The Domain Name System:
An Integral Part of the Internet

By Keiko Ishioka

The Domain Name System (otherwise known as the Domain Name Server system) (DNS) is a distributed database that is accessed by anyone who connects to the Internet or uses an email address. Although most users may be unaware that they are actually implementing the DNS system each time they connect to the Internet, they in fact are. Computers do not understand domain names, but they do understand IP addresses. Thus, each time a user navigates to a web page or checks an email account, they are accessing the DNS database. Without its existence, there would be a chaotic process of mapping hostnames to their proper IP addresses, and vice versa. Because of this, it is important for a system administrator to have an understanding of the Domain Name System when dealing with the Internet.

Before the Internet evolved into its present state, there was the ARPAnet, which began in the late 1960s, funded by the U.S. Department of Defense's (DOD) Advanced Research Projects Agency (ARPA). The ARPAnet was originally developed to enable the sharing of computing resources among various government contractors (Albitz & Liu 1). To achieve this goal, the concept of a network emerged so that these resources could be exchanged between two computers that were physically located in separate rooms, or buildings. After the ARPAnet project was cancelled by the Department of Defense in 1988, the National Science Foundation funded a new network named the NSFNET (2). In 1995, the Internet made a transition from being a publicly funded network to a collection of networks offered by commercial long-distance carriers, for example, MCI and Sprint, and long-time commercial internetworks, for example, PSINet and UUNET (2). Presently, the Internet is extremely popular among anyone of any age, including those that may not even be familiar with its actual implementation. Due to the large amount of hosts that have emerged with it, the Domain Name System was developed to satisfy the demands of mapping a hostname to an IP address (called a "forward mapping") and also, mapping an IP address back to its hostname (called a "reverse mapping") in an efficient manner.

In order for the ARPAnet to operate properly, a database was needed to map a hostname, or rather, a domain name (created for the ease of human readability) to an IP address (which consist of a set of four numbers, each separated by a dot, in the range of 0 to 255), a language that the computer understands. The first implementation of this database was addressed with the use of a file called *HOSTS.TXT*. This one file contained all the necessary information needed about hosts that were connected to the ARPAnet (3). (The Unix host table, */etc/hosts*, was actually derived from *HOSTS.TXT* (3)). At that time, SRI's *Network Information Center* (referred to as "the NIC") maintained the *HOSTS.TXT* file, which was distributed from SRI-NIC, a single host (3).

But as the ARPAnet increased in popularity and grew significantly in size, the previous method of emailing a change to the NIC, then FTPing to SRI-NIC to access the *HOSTS.TXT* file became extremely difficult to manage and maintain (3). This was because an entry for each host that connected to the ARPAnet was listed on a line in the file. Eventually, the issues of "traffic and load," "name collisions" and "consistency" emerged (3). "Traffic and load" was a problem because of the increasing amount of network users. As more people connected to the network, traffic became heavier and as a consequence, the load time increased. Similarly, as the number of hosts that connected to the network increased in size, it became more difficult to ensure that there were no identical names in the *HOSTS.TXT* file. Keeping the file consistent became more challenging as more hostnames were created and added to the ARPAnet.

A change was needed, and thus, Paul Mockapetris, of the USC's Information Sciences Institute, developed the idea of the Domain Name System in 1984 when he described it in his release of RFCs 882 and 883 (4). JEEVES, also written by Paul Mockapetris, became the first implementation of the Domain Name System. Later, in the late 1980s (Nemeth et al. 402), Kevin Dunlap developed BIND (Berkeley Internet Name Domain) (Albitz & Liu 9). BIND is currently the most popular implementation of the Domain Name System on Unix and Windows NT systems (Nemeth et al. 402). It is currently maintained by the Internet Software Consortium (ISC) (402).

