Hentzenwerke Whitepaper Series

DNS Explained

By Whil Hentzen

DNS. In a general sense, it's what tells your computer that www.example.com is attached to 192.0.34.166, so that you can enter 'www.example.com in your Web browser instead of the IP address itself. You can live a long time without knowing anything about DNS, and, in a perfect world, that's how it should be, just as you don't have to know the mix of fuel to oxygen that your carburetor delivers to let your car engine run. But it's not a perfect world, not quite yet, and so there are times, particularly as a Web site developer, that you'll need to know a bit, or a lot, about DNS.

Hentzenwerke Publishing, Inc. • articles@hentzenwerke.com • www.hentzenwerke.com

1. Preface

1.1 Copyright

1.2 Revisions

1.2.1 History

Version	Date	Synopsis	Author
1.0.0	2006/11/21	Original	WH

1.2.2 New version

The newest version of this document will be found at www.hentzenwerke.com.

1.2.3 Feedback and corrections

If you have questions, comments, or corrections about this document, please feel free to email me at 'articles@hentzenwerke.com'. I also welcome suggestions for passages you find unclear.

1.3 References and acknowledgments

Thanks to the folks on the Milwaukee Linux User Group list (www.milwaukeelug.org), particularly Aaron Schrab, who reviewed multiple copies of this document, as well as all that amazing stuff that one can find on the Internet.

1.4 Disclaimer

No warranty! This material is provided as is, with no warranty of fitness for any particular purpose. Use the concepts, examples and other content at your own risk. There may be errors and inaccuracies that in some configurations may be damaging to your system. The author(s) disavows all liability for the contents of this document.

Before making any changes to your system, ensure that you have backups and other resources to restore the system to its state before making those changes.

All copyrights are held by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark. Naming of particular products or brands should not be seen as endorsements.

1.5 Prerequisites

This document was written for folks who want to develop Web sites and put them on the Internet. It is not intended for folks who would be administering a DNS server, say, for an ISP. Rather, it provides a high-level overview of the technology and what a Web site developer needs to know. As such, it glosses over some of the in-depth technical issues and simplifies others, eliminating details that are unimportant for this level of discussion.

2. The four players in the DNS architecture

Strictly speaking, DNS = Domain Name Service, the architecture for mapping IP addresses to hostnames. Unfortunately, sloppiness, slang, and jargon has usurped the term to mean everything from the architecture to the software that implements the architecture to the database that holds the actual hostname-IP address mappings.

There are several players involved in the architecture - the DNS database, a DNS server, and a DNS client.

The first player is the "DNS database", the database where the actual mappings of hostnames and IP addresses are stored. The second player is a DNS server, the software that dishes out info from the DNS database when asked. The DNS database and the DNS server software both reside, obviously, on a DNS server machine. The DNS client is the program (in a loose sense of the word) that sits on an end-user's computer that does the asking of a DNS server when the end-user is trying to connect to another computer.

2.1 DNS database

Each mapping is stored in a (very large) database that is distributed across a collection of special servers connected to the Internet, so that only part of the database is on any one server. These servers are called DNS servers, or, sometimes, DNS name servers. There are different DNS servers for different top level domains – those ending in '.com' are located in one database, for example, while those ending in '.edu' are located in another database. When you want to access content on example.com, you'll type that URL into your browser. Your browser looks up the IP address of example.com's host via the appropriate DNS database. This process is called 'resolving' an address. If everything works (all the correct data is found), voila, your browser is now displaying data sent from example.com's Web server software.

In the early days of the Internet, there were only a few DNS servers. The entire list of domain names and IP addresses was contained in a simple text file, and every computer on the Internet had a copy of that text file. It was relatively easy to keep all of the copies of this text file in sync. Decades later, however, things had gotten busier. Imagine if the tens of millions of domains on the Internet were all listed in a single text file, and every one of the "billyuns and billyuns" of computers on the Internet had to keep an up-to-date copy of that enormous text file.

2.2 DNS server - architecture

Even after splitting up the DNS database into subsets for the various domains, it could get very crowded at each of the DNS servers if there was only one set – particularly if there was only one DNS server for .com domains! In addition, having all of the DNS information in one place would create a single point of failure that, if it did indeed fail, would bring the entire Internet to a crashing halt. As a result, the architects of the Internet created the ability for multiple copies of the DNS databases to be available to users around the world. That means there are many DNS servers scattered around the Internet. So when your computer tries to resolve a URL, it likely uses a copy (or 'mirror', or 'slave') of the primary DNS server for the type of domain in question, instead of going to the master copy. This system of multiple DNS servers also provides redundancy – if one of them goes down, your computer can use a different DNS server instead of getting stalled. Kind of like having a spare copy of the phone book at home when the master copy has disappeared somewhere in your teenager's room.

There are logically two types of DNS servers - authoritative and caching (also known as 'recursive'). A single physical DNS server can serve both roles at once, but for simplicity's sake, let's assume that a host is one or the other.

