DNS

The Domain Name Service, or DNS, is integral to network computing. It provides a mechanism by which IP addresses can be looked up using host names, and vice versa. If it weren't for DNS, you wouldn't be able to type "google.com" into your browser and get to the search engine. Instead, you would have to remember the IP address that points to Google's web server. It would be like having to remember the phone numbers of every person you know! DNS is like the phonebook of internet addresses.

A Brief History

When the internet was first created, DNS did not exist. Instead, each internet-connected machine had a file that mapped host names to IP addresses. Most systems still have this file, though it's usually empty, or only contains a couple of entries. This file is located at /etc/hosts. When new hosts were added to the internet, their address had to be added to the hosts file, and the new file had to be sent out to all computer users to install on their machines. As the internet grew, it became increasingly difficult to keep the hosts file current. This problem was solved by the creation of DNS.

DNS allows for centrally managed databases that map host names to address. Each domain is responsible for managing its own DNS entries. If you send a query to your domain's DNS server for a host that is not within your domain, your DNS server has the ability to query the DNS server for the correct domain and return an answer to you. This structure helps to reduce the work associated with passing around the /etc/hosts file. In the next course we'll discuss DNS servers at length, but for now we'll just cover DNS on the client side.

Configuring Your DNS Client

By default, almost all machines can query DNS, but they may not be set up to do it automatically. This is the case with your machine. Configuring the DNS client to query a DNS server involves adding a few lines to the **/etc/resolv.conf** file. The name of this file is derived from the term used to describe the response to a DNS query. If you were to look up the host "oreillyschool.com," and get back the address "199.27.144.89," you would say that "the host oreillyschool.com resolved to the address 199.27.144.89." Create the file **/etc/resolv.conf** with these contents (you'll need to use root privileges to do this):

CODE TO TYPE:

```
domain useractive.com
nameserver 172.16.0.1
nameserver 172.16.0.2
```

Note We use the "useractive.com" domain here partly as a tribute to our humble beginnings.

When you define a domain parameter, you can use shorter hostnames later (for example, "cold" rather than "cold.useractive.com").

After you've written the file, you can use the **host** command to see whether DNS is working:

INTERACTIVE SESSION:

```
[username@username-m0 ~]$ host oreillyschool.com
oreillyschool.com has address 199.27.144.89
oreillyschool.com mail is handled by 20 ALT2.ASPMX.L.GOOGLE.com.
oreillyschool.com mail is handled by 10 ASPMX.L.GOOGLE.com.
oreillyschool.com mail is handled by 20 ALT1.ASPMX.L.GOOGLE.com.
```

Although there's no indication of it here, the name for the IP address that's returned by this query is an *A record*. There are several classes of records for hosts and domains. The other three entries you see here are *MX* or *Mail eXchanger*records. These tell you which machines handle the email for the given host or domain. We'll cover a few more record types in the next section when we introduce tools that enable you to make more detailed DNS queries.

DNS Client Tools

Now try running a DNS lookup on "google.com" using **host**. You'll see nearly a dozen IP addresses returned, as well as several MX records. DNS allows one host name to have multiple A records. This is primarily done to achieve round-robin load balancing. Each time the DNS server is queried for a host with multiple A records, the list of A records is returned in a different order, that way multiple clients can be directed to various servers, allowing the load to be spread among them.

It's also possible to have multiple MX records (as you can see in both the oreillyschool.com and google.com examples). In the MX lines, you'll see a number directly before the host name of the server. This number indicates the priority of the given server. The lower the number, the higher the priority. The highest priority server will be tried first, and if that fails, mail will then be sent to the next higher priority server and so on, until it is successfully sent or the list of servers is exhausted.