A *domain name space* is similar to the hierarchical structure of the Unix filesystem. Correspondingly, a *domain name* is like an "absolute pathname" (Albitz & Liu 11). But instead of representing the root with a forward slash ("/"), like in the Unix filesystem, a dot (".") is used to represent the DNS's root (11). Similarly, a *fully qualified domain name* (FQDN) is like the Unix filesystem's representation of an absolute pathname, but the difference is that the root of the DNS system is located at the end of the domain name, whereas in a Unix filesystem, the root is located at the beginning of an absolute pathname (12). Although the DNS system actually interprets a fully qualified domain name with a dot at the end, for example, "admiral.umsl.edu.", this implementation is normally hidden from the user. In fact, in some cases, if the user attempts to use the final dot on certain systems, it will cause the system to crash. But, from the DNS system's standpoint, domains that do not end with a dot are interpreted as a relative address (Nemeth et al. 399).

In contrast to the Unix filesystem, there is a restriction of 127 levels in the Domain Name System's tree (Albitz & Liu 12). Common names of the root's children (referred to as a top-level, root-level, or a first-level domain) are called *arpa* (for ARPAnet hosts), *com* (for commercial organizations), *edu* (for accredited educational organizations), *gov* (for government organizations), *mil* (for military organizations), *net* (originally for organizations that provided a network infrastructure), *org* (for noncommercial organizations) and *int* (for international organizations) (17-18). Country-coded top-level domains (ccTLDs), represented by a two-letter ISO code, are also now commonly used for domains that are created outside of the United States, for example, *ca* for Canada and *fr* for France (Nemeth et al. 398). Although the top-level domain names were originally used to represent different types of organizations, certain ones, such as *com* and *net*, are now known to not necessarily follow their original conventions.

Note that most of the top-level domains pertained to primarily U.S. organizations. This is due to the fact that the Internet began as a small U.S.-funded research project, the ARPAnet (Albitz & Liu 18). At that time, no one anticipated that the Internet would become such a huge success and would become as popular as it has become today. Presently, the original top-level domains are now called *generic top-level domains*, or gTLDs (18). In addition, a few more TLD's (top-level domains) have been created since then, including *biz* (for businesses), *coop* (for cooperatives), *info* (for informational services), *museum* (for museums), *name* (for individuals), and *pro* (for licensed professionals) to satisfy the rapid growth of the Internet (18). The Internet Corporation for Assigned Names and Numbers (ICANN) is currently responsible for the management and control of the Internet's domain names (18). Information on ICANN's work and new TLDs can be found at *http://www.icann.org*.

To begin, some type of software is needed to implement the DNS system. Today, BIND is the most popular software available on Unix and Windows NT machines. Because the DNS protocol is now standardized, data can be shared between Unix and non-Unix DNS implementations (Nemeth et al. 395). In fact, most sites provide DNS service for both types of systems by running BIND on Unix servers (395). The open source code for the most recent versions of BIND can be found in */isc/bind/src/cur/bind-8/bind-src.tar.gz* (for 8.2.3 version) or */isc/bind9/9.1.0/bind-9.1.0.tar.gz* (for 9.1.0 version) at *ftp.isc.org*, the URL for the Internet Software Consortium (Albitz & Liu 38). But normally, the software for BIND is included as a standard part of the package on most Unix-based operating systems (37). Instructions on how to install and compile BIND on a particular operating system is available in the sections of *src/INSTALL* (38). To determine the version of BIND that is currently installed on the operating system, issue the following command (Nemeth et al. 403), under the assumption that *admiral.umsl.edu* is the server being used:

```
admiral% dig @admiral.umsl.edu version.bind txt chaos

; <<>> DiG 9.2.1 <<>> @admiral.umsl.edu version.bind txt chaos
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45296
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;version.bind.                  CH      TXT

;; ANSWER SECTION:
version.bind.           0       CH      TXT     "9.2.1"

;; Query time: 27 msec
;; SERVER: 134.124.15.13#53(admiral.umsl.edu)
;; WHEN: Mon Nov 10 01:56:49 2003
;; MSG SIZE  rcvd: 48
```

Sometimes, the version number may be hidden. If this is the case, the BIND version can be located in the log files in */var/log* or its equivalence, depending on the system (403).

