

## **DNS—The Domain Name System**

Although programs theoretically could refer to hosts, mailboxes, and other resources by their network (e.g., IP) addresses, these addresses are hard for people to remember. Also, sending e-mail to tana@128.111.24.41 means that if Tana's ISP or organization moves the mail server to a different machine with a different IP address, her e-mail address has to change.

Way back in the ARPANET, there was simply a file, hosts.txt, that listed all the hosts and their IP addresses. Every night, all the hosts would fetch it from the site at which it was maintained. For a network of a few hundred large timesharing machines, this approach worked reasonably well.

### **The DNS Name Space**

the Internet is divided into over 200 top-level domains, where each domain covers many hosts. Each domain is partitioned into subdomains, and these are further partitioned, and so on. All these domains can be represented by a tree, as shown in [1](#). The leaves of the tree represent domains that have no subdomains (but do contain machines, of course). A leaf domain may contain a

single host, or it may represent a company and contain thousands of hosts.

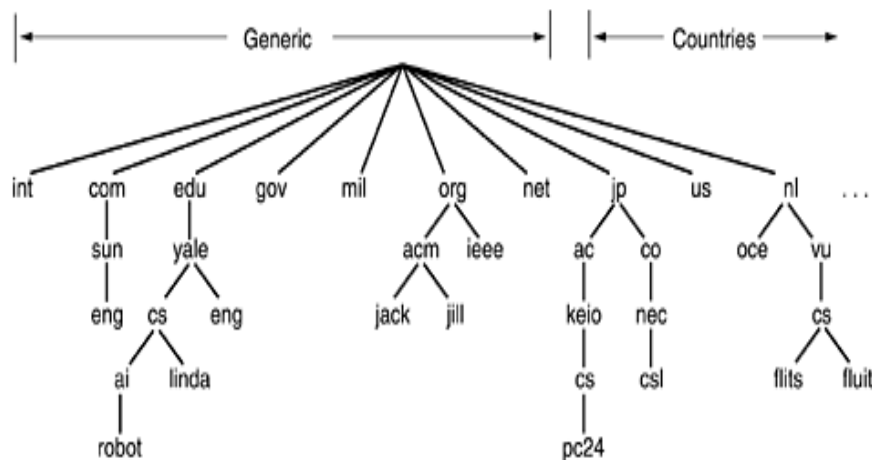


Fig 1A portion of the Internet domain name space.

The top-level domains come in two flavors:

- **Generic:** The original generic domains were com (commercial), edu (educational institutions), gov (the U.S. Federal Government), int (certain international organizations), mil (the U.S. armed forces), net (network providers), and org (nonprofit organizations).
- **Countries:** The country domains include one entry for every country, as defined in ISO 3166.

In November 2000, ICANN approved four new, general-purpose, top-level domains, namely, biz (businesses), info

(information), name (people's names), and pro (professions, such as doctors and lawyers). In addition, three more specialized top-level domains were introduced at the request of certain industries. These are aero (aerospace industry), coop (co-operatives), and museum (museums). Other top-level domains will be added in the future.

- ❖ Each domain is named by the path upward from it to the (unnamed) root. The components are separated by periods (pronounced "dot"). Thus, the engineering department at Sun Microsystems might be eng.sun.com., rather than a UNIX-style name such as /com/sun/eng. Notice that this hierarchical naming means that eng.sun.com. does not conflict with a potential use of eng in eng.yale.edu., which might be used by the Yale English department.
- ❖ Domain names are case insensitive, so edu, Edu, and EDU mean the same thing. Component names can be up to 63 characters long, and full path names must not exceed 255 characters.
- ❖ In principle, domains can be inserted into the tree in two different ways. For example, cs.yale.edu could equally well be listed under the us country domain

as `cs.yale.ct.us`. In practice, however, most organizations in the United States are under a generic domain, and most outside the United States are under the domain of their country. There is no rule against registering under two top-level domains, but few organizations except multinationals do it (e.g., `sony.com` and `sony.nl`).

1. `cs.yale.edu` (Yale University, in the United States)
2. `cs.vu.nl` (Vrije Universiteit, in The Netherlands)
3. `cs.keio.ac.jp` (Keio University, in Japan)

❖ To create a new domain, permission is required of the domain in which it will be included. For example, if a VLSI group is started at Yale and wants to be known as `vlsi.cs.yale.edu`, it has to get permission from whoever manages `cs.yale.edu`. Similarly, if a new university is chartered, say, the University of Northern South Dakota, it must ask the manager of the `edu` domain to assign it `unsd.edu`. In this way, name conflicts are avoided and each domain can keep track of all its subdomains. Once a new domain has been created and registered, it can create subdomains, such

as cs.unsd.edu, without getting permission from anybody higher up the tree.

- ❖ Naming follows organizational boundaries, not physical networks. For example, if the computer science and electrical engineering departments are located in the same building and share the same LAN, they can nevertheless have distinct domains. Similarly, even if computer science is split over Babbage Hall and Turing Hall, the hosts in both buildings will normally belong to the same domain.

## Resource Records

- ❖ Every domain, whether it is a single host or a top-level domain, can have a set of resource records associated with it. For a single host, the most common resource record is just its IP address, but many other kinds of resource records also exist. When a resolver gives a domain name to DNS, what it gets back are the resource records associated with that name. Thus, the primary function of DNS is to map domain names onto resource records.