Before we introduce any more record types, we'll need to learn about the tool we use to query all record types: **dig**. Dig allows you to perform a DNS lookup and specify one or more record types to return. The format of the command is:

OBSERVE:

```
dig host recordtype
```

Let's make our first dig query using a familiar record type, the A record:

INTERACTIVE SESSION:

```
[username@username-m0 ~]$ dig google.com A

; <<>> DiG 9.7.3-P3-RedHat-9.7.3-8.P3.el6_2.2 <<>> google.com A
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20502
;; flags: qr rd ra; QUERY: 1, ANSWER: 11, AUTHORITY: 4,
ADDITIONAL: 4

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

;; ANSWER SECTION:
google.com.             300     IN      A       74.125.224.133
google.com.             300     IN      A       74.125.224.134
google.com.             300     IN      A       74.125.224.135
google.com.             300     IN      A       74.125.224.136
google.com.             300     IN      A       74.125.224.137
google.com.             300     IN      A       74.125.224.142
google.com.             300     IN      A       74.125.224.128
google.com.             300     IN      A       74.125.224.129
google.com.             300     IN      A       74.125.224.130
google.com.             300     IN      A       74.125.224.131
google.com.             300     IN      A       74.125.224.132

;; AUTHORITY SECTION:
google.com.             216604  IN      NS      ns3.google.com.
google.com.             216604  IN      NS      ns4.google.com.
google.com.             216604  IN      NS      ns1.google.com.
google.com.             216604  IN      NS      ns2.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         15102   IN      A       216.239.32.10
ns2.google.com.         15102   IN      A       216.239.34.10
ns3.google.com.         187530  IN      A       216.239.36.10
ns4.google.com.         187530  IN      A       216.239.38.10

;; Query time: 51 msec
;; SERVER: 172.16.0.1#53(172.16.0.1)
;; WHEN: Fri Feb 24 10:20:01 2012
;; MSG SIZE  rcvd: 340
```

That's a lot of information to parse! Fortunately, dig splits the code into sections to make it easier to read. Take a look at the QUESTION SECTION first. It tells us what we've asked for, in this case, an A record. The next section is the ANSWER SECTION. This list looks a lot like the list you got back when you ran the **host** command against

google.com. However, it gives us some information that **host** did not. After the host name listed on each line, you see the number "300." This is the *Time To Live* or *TTL*;it's a length of time in seconds. In order to understand what the TTL tells us, we need to understand how DNS caching works.

One DNS server can query another DNS server that acts on behalf the domain of the host you're seeking. The server that acts for the target domain is called the *authoritative DNS server* for that domain. When your DNS server receives a response from the authoritative DNS server for the domain you are querying, it not only responds to your query, it also stores the information in its own database. This is called *caching*. Caching allows your DNS server to respond to any additional queries for that host without having to make a query of the authoritative server for that domain. It saves time and reduces load on authoritative DNS servers. The cached result remains cached until the entry has exceeded its TTL. We'll discuss DNS TTLs in depth in the next course when we learn about setting up a DNS server.

Moving on, we see the AUTHORITY SECTION. This section lists the authoritative DNS servers for the domain you've queried. As you can see in this example, it's possible to have multiple authoritative DNS servers for a particular domain. The IP addresses are included in the ADDITIONAL SECTION, which follows the AUTHORITY SECTION. Much like MX records, NS (**N**ame **S**erver) records contain a host name, not an IP address. That's the reason that the ADDITIONAL SECTION contains the A records that correspond to the hosts in the NS and MX records. In this example, we don't see an MX record in the ADDITIONAL SECTION, but if you perform an MX query on a domain, you see A records for the mail exchangers (highlighted in **red** here):

INTERACTIVE SESSION:

```
[username@username-m0 ~]$ dig google.com MX

; <<>> DiG 9.7.3-P3-RedHat-9.7.3-8.P3.el6_2.2 <<>> google.com MX
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12484
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 4,
ADDITIONAL: 9

;; QUESTION SECTION:
;google.com.                    IN      MX

;; ANSWER SECTION:
google.com.                 600        IN         MX         40
alt3.aspmx.l.google.com.
google.com.                 600        IN         MX         50
alt4.aspmx.l.google.com.
google.com.                 600        IN         MX         10
aspmx.l.google.com.
```

```
google.com.                     600        IN         MX         20
alt1.aspmx.l.google.com.
google.com.                     600        IN         MX         30
alt2.aspmx.l.google.com.

;; AUTHORITY SECTION:
google.com.             309107  IN      NS      ns3.google.com.
google.com.             309107  IN      NS      ns4.google.com.
google.com.             309107  IN      NS      ns1.google.com.
google.com.             309107  IN      NS      ns2.google.com.

;; ADDITIONAL SECTION:
aspmx.l.google.com.     293     IN      A       74.125.53.26
alt1.aspmx.l.google.com. 293    IN      A       74.125.45.26
alt2.aspmx.l.google.com. 293    IN      A       74.125.113.26
alt3.aspmx.l.google.com. 293    IN      A       173.194.67.26
alt4.aspmx.l.google.com. 293    IN      A       74.125.79.26
ns1.google.com.         40309   IN      A       216.239.32.10
ns2.google.com.         40309   IN      A       216.239.34.10
ns3.google.com.         40309   IN      A       216.239.36.10
ns4.google.com.         40309   IN      A       216.239.38.10

;; Query time: 41 msec
;; SERVER: 172.16.0.1#53(172.16.0.1)
;; WHEN: Mon Mar  5 15:03:24 2012
;; MSG SIZE  rcvd: 352
```