Once BIND is installed and compiled on the operating system, a second-level domain name can be chosen. Second-level domain names are registered through a registrar, which is a company, or an organization, that communicates between a customer and a registry. It is the registrar's responsibility to register the customer's domain name by submitting the customer's zone data files and contact information to the registry when a customer pays for the *registration* of a domain name. Basically, the registrar is in charge of supplying a top-level (also called first-level) domain name. In the beginning, there was only one company that could act as a registrar. This company was called Network Solutions, Inc. (currently, a VeriSign, Inc. company). At that time, under a contract with the National Science Foundation, all registrations of a subdomain of any generic top-level domain (*com, net, org*, and *edu*) were maintained by Network Solutions, Inc. (399). But, in June 1999, ICANN allowed other companies to assume the role of a registrar, and thus, introduced competition among the companies. Because of this, there are now many more accredited registrars, and thus, more options (including cost) for selecting a company to

register a domain name from. An updated list of ICANN's accredited registrars can be found at *http://www.icann.org/registrars/accredited-list.html*. Although the main TLDs can be registered from the list of the registrars found on ICANN's site, certain TLDs must be registered through a different organization. For example, the top-level domain, *edu*, can only be registered through Educause, a nonprofit association. The URL for registering *edu* can be found at http://www.educause.edu/edudomain/.

A *registry*, then, is an organization that is responsible for maintaining data files for a top-level domain. The data files contain information of the delegation of each subdomain for each top-level domain. Producing a subdomain is similar to creating a second-level domain name. The only difference is that a subdomain's authority is within a domain's organization, whereas, a second-level domain's authority is required to be authorized by an accredited registrar. A subdomain is required to have two hosts to act as servers for the domain, as does a second-level domain. But, it is the responsibility of the parent domain to check that its child domain's name servers are running properly to avoid a "lame delegation" (402). A lame delegation occurs when a user attempts to contact a host that is associated with a domain that is invalid (a "lame domain"). If this occurs, the name server will refuse the user's query. But because the Domain Name System will continue to query the host hundreds of times, the network traffic will increase, forcing extra work on the master and root servers (478).

A second-level domain name must be chosen carefully because they must be unique within a top-level's domain. But because there is a limited amount of top-level domains available, most of the common names have already been taken. In fact, there is also a common issue of "domain squatting," which means that a domain name was registered for the sole purpose of reselling it later to an interested party at a much higher price. Therefore, the first step in obtaining a domain name is to check if the domain name is already in use, or rather, taken by someone else. Domain names can consist of up to 63 characters in each subdomain's component and up to 255 characters in the complete name (400). Commands that can be used to query a domain name include **nslookup** and **dig**. **nslookup** is the older command that was most commonly used, but as the note in the following example suggests, is now deprecated. The commands are written in **bold**.

```
admiral% nslookup
Note:  nslookup is deprecated and may be removed from future releases.
Consider using the `dig' or `host' programs instead.  Run nslookup with
the `-sil[ent]' option to prevent this message from appearing.
> set type=any
> doesthisdomainexist.com
Server:         127.0.0.1
Address:        127.0.0.1#53

** server can't find doesthisdomainexist.com: NXDOMAIN
```

This example displays the output of a query on the domain name, *doesthisdomainexist.com*. The bottom line specifies that the domain name is an *NXDOMAIN*, which simply means that the domain name does not exist. But although the domain name does not exist, it does not necessarily mean that it is available for registering. To ensure that it is, attempt to register the domain name through a registrar. A domain name may be an NXDOMAIN but may not

currently be available for registration due to several reasons. For example, the domain name may currently be in a dispute or it may not have been flushed from the previous registrar's database yet.