A resource record is a five-tuple. Although they are encoded in binary for efficiency, in most expositions, resource records are presented as ASCII text, one line per resource record. The format we will use is as follows:

Domain_name	Time_to_live	Class	Type	Value
-------------	--------------	-------	------	-------

## Name Server

In theory at least, a single name server could contain the entire DNS database and respond to all queries about it. In practice, this server would be so overloaded as to be useless. Furthermore, if it ever went down, the entire Internet would be crippled.

To avoid the problems associated with having only a single source of information, the DNS name space is divided into nonoverlapping zones. One possible way to divide the name space of [Fig. 1](#) is shown in [Fig. 2](#). Each zone contains some part of the tree and also contains name servers holding the information about that zone. Normally, a zone will have one primary name server, which gets its information from a file on its disk, and one or more secondary name servers, which get their information from the primary name server. To improve reliability, some servers for a zone can be located outside the zone.

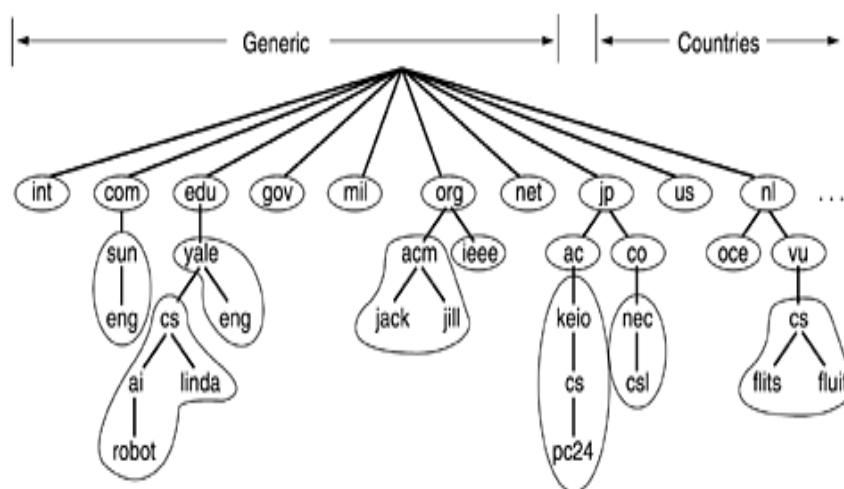


Fig2 Part of the DNS name space showing the division into zones

Where the zone boundaries are placed within a zone is up to that zone's administrator. This decision is made in large part based on how many name servers are desired, and where. For example, in [Fig. 2](#), Yale has a server for yale.edu that handles eng.yale.edu but not cs.yale.edu, which is a separate zone with its own name servers. Such a decision might be made when a department such as English does not wish to run its own name server, but a department such as computer science does. Consequently, cs.yale.edu is a separate zone but eng.yale.edu is not.

When a resolver has a query about a domain name, it passes the query to one of the local name servers. If the domain being sought falls under the jurisdiction of the name server, such as ai.cs.yale.edu falling under cs.yale.edu, it returns the authoritative resource records. An authoritative record is one that comes from the authority that manages the record and is thus always correct. Authoritative records are in contrast to cached records, which may be out of date.

If, however, the domain is remote and no information about the requested domain is available locally, the name server sends a query message to the top-level name server for the domain requested. To make this process clearer, consider the example of [Fig. 3](#). Here, a resolver

on flits.cs.vu.nl wants to know the IP address of the host linda.cs.yale.edu. In step 1, it sends a query to the local name server, cs.vu.nl. This query contains the domain name sought, the type (A) and the class (IN).

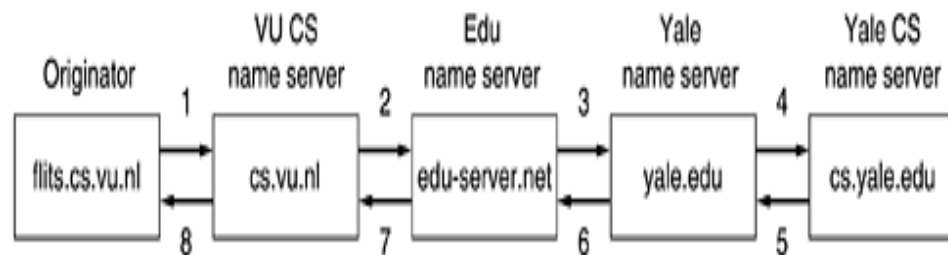


Figure 3. How a resolver looks up a remote name in eight steps.

Let us suppose the local name server has never had a query for this domain before and knows nothing about it. It may ask a few other nearby name servers, but if none of them know, it sends a UDP packet to the server for edu given in its database (see [Fig. 3](#)), edu-server.net. It is unlikely that this server knows the address of linda.cs.yale.edu, and probably does not know cs.yale.edu either, but it must know all of its own children, so it forwards the request to the name server for yale.edu (step 3). In turn, this one forwards the request to cs.yale.edu (step 4), which must have the authoritative resource records. Since each request is from a client to a server, the resource record requested works its way back in steps 5 through 8.