2.2.1. Authoritative

Authoritative servers are the DNS servers of last resort, so to speak. A purely authoritative server gets entered with data about hostname-IP address mappings (called 'zone files', which I'll discuss shortly) and then responds to requests for that information - it never relies on other DNS servers for information that it is missing. Two special cases of authoritative servers are root and TLD servers.

The root servers form what you might think of as the apex of the DNS pyramid of servers. They contain information about the TLD - top level domain - servers. The next level are the parent, or TLD - top level domain - servers. Each of these

is authoritative for one or more top level domains. For example, .com and .net are both handled by *.gtld-servers.net. When you register your domain, one of the TLD servers will know where to find information for your domain.

2.2.2. Caching

Caching/recursive servers, on the other hand, are the type that get listed as name servers on an end-user (client) computer. When you set up your Internet connection, you likely have to enter two or more "DNS server" or "name server" addresses. These caching servers can be thought of as the worker bees - millions of machines scattered across the globe where the actual DNS mappings of hostnames and IP addresses are stored. Many are hosted at ISPs and similarly secure and reliable locations, but they could also be at privately owned "DNS 'r us" type companies, located in a trailer park skirting a downtown's red light district. You can even run a caching server on your own computer.

Roughly speaking, the TLD servers point to these 'worker bee' DNS servers when asked where your domain's information is.

2.3 DNS server - software

DNS server software is a program running on a computer that gets queries (in the form of URLs) from folks looking for your domain and dishes out responses (in the form of IP addresses) in return, using one of those 'worker bee' DNS databases. Your friend 'Bob' hears you have a Web site, www.example.com, and enters the URL into his Web browser. The Web browser looks up the DNS servers that he entered in his network card settings, and asks one of them what the IP address for your website is. If that DNS server has the mapping for www.example.com, it'll look it up and return the IP address, 192.0.34.166. In some cases, the DNS server he is using won't have www.example.com in its own database, but it knows where to go look - the TLD server that for '.com'.

This DNS server software program is running constantly, and is typically configured to be a 'service', so that it starts up when the computer is started. This is similar to a database server or a Web server (both of which, interestingly enough, also lie in wait for requests from users and then dish out responses in return.) There are specific instances of DNS programs, just like there are specific instances of database servers (MySQL, PostgreSQL, Oracle) and Web servers (Apache, IIS, etc.). Common DNS programs include BIND, tinydns, and djbdns.

2.4 DNS client

The third player in this scheme is a DNS 'client'. You can think of this client as a program running on your desktop (laptop) computer that fetches the IP address from a DNS server. When you enter a URL into your browser, your browser then talks to your DNS client, which then goes out onto the Internet to find one of the DNS servers that were identified in the network card settings on your computer. When the DNS client gets an answer, it then returns that answer back to whoever requested it, such as the Web browser.

Strictly speaking, the DNS client isn't actually a separate program on your computer, like a Web browser or an email client. Instead, it's a module or part of a larger program - often the operating system - that handles the work. There isn't a separate program that can be started up and shut down. A Web browser, for instance, would just ask the OS to do the lookup.

2.5 Zone file

Finally, although I said, 'three', there's one more player that I should mention now - the zone file. The DNS database consists of millions of hostname-IP address mappings. At first glance, you might think the DNS database just looks like this:

```
anacondasteel.com  99.20.4.160
bestbuy.com        68.7.7.14
crazyeddie.com     100.203.40.57
digg.com           8.12.66.48
example.com        192.0.34.166
ford.com           44.29.151.40
```

In reality, the database is considerably more complex than this. A domain has more than just the IP address for the Web server. There could be other servers involved, such as an FTP server or a mail server. There could be subdomains (the 'www.' part of the URL), and additional information, such as time for updates to be checked, are also needed. As a result, each domain has a set of records that together are called a 'zone file'. A single zone file looks something like this:

```
$TTL    86400
$ORIGIN example.com.
@  1D  IN    SOA ns1.example.com.      hostmaster.example.com. (
                  2005010101 ; serial
                  3H ; refresh
                  15 ; retry
                  1w ; expire
                  3h ; minimum
                 )
       IN  NS     ns1.example.com.
       IN  NS     ns2.example.com.
           MX  10 mail.example.com.
           MX  20 mail.another.com.
                  A      192.0.34.166
     www    IN  A      192.0.34.167
       IN  CNAME  www.example.com.
```

While this may look confusing, this information represents what would be needed for a rather simple domain. A zone file for a big domain, such as for Ford Motor Company or General Electric, is considerably more involved. The key point to bring away from this is that the DNS records for a single domain are more involved than a simple one to one hostname-IP address mapping. Additionally, now you know that the DNS database is a collection of millions of zone files.

Revisiting the terminology... your zone file contains your DNS records. Each DNS record is for a specific service, such as 'www' for a Web server or "MX" for a mail server. I'll use the terms 'zone file' and 'DNS records' somewhat interchangably, as they refer to the same general 'chunk of data'.

Let's look at the path that DNS plays both on the end of an individual surfing the Web or getting their email, and as an administrator of a Web site.