As I mentioned earlier, it's possible to look up a host name based on an IP address. Give it a try using the **host** command to look up 199.27.144.89—you'll get "www.oreillyschool.com." This is called a *reverse lookup*. Now, try using dig to do the same thing. It doesn't work, does it? In the QUESTION SECTION, you'll see that dig has made a query for an A record, but A records are used to map host names to IP addresses, not the other way around. There is a special kind of DNS record for mapping IP addresses to host names: the PTR record. Try using dig to make a PTR query for 199.27.144.89. Still no answer! What's going on here? Let's take a step back and discuss the inner workings of DNS queries.

When you ask for the A record for "www.oreillyschool.com," your DNS server doesn't know who the authoritative DNS server is for the oreillyschool.com domain. In order to find out that information, the DNS server must query the *root servers* first. This group of servers holds DNS information for all of the *top-level domains*, such as ".com," ".net," ".gov," and so on. Next, because "www.oreillyschool.com" is a part of the ".com" top-level domain, the root DNS server will tell your resolver to query a DNS server that is responsible for the ".com" domain. Then your resolver can query the ".com" server for the NS record for the "oreillyschool" domain. Now your resolver knows the authoritative nameserver for the "oreillyschool.com" domain, so it can make a query of that server for the host "www." The information that your resolver obtains in the process of resolving

your query is cached, so the next time you want to look something up in the ".com" domain, you won't have to ask the root servers who's responsible for ".com" first.

So what does this have to do with reverse lookups? Excellent question! You may have noticed that the series of DNS queries involved in resolving a host name starts at the broadest level at the right end of the host name, working to the most specific level at the left end of the host name. In order to make reverse lookups fit into this schema, the IP address must be reversed. That's not all, though. On the right side of the query, you must also append ".in-addr.arpa." The "arpa" domain is another top-level domain, and the "in-addr" subdomain is used specifically for executing IPv4 reverse DNS lookups. Once the query reaches the "in-addr" authoritative servers, it begins locating the authoritative nameservers for increasingly more specific IP address blocks. In our case, these would be 199.in-addr.arpa, 27.199.in-addr.arpa and then finally, 144.27.199.in-addr.arpa. When the resolver locates that last authoritative nameserver, then it can inquire about where 199.27.144.89 (or 89.144.27.199.in-addr.arpa) points. Here's what it looks like when we ask for the PTR record for 89.144.27.199.in-addr.arpa:

INTERACTIVE SESSION:

```
[username@username-m0 ~]$ dig 89.144.27.199.in-addr.arpa PTR

;     <<>>     DiG     9.7.3-P3-RedHat-9.7.3-8.P3.el6_2.2     <<>>
89.144.27.199.in-addr.arpa PTR
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62613
;;  flags:  qr  aa  rd  ra;  QUERY:  1,  ANSWER:  1,  AUTHORITY:  2,
ADDITIONAL: 2

;; QUESTION SECTION:
;89.144.27.199.in-addr.arpa.    IN      PTR

;; ANSWER SECTION:
89.144.27.199.in-addr.arpa.    600    IN                      PTR
www.oreillyschool.com.

;; AUTHORITY SECTION:
144.27.199.in-addr.arpa.    600            IN                    NS
ns2.useractive.com.
144.27.199.in-addr.arpa.    600            IN                    NS
ns1.useractive.com.

;; ADDITIONAL SECTION:
ns1.useractive.com.     600     IN      A       199.27.148.66
ns2.useractive.com.     600     IN      A       199.27.148.67

;; Query time: 27 msec
;; SERVER: 172.16.0.1#53(172.16.0.1)
```

```
;; WHEN: Mon Mar  5 15:45:04 2012
;; MSG SIZE  rcvd: 158
```

