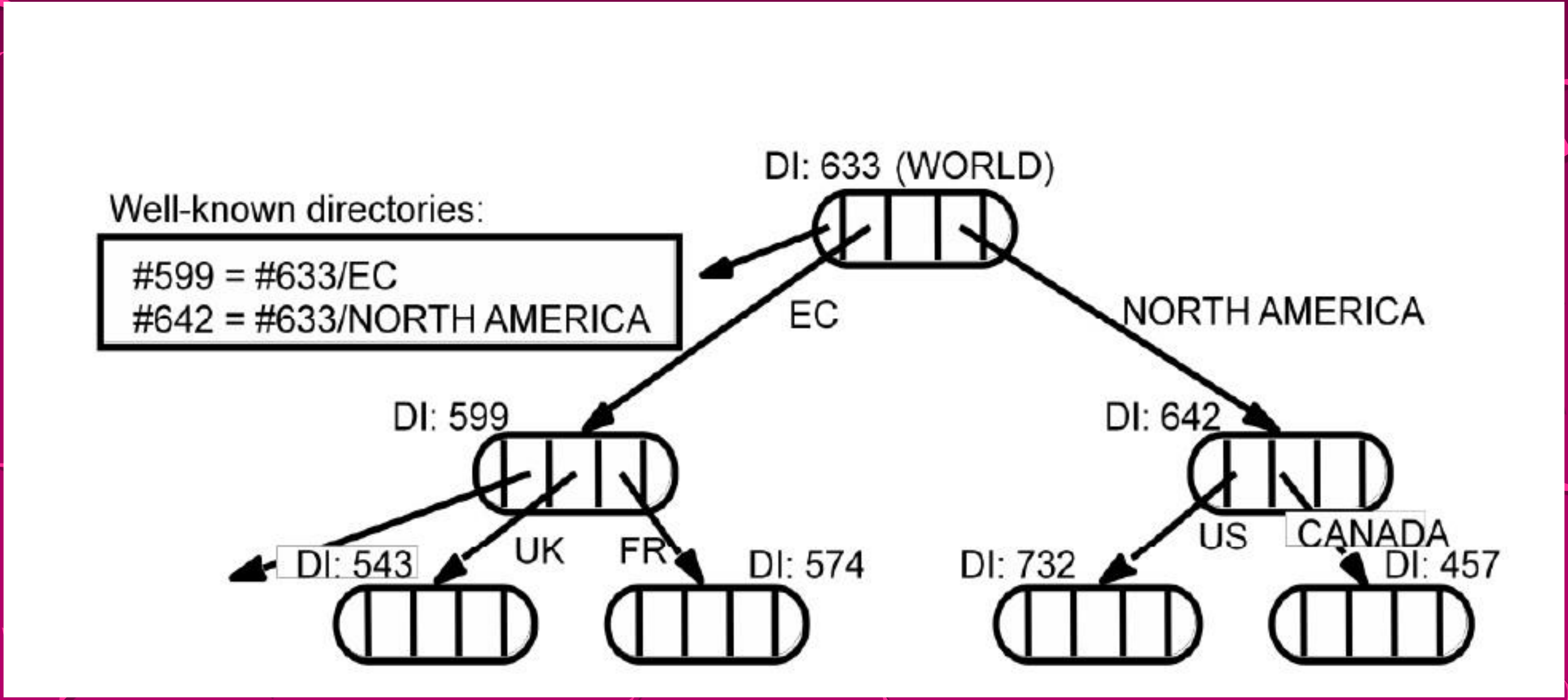
Accommodating Changes

- How to integrate naming trees of two previously separate GNS services
- But what is for '/UK/AC/QMV, Peter.Smith' ?



