DNS and BIND Primer

Pete Nesbitt
pete @ linux1.ca

April 2012

When we access the Internet we typically do so by accessing systems using a somewhat meaningful **hostname** often in the form of a web based URL. Network and computer equipment reference a system by it's **IP Address**, and in some cases by the machines Physical Address which is not covered in this article. This DNS primer is intended to shed some light on the workings behind resolving a domain name to an IP Address.

Domain Name System (DNS) is the system that matches host names to IP addresses and sometimes the reverse. DNS allows a client program, called a resolver, to locate the IP address of any device that is defined on a DNS server anywhere on the Internet or any other accessible DNS managed network. Understanding the fundamentals of how DNS works is essential for any administrator in troubleshooting network issues, and in the case of a user just understanding why things fail can relieve some of the stress.

Like a good Systems Administrator, when everything is working as expected, DNS is rarely on anyone's mind. It's a different story when DNS fails and you can't access any of your favourite sites. Even accessing by IP Address these days can fail because many web servers host a number of different web sites based on the host name within the request.

Berkeley Internet Name Domain (BIND) is the dominant name server used on the Internet and other networks.

DNS Organization

Domain Names are a sequence of at least two sections joined by a '.' (dot). In most cases the left most part is a computer hostname, the most commonly seen when browsing is *www*. Sometimes the hostname is not given, just the domain name like *example.com* which is somewhat misleading but simpler to read and type. In these cases the domain itself is represented by the same IP address as one of it's hosts. So entering example.com in your browser would be the same as entering www.example.com as that is the server you will be accessing.

In large scale environments several web servers are often represented by a single public IP address. In these cases the requests are load balanced among

the group of servers. This can be achieved through DNS entries or more often managed by a hardware device such as F5 Networks Big-IP. [1]

The information stored on a DNS server is broken into *DNS Zones*. *Zone Files*, typically one file for each domain the server is responsible for (see *Authority* section page 5), come in two formats. The Primary Master server for the domain (see *Name Servers* section page 5) has the original file, often directly edited by the DNS Administrator. This file can include comment lines and spacing for readability. There is also an auto generated version of the zone file created on the master and sent to the slave servers. There are a few mechanisms related to how the slaves are notified of the change but what is important is that changes are automatically distributed to all the servers answering DNS requests for the given zone.

What's in a Name?

For resolution purposes, host names are read from right to left. Similar to finding a city on a world map where you first must find the country in which it is located, in DNS the first information needed is which Domain Server has the records for that top level. For this we turn to the root servers, like a world map shows countries, they tell us where to find the server for the *right most* part, such as .com .org etc. With that information the request is sent again but to the server returned by the root server query. This process continues until we have drilled down to the target name/IP match. That information is passed back to the client which can proceed with it's task like loading the web page, the process which started the whole DNS request. This process is called *recursion* or a *recursive query* and is usually conducted between a users first listed name server and the remote servers, not by the users DNS client itself. Although there can be many requests and responses exchanged to resolve a given host on the Internet, the whole process typically takes only a fraction of a second.

DNS Tree:

```
root servers
  top level domains
    second level domains
      third level domains
        . . .
          local domain for the target host
            local host
```

Here is an explanation of the above listing

root servers -root servers are public servers that know about the top level domain servers. Although they are represented by a trailing dot in a URL, the '.' is implied and generally not required.

top level domains -these are the right hand side of the name, such as com or net

second level domains -these often represent an organization, such as example.com

third level domains -third level domains are the first ones that can be a domain or if it is also the *left most* part it could represents a computer.

additional levels -there can be any number of subdomains depending on an organizations needs.

local domain -this is the most qualified domain that a computer is a member of. It can be any level other than root or top, but is always the *second from left most* part.

local host -the *left most* part is usually a computer name, although it could be a domain. Internally, an operating system can be referenced as localhost.localdomain instead of it's formal name.

Name Server Types

Master Name Server -gets zone information from local files
-formerly called the Primary Name Server, sometimes called a Primary Master

Slave Name Server -gets zone information via a zone transfer from an authoritative server, usually a master
-formerly called the Secondary Name Server, sometimes referred to as a Secondary Master

Caching-only Name Servers -not authoritative for any zones
-have no zone files only cached information

There are some other types of Name Servers such as Forward-Only and Stealth but these are not accessed by the general public and are usually part of corporate or private networks.

There are some internal zones that are not discussed here because they apply to the computer itself. These are referred to as localhost.localdomain and 127.0.0.0

Authority

When you do a look-up with DNS tools such as *dig* you may notice the results include information about *authority*. A DNS Server is considered to be Authoritative if it has local zone files containing the DNS records for the request. It is considered Non-Authoritative if it does not have the zone file for the requested zone. Regardless of whether or not the requested information is in it's DNS cache, if the server does not have the zone file then the reply is non-authoritative.

Expiring Zone Data

To make all this interaction work properly there are mechanisms to help reduce the chances of having different versions of zone data cached on DNS servers

around the globe. Although there are a number of aspects and solutions to this issue, the main players are expiry times.

Here are some key times recorded in the zone file:

- time to start checking for updated zones (used by Slave Server)
- time to discard zone data because it is too old and no Master could be contacted (used by Slave Server)
- time to expire cached records (used by Caching-only Server)
- time to wait before resending a failed lookup where server reported hostname does not exist (used by Caching-only Server)

The purpose of the the last item above, known as a "negative cache" time is so in cases like a typo in a web link, remote servers don't continually request the same bad hostname.

It's important to remember that a Non-Authoritative answer, even within it's expiry time, may not reflect the latest changes to the zone records if they have been recently modified.

