How Domain Name Servers Work

by Marshall Brain

If you spend any time on the Internet sending e-mail or browsing the Web, then you use **domain name servers** without even realizing it. Domain name servers, or DNS, are an incredibly important but completely hidden part of the Internet, and they are fascinating! The DNS system forms one of the largest and most active distributed databases on the planet. Without DNS, the Internet would shut down very quickly.

In this edition of **HowStuffWorks**, we will take a look at the DNS system so you can understand how it works and appreciate its amazing capabilities.

The Basics

When you use the Web or send an e-mail message, you use a **domain name** to do it. For example, the URL "http://www.howstuffworks.com" contains the domain name **howstuffworks.com**. So does the e-mail address "iknow@howstuffworks.com."

Human-readable names like "howstuffworks.com" are easy for people to remember, but they don't do machines any good. All of the machines use names called **IP addresses** to refer to one another. For example, the machine that humans refer to as "www.howstuffworks.com" has the IP address **216.183.103.150**. Every time you use a domain name, you use the Internet's domain name servers (DNS) to translate the human-readable domain name into the machine-readable IP address. During a day of browsing and e-mailing, you might access the domain name servers hundreds of times!

Domain name servers translate domain names to IP addresses. That sounds like a simple task, and it would be -- except for five things:

- There are billions of IP addresses currently in use, and most machines have a human-readable name as well.
- There are many billions of DNS requests made every day. A single person can easily make a hundred or more DNS requests a day, and there are hundreds of millions of people and machines using the Internet daily.
- Domain names and IP addresses change daily.
- New domain names get created daily.
- Millions of people do the work to change and add domain names and IP addresses every day.

The DNS system is a **database**, and no other database on the planet gets this many requests. No other database on the planet has millions of people changing it every day, either. That is what makes the DNS system so unique!

IP Addresses

To keep all of the machines on the Internet straight, each machine is assigned a unique address called an **IP address**. IP stands for **Internet protocol**, and these addresses are 32-bit numbers normally expressed as four "octets" in a "dotted decimal number." A typical IP address looks like this:

```
216.183.103.150
```

The four numbers in an IP address are called **octets** because they can have values between 0 and 255 (2^8 possibilities per octet).

Every machine on the Internet has its own IP address. A server has a static IP address that does not change very often. A home machine that is dialing up through a modem often has an IP

address that is assigned by the ISP when you dial in. That IP address is unique for your session and may be different the next time you dial in. In this way, an ISP only needs one IP address for each modem it supports, rather than for every customer.

If you are working on a Windows machine, you can view your current IP address with the command **WINIPCFG.EXE** (**IPCONFIG.EXE** for Windows 2000/XP). On a UNIX machine, type **nslookup** along with a machine name (such as "nslookup www.howstuffworks.com") to display the IP address of the machine (use the command **hostname** to learn the name of your machine).

For more information on IP addresses, see IANA.

As far as the Internet's machines are concerned, an IP address is all that you need to talk to a server. For example, you can type in your browser the URL **http://216.183.103.150** and you will arrive at the machine that contains the Web server for HowStuffWorks. Domain names are strictly a human convenience.

Domain Names

If we had to remember the IP addresses of all of the Web sites we visit every day, we would all go nuts. Human beings just are not that good at remembering strings of numbers. We are good at remembering words, however, and that is where domain names come in. You probably have hundreds of domain names stored in your head. For example:

- www.howstuffworks.com - a typical name
- www.yahoo.com - the world's best-known name
- www.mit.edu - a popular EDU name
- encarta.msn.com - a Web server that does not start with www
- www.bbc.co.uk - a name using four parts rather than three
- ftp.microsoft.com - an FTP server rather than a Web server

The COM, EDU and UK portions of these domain names are called the **top-level domain** or **first-level domain**. There are several hundred top-level domain names, including COM, EDU, GOV, MIL, NET, ORG and INT, as well as unique two-letter combinations for every country.