3. Individual Surfer

When you configure your computer's network card or modem to connect to the Internet, your Internet Service Provider (ISP) gives you the addresses for your DNS settings. Typically you are given at least two, but sometimes three. This duplication is so that if one of the DNS servers is unreachable or slow, your machine will have a second choice (and a third) to try.

In Windows 2000, you get to this dialog via Start | Settings | Network and Dial-up Connections | Properties | TCP/IP component | Properties, as shown in **Figure 1**.

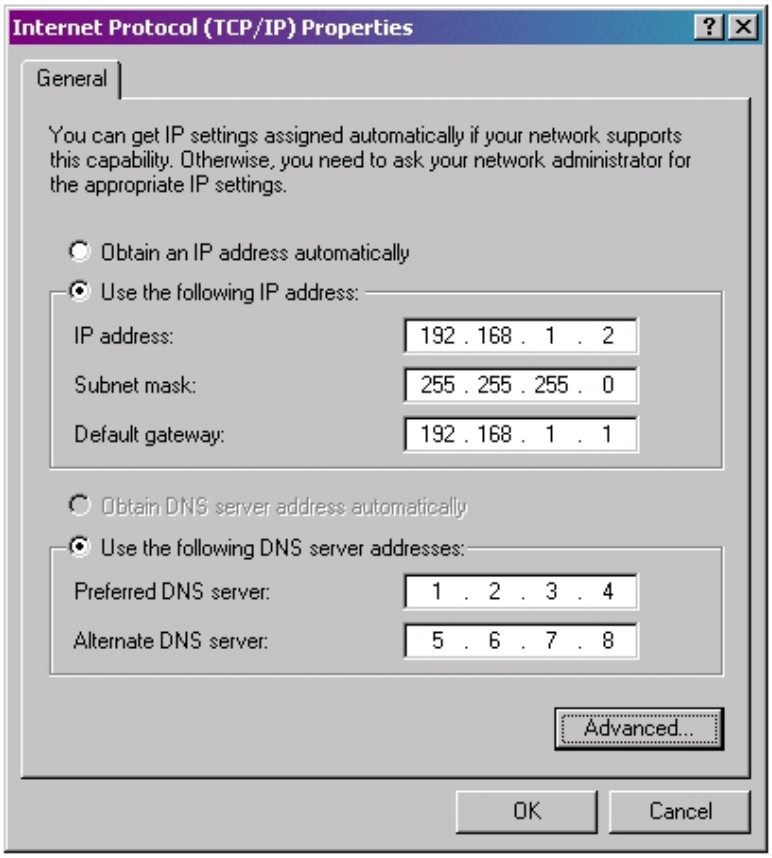

***Figure 1**. The Windows 2000 dialog for entering DNS name servers.*

In Fedora Core 6, you get to the dialog via F | Administration | Network | Network Configuration dialog | DNS tab, as shown in **Figure. 2**.

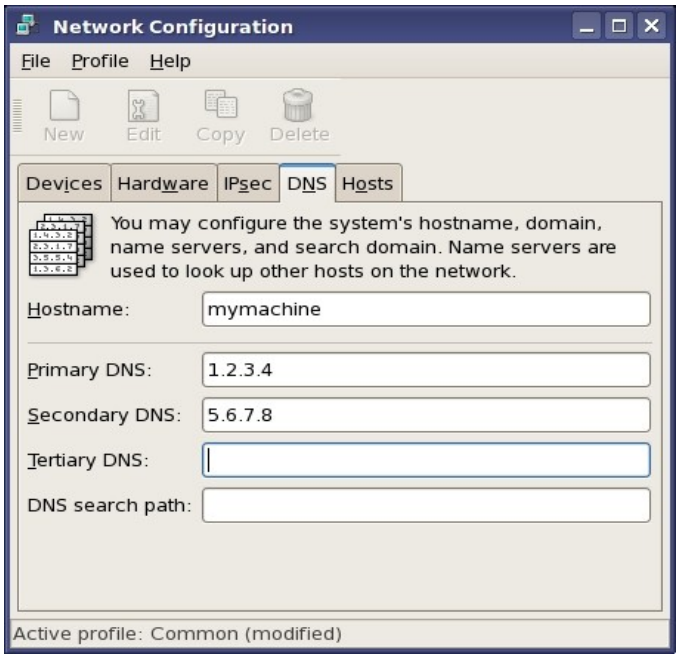

***Figure 2**. The Fedora Core 6 dialog for entering DNS name servers.*

When you enter a URL in your Web browser, or connect to your mail server via your email client, or when your IM program tries to connect to another server, your computer will send a query to the DNS servers in your DNS settings to look up the domain name. A bunch of stuff happens, and eventually a response will come back with the IP address of the machine you're trying to reach, and your computer connects to that machine. There's obviously a lot more going on under the hood, but for our purposes, this is close enough.

Some network card configuration programs call the DNS server a 'nameserver' (typically, Linux distros do.)

That's all an end-user really needs to know. Let's take a look at what an administrator needs to know, behind the scenes.

