7 Names and Naming Services

7.1 Main Points

- Use of names
- Structure of names
- Name Services
- Domain Name Service (DNS) - The Internet Name Service

Definitions:-

Names - what its called

Address - where it is

Route - how to get there

7.2 Why names?

Object Identification a service or resource we want to use, eg a filename, a telecommunications provider, a person

Allow Sharing Communicating processes can pass names and thus share resources

Location Independence If we separate the name from the address, can migrate object transparently

Security If large number of possible names, knowing the name of the object means that it must explicitly have been passed. If entire system constructed so that names are passed with authorisation then knowing name means chain of trust to allow access to object.

7.3 What does one do with names?

- Use as arguments to functions, eg to call a service
- Create names. If an object comes into creation it must be named. Name should meet rules for system, eg be unique, therefore naming authority (possibly object) must keep track of what names it has allocated
- Delete names. When an object disappears from the system, at some point may wish to delete name and allow re-use.

7.4 What's a name?

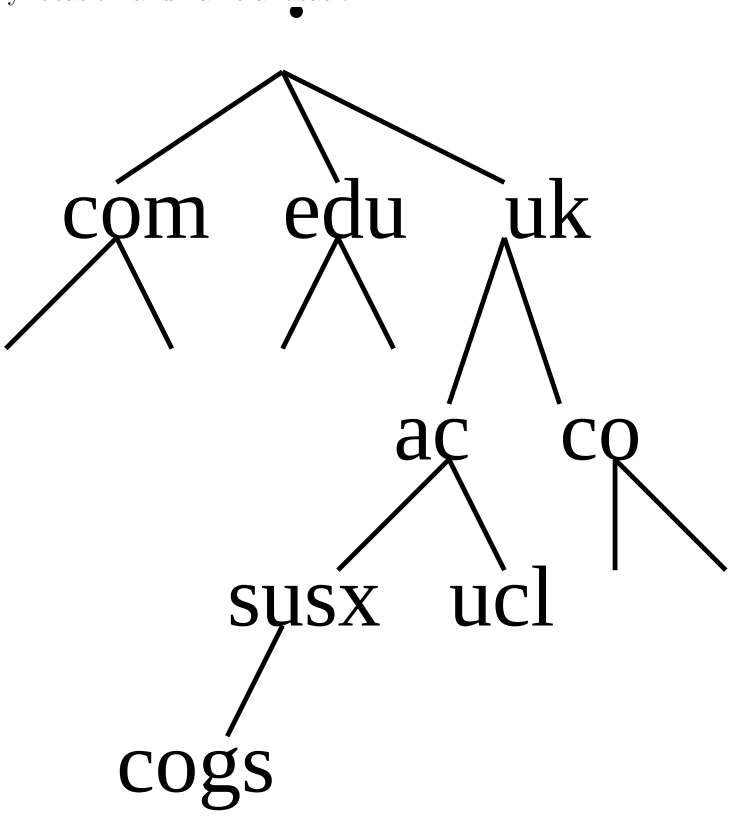
- Part of the design space for any distributed system is how to construct the name.
- Choice of design has ramifications for design of rest of system and performance of service.

7.4.1 Unique Identifier

- **Unique Identifier** (UID) aka flat names, primitive names.
- No internal structure, just string of bits.
- Only use is comparison against other UIDs eg for lookup in table containing information on named object.
- Provide location independence and uniformity
- *But* Difficult to name different versions of the same object eg Distributed systems edition 1 vs Distributed Systems edition 2. Real objects have relationships to each other - useful to reflect in naming practice
- Difficult to discover address from name - need to search entire name space

7.5 Partitioned names

Add partition structure to name to enable some function to be more efficient, typically location and name allocation



- Domain Name Service (DNS) name - tsunb.ctn.cogs.susx.ac.uk. Each part of name comes from flat name space. Division of name space and possible objects into smaller space. "uk" reduces objects to those within the uk, "ac" to those administered by academic networks, and so on.
- When allocating names, simply add to lowest part of partition. Low risk of collision, since small number of other objects, all administered by same authority
- When looking up name, can guess where information will reside.

7.6 Descriptive names

- Necessary to have a unique name.
- But useful to name objects in different ways, such as by service they offer eg Postman Pat, John of Gwent, www.cogs.susx.ac.uk.
- Create name using attributes of object.
- Note that objects can therefore have multiple names, not all of which are unique
- Choice of name structure depends on system. DNS chooses partition according to administration of creation, with aliases to allow naming by service eg ftp.cogs.susx.ac.uk is also doc-sun.crn.cogs.susx.ac.uk

7.7 Object Location from name - broadcast

- Ask all possible objects within the system if they respond to that name. Can be efficient if network supports broadcast eg Ethernet and other LANs.
- Equivalent to distributing name table across all objects, objects storing names referring to them.
- Only want positive responses - all responses would generate a lot of traffic.
- Scaling problem when move into wide area.
  - Greater number of hosts imply greater probability of failure
  - Broadcasts consume higher proportion of bandwidth, made worse due to higher failures needing more location requests
- Broadcast used only on small LAN based systems (and in initial location of directory)

7.8 Location through database

- Keep information about object in a table, indexed by a name. Location is just another piece of information.
- In DNS, table is stored as {name,attribute} pairing, in a *resource record*
- If database centralised, then
  - Whole system fails if database machine fails
  - Database machine acts as bottleneck for performance of system as whole
  - In wide area systems, authority should be shared amongst controlling organisations

  So name information usually distributed.