The next example gives the result of a domain name that is currently registered, *umsl.edu*.

```
> umsl.edu
Server:         127.0.0.1
Address:        127.0.0.1#53

umsl.edu
        origin = epsilon3.umsl.edu
        mail addr = eckert.JINX.umsl.edu
        serial = 965980
        refresh = 3600
        retry = 900
        expire = 604800
        minimum = 86400
umsl.edu        mail exchanger = 0 stl-msx-nlb.umsl.edu.
umsl.edu        nameserver = epsilon3.umsl.edu.
umsl.edu        nameserver = admiral.umsl.edu.
umsl.edu        nameserver = jupiter.cc.umr.edu.
Name:   umsl.edu
Address: 134.124.1.234
```

As mentioned before, **nslookup** is now a deprecated command. Thus, another command, **dig**, is more popular to use for querying a domain name. The next two examples display the output of querying the same domain names that were used in the previous example using **dig**.

```
admiral% dig doesthisdomainexist.com

; <<>> DiG 9.2.1 <<>> doesthisdomainexist.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 50726
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;doesthisdomainexist.com.       IN      A

;; AUTHORITY SECTION:
com.                    9880    IN      SOA     a.gtld-servers.net. nstld.verisign-grs.com. 2003110800 1800 900
604800 86400

;; Query time: 10 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Nov  8 18:06:48 2003
;; MSG SIZE  rcvd: 114

admiral% dig umsl.edu

; <<>> DiG 9.2.1 <<>> umsl.edu
;; global options:  printcmd
;; Got answer:
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49867
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 2

;; QUESTION SECTION:
;umsl.edu.                IN      A

;; ANSWER SECTION:
umsl.edu.           86400   IN      A       134.124.1.234

;; AUTHORITY SECTION:
umsl.edu.           86400   IN      NS      epsilon3.umsl.edu.
umsl.edu.           86400   IN      NS      admiral.umsl.edu.
umsl.edu.           86400   IN      NS      jupiter.cc.umr.edu.

;; ADDITIONAL SECTION:
admiral.umsl.edu.       86400   IN      A       134.124.15.13
epsilon3.umsl.edu.      86400   IN      A       134.124.15.136

;; Query time: 3 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Nov  8 18:07:06 2003
;; MSG SIZE  rcvd: 148
```

Note that the status of the domain is explicitly given in both outputs when the **dig** command is used. Additionally, there is an *Answer Section* for the domain name that is currently connected the Internet. Within this section, the IP address of *umsl.edu* is given. On the other hand, three name servers are listed for the domain name in the *Authority Section*. These name servers (specifically, the master and slave servers) identify the zone for which it is authoritative (440).

After a domain name has been selected and registered, the next step is to ensure that the IP network or networks are also registered. It is not uncommon for a registrar to refuse delegation of a subdomain to name servers or networks that are not registered. (A *delegation* occurs when the authority of a subdomain is given to another organization, which, in turn, is then held responsible for maintaining that subdomain's data files (Albitz & Liu 20)). Originally, the InterNIC was the official source for all IP addresses. Today, Internet service providers (ISPs) have assumed the role of the InterNIC by allocating space from their own networks (52). If a network is with an ISP, it is the responsibility of the ISP to register it. Otherwise, the network registry, in accordance to the domain name's region, must be contacted to obtain an IP address space and to register the network. The American Registry of Internet Numbers (ARIN) is responsible for the western hemisphere. They can be found at *http://www.arin.net*. For Asia and the Pacific, the Asia Pacific Network Information Center (APNIC) can be found at *http://www.apnic.net*. For Europe, the RIPE Network Coordination Centre can be found at *http://www.ripe.net* (53-54). In addition, network registries offer a *whois* service to detect if a network is currently registered. The following URLs correspond to each registry's *whois* page: *http://www.arin.net/whois/index.html* for ARIN, *http://whois.apnic.net* for APNIC and *http://www.ripe.net/cgi-bin/whois* for RIPE. Registering a network is required before it is possible to set up the *in-addr.arpa* zones (54).