4. Setting up a Web server/Web site

Setting up a Web server is as involved as being a casual Internet user is easy. We'll start by assuming you don't even have a domain name yet.

4.1 Domain name

You'll need a domain name, such as example.com. You may be used to thinking of domain names as 'www.example.com', but in actuality, the 'www' is an add-on that we'll deal with later. When you get a domain name, you'll just be getting the 'example.com' part. (Once you have the 'example.com' part, you can set up multiple sites, called subdomains, with names like 'www.example.com', 'customers.example.com', 'private.example.com' and 'bob.example.com'.)

You'll get a domain name from a registrar. There are many, many registrars out there, but most of them are simply resellers for the primary domain name registrars. Primary registrars include networksolutions.com, register.com, godaddy.com, and so on. For purposes of this article, I'll use godaddy.com as the sample registrar, both because they're the one I use (but, no, I don't get a commission for referring them) and because I've used networksolutions.com and register.com, and find them lacking in many significant areas. They've both been around for a long time, but I wouldn't ever use them again. You may choose differently, but caveat emptor.

(The one big problem with godaddy.com is that they are very, very pushy about trying to sell you extra stuff that you really don't need. Ignore it all for the time being; you can always add it to your site later if you want, and they're constantly running specials to give you deals on doing so.)

4.2 Navigating to a brand new domain

OK, so now you've plunked down your $8.95 at godaddy and now have your very own domain name, say, 'example.com'. What happens when your mom, all proud that her son/daughter has their very own website, types in 'www.example.com' into her browser? Absolutely nothing. That's because there is no website attached to that domain name yet, and so your mom's browser ends up in '404 - page not found' land. So you need to do four more steps. I'll describe in general terms what those four things are, and then I'll walk through a specific example.

First, create a website and put it on a Web server somewhere. Second, find a DNS server (one of those authoritative DNS servers I mentioned earlier) that you can use. Third, create a DNS record (in your 'zone file') that maps 'www.example.com' to the IP address of the Web server from Step 1, and stuff it into that authoritative DNS server that you found in Step 2. And fourth, tell your registrar, godaddy.com, that the DNS server that contains your zone file for www.example.com, is the authoritative DNS server you picked in Step 2.

If you used godaddy.com (or most any other popular registrar), you'll find that the preceding description isn't completely accurate - when your mom typed 'www.example.com' in her browser, she didn't come across a 'page not found' page. Instead, she was directed to a "This site has recently been registred with godaddy.com" page. When you registered the domain name with godaddy, they didn't just let it sit off there in the ether. They create a dummy Web page for your domain on one of their servers, and then they create a zone file with records in it that point the domain to that temporary page. This is both to inform new visitors that the site is, indeed, OK (else a visitor might think they just typed the domain name wrong), and, natch, to advertise themselves at the same time. In other words, they provided default values for Steps 2, 3, and 4 for you, and they'll stay that way until you choose to change them.

OK, now lets' do this same thing, but with 'real data'.

4.3 Live example

Suppose you've registered 'example.com' with godaddy. (Godaddy has a Web server and an authoritative DNS server that they use for newly registered domains like yours.) A zone file was created, and they've put that zone file in a pair of their DNS servers. They've also created a dummy Web page for 'www.example.com' for you and put it on one of their Web servers as well. So then, when you (or your mom) navigate to www.example.com, you arrive at the 'temporarily parked here...' page. You can log into your godaddy account and navigate to the 'Manage Domains' link; doing so will display a page that lists your nameservers, like so:

```
Name Servers: (Last update 1/1/1980)
PARK21.SECURESERVER.NET
PARK22.SECURESERVER.NET
```

These are the names for a couple of DNS servers that godaddy runs. The DNS records in your zone file on them point to godaddy's Web server. The actual Web page for administering them looks like **Figure 3**. You can see the nameservers displayed in the very middle of the page, slightly under and the left of the red "Cancel" button.