7.9 Distributed Name Servers

Parts of the name table are distributed to different servers eg. in Domain Name Service, servers are allocated portions of the name space below certain domains such as the root, ac.uk, susx.ac.uk

Names can be partitioned between servers based on

algorithmic clustering eg apply well-known hash function on name to map to server. May result in server being remote from object. Only technique for UIDs

structural clustering if name is structured, use structure to designate names to particular server, such as in DNS

attribute clustering if names are constructed using attributes, then servers take responsibility for certain attributes.

7.10 Availability and performance

If a single server is responsible for name space, there is still single point of failure, and a performance bottleneck.

Most systems therefore

- Replicate name list to other servers. Also increases performance for heavily accessed parts of name space eg secondary servers in DNS
- Cache information received by lookup. No need to repeat lookup if asked for same information again. Increases performance. Implemented in both client side and server (in recursive calls) in DNS.

If information is cached, how do we know when its invalid? May attempt to use inconsistent information.

7.11 Maintaining consistency for distributed name services

Alleviated by following features of some distributed systems

- In most systems objects change slowly, so names live for a long time, and are created infrequently
- If address of an object is wrong, it causes an error. Address user can recover if it assumes one of the possible problems is inconsistent information.
- Obsolete information can be fixed by addressed object leaving redirection pointers. Equivalent to leaving a forwarding address to new home.

However, there are always systems which break these assumptions eg highly dynamic distributed object system, creating lots of objects and names and deleting lots of names.

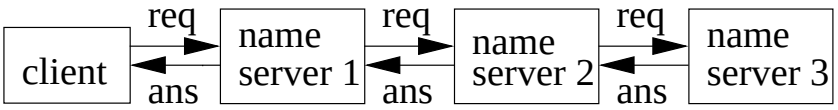
7.12 Client and Name server interaction.

- Hidden behind RPC interface.
- Client knows of server to ask (either installed in file, or through broadcast location).
- Client calls with arguments of name and required attribute, eg address. In DNS arguments are name and the type of requested attribute.
- Server will return result with either required attribute or error message

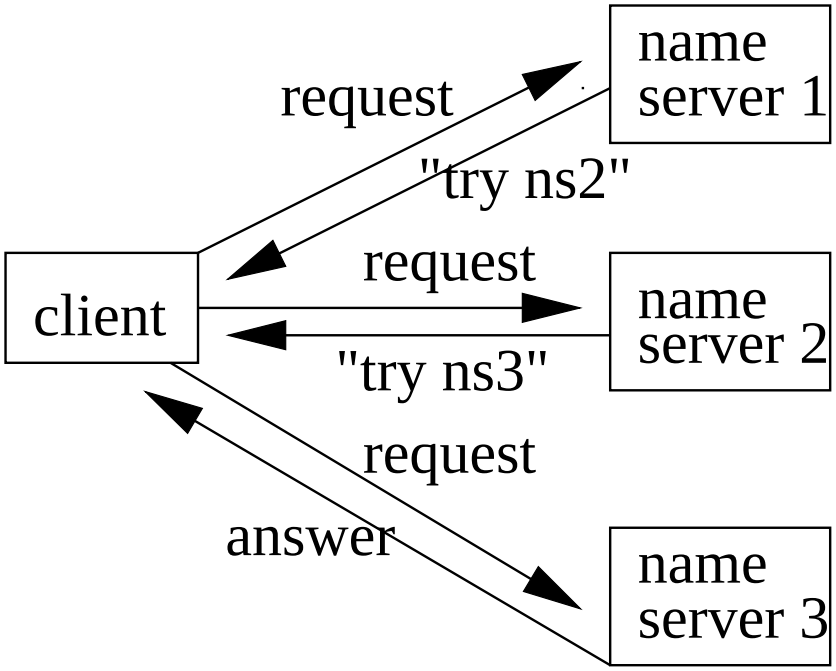
7.12.1 Lookup Modes

If name not stored on server, may be stored on other server. Two options

Recursive server asks other possible server about name and attribute, which may then have to ask another server and so on



Iterative server returns address of other possible server to client, who then resends request to new server.



7.13 Summary

- Names can be flat, or they can be structured
- Centralised name servers suffer from availability - distributed name servers suffer from inconsistency
- Interaction with name server best modelled by RPC