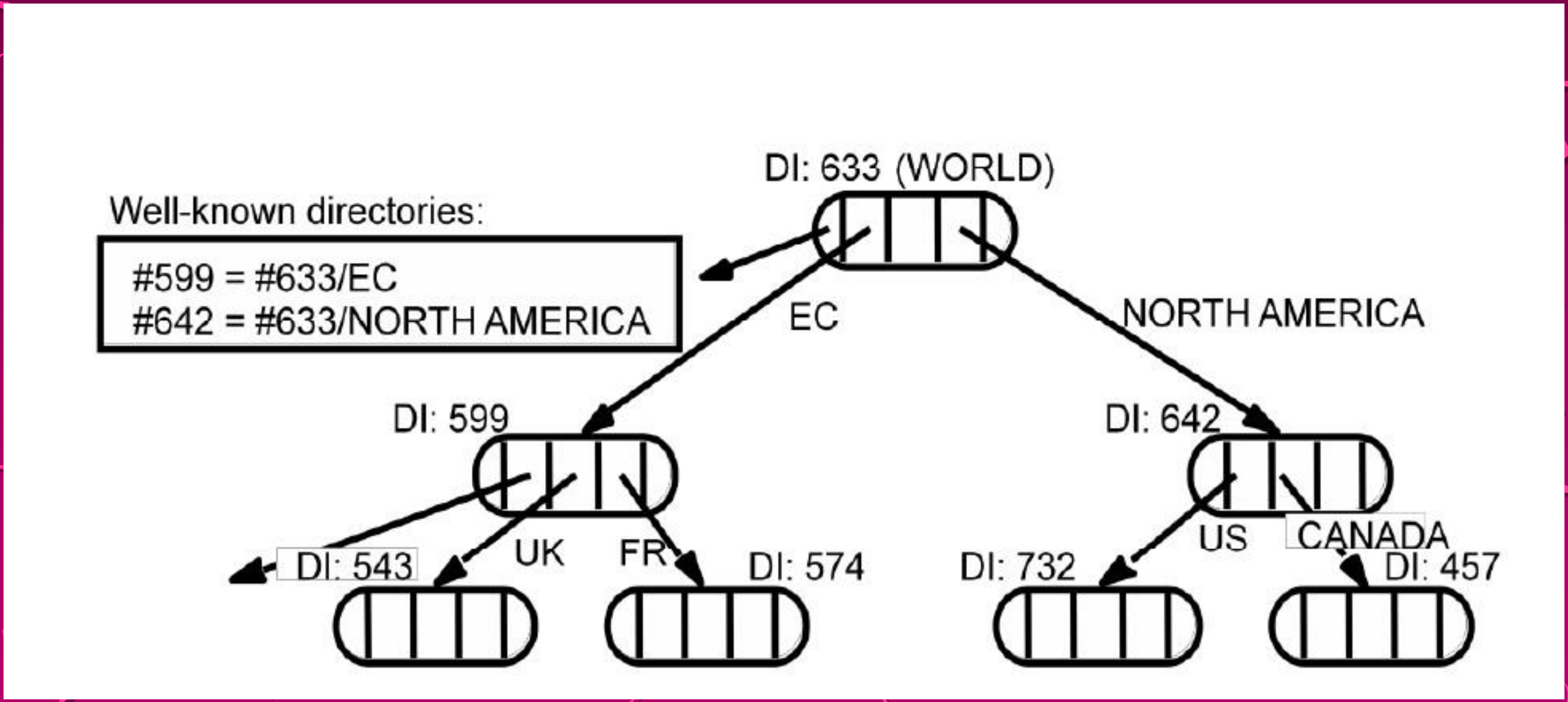
Using DI to solve changes

- Using the name '#599/UK/AC/QMV, Peter.Smith'