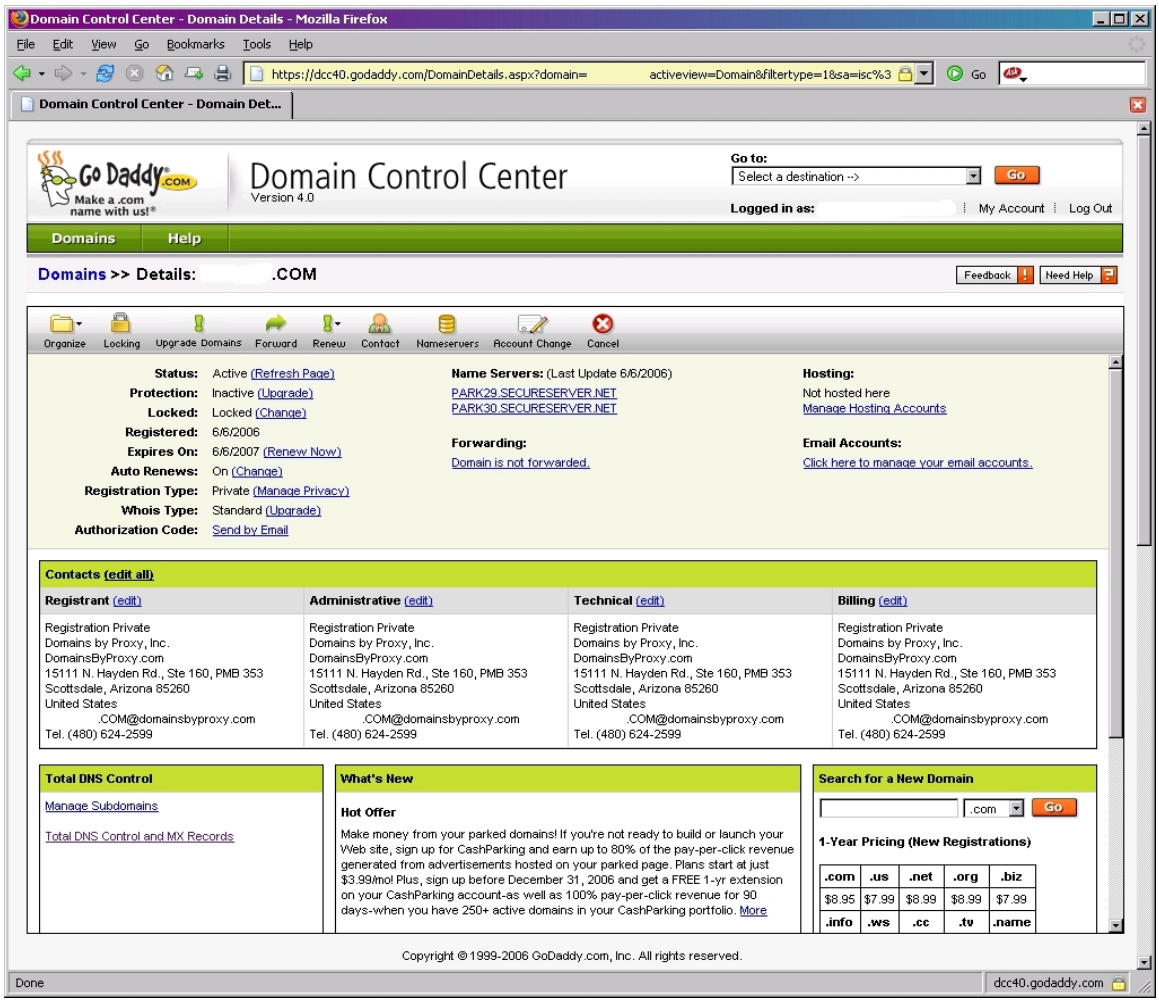

***Figure 3**. The GoDaddy Web page for managing domain DNS information.*

4.4 Hosting your site at a third party site

Now what? First, you CAN continue to host your DNS zone file and your Web site at godaddy. Use of the godaddy DNS server to store your DNS records comes with your registration fee (other registrars may or may not include DNS with the registration). Note that typically hosting your Web site will cost you a couple of bucks extra.

If that's the case, your job is nearly done. After purchasing the Web site option, they'll give you some space on a server of theirs. (They have acres and acres of servers.) They'll also give you the IP address for your space on that server. You'll create your Web pages and then FTP those pages up to their server. They'll also change your zone file so that it points to the root directory of your space on the server, and you're done.

4.5 Hosting your site yourself

Suppose you want to host your own Web server. There can be several reasons for this, such as the need to run software that isn't supported on godaddy's servers. In this case, you'd create your own Web server, installing Linux and Apache, say, on it, and then move your Web pages to that machine. But at this point, this Web server machine is an island without a bridge connecting it to the rest of the Internet. Time to get connected.

You'll need a connection to the Internet, similar to the one you already use - maybe even the same one. Your ISP will give you a public IP address that becomes the gateway for your server. (I'm taking complications like routers and firewalls out of the equation right for the time being.) When your ISP became an ISP (from its humble beginnings as a bait and tackle shop or a telephone company), they received a block of IP addresses (their 'netblock') that they in turn dish out to their customers. The IP address that they give you is one of the addresses in their netblock. While it's 'your' IP address, they're actually just lending it to you for as long as you're their customer; if you change ISPs, you'll lose that IP address and need to get one from your new ISP. I mention this because the concept that the IP address is under the ISP's control will be important later on.

Back to your new public IP address and your Web server. You would need to change your DNS records on godaddy's DNS servers to point 'www.example.com' to that new public IP address from your ISP. It'll take anywhere from a few minutes to the better part of a day for the changes to your zone file to filter throughout all of the DNS servers on the Web (this is called 'propagation'), but soon enough, people surfing to 'www.exmaple.com' will end up at your web site!

At this point, your job is done. Well, except for the trivial matter of building your Web site, garnering traffic, that sort of thing. Your zone file still resides on godaddy's DNS servers, but the Web server record in it points to the IP address of the Web server sitting in your basement or in the server room of your company.

Suppose you had a separate email server, running on a box sitting right next to the Web server? You would change your zone file to point your MX records to the IP address for that machine. What if you didn't host your own email server, though? Instead, you used the server of an email service provider, such as godaddy or your ISP. In those cases, the MX record in your zone file would point to THAT email server's IP address instead.

