
# Distributed Systems: Concepts and Design

# Chapter 13 Exercise Solutions

---

13.1 Describe the names (including identifiers) and attributes used in a distributed file service such as NFS (see Chapter 12).

*13.1 Ans.*

Names: hierarchical, textual file names; local identifiers of open files (file descriptors); file handles; file system identifiers; inode numbers; textual hostnames; IP addresses; port numbers; physical network addresses; user and group identifiers; disk block numbers; physical disk addresses.

Attributes: file metadata, including physical storage information and user-visible file attributes.

See Section 12.3 for more information about NFS.

---

13.2 Discuss the problems raised by the use of aliases in a name service, and indicate how, if at all, these may be overcome.

*13.2 Ans.*

Firstly, if aliases are private and not publicly defined, then there is a risk of misunderstanding through a user referring to an object using an alias. The alias might refer to an (incorrect) object in the other user's name space.

The second problem with aliases is that they may introduce cycles into the naming graph. For example, a name */users/fred* can in principle be made an alias for */users*. A resolver will potentially cycle infinitely in attempting to resolve this name. A solution is to place a limit on the number of aliases that a resolver is prepared to encounter when it resolves a name.

---

13.3 Explain why iterative navigation is necessary in a name service in which different name spaces are partially integrated, such as the file naming scheme provided by NFS.

*13.3 Ans.*

The reason why iterative navigation is necessary is that when a client encounters a symbolic link, then this symbolic link should be resolved with respect to the client's name space, even when the server stores the link. For example, suppose that the server's directory */jewel* is mounted on the client's directory */ruby/red*. Suppose that */ruby/red/stone* (stored in the server's name space as */jewel/stone*) is a symbolic link to */ruby/stone*. The server passes back this link to the client, which then must continue to resolve it. The pathname */ruby/stone* might refer to a file stored at the client, or it might be a mount point to another server.

---

13.4 Describe the problem of unbound names in multicast navigation. What is implied by the installation of a server for responding to lookups of unbound names?

*13.4 Ans.*

In multicast navigation, a client multicasts a name to a group of servers for resolution. If a server can resolve the name, it replies to the client. To minimise messages, a server that cannot resolve the name does not respond. However if no server can resolve the name – the name is unbound – then the client will be greeted with silence.

It must re-send the request, in case it was dropped. The client cannot distinguish this case from that of the failure of a server that can resolve the name.

A solution to this problem is to install a member of the group which keeps track of all bound names, but does not need to store the corresponding attributes. When a request is multicast to the group, this server looks for the name in its list of bound names. If the name appears in the list, it does nothing. If, however, the name is not in its list, then it sends a 'name unbound' response message to the client. The implication is that this special server must be notified whenever a client binds or unbinds a name, increasing the overheads for these operations.

---

13.5 How does caching help a name service's availability?

*13.5 Ans.*

Clients cache both object attributes and the addresses of servers that store directories. This helps the service's availability because the client may still access cached attributes even if the server that stores them crashes (although the attributes may have become stale). And if, for example, the server that stores the director */emerald* has crashed, a client can still look up the object */emerald/green/stone* if it has cached the location of the directory */emerald/green*.

---

13.6 Discuss the absence of a syntactic distinction (such as use of a final '.') between absolute and relative names in DNS.

*13.6 Ans.*

DNS servers only accept complete domain names without a final '.', such as *dcs.qmul.ac.uk*. Such names are referred to the DNS root, and in that sense are absolute. However, resolvers are configured with a list of domain names which they append to client-supplied names, called a domain suffix list. For example, when supplied with a name *fred* in the department of Computer Science at Queen Mary and Westfield College, a resolver appends *.dcs.qmul.ac.uk* to get *fred.dcs.qmul.ac.uk*, which it then submits to a server. If this should be unbound, the resolver tries *fred.qmul.ac.uk*. Eventually, if necessary, the resolver will submit the name *fred* to a server. Some resolvers accept a final after a domain name. This signifies *to the resolver* that the name is to be sent directly to the server as it is (but stripped of its final '.'); the final '.' is not acceptable domain name syntax.

In practice the lack of syntactic distinction between relative names (*fred*) and absolute names (*fred.dcs.qmul.ac.uk*) is not a problem because of the conventions governing first-level domain names. No-one uses single-component names referred to the root (such as *gov*, *edu*, *uk*), so a single-component name is always relative to some subdomain. In principle, a multi-component name such as *ac.uk* uttered in the domain *elvis.edu* could refer to a (bound) domain *ac.uk.elvis.edu*, but normally organisations neither need to nor want to install such confusing names in their subdomains.