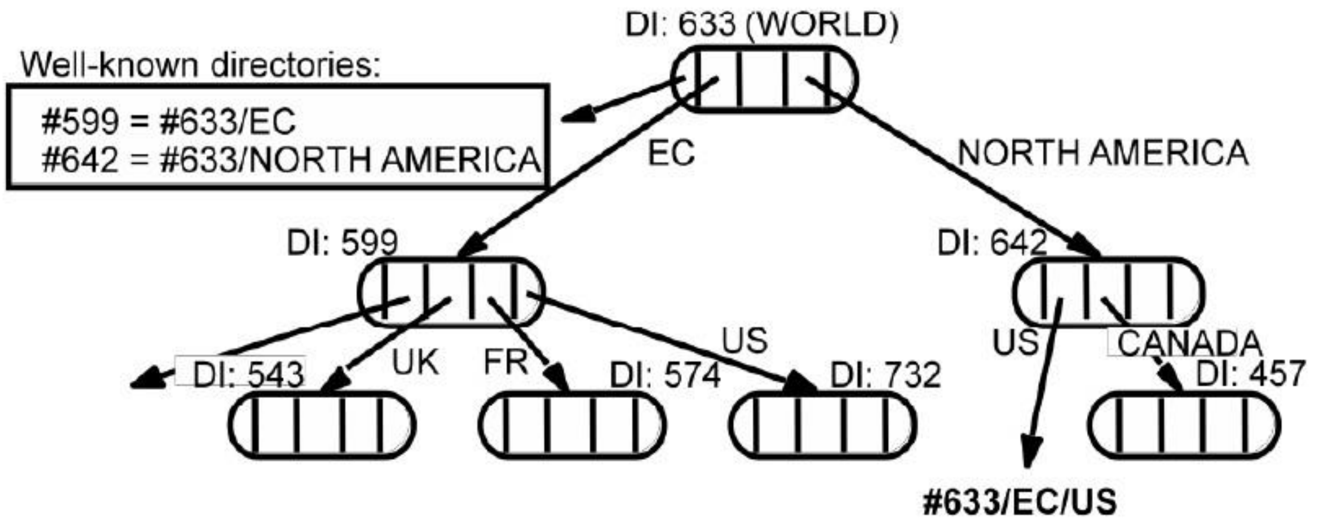
Using DI to solve changes

- But how GNS locate a directory given only DI?



Restructuring of Database

- Using symbolic links



GNS Issues

- The GNS is descended from Grapevine and Clearinghouse, two successful naming systems developed primarily for the purposes of mail delivery by the Xerox Corporation.
- GNS addresses needs for scalability and reconfigurability for merging and moving directory trees
- However, a database (the table of well-known directories) that must be replicated at every node
- In a largescale network, reconfigurations may occur at any level, and this table could grow to a large size, conflicting with the scalability goal.

X.500 Directory Services

- X.500 is a directory service used in the same way as a conventional name service
  - Primarily used to satisfy descriptive queries
  - Used to discover the names and attributes of other users or system resources.
- Users may have a variety of requirements for searching and browsing in a directory of network users, organizations and system resources to obtain information about the entities that the directory contains.
- The uses for such a service are likely to be quite diverse.

X.500 Directory Services (Cont.)

- Enquiries that are directly analogous to the use of telephone directories
  - ‘white pages’ access to obtain a user’s electronic mail address
  - ‘yellow pages’ query aimed at obtaining the names and telephone numbers of garages specializing in the repair of a particular make of car
- The use of the directory to access personal details
  - Job roles, dietary habits, and photographic images of the individuals.
- Queries may originate from users or from **processes**
  - used to identify services to meet a functional requirement

X.500 Directory Services (Cont.)

- Individuals and organizations can use a directory service to make available
  - Wide range of information about themselves
  - Resources that they wish to offer for use in the network.
- Users can search the directory for specific information with only partial knowledge of its name, structure or content.

X.500 Directory Services (Cont.)

- Standard of ITU and ISO organizations
- General-purpose directory service
- A service for access to information about ‘real-world entities’
- Used for access to information about hardware and software services and devices
- Application-level service in the Open Systems Interconnection (OSI) set of standards
- Does not depend to any significant extent on the other OSI standards

X.500 Service Architecture

- The data stored in X.500 servers is organized in a tree structure with named nodes with
  - a wide range of attributes are stored at each node in the tree
  - access is possible not just by name and by searching for entries with any required combination of attributes
- X.500 name tree is called the *Directory Information Tree* (DIT)
- The entire directory structure including the data associated with the nodes, is called the *Directory Information Base* (DIB)

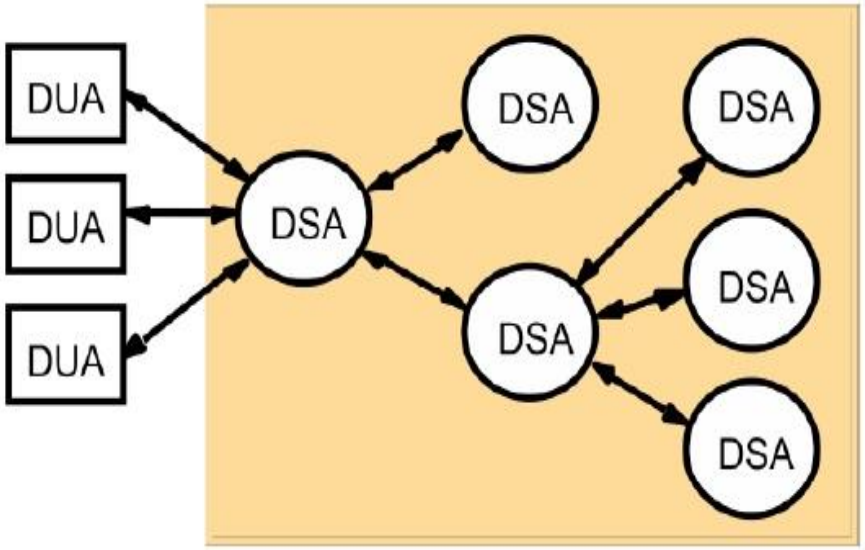
X.500 Service Architecture

- There is intended to be a single integrated DIB containing information provided by organizations throughout the world, with portions of the DIB located in individual X.500 servers.
- Typically, a medium-sized or large organization would provide at least one server.
- Clients access the directory by establishing a connection to a server and issuing access requests.
- Clients can contact any server with an enquiry.