4.6 Turning over your DNS to a third party

Let's take things a step further. Suppose you don't want to keep your DNS zone file on godaddy's DNS servers? Just like you aren't required to host your Web site or email on godaddy's servers, you're not required to host your DNS on their servers. First, you'll need a new DNS server. This can be a box of yours running the DNS server software, just like you've got a box running Web server software. Or you could outsource it to any one of a number of companies that provide DNS services, just like there are companies that provide Web site hosting and email services. You can move your zone file to one of those companies. Suppose you decided to use zoneedit.com to host your zone file. When you set up an account with them, they'll give you the names of the DNS servers (much like godaddy's were "PARK21.SECURESERVER.NET" and "PARK22.SECURESERVER.NET").

You just need to tell your registrar, godaddy in this case, that your domain's name server is now zoneedit.com, and change your name servers from "PARK21.SECURESERVER.NET" and "PARK22.SECURESERVER.NET" to the DNS servers that zoneedit's gave you.

4.7 "It's a lot worse than that"

There's more to DNS than this. The DNS database is distributed across many servers throughout the world, and there's a complex and sophisticated mechanism to keep them all in sync. For all practical purposes, you don't need to know any of this. You just need to keep the zone file on your own DNS server set up, and your machines configured to use your DNS servers, and you'll be OK.

4.8 Summary of authoritative and caching server roles

In summary, when you create a domain name, you tell your registrar where the authoritative DNS servers for your domain are. Your zone file sits on these authoritative servers. This zone file is copied (or 'propagated') to caching servers all over the Internet. The timing values in the zone file tells the caching servers how often they need to refresh their data. User machines all over the Internet point to caching servers. When a user machine requests a record, it'll look at its caching servers. If the data is not found, there's a clearly defined path for it to look at; the details of which aren't important to us.

Suppose one of your user machine's caching servers had just started up, and hadn't gone out to populate its database yet. It'll only know about the root servers. So in this case, the caching server would ask a root server for the information about www.example.com. The root server would respond, saying that the root server doesn't know, but information on .com can be found at the servers for that domain. The caching server asks one of the .com servers for the information. The .com server responds saying that it only knows where to find information for example.com. If the URL was longer and had more levels, the caching server continues on like that until it finally gets the all of the info or ends up with a permanent error. Finally, the caching server returns the info to the client that requested it.

That said, it's now time to look at those zone files in more detail.

5. Zone files

Zone files control two basic things - the mappings of various types of servers, such as web servers, mail servers, ftp servers, and so on. This way, mail to bob@example.com knows where the example.com mail server is located - since it might not be on the same machine - or in the same country! - that serves up Web pages for www.example.com.

The second thing that a zone file controls is how often changes to the zone file are propagated throughout the Internet. Unless you have special needs, the default settings that came with your zone file when it was set up are probably good enough. Most people (particularly the folks reading this) are typically going to set up their zone file, pointing to their web site, mail server, and so on, and then leave it alone. If you were in the business of running multiple websites for many people or organizations, you'd probably be doing more tweaking.

Fortunately, the arcane syntax used in a zone file is usually hidden away from you. Most DNS server providers provide an editing facility that allows you to make changes using a simple interface as shown in **Figure 4**.

