DNS Tips and Tricks

DNS makes the world go round, it's also pretty interesting stuff. Typically DNS just works, but there are times when you want to do something special or you're not getting the results you think you should. This document is for those times.

1. DiG

Are you still using **nslookup** to query DNS? Stop it! The **nslookup** utility has some basic functionality but once you get beyond simple resolution you really want to start using **dig** from the ISC BIND (http://www.isc.org/software/bind) software package. You'll see that most of the sections in this document work with **dig**.

2. Who's Your DNS?

Do you know what your DNS server is? That's easy, just run **cat /etc/resolv.conf**. Except that you're using DHCP which will often set the DNS server to localhost (127.0.0.1). You can just run **dig 127.0.0.1 | grep ';; SERVER'**, but that essentially displays the same thing. OK, we know that we've got DHCP enabled, and DHCP is setting the DNS server, so we just need to find the DHCP lease. Let's look in `/var/lib/dhcp/`, find the newest file (since we may have leases from other networks), find the DNS server specification, and figure out the latest entry using the svelte command **cat $(find /var/lib/dhcp/ -maxdepth 1 -type f -printf "%T@ %p\n" | sort -n | tail -n 1 | cut -d' ' -f 2-) | grep domain-name-servers | tail -n1**. And our DNS server is...our router at 192.168.0.1? Or maybe our corporate DNS server at 10.a.b.c? That's not right.

Even if you have access to the router you're talking to (which you do at home, but not always at work) you may be surprised to find that the IP address listed there isn't the IP address that your DNS server actually is configured with. Let's take Google's DNS service (https://developers.google.com/speed/public-dns/) as an example. You set up your DNS server to be 8.8.8.8, however this is an Anycast IP Address (http://en.wikipedia.org/wiki/Anycast) which means that the request is forwarded to a local datacenter. If you're in New York, 8.8.8.8 probably goes to their North Carolina datacenter. If you're in Stockholm, it goes to Hamina, Finland. Even with an IP address you have no idea where your DNS server actually is.

Why does this matter? Many companies use your DNS server's location to figure out where to map you to. If you're sitting in Arizona and the website you're trying to reach has datacenters in NYC and LA you'll probably be sent to the LA datacenter. The problem is centralized DNS. If you're at work in your office in California, but your corporate headquarters is in London there's a decent chance all of your

DNS requests go out of the London office. This means that companies who use DNS to choose the closest datacenter will map you to London instead of California and your performance will suffer. So knowing your DNS server is the first step towards troubleshooting performance problems.

Especially if you're having performance problems with a Content Delivery Network like Akamai's. This is common enough that Akamai has a tool to help you figure out your DNS server. If you run **dig +short whoami.akamai.net** the response you get will be the IP address Akamai saw your request from. You may want to try this a few times, a lot of people use a round robin DNS rotation so you may see it appearing from various parts of the Internet randomly.

An adjunct tool is **curl whatismyip.akamai.com** (or the less recommended **wget -qO-whatismyip.akamai.com** - curl:wget::dig:nslookup) which will show your public IP address. To find a rough guess as to the performance you get on CDNs you can use a tool like this one (http://www.ip-adress.com/ipaddressdistance/) to figure out how far away your IP address is from your DNS. It's not quite as accurate as Akamai's own EdgeScape (http://www.akamai.com/html/technology/products/edgescape.html) service but it is slightly cheaper...

3. EDNS Client Subnet

Google's mission is to make the Internet faster. They're also a sort of Content Delivery Network of their own, they will route you to their nearest datacenter (http://www.google.com/about/datacenters/locations/index.html). They understand the centralized DNS issue, and Wilmer van der Gaast (https://plus.google.com/117161704068825702652) submitted an IETF draft (http://www.afasterinternet.com/ietfdraft.htm) to help fix this problem. I encourage you to check out the A Faster Internet (http://www.afasterinternet.com/howitworks.htm) webpage to figure out how this works.