An *in-addr.arpa* zone is a special type of a top-level domain. "The *in-addr.arpa* domain was created to allow one set of software modules and one naming tree to map from IP addresses to

hostnames as well as from hostnames to IP addresses" (Nemeth et al. 442). This is required because a fully qualified hostname is interpreted from the right-hand side. For example, in the hostname, *admiral.umsl.edu*, *edu* is the parent domain of *umsl*. Similarly, *umsl.edu* is the parent domain of *admiral*. On the other hand, an IP address is interpreted with its most significant part on the left-hand side. For example, the host 13 is on a subnet of 15, which is on a part of the 134.124 network for the IP address 134.124.15.13 (442). The *in-addr.arpa* zone contains domains that are named like IP addresses, but with the bytes listed in the reverse order (442). Continuing from the previous example, 15.124.134.in-addr.arpa represents the zone for the 15 subnet. With this type of format, programs that authenticate an inbound network request, for example, **rlogind, sendmail,** and **syslogd**, can use the Domain Name System to perform a reverse mapping from an IP address to a hostname (443). The *PTR record* in the *in-addr.arpa* zone lists the IP address's corresponding hostname (442). It is important that the *A record* (which specifies the forward mapping from a hostname to an IP address) must match its corresponding PTR record to avoid an authentication failure (443).

A system administrator is responsible for maintaining the zone files, a set of text files that are kept in the Domain Name System's database. Parser commands and resource records (RRs) are the two types of entries found in the zone files (436). Parser commands are basically provided to enable the use of entering records with a shortcut method. On the other hand, resource records define an attribute for a domain name, such as an IP address. If a listed domain name does not end with a dot, which specifies that it is a fully qualified domain name, it is interpreted as a relative name. This results in appending the default domain and a dot to the end of the given domain name (437). For example, if "admiral.umsl.edu" is specified in the *name* field, a disastrous result of "admiral.umsl.edu.umsl.edu." would happen. The resource record's basic format is as follows:

[name] [ttl] class type data

The *name* field identifies the host or domain, the *ttl* (time to live) field specifies the amount of seconds a data item can be cached. The *class* field specifies the type of network. The *type* field specifies the type of DNS record (zone, basic, security or optional). And the *data* field consists of the actual data that corresponds to the record type (436-437). Special characters allowed in resource records are ";" for introducing a comment, "@" to represent the current domain name, "()" to allow the data to span multiple lines, and "*" (in the *name* field only) to represent a wild card (436). The resource records must be in the order of *SOA record*, which specifies the *authority* for the zone, *NS record*, which lists a *name server* for the zone, and *other records*, which contains data about the hosts in the zone. Some other records include *A*, a name-to-address mapping, *PTR*, an address-to-name mapping, *CNAME*, a canonical name (for aliases) (Albitz & Liu 59), and *MX*, a mail exchanger (94). DNS lookups are not case-sensitive, but the case is preserved in the domain name (58).

As mentioned previously, the Domain Name System is essentially a distributed database. Each site stores data about its own computers and when accessed, the two sites act as a client/server model, communicating with one another to exchange data. On the client side, the resolver library is used to access the name servers. The file called */etc/resolv.conf* lists the DNS servers that a host on the network should query when its corresponding domain name is accessed

(Nemeth et al. 411). The following is an example of what is listed in */etc/resolv.conf*, using an account on *hoare.cs.umsl.edu*:

```
hoare% cat /etc/resolv.conf
domain mathcscs.umsl.edu
nameserver 134.124.30.125
search cs.umsl.edu umsl.edu
```

Although there is only one name server listed in this example, there can be a maximum of three name servers (one per line). A single domain name is listed as the argument to the *domain* line. This is considered to be the default domain. It is appended to a name that was not fully qualified (does not end with a dot) (412). But although this file contains a *domain* line and a *search* line, it is recommended to solely use the *search* line. If both directives are listed in the file, the directive that is listed last is the one that is actually used in the query because the two directives are mutually exclusive (412). Unlike the *domain* line, multiple arguments (up to eight) can be supplied in the *search* line. In this example, a query for *hoare* will result in a search for the hostname *hoare.cs.umsl.edu*. If that search fails, *hoare.umsl.edu* will be queried next.