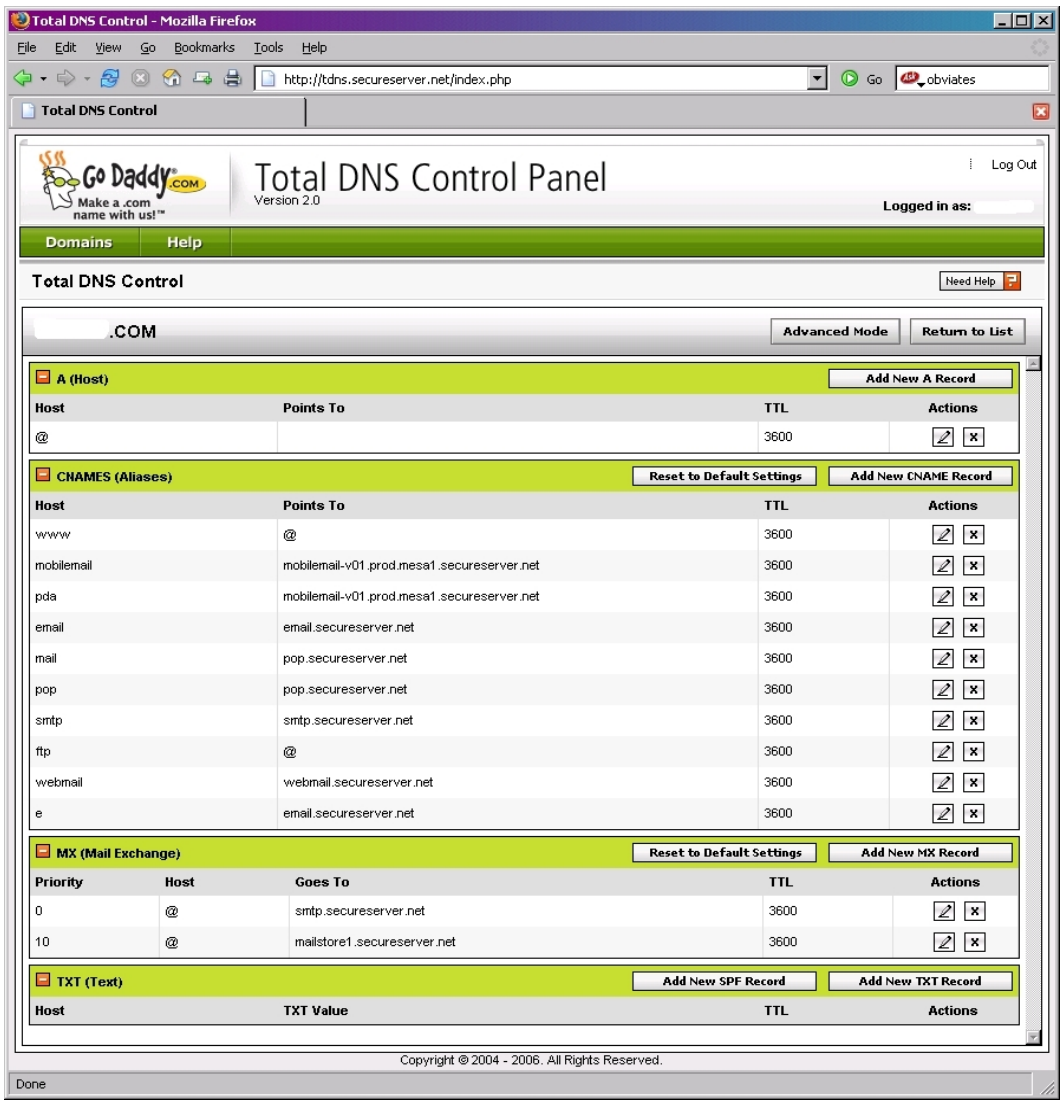

***Figure 4**. GoDaddy's Web interface for changing DNS records.*

What you do need to know is the concepts behind the changes you're making. Let's look at our dummy zone file again.

```
$TTL    86400
$ORIGIN example.com.
@  1D  IN    SOA ns1.example.com.      hostmaster.example.com. (
                  2005010101 ; serial
                  3H ; refresh
                  15 ; retry
                  1w ; expire
                  3h ; minimum
                 )
           NS     ns1.example.com.
           NS     ns2.example.com.
           MX  10 mail.example.com.
           MX  20 mail.another.com.
                  A      192.0.34.166
     www     IN  A      192.0.34.167
     bob     IN  A      192.0.34.168
       IN  CNAME  www.example.com.
```

SOA is an acronym for "Start of Authority". The DNS database consists of zillions of zone files, each of which is the responsibility of someone. Everything in that zone file is that person's responsibility. When you think of the DNS database as a pyramid, your zone file makes up one very small brick in that construction. The SOA indicates where in that stack of bricks your responsibility starts. The rest of the DNS database - the root servers, TLD servers, and worker bee servers can only point to your zone file. Within your zone file, you are the master. The SOA record points to the 'start' of your zone file.

You don't have to worry about most of the pieces of the SOA record. The first piece, the '@' sign, is simply a pointer to the current zone - sort of like you're pointing to yourself. The next important piece is 'ns1.example.com' - it's the primary nameserver for your domain. This entry must be followed by a period. And the last part of the line is the email address of the person responsible for the domain, except that the '@' sign is replaced by a period, and the address is followed by a period.

The second line is the serial number of the zone file - the date that the zone file was last updated, followed by a sequence number so that if the zone file is updated more than once a day, it's clear to other servers that get updated with this zone file's information whether or not they have the most recent update. In other words, suppose you update your zone file in the morning. The serial number becomes '2006103101'. Later that day, the changes you've made are propagated throughout the Internet - and now other DNS servers have the most recent date, including the '2006103101' serial number. If you then update your zone file again that day (some people can never make up their minds...), the serial number becomes '2006103102'. When another server checks in on your zone file, it will see that its serial number ends in '01' while yours ends in '02' and thus knows to grab a fresh copy of your zone file.

The next four lines involve time spans. The values can be described in seconds, in which case no unit is needed to be displayed, or in other time units, in which case time units - 'H' = hours, 'w' = weeks, 'm' = minutes - are required. 86,400 = seconds in one day, 28800 = 8 hours, 604,800 = 1 week, 7200 = two hours.

The third line, refresh, tells another DNS server how often it should check your zone file for updates. The fourth line, retry, tells the other server how often it should try to connect to your server in the event of a connection failure. The fifth line, expiry, is the total amount of time that the other server should try checking before it gives up. If it gives up, it will flag your zone file on its database as expired and then begin to redirect requests for your DNS information to the root servers. Finally, the sixth line, TTL (time to live) represents the amount of time that another server will cache answers from your server. As I said, for the most part, you'll not want to mess with these values.