These expiry times vary depending on the expected rate of change to data on the target server but some typical times are 1 hour or 24 hours or even two weeks. There are no set rules around these times which are set by the DNS Administrator (or possibly some BIND frontend).

It is possible for a DNS Slave (a slave is authoritative) to return the wrong information if it has not received an updated copy of the zone file from the Master, however this is not usually an issue as BIND looks after this aspect and a DNS Administrator also has a few ways to force updates of critical changes.

master and slave servers are considered to be *authoritative* for the zone

any name server can serve a mix of primary and slave roles for different zones

all name servers cache their query results. This increases performance by magnitudes over reading a file.

Zone File Types

Forward -normal resolution, matches a known hostname to an unknown IP address

Reverse -matches a known IP address to an unknown hostname

Answer Types

Authoritative -answer is contained in servers zone files (master or slave) although response comes from it's cached data for performance reasons

Non-Authoritative -answer comes from the name servers cache which is populated by replies to queries it has sent to Authoritative servers
-these servers are most often used in private networks to improve performance and reduce outbound DNS requests

Name Resolution Process

1. The resolver (dns client) sends a DNS query to the first name server listed in the resolv.conf file (Linux/Unix).
   If there is no response in n seconds (see chart page 9, try next server in resolv.conf. . .

2. If the name server is authoritative or has a cached answer it answers the query directly.

3. If the name server can not resolve the query, it sends a query to a root name server which is authoritative for or knows of authoritative servers for all top level domains.

4. The root server answers with a referral to the name of the top level domain server for the requested domain or possibly the next second-level domain name server.

5. The local name server then queries the name server it received from the last query and the process continues until the final answer is received. The answer is sent back to the client.

Recursion

Most client resolvers are not capable of following referrals, which is why the first name server will normally process all query referrals, passing the final answer back to the client resolver. This process is called DNS Recursion and is all done behind the scenes.

If you manually send a recursive query to a remote DNS server (say you ask IBM's servers to resolve www.microsoft.com) you will normally get an error similar to "WARNING: recursion requested but not available". This is a security aspect that prevents DoS and other possible issues. Most DNS Servers are configured to only handle recursive requests for a fixed group of IP addresses such as the companies internal network.

Common Tools

There are three primary tools used to interact with DNS servers at a command line

dig this is a very full featured tool which can be found on any Linux system.

host this is the simplest tool to use which presents clear, easy to understand results.

nslookup although available on most systems, it can be a bit tricky to use and generally considered deprecated.

In their simplest form, each can be used by entering the command followed by a host or domain name to look up.

e.g. 'host example.com'

See the individual Linux man pages for usage details (e.g. 'man dig').

Query Timing

Depending on how many name servers you have defined (up to three), it can take 15 to 24 seconds before a look-up fails and returns a "Host Not Found" error. The first listed server is queried, if it times out it retries once, then tries the next server, again with one retry. The time out periods are progressively reduced as it goes through the name servers defined in /etc/resolv.conf. To follow the process for the maximum 3 name server definitions, read this chart from top to bottom moving left to right.

```
BIND 8.2.1 and newer
   Retry         NS 1          NS 2           NS 3
   0              5 sec        (2x) 5 sec     (3x) 5 sec
   1             10 sec        (2x) 5 sec     (3x) 3 sec
   Total Time    15 seconds    20 seconds     24 seconds
```

O'Reilly DNS and BIND [2]

```
BIND 8.2.0 and earlier
   Retry        NS 1          NS 2           NS 3
   0             5 sec        (2x)  5 sec    (3x)  5 sec
   1            10 sec        (2x)  5 sec    (3x)  3 sec
   2            20 sec        (2x) 10 sec    (3x)  6 sec
   3            40 sec        (2x) 20 sec    (3x) 13 sec
   Total Time   75 seconds     80 seconds     81 seconds
```

O'Reilly DNS and BIND [3]

Note: in BIND versions prior to 8.2.1 the resolver sent 3 retries instead of just 1 (4 attempts including the initial request). Normally things resolve in a second or less but in a worst case scenario on an early versions of BIND Resolver it could take up to 81 seconds to give up.

Conclusion

DNS and the BIND server both play essential rolls in today's Internet and in many private networks around the world, so it is beneficial for anyone using a computer or networked device to at least have a general understanding of how it all works. Few services effect so many systems and users of those systems as much as DNS, which is involved virtually every time someone accesses a network.

Here the the key points we covered:

- a hostname is only there for human use and the IP Address is what matters to the machines attached to a network
- there are several types of DNS servers but they all do serve the same purpose which is mapping hostnames to IP addresses
- a domain name is comprised of two or more sections separated by a dot and they are interpreted right to left
- domain and host names are matched to their IP Address by searching in a top-down cycle of queries and responses

The BIND DNS Server is very configurable and the DNS system is complex in it's details. Whether you see a future in Systems Administration or just want to know, the best way to understand BIND is to install a BIND Server on a test Linux server.

DNS for Rocket Scientists [4] is a great resource for everything DNS or BIND.

I hope you found this paper of value and that maybe it has peeked your interest enough to continue to learn more about this essential service.

References

[1] F5 network big-ip. `http://www.f5.com/products/big-ip/`. Features include advanced Load Balancing.

[2] Cricket Liu and Paul Albitz. *DNS and BIND*. O'Reilly & Associates, Inc., 2006.

[3] Paul Albitz and Cricket Liu. *DNS and BIND*. O'Reilly & Associates, Inc., 1998.

[4] Dns for rocket scientists. `http://www.zytrax.com/books/dns/`. Extensive coverage of DNS and BIND configuration.