An advantage to the lack of syntactic distinction between absolute and relative names is that the DNS name space could, in principle, be reconfigured. We could, for example, transform *edu*, *gov*, *com* etc. into *edu.us*, *gov.us*, *com.us* etc. and still correctly resolve names such as *purdue.edu* in the USA by configuring all resolvers in the USA to include *.us* in their domain suffix list.

---

13.7 Investigate your local configuration of DNS domains and servers. You may find a program such as *nslookup* installed on UNIX systems, which enables you to carry out individual name server queries.

*13.7 Ans.*

Left to the reader.

---

13.8 Why do DNS root servers hold entries for two-level names such as *yahoo.com* and *purdue.edu*, rather than one-level names such as edu and *com*?

*13.8 Ans.*

First-level domain names such as *edu* and *com* refer to abstract categories of organizations and administrative units, and do not refer to any actual body. Second-level domain names such as *yhaoo.com* and *purdue.edu* are

not so many in number as to need to be divided between separate *com* and *edu* servers. Such a division would bring extra complexity and overheads. Although uk etc. do have separate servers.

---

13.9 Which other name server addresses do DNS name servers hold by default, and why?

*13.9 Ans.*

A DNS name server holds the addresses of one or more root servers, so that all parts of the name space can be reached. It stores the addresses of servers storing subdomains (thus a server for *qmul.ac.uk* stores addresses of servers for *dcs.qmul.ac.uk*). Thirdly, it is often convenient for it to store the addresses of servers storing its parent domain (thus a server in *dcs.qmul.ac.uk* knows the addresses of servers storing *qmul.ac.uk*).

---

13.10 Why might a DNS client choose recursive navigation rather than iterative navigation? What is the relevance of the recursive navigation option to concurrency within a name server?

*13.10 Ans.*

A DNS client may choose recursive navigation simply because it is too basic to perform iterative navigation. A server that performs recursive navigation must await a reply from another server before replying to the client. It is preferable for a server to deal with several outstanding client requests at one time rather than holding off other requests until each one is completed, so that clients are not unduly held up. The server will, in general, refer resolution to several other servers rather than just one, so client requests will be satisfied in parallel to some extent.

---

13.11 When might a DNS server provide multiple answers to a single name lookup, and why?

*13.11 Ans.*

A DNS server provides several answers to a single name lookup whenever it possesses them, assuming that the client has requested multiple answers. For example, the server might know the addresses of several mail servers or DNS servers for a given domain. Handing back all these addresses increase the availability of the mail service and DNS respectively.

---

13.12 GNS does not guarantee that all copies of entries in the naming database are up-to-date. How are clients of GNS likely to become aware that they have been given an out-of-date entry? Under what circumstances might it be harmful?

*13.12 Ans.*

Clients will become aware of the use of an out-of-date entry if a name that they have obtain is no longer a valid communication identifier (such as for example when a user's email address has changed and no forwarding address exists). This is not normally harmful, since the client can recover gracefully by making a delayed request to GNS. However, it may be harmful if the communication identifier obtained from GNS provides access to some protected resource, and the name used to obtain it should no longer be bound to that resource. For example, when a user ceases to be a member of the organisation, GNS may continue to supply information about his or her organisational role, leading users or applications to accord privileges to the user.

---

13.13 Discuss the potential advantages and drawbacks in the use of an X.500 directory service in place of DNS and the Internet mail delivery programs. Sketch the design of a mail delivery system for an internetwork in which all mail users and mail hosts are registered in an X.500 database.

*13.13 Ans.*

For access to conventional email addresses (based on Internet Domain Names), X.500 would provide a similar facilities to the DNS service. X.500 is designed to be scalable. If this is achieved in practice, then it should meet the future needs of large-scale networking better than DNS.

The main advantage of X.500 is that it is an attribute-based directory service. In principle, users could address messages to people by quoting their real names and their organisational affiliations, instead of the Domain Name based addresses currently used. The mail system would make a *search* request of X.500 to find the corresponding DNS or other network address of the user's mailbox. A drawback is that searching with a wide scope is quite slow and costly in computing resources, the scope could be limited by the use of the

organisational affiliation. Several alternate mailboxes could be held in the directory server, providing fault-tolerant mail delivery.

---

13.14 What security issues are liable to be relevant to a directory service such as X500 operating within an organization such as a university?

*13.14 Ans.*

There are two main security issues of relevance to a name service. The first is the question of who may create and modify an entry for a given name. It is important that malicious users cannot create bogus entries or alter stored attributes without permission. It is equally important that administrative boundaries in the name space are respected.

The second issue is privacy. In general, users may want only privileged principals to read their attributes.