X.500 Service Architecture

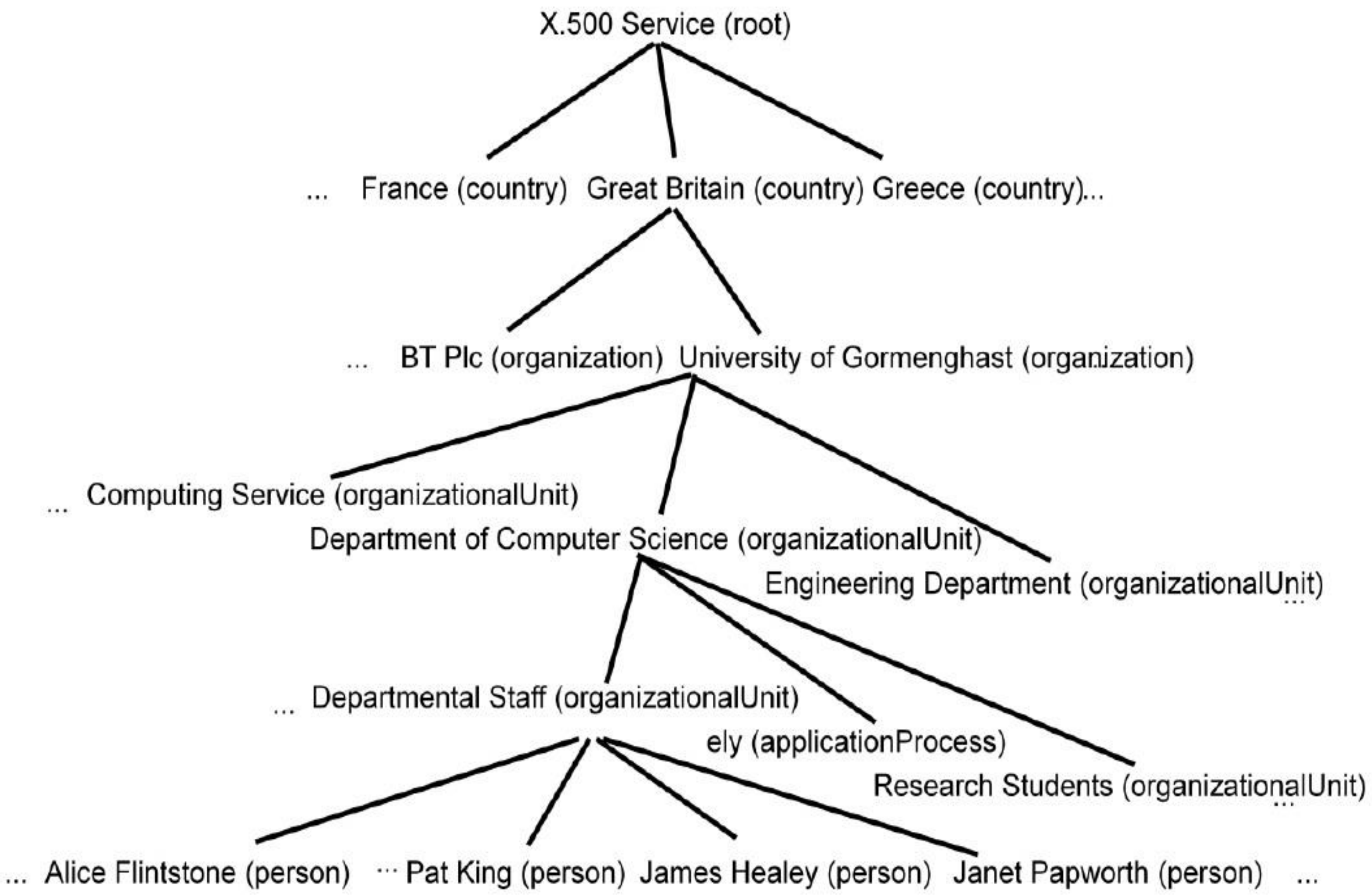
- If the data required are not in the segment of the DIB held by the contacted server, it will either invoke other servers to resolve the query or redirect the client to another server.
- In the terminology of the X.500 standard
  - servers are *Directory Service Agents* (DSAs)
  - clients are termed *Directory User Agents* (DUAs)