Within every top-level domain there is a huge list of **second-level domains**. For example, in the COM first-level domain, you've got:

- howstuffworks
- yahoo
- msn
- microsoft
- plus millions of others...

Every name in the COM top-level domain must be **unique**, but there can be duplication across domains. For example, **howstuffworks.com** and **howstuffworks.org** are completely different machines.

In the case of bbc.co.uk, it is a third-level domain. Up to **127 levels** are possible, although more than four is rare.

The left-most word, such as **www** or **encarta**, is the **host name**. It specifies the name of a specific machine (with a specific IP address) in a domain. A given domain can potentially contain millions of host names as long as they are all unique within that domain.

Distributing Domain Names

Because all of the names in a given domain need to be unique, there has to be a single entity that controls the list and makes sure no duplicates arise. For example, the COM domain cannot contain any duplicate names, and a company called **Network Solutions** is in charge of maintaining this list. When you register a domain name, it goes through one of several dozen registrars who work with Network Solutions to add names to the list. Network Solutions, in turn, keeps a central database known as the whois database that contains information about the owner and name servers for each domain. If you go to the whois form, you can find information about any domain currently in existence.

While it is important to have a central authority keeping track of the database of names in the COM (and other) top-level domain, you would not want to centralize the database of all of the information in the COM domain. For example, Microsoft has hundreds of thousands of IP addresses and host names. Microsoft wants to maintain its own domain name server for the **microsoft.com** domain. Similarly, Great Britain probably wants to administrate the **uk** top-level domain, and Australia probably wants to administrate the **au** domain, and so on. For this reason, the DNS system is a **distributed database**. Microsoft is completely responsible for dealing with the name server for microsoft.com -- it maintains the machines that implement its part of the DNS system, and Microsoft can change the database for its domain whenever it wants to because it owns its domain name servers.

Every domain has a domain name server somewhere that handles its requests, and there is a person maintaining the records in that DNS. This is one of the most amazing parts of the DNS system -- it is completely distributed throughout the world on millions of machines administered by millions of people, yet it behaves like a single, integrated database!

The Distributed System

Name servers do two things all day long:

- They accept requests from programs to convert domain names into IP addresses.
- They accept requests from other name servers to convert domain names into IP addresses.

When a request comes in, the name server can do one of four things with it:

- It can answer the request with an IP address because it already knows the IP address for the domain.
- It can contact another name server and try to find the IP address for the name requested. It may have to do this multiple times.
- It can say, "I don't know the IP address for the domain you requested, but here's the IP address for a name server that knows more than I do."
- It can return an error message because the requested domain name is invalid or does not exist.

When you type a URL into your browser, the browser's first step is to convert the domain name and host name into an IP address so that the browser can go request a Web page from the machine at that IP address (see How Web Servers Work for details on the whole process). To do this conversion, the browser has a conversation with a name server.

When you set up your machine on the Internet, you (or the software that you installed to connect to your ISP) had to tell your machine what name server it should use for converting domain names to IP addresses. On some systems, the DNS is dynamically fed to the machine when you connect to the ISP, and on other machines it is hard-wired. If you are working on a Windows 95/98/ME machine, you can view your current name server with the command **WINIPCFG.EXE** (IPCONFIG for Windows 2000/XP). On a UNIX machine, type **nslookup** along with your machine name. Any program on your machine that needs to talk to a name server to resolve a domain

name knows what name server to talk to because it can get the IP address of your machine's name server from the operating system.

The browser therefore contacts its name server and says, "I need for you to convert a domain name to an IP address for me." For example, if you type "www.howstuffworks.com" into your browser, the browser needs to convert that URL into an IP address. The browser will hand "www.howstuffworks.com" to its default name server and ask it to convert it.

The name server may already know the IP address for www.howstuffworks.com. That would be the case if another request to resolve www.howstuffworks.com came in recently (name servers cache IP addresses to speed things up). In that case, the name server can return the IP address immediately. Let's assume, however, that the name server has to start from scratch.