Since nobody actually reads the link, in a nutshell let's say you use Google DNS. You send a DNS request to 8.8.8.8 and it reaches the nearest Google server. Problem is you're in South Africa and it hits the Belgium datacenter. Google's DNS server sees that you're looking for www.acceleratedwebsite.com so it asks the DNS server at acceleratedwebsite.com where to go. Since acceleratedwebsite.com sees a Belgium IP address they route you to their Paris datacenter.

Now let's assume that acceleratedwebsite.com supports this edns-client-subnet option. When they get the DNS request from Google, they still see a Belgium IP address but they also see a subnet representing your South African location. They use this South African subnet to route you to their server in Johannesburg.

The problem with this solution is that your local DNS server needs to support it and whatever is making the decision where you should be mapped needs to support it. Your local ISP probably doesn't support this extension, but they also probably put their DNS server very close to you so it doesn't matter. Your work DNS probably doesn't support this either, and they may have a centralized DNS server. However, both Google DNS and OpenDNS do support this.

It also needs buy-in from the service making the mapping decision. Most home-grown systems use a Global Traffic Manager (GTM) solution from a vendor who probably doesn't support this draft yet. Unfortunately, the largest CDN player out there (Akamai) also doesn't support it. However, some players like CDNetworks and others in this list (http://www.afasterinternet.com/participants.htm) do. If the website you're trying to reach uses one of these participants and you're using Google DNS or OpenDNS you'll be reasonably certain of getting a good mapping.

The one thing that nobody can fix is bad routing in corporate networks. Some offices actually route all traffic to HQ before it reaches the Internet. The best you can do then is to be mapped close to headquarters. Most VPNs also don't support split-horizon routing, this means that if you're traveling halfway around the world you may get reasonable downloads until you connect to your VPN at which point all traffic goes to the VPN access concentrator you connected to. To help fix this, I don't use a VPN but instead rely on an SSH tunnel with a SOCKS proxy - but I'm in a minority who have the knowledge to do so and who work at a company with a public SSH server.

3.1. Building Dig

Want to see if your DNS supports the Google extension? Or just want to figure out which server someone else would be routed to? You can recompile dig to do this. I'm going to assume Ubuntu Precise for this, but it should work well with minor adjustments on most distributions.

First you'll need to set up the dependencies. Run **sudo apt-get install build-essential libssl-dev** to get started.

Now you can download the BIND source here (https://www.isc.org/software/bind/) and the patch here (http://wilmer.gaa.st/edns-client-subnet/bind-9.7.3-dig-edns-client-subnet.diff). Create a directory someplace and copy these files to it. Now run these commands:

```
tar -vzxf bind-9.9.1*.tar.gz
cd bind-9.9.1*/
patch -p0 -F3 < ../bind-9.7.3-dig-edns-client-subnet.diff
./configure
make
```

You can now run **bin/dig/dig @ns1.google.com www.google.com +client=130.89.89.130** to verify that it works. If you like it, you can copy it to your path, but make sure you either replace the existing **dig** (which means if Ubuntu updates it for any reason you'll get the Ubuntu flavor) or put it someplace before `/usr/bin/` in your path otherwise you'll be running the built-in version. Another option is to rename it to **dig-client** to keep things separate.

Once you've moved the binary file out you can delete the directory with the patch and source code.

A. About Me

My name is Jeff Bower, I'm a technology professional (http://www.linkedin.com/in/jdbower) with more years of experience in the telecommunications industry than I'd care to admit. I tend to post with the username jdbower on various forums, including various Android, virtualization, BBQ, and Linux forums around the Internet. Writing these documents is a hobby of mine, I hope you find them useful and feel free to browse more at https://www.ebower.com/docs.

I also enjoy cooking, especially outdoors with my Komodo Kamado (http://www.komodokamado.com) and using my Stoker (https://www.rocksbarbque.com). Take a look at my recipes stored at https://www.ebower.com/recipes.

If you've got any questions or feedback please feel free to email me at docs@ebower.com (mailto:docs@ebower.com) or follow me on Google+ (https://profiles.google.com/100268310848930740059).