- Directory Server Agent (DSA)
- Directory User Agent (DUA)

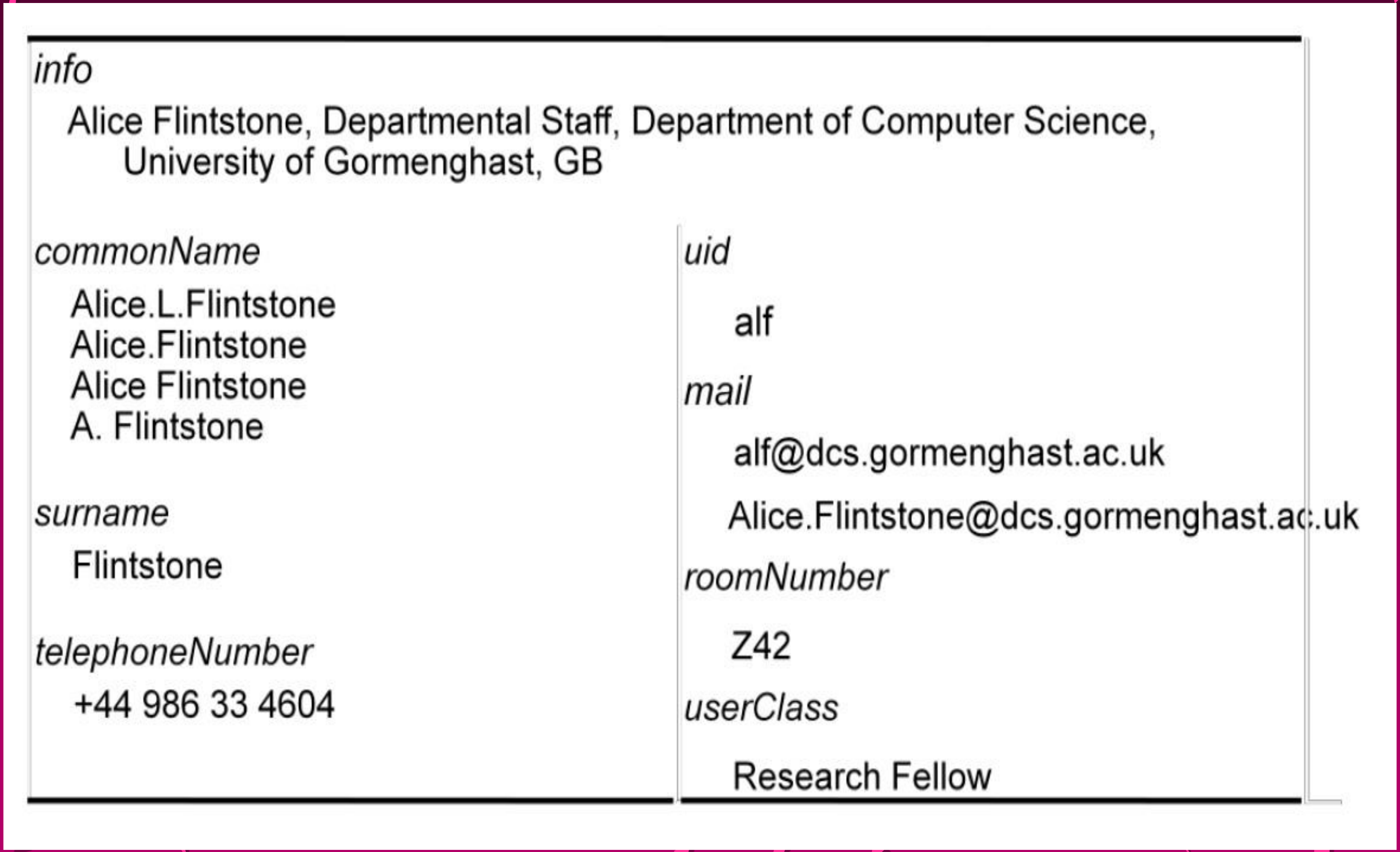


X.500 Service Architecture (Cont.)