A name server would start its search for an IP address by contacting one of the **root name servers**. The root servers know the IP address for all of the name servers that handle the top-level domains. Your name server would ask the root for www.howstuffworks.com, and the root would say (assuming no caching), "I don't know the IP address for www.howstuffworks.com, but here's the IP address for the COM name server." Obviously, these root servers are vital to this whole process, so:

- There are many of them scattered all over the planet.
- Every name server has a list of all of the known root servers. It contacts the first root server in the list, and if that doesn't work it contacts the next one in the list, and so on.

Here is a typical list of root servers held by a typical name server:

```
; This file holds the information on root name servers
; needed to initialize cache of Internet domain name
; servers (e.g. reference this file in the
; "cache  .  <file>" configuration file of BIND domain
: name servers).
;
; This file is made available by InterNIC registration
; services under anonymous FTP as
;     file             /domain/named.root
;     on server        FTP.RS.INTERNIC.NET
; -OR- under Gopher at RS.INTERNIC.NET
;     under menu     InterNIC Registration Services (NSI)
;        submenu     InterNIC Registration Archives
;     file           named.root
;
; last update:    Aug 22, 1997
; related version of root zone:   1997082200
;
;
; formerly NS.INTERNIC.NET
;
.                       3600000 IN  NS  A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000     A   198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                       3600000     NS  B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000     A   128.9.0.107
;
; formerly C.PSI.NET
;
.                       3600000     NS  C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.     3600000     A   192.33.4.12
;
```

```
; formerly TERP.UMD.EDU
;
.                       3600000     NS  D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.     3600000     A   128.8.10.90
;
; formerly NS.NASA.GOV
;
.                       3600000     NS  E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.     3600000     A   192.203.230.10
;
; formerly NS.ISC.ORG
;
.                       3600000     NS  F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.     3600000     A   192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
.                       3600000     NS  G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.     3600000     A   192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                       3600000     NS  H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.     3600000     A   128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                       3600000     NS  I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.     3600000     A   192.36.148.17
;
; temporarily housed at NSI (InterNIC)
;
.                       3600000     NS  J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.     3600000     A   198.41.0.10
;
; housed in LINX, operated by RIPE NCC
;
.                       3600000     NS  K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.     3600000     A   193.0.14.129
;
; temporarily housed at ISI (IANA)
;
.                       3600000     NS  L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.     3600000     A   198.32.64.12
;
; housed in Japan, operated by WIDE
;
.                       3600000     NS  M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.     3600000     A   202.12.27.33
; End of File
```

The formatting is a little odd, but basically it shows you that the list contains the actual IP addresses of 13 different root servers.

The root server knows the IP addresses of the name servers handling the several hundred top-level domains. It returns to your name server the IP address for a name server for the COM domain. Your name server then sends a query to the COM name server asking it if it knows the IP address for www.howstuffworks.com. The name server for the COM domain knows the IP addresses for the name servers handling the HOWSTUFFWORKS.COM domain, so it returns those. Your name server then contacts the name server for HOWSTUFFWORKS.COM and asks

if it knows the IP address for www.howstuffworks.com. It does, so it returns the IP address to your name server, which returns it to the browser, which can then contact the server for www.howstuffworks.com to get a Web page.

One of the keys to making this work is **redundancy**. There are multiple name servers at every level, so if one fails, there are others to handle the requests. There are, for example, three different machines running name servers for HOWSTUFFWORKS.COM requests. All three would have to fail for there to be a problem.

The other key is **caching**. Once a name server resolves a request, it caches all of the IP addresses it receives. Once it has made a request to a root server for any COM domain, it knows the IP address for a name server handling the COM domain, so it doesn't have to bug the root servers again for that information. Name servers can do this for every request, and this caching helps to keep things from bogging down.

Name servers do not cache forever, though. The caching has a component, called the **Time To Live** (TTL), that controls how long a server will cache a piece of information. When the server receives an IP address, it receives the TTL with it. The name server will cache the IP address for that period of time (ranging from minutes to days) and then discard it. The TTL allows changes in name servers to propagate. Not all name servers respect the TTL they receive, however. When HowStuffWorks moved its machines over to new servers, it took three weeks for the transition to propagate throughout the Web. We put a little tag that said "new server" in the upper left corner of the home page so people could tell whether they were seeing the new or the old server during the transition.