Domains are queried with the root as the beginning request. Thus, it is the responsibility of the root name servers to locate the authoritative name servers for each of the top-level zones (Albitz & Liu 26). For example, if a query is given for a domain name, *admiral.umsl.edu*, the root name server responds with a referral consisting of the names and addresses for the *edu* domain. Next, the top-level name server *edu* responds with another referral consisting of the names and addresses for the *umsl.edu* domain and so on, until finally, the name server that is authoritative for the requested information is found (Nemeth et al. 408).

On the server side of the DNS system, two types of name servers, a primary master (authoritative server) and a secondary master (slave server), are required. A *name server* implements the Domain Name System by running the BIND server daemon called **named** (403). The configuration of **named** is specified in a file called */etc/named.conf*. Each type of statement listed in the file must be terminated with a semicolon (415). The main types of statements are *include*, *options*, *acl*, *server*, and *zone*. An *include* statement specifies a subsidiary file to be included in the configuration file (417). An *options* statement is used to set global options and defaults (417). An *acl* statement specifies the access control list (422). A *server* statement is used to override a specified server-related configuration option (423). And finally, a *zone* statement is used to inform **named** "about the zones for which it is authoritative and set the options that are appropriate for managing each zone" (424).

The *primary master* name server for a zone is responsible for reading data for the zone from a disk file that is located on its host. It is the system administrator's responsibility to edit the master server's data file when a zone's data needs to be updated or changed. Therefore, it is important to keep the master server in a secure and stable location (405). On the other hand, it is the *secondary master* name server's job to retrieve the zone's data from the master server via a "zone transfer" operation (405). A *zone transfer* occurs when a slave begins running and contacts its master server and, if necessary, copies the zone data files (Albitz & Liu 25), whereas a *zone* refers to a section of the domain name space (21). But in spite of what a slave name server suggests, it is not a "second-class name server" (25). In fact, a slave name server in one

zone can be a master name server for another zone. Similarly, a master server in one zone can be a slave server to a different zone. This is possible because a name server can be authoritative for more than just one zone (25). Slave name servers are primarily used to distribute the work load to ensure that there is a name server nearby all the hosts in its zone (25). Because of this, it is recommended to have at least two slave servers, one of which is off-site (Nemeth et al. 405). Slave servers are normally configured to back up the zone data files they have transferred from a master name server. In effect, a slave server will first read the backed up data files if it is killed and/or restarted. If it finds that the backed up data is current, it will simply use those files, thus eliminating the need for any unnecessary work (Albitz & Liu 25).

To check that a name server is alive, the **ping** command can be used. Although the next example uses **ping** on hostnames, an IP address can be supplied instead.

```
admiral% /usr/sbin/ping doesthisdomainexist.com
/usr/sbin/ping: unknown host doesthisdomainexist.com
admiral% /usr/sbin/ping admiral.umsl.edu
admiral.umsl.edu is alive
```

In conclusion, accessing the Internet would be a chaotic chore without the implementation of the Domain Name System. Computers understand IP addresses, but domain names were created for the ease of human readability. Without them, sites would be difficult to remember. For example, it is much easier to remember *admiral.umsl.edu* rather than its corresponding IP address, 134.124.15.13. But in order for a mapping of a hostname with an IP address be possible, some type of file to keep the host information was needed. Although *HOSTS.TXT* served its purpose for the ARPAnet when it was a small network, managing the file became unworkable as more hosts were created. Therefore, the Domain Name System was developed. Currently, it is the most widely-used (and probably the most accessed) database in existence.

References

Albitz, Paul, and Cricket Liu. DNS and BIND. 4th ed. Sebastopol, CA: O'Reilly & Associates, Inc. 2001

Nemeth, Evi, et al. UNIX System Administration Handbook. 3rd ed. Upper Saddle River, NJ: Prentice-Hall PTR. 2001