- Each entry in the DIB consists of a name and a set of attributes.
- Full/absolute name of an entry corresponds to a path through the DIT from the root of
  the tree to the entry
- DUA can establish a context, which includes a base node,
  - Use shorter relative names that give the path from the base node to the named entry


X.500 Service (root)
... France (country) Great Britain (country) Greece (country)...
... BT Plc (organization) University of Gormenghast (organization)
... Computing Service (organizationalUnit)
Department of Computer Science (organizationalUnit)
Engineering Department (organizationalUnit)
...
... Departmental Staff (organizationalUnit)
ely (applicationProcess)
Research Students (organizationalUnit)
...
... Alice Flintstone (person) ... Pat King (person) James Healey (person) Janet Papworth (person) ...

An X.500 DIB Entry



info

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

commonName	*uid*
Alice.L.Flintstone Alice.Flintstone Alice Flintstone A. Flintstone	alf
	mail
	alf@dcs.gormenghast.ac.uk Alice.Flintstone@dcs.gormenghast.ac.uk
surname	*roomNumber*
Flintstone	Z42
telephoneNumber	*userClass*
+44 986 33 4604	Research Fellow

An X.500 DIB Entry (Cont.)

- The data structure for the entries in the DIB and the DIT is very flexible.
- A DIB entry consists of a set of attributes, where an attribute has a *type* and one or more *values*.
- The type of each attribute is denoted by a type name
  - *countryName*, *organizationName*, *commonName*, *telephoneNumber*, *mailbox*, *objectClass*).
- For each distinct type name there is a corresponding type definition
  - includes a type description and a syntax definition in the ASN.1

An X.500 DIB Entry (Cont.)

- DIB entries are classified in a manner similar to the object class structures found in object-oriented programming languages.
- Each entry includes an *objectClass attribute*, *which determines* the class (or classes) of the object to which an entry refers.
- *Organization*, *organizationalPerson* and *document are all examples of objectClass values.*
- Further classes can be defined as they are required.

An X.500 DIB Entry (Cont.)

- The definition of a class determines which attributes are mandatory and which are optional for entries of the given class.
- The definitions of classes are organized in an inheritance hierarchy in which all classes except one (called *topClass*) must contain an *objectClass* attribute, and the value of the *objectClass* attribute must be the names of one or more classes.
- If there are several *objectClass* values, the object inherits the mandatory and optional attributes of each of the classes.

An X.500 DIB Entry (Cont.)

- The name of a DIB entry (the name that determines its position in the DIT) is
  determined by selecting one or more of its attributes as *distinguished attributes*.
- The attributes selected for this purpose are referred to as the entry’s *Distinguished Name* (DN)

Directory Access Method

- *read*
- An absolute or relative name (a *domain name* in X.500 terminology) for an entry is given, together with a list of attributes to be read (or an indication that all attributes are required).
- The DSA locates the named entry by navigating in the DIT, passing requests to other DSA servers where it does not hold relevant parts of the tree.
- It retrieves the required attributes and returns them to the client.

Directory Access Method (Cont.)

- *search*
- This is an attribute-based access request.
  - A base name and a filter expression are supplied as arguments.
  - The base name specifies the node in the DIT from which the search is to commence.
  - The filter expression is a boolean expression that is to be evaluated for every node below the base node
- The *search* command returns a list of names (domain names) for all of the entries below the base node for which the filter evaluates to *TRUE*.

Administration and Updating of the DIB

- The DSA interface includes operations for adding, deleting and modifying entries.
- Access control is provided for both queries and updating operations, so access to parts of the DIT may be restricted to certain users or classes of user
- The DIB is partitioned, with the expectation that each organization will provide at least one server holding the details of the entities in that organization.
- Portions of the DIB may be replicated in several servers.

Lightweight Directory Access Protocol

- X.500’s assumption that organizations would provide information about themselves in public directories within a common system has proved largely unfounded.
- Group at the University of Michigan proposed a more lightweight approach called the *Lightweight Directory Access Protocol (LDAP)*
  - a *DUA* accesses X.500 directory services directly over TCP/IP instead of the upper layers of the ISO protocol stack.
- LDAP also simplifies the interface to X.500
  - provides a relatively simple API and it replaces ASN.1 encoding with textual encoding.

Lightweight Directory Access Protocol (Cont.)

- Although the LDAP specification is based on X.500, LDAP does not require it
- An implementation may use any other directory server that obeys the simpler LDAP specification, as opposed to the X.500 specification.
  - Microsoft’s Active Directory Services provides an LDAP interface.
- Unlike X.500, LDAP has been widely adopted, particularly for intranet directory services.
- It provides secure access to directory data through authentication.