After the time span values, the next records specify the nameservers for the domain - the records to the right of the 'NS' record type. After that comes an 'MX' record - which stands for 'mail exchanger'. You can have more than one MX record, each of which points to a different mail server. The order in which mail is directed to a server is controlled by the two digit number between the 'MX' record type and the URL of the mail server itself, which the lower number being a higher priority.

The A records map host names to IP addresses. You'll usually have one 'A' record that is mapped to a 'catch-all' IP address, and then, possibly, subdomains mapped to other IP addresses. For example, 'www.example.com' is mapped to '192.0.34.167' while 'bob.example.com' is mapped to '192.0.34.168'. If your mom typed 'http://example.com' into her browser, however, she would be directed to the 'catch-all' '192.0.34.166' address.

The CNAME record (CNAME is short for "canonical name") is an alias for an A record.

Finally, TXT records (not shown here) are used for SPF (Sender Policy Framework) records. These are records that specify which machines are allowed to send mail with the sender set to your domain. If the sender domain does not have an SPF record, or if the sender domain is sending from a machine that is not listed in the SPF record, then the mail is classified as spam. (More info on SPF can be found at openspf.org.)

6. Reverse DNS

The DNS records allow programs to look up the IP address for 'example.com' in all its glory (as well as anything else related, such as FTP or mail servers). You may be wondering about the reverse - if you had an IP address, could you look up the domain name? Yes, you could, and this is called 'reverse DNS'. This is actually important because some programs

(particularly mail servers) will refuse email from domains if the reverse DNS results do not match the regular (also called 'forward') DNS.

A reverse DNS record looks like this:

```
zone "34.0.192.in-addr.arpa" {
     type master;
     file "pri.34.0.192.in-addr.arpa";
};
```

You'll see that the first part of the address, '34.0.192.in-addr.arpa' begins with the reverse of the first three parts of the example.com IP address. The third line contains the name of the reverse zone file, 34.0.192.in-addr.arpa. The reverse zone file itself contains, instead of A or MX records, PTR records (PTR stands for 'pointer'.)

A reverse zone file would consist of individual records for each IP address associated with the domain. Following our example, our reverse zone file might include PTR records like this:

```
166  PTR  example.com
167  PTR  ns1.example.com
168  PTR  mail.example.com
```

Reverse DNS is something that many ISPs don't have a clear handle on, or that they don't bother to do until you nag them to. You can tell if your reverse DNS is set up through the Linux command line tool, 'dig'. First, let's look at a domain where the reverse DNS is not set up properly. Find the IP address for an example domain:

```
> dig bozo_dns.com
<some stuff>
QUESTION SECTION:
bozo_dns.com        IN     A
ANSWER SECTION:
bozo_dns.com               12837         IN    A    1.2.3.4
```

And then, with the "-x" switch on the IP address, you can see that the reverse DNS is not set up properly.

```
> dig -x 1.2.3.4
<some stuff>
QUESTION SECTION:
4.3.2.1.in-addr.arpa.   IN PTR
ANSWER SECTION:
4.3.2.1.in-addr.arpa.   IN PTR  1.2.3.4.ded.pacbell.net.
```

As you can see from the last line, the IP address does not resolve back to the domain name; instead, it resolves to the ISP who owns the netblock.

This example shows DNS set up properly:

```
> dig example.com
<some stuff>
QUESTION SECTION:
example.com        IN     A
ANSWER SECTION:
example.com               8702         IN    A    192.0.34.166
```

And then, with the "-x" switch on the IP address, you can see that the reverse DNS is set up properly as well.

```
> dig -x 192.0.34.166
<some stuff>
```

```
QUESTION SECTION:
166.34.0.192.in-addr.arpa.   IN PTR
ANSWER SECTION:
166.34.0.192.in-addr.arpa.   IN PTR  www.example.com.
```

Reverse DNS records are stored by your ISP on servers similar to the authoritative DNS servers we've already looked at, because the ISP controls the netblock of IP addresses. In order to get reverse DNS set up, you have to request your ISP to do it, since they control the records. While in theory you could have your ISP delegate authority for your reverse DNS to another service, most ISPs won't as a matter of convenience and consistency.

7. Where to go for more information

This free whitepaper is published and distributed by Hentzenwerke Publishing, Inc. We have the largest lists of “Moving to Linux”, OpenOffice.org, and Visual FoxPro books on the planet.

We also have oodles of free whitepapers on our website and more are being added regularly. Our Preferred Customer mailing list gets bi-monthly announcements of new whitepapers (and gets discounts on our books, first crack at special deals, and other stuff as we think of it.)

Click on “Your Account” at www.hentzenwerke.com to get on our Preferred Customer list.

If you found this whitepaper helpful, check out these Hentzenwerke Publishing books as well:

Linux Transfer for Windows® Network Admins:
A roadmap for building a Linux file and print server
Michael Jang

Linux Transfer for Windows® Power Users:
Getting started with Linux for the desktop
Whil Hentzen