Creating a New Domain Name

When someone wants to create a new domain, he or she has to do two things:

- **Find a name server for the domain name to live on.**
- **Register the domain name.**

Technically, there does not need to be a machine in the domain -- there just needs to be a name server that can handle the requests for the domain name.

There are two ways to get a name server for a domain:

- You can create and administer it yourself.
- You can pay an ISP or hosting company to handle it for you.

Most larger companies have their own domain name servers. Most smaller companies pay someone.

The history of HowStuffWorks is typical. When howstuffworks.com was first created, it began as a **parked domain**. This domain lived with a company called www.webhosting.com. Webhosting.com maintained the name server and also maintained a machine that created the single "under construction" page for the domain.

To create a domain, you fill out a form with a company that does domain name registration (examples: register.com, verio.com, networksolutions.com). They create an "under construction page," create an entry in their name server, and submit the form's data into the whois database. Twice a day, the COM, ORG, NET, etc. name servers get updates with the newest IP address information. At that point, a domain exists and people can go see the "under construction" page.

HowStuffWorks then started publishing content under the domain www.howstuffworks.com. We set up a hosting account with Tabnet (now part of Verio, Inc.), and Tabnet ran the DNS for

HowStuffWorks as well as the machine that hosted the HowStuffWorks Web pages. This type of machine is called a **virtual Web hosting machine** and is capable of hosting multiple domains simultaneously. Five-hundred or so different domains all shared the same processor.

As HowStuffWorks became more popular, it outgrew the virtual hosting machine and needed its own server. At that point, we started maintaining our own machines dedicated to HowStuffWorks, and began administering our own DNS. We have a primary server and a secondary:

- AUTH-NS1.HOWSTUFFWORKS.COM 209.116.69.78
- AUTH-NS2.HOWSTUFFWORKS.COM 209.116.69.79

Our primary DNS is **auth-ns1.howstuffworks.com**. Any changes we make to it propagate automatically to the secondary, which is also maintained by our ISP.

All of these machines run name server software called BIND. BIND knows about all of the machines in our domain through a text file on the main server that looks like this:

```
@          NS    auth-ns1.howstuffworks.com.
@          NS    auth-ns2.howstuffworks.com.
@          MX 10 mail

mail       A     209.170.137.42

vip1       A     216.183.103.150
www        CNAME vip1
```

Decoding this file from the top, you can see that:

- The first two lines point to the **primary and secondary name servers**.
- The next line is called the **MX record**. When you send e-mail to anyone at howstuffworks.com, the piece of software sending the e-mail contacts the name server to get the MX record so it knows where the SMTP server for HowStuffWorks is (see How E-mail Works for details). Many larger systems have multiple machines handling incoming e-mail, and therefore multiple MX records.
- The next line points to the machine that will handle a request to **mail.howstuffworks.com**.
- The next line points to the IP address that will handle a request to **oak.howstuffworks.com**.
- The next line points to the IP address that will handle a request to **howstuffworks.com** (no host name).

You can see from this file that there are several physical machines at separate IP addresses that make up the HowStuffWorks server infrastructure. There are aliases for hosts like mail and www. There can be aliases for anything. For example, there could be an entry in this file for **scoobydoo.howstuffworks.com**, and it could point to the physical machine called walnut. There could be an alias for **yahoo.howstuffworks.com**, and it could point to yahoo. There really is no limit to it. We could also create multiple name servers and segment our domain.

The Beauty of DNS

As you can see from this description, DNS is a rather amazing distributed database. It handles billions of requests for billions of names every day through a network of millions of name servers administered by millions of people. Every time you send an e-mail message or view a URL, you are making requests to multiple name servers scattered all over the globe. What's amazing is that the process is usually completely invisible and extremely reliable!