Take a few minutes and use **dig** to go through the same process manually that your resolver would go through automatically when it tries to locate the PTR record for "208.201.239.100." That is, first query the "arpa" top level domain for the NS record for "in-addr.arpa", then ask the "in-addr.arpa" nameserver for the NS record for "208.in-addr.arpa" and so on. Practice using and reading the output of dig. Remember, there may be multiple authoritative nameservers for a particular domain.

Another useful tool is **nslookup**. In addition to the servers that you have defined, it gives you the option of querying a specific DNS server that may not be defined in your resolv.conf file. By default, running **nslookup** with a hostname causes it to behave just like any of the other DNS utilities we've discussed so far (that is, it queries your configured DNS server). However, when invoked as **nslookup *hostname dns-server***, nslookup will use the server you specify. First, we will try using nslookup without specifying a nameserver.

INTERACTIVE SESSION:

```
[username@username-m0 ~]$ nslookup www.oreillyschool.com
Server:         172.16.0.1
Address:        172.16.0.1#53

Non-authoritative answer:
www.oreillyschool.com   canonical name = blogs.oreilly.com.
Name:   blogs.oreilly.com
Address: 67.222.96.148
```

As you can see in our example, in addition to the answer it received, **nslookup** tells you which DNS server it used to complete the query. A particularly interesting bit of output has been highlighted in **red**. By querying a DNS server other than the O'Reilly School of Technology's DNS server, we have gotten a *Non-authoritative answer*. Now would be a great time to figure out what the authoritative nameservers for the oreillyschool.com domain are! We can then use this information with nslookup to get an authoritative answer. We will use dig again for this purpose, but this time we will be asking for NS records. The important information is highlighted in **red**:

INTERACTIVE SESSION:

```
[username@username-m0 ~]$ dig oreillyschool.com NS

;     <<>>     DiG     9.7.3-P3-RedHat-9.7.3-8.P3.el6_2.2     <<>>
oreillyschool.com NS
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62613
;;  flags:  qr  rd  ra;  QUERY:  1,  ANSWER:  3,  AUTHORITY:  0,
ADDITIONAL: 3
```

```
;; QUESTION SECTION:
;oreillyschool.com.                IN      NS

;; ANSWER SECTION:
oreillyschool.com.                600             IN                NS
ns1.useractive.com.
oreillyschool.com.                600             IN                NS
ns2.useractive.com.
oreillyschool.com.                600             IN                NS
ns3.useractive.com.

;; ADDITIONAL SECTION:
ns1.useractive.com.     78839   IN      A        199.27.148.66
ns2.useractive.com.     78839   IN      A        199.27.148.67
ns3.useractive.com.     78839   IN      A        54.225.197.215

;; Query time: 13 msec
;; SERVER: 172.16.0.1#53(172.16.0.1)
;; WHEN: Thu Oct 17 12:58:07 2013
;; MSG SIZE  rcvd: 148
```

Looking at our dig query here, we have a list of nameservers in the ANSWER SECTION. Luckily, dig also gives us the A records for each nameserver in the ADDITIONAL SECTION, so we can see what their IP addresses are. We can take this information and plug it in to an nslookup command to find an authoritative answer:

INTERACTIVE SESSION:

```
[username@username-m0    ~]$    nslookup    www.oreillyschool.com
199.27.148.66
Server:         199.27.148.66
Address:        199.27.148.66#53

www.oreillyschool.com   canonical name = blogs.oreilly.com.
Name:   blogs.oreilly.com
Address: 67.222.96.148
```

This time we do not get a message about having a non-authoritative answer.

Source: http://courses.oreillyschool.com/sysadmin2/dns.html