Anatomy of a Linux DNS Lookup – Part III

5 Minutes

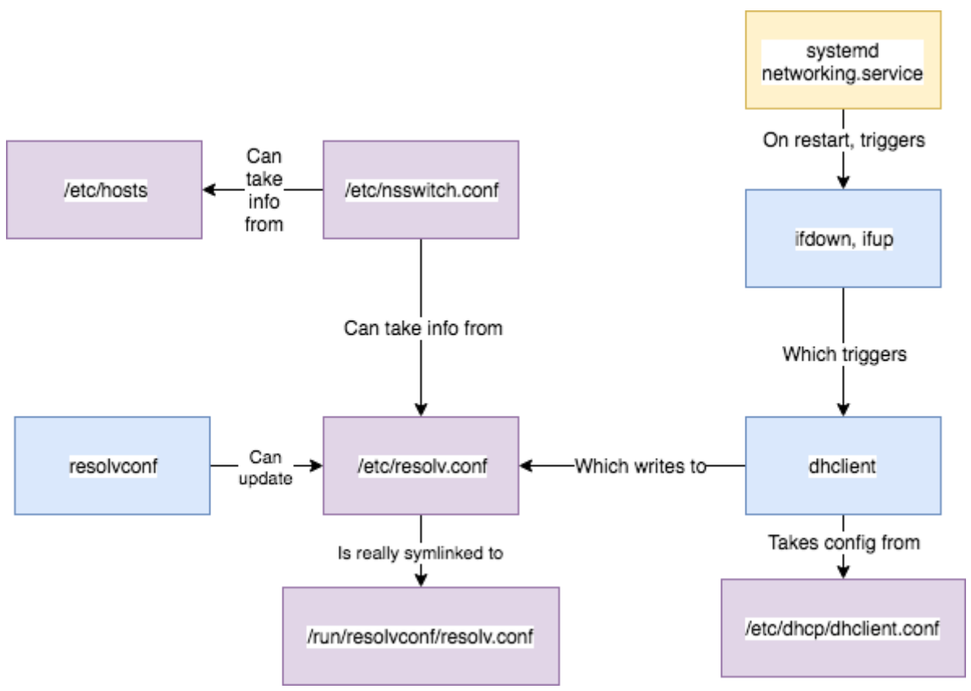
In Anatomy of a Linux DNS Lookup – Part I (https://zwischenzugs.com/2018/06/08/anatomy-of-a-linux-dns-lookup-part-i/) I covered:

- `nsswitch`
- `/etc/hosts`
- `/etc/resolv.conf`
- `ping` vs `host` style lookups

and in Anatomy of a Linux DNS Lookup – Part II (https://zwischenzugs.com/2018/06/18/anatomy-of-a-linux-dns-lookup-part-ii/) I covered:

- `systemd` and its `networking` service
- `ifup` and `ifdown`
- `dhclient`
- `resolvconf`

and ended up here:



A (roughly) accurate map of what's going on

Unfortunately, that's not the end of the story. There's still more things that can get involved. In Part III, I'm going to cover NetworkManager and dnsmasq and briefly show how they play a part.

Other posts in the series:

Anatomy of a Linux DNS Lookup – Part I (https://zwischenzugs.com/2018/06/08/anatomy-of-a-linux-dns-lookup-part-i/)

Anatomy of a Linux DNS Lookup – Part II (https://zwischenzugs.com/2018/06/18/anatomy-of-a-linux-dns-lookup-part-ii/)

Anatomy of a Linux DNS Lookup – Part IV (https://zwischenzugs.com/2018/08/06/anatomy-of-a-linux-dns-lookup-part-iv/)

Anatomy of a Linux DNS Lookup – Part V – Two Debug Nightmares (https://zwischenzugs.com/2018/09/13/anatomy-of-a-linux-dns-lookup-part-v-two-debug-nightmares/)

1) NetworkManager

As mentioned in Part II, we are now well away from POSIX standards and into Linux distribution-specific areas of DNS resolution management.

In my preferred distribution (Ubuntu), there is a service that's available and often installed for me as a dependency of some other package I install called NetworkManager (https://en.wikipedia.org/wiki/NetworkManager). It's actually a service developed by RedHat in 2004 to help manage network interfaces for you.

What does this have to do with DNS? Install it to find out:

```
$ apt-get install -y network-manager
```

In my distribution, I get a config file.

```
$ cat /etc/NetworkManager/NetworkManager.conf
[main]
plugins=ifupdown,keyfile,ofono
dns=dnsmasq

[ifupdown]
managed=false
```

See that `dns=dnsmasq` there? That means that NetworkManager will use `dnsmasq` to manage DNS on the host.

2) dnsmasq

The dnsmasq program is that now-familiar thing: yet another level of indirection for `/etc/resolv.conf` .

Technically, dnsmasq can do a few things, but is primarily it acts as a DNS server that can cache requests to other DNS servers. It runs on port 53 (the standard DNS port), on all local network interfaces.

So where is `dnsmasq` running? NetworkManager is running:

```
$ ps -ef | grep NetworkManager
root     15048     1  0 16:39 ?        00:00:00 /usr/sbin/NetworkManager --no-daemon
```

But no `dnsmasq` process exