DNS

When DNS was not into existence, one had to download a Host file containing host names and their corresponding IP address. But with increase in number of hosts of internet, the size of host file also increased. This resulted in increased traffic on downloading this file. To solve this problem the DNS system was introduced.

Domain Name System helps to resolve the host name to an address. It uses a hierarchical naming scheme and distributed database of IP addresses and associated names. To map a name onto an IP address, an application program calls a library procedure called the **resolver**, passing it the name as a parameter.

IP address is a unique logical address assigned to a machine over the network. An IP address exhibits the following properties:

- IP address is the unique address assigned to each host present on Internet.
- IP address is 32 bits (4 bytes) long.
- IP address consists of two components: network component and host component.
- Each of the 4 bytes is represented by a number from 0 to 255, separated with dots. For example 137.170.4.124

IP address is 32-bit number while on the other hand domain names are easy to remember names. For example, when we enter an email address we always enter a symbolic string such as webmaster@gmail.com

The Domain name system comprises of Domain Names, Domain Name Space, Name Server that have been described below:

Domain Name Space

The domain name space refers a hierarchy in the internet naming structure. This hierarchy has multiple levels (from 0 to 127), with a root at the top

Domain Name is a symbolic string associated with an IP address. There are several domain names available; some of them are generic such as com, edu, gov, net etc, while some country level domain names such as au,

in, za, us etc. The top-level domains come in two flavors: generic and countries.

. The following diagram shows the domain name space hierarchy:

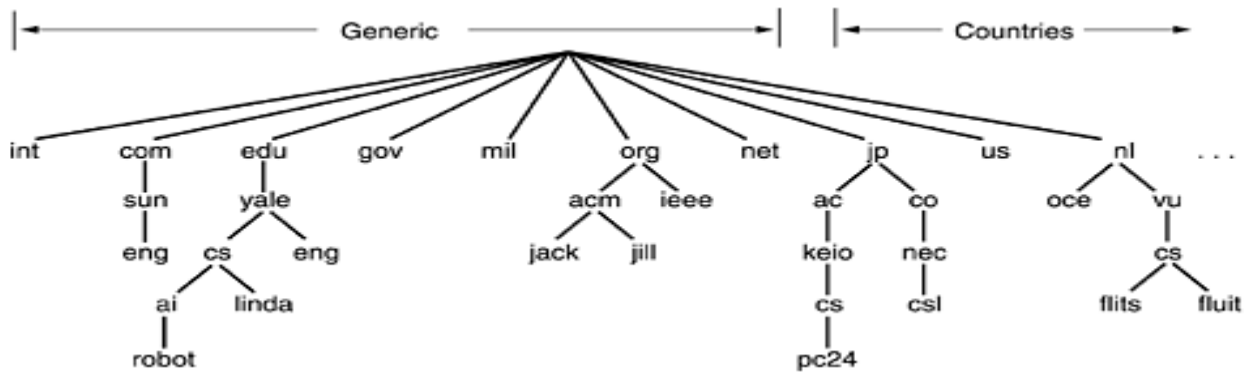


Figure 1 A portion of the Internet domain name space

The following table shows the Generic Top-Level Domain names:

Domain Name	Meaning
Com	Commercial business
Edu	Education
Gov	U.S. government agency
Int	International entity
Mil	U.S. military
Net	Networking organization
Org	Non profit organization

The following table shows the Country top-level domain names:

Domain Name	Meaning
au	Australia
in	India
cl	Chile
fr	France
us	United States
za	South Africa
uk	United Kingdom
jp	Japan
es	Spain
de	Germany
ca	Canada
ee	Estonia
hk	Hong Kong

In general, getting a second-level domain, such as *name-of-company*.com, is easy. It merely requires going to a registrar for the corresponding top-level domain (*com* in this case) to check if the desired name is available and not somebody else's trademark. If there are no problems, the requester pays a small annual fee and gets the name. By now, virtually every common word in English has been taken in the *com* domain. Try household articles, animals, plants, body parts, etc. Nearly all are taken. 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 can be either absolute or relative. An absolute domain name always ends with a period (e.g., *eng.sun.com.*), whereas a relative one does not. Relative names have to be interpreted in some context to uniquely determine their true meaning. In both cases, a named

domain refers to a specific node in the tree and all the nodes under it. 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.

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 is

Domain_name Time_to_live Class Type Value

The ***Domain_name*** tells the domain to which this record applies. Normally, many records exist for each domain and each copy of the database holds information about multiple domains. This field is thus the primary search key used to satisfy queries. The order of the records in the database is not significant.

The ***Time_to_live*** field gives an indication of how stable the record is. Information that is highly stable is assigned a large value, such as 86400 (the number of seconds in 1 day). Information that is highly volatile is assigned a small value, such as 60 (1 minute).

The third field is the ***Class***. For Internet information, it is always *IN*. For non-Internet information, other codes are used.

The ***Type*** field tells what kind of record this is. The most important types are listed below.

Type	Meaning	Value
SOA	Start of Authority	Parameters for this zone
A	IP address of a host	32-Bit integer
MX	Mail exchange	Priority, domain willing to accept e-mail
NS	Name Server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
HINFO	Host description	CPU and OS in ASCII
TXT	Text	Uninterpreted ASCII text

Figure 2 . The principal DNS resource record types for IPv4.

SOA

An *SOA* record provides the name of the primary source of information about the name server's zone (described below), the e-mail address of its administrator, a unique serial number, and various flags and timeouts.

A

The most important record type is the *A* (Address) record. It holds a 32-bit IP address for some host. Every Internet host must have at least one IP address so that other machines can communicate with it. Some hosts have two or more network connections, in which case they will have one type *A* resource record per network connection (and thus per IP address). DNS can be configured to cycle through these, returning the first record on the first request, the second record on the second request, and so on.

MX

The next most important record type is the *MX* record. It specifies the name of the host prepared to accept e-mail for the specified domain. It is used because not every machine is prepared to accept e-mail. If someone wants to send e-mail to, for example, *bill@microsoft.com*, the sending host needs to find a mail server at *microsoft.com* that is willing to accept e-mail. The *MX* record can provide this information.

NS

The *NS* records specify name servers. For example, every DNS database normally has an *NS* record for each of the top-level domains, so, for example, e-mail can be sent to distant parts of the naming tree.

CNAME

CNAME records allow aliases to be created. For example, a person familiar with Internet naming in general and wanting to send a message to someone whose login name is *paul* in the computer science department at M.I.T. might guess that *paul@cs.mit.edu* will work. Actually, this address will not work, because the domain for M.I.T.'s computer science department is *lcs.mit.edu*. However, as a service to people who do not know this, M.I.T. could create a *CNAME* entry to point people and programs in the right direction. An entry like this one might do the job:

```
cs.mit.edu        86400    IN    CNAME lcs.mit.edu
```

Like *CNAME*, *PTR* points to another name. However, unlike *CNAME*, which is really just a macro definition, *PTR* is a regular DNS datatype whose interpretation depends on the context in which it is found.
Always a name is associated with an IP address to allow lookups of the IP address and return the name of the corresponding machine. These are called **reverse lookups**.

HNIFO

HINFO records allow people to find out what kind of machine and operating system a domain corresponds to.

TXT

TXT records allow domains to identify themselves in arbitrary ways. Both of these record types i.e HNIFO and TXT are for user convenience. Neither is required, so programs cannot count on getting them (and probably cannot deal with them if they do get them).

Value field, can be a number, a domain name, or an ASCII string. The semantics depend on the record type.

An example of the kind of information one might find in the DNS database of a domain.This figure depicts part of a (semihypothetical) database for the *cs.vu.nl* domain

***Figure 3. A portion of a possible DNS database for* cs.vu.nl**

```
; Authoritative data for cs.vu.nl
cs.vu.nl.        86400   IN  SOA     star boss (9527,7200,7200,241920,86400)
cs.vu.nl.        86400   IN  TXT     "Divisie Wiskunde en Informatica."
cs.vu.nl.        86400   IN  TXT     "Vrije Universiteit Amsterdam."
cs.vu.nl.        86400   IN  MX      1 zephyr.cs.vu.nl.
cs.vu.nl.        86400   IN  MX      2 top.cs.vu.nl.

flits.cs.vu.nl.  86400   IN  HINFO   Sun Unix
flits.cs.vu.nl.  86400   IN  A       130.37.16.112
flits.cs.vu.nl.  86400   IN  A       192.31.231.165
flits.cs.vu.nl.  86400   IN  MX      1 flits.cs.vu.nl.
flits.cs.vu.nl.  86400   IN  MX      2 zephyr.cs.vu.nl.
flits.cs.vu.nl.  86400   IN  MX      3 top.cs.vu.nl.
www.cs.vu.nl.    86400   IN  CNAME   star.cs.vu.nl
ftp.cs.vu.nl.    86400   IN  CNAME   zephyr.cs.vu.nl

rowboat                  IN  A       130.37.56.201
                         IN  MX      1 rowboat
                         IN  MX      2 zephyr
                         IN  HINFO   Sun Unix

little-sister            IN  A       130.37.62.23
                         IN  HINFO   Mac MacOS

laserjet                 IN  A       192.31.231.216
                         IN  HINFO   "HP Laserjet IIISi" Proprietary
```

The first line gives some basic information about the domain further. The next two lines give textual information about where the domain is located. Then come two entries giving the first and second places to try to deliver e- mail sent to *person@cs.vu.nl*. The *zephyr* (a specific machine) should be tried first. If that fails, the *top* should be tried as the next choice.

After the blank line, added for readability, come lines telling that the *flits* is a Sun workstation running UNIX and giving both of its IP addresses. Then three choices are given for handling e- mail sent to *flits.cs.vu.nl*. First choice is naturally the *flits* itself, but if it is down, the *zephyr* and *top* are the second and third choices. Next comes an alias, *www.cs.vu.nl*, so that this address can be used without designating a specific machine. Creating this alias allows *cs.vu.nl* to change its World Wide Web server without invalidating the address people use to get to it. A similar argument holds for *ftp.cs.vu.nl*.

The next four lines contain a typical entry for a workstation, in this case, *rowboat.cs.vu.nl*. The information provided contains the IP address, the primary and secondary mail drops, and information about the machine. Then comes an entry for a non-UNIX system that is not capable of receiving mail itself, followed by an entry for a laser printer that is connected to the Internet.

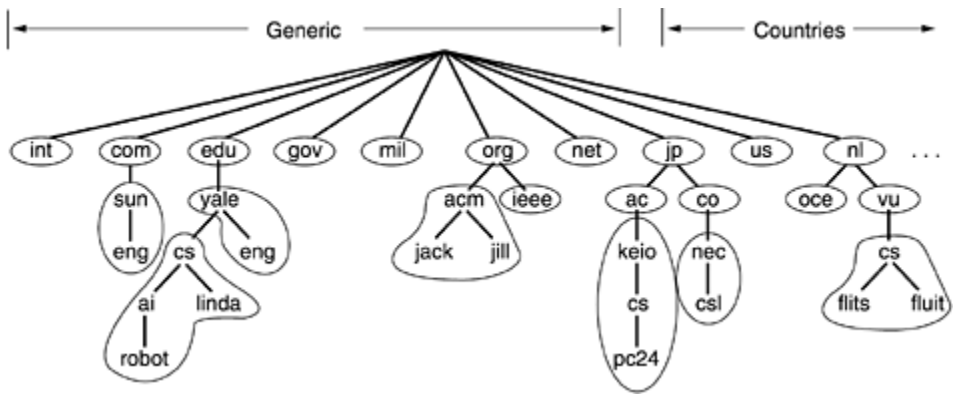
The IP addresses that are used to look up the top-level domains are not shown. These are needed to look up distant hosts, but since they are not part of the *cs.vu.nl* domain, they are not in this file. They are supplied by the root servers, whose IP addresses are present in a system configuration file and loaded into the DNS cache when the DNS server is booted. There are about a dozen root servers spread around the world, and each one knows the IP addresses of all the top-level domain servers. Thus, if a machine knows the IP address of at least one root server, it can look up any DNS name.

Name Servers

A single name server contains the entire DNS database and respond to all queries about it. This server would be so overloaded as to be useless.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 non overlapping **zones**. One possible way to divide the name space is shown in the figure. 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.

Figure 4. 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, 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 one example. 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 5. 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, *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.

Once these records get back to the *cs.vu.nl* name server, they will be entered into a cache there, in case they are needed later. However, this information is not authoritative, since changes made at *cs.yale.edu* will not be propagated to all the caches in the world that may know about it. For this reason, cache entries should not live too long. This is the reason that the *Time_to_live* field is included in each resource record. It tells remote name servers how long to cache records. If a certain machine has had the same IP address for years, it may be safe to cache that information for 1 day. For more volatile information, it might be safer to purge the records after a few seconds or a minute.

The query method described here is known as a **recursive query**, since each server that does not have the requested information goes and finds it somewhere, then reports back. An alternative form is also possible. In this form, when a query cannot be satisfied locally, the query fails, but the name of the next server along the line to try is returned. Some servers do not implement recursive queries and always return the name of the next server to try.

When a DNS client fails to get a response before its timer goes off, it normally will try another server next time. The assumption here is that the server is probably down, rather than that the request or reply got lost.

While DNS is extremely important to the correct functioning of the Internet, all it really does is map symbolic names for machines onto their IP addresses. It does not help locate people, resources, services, or objects in general. For locating these things, another directory service has been defined, called **LDAP** (**Lightweight Directory Access Protocol**). It is a simplified version of the OSI X.500 directory service and is described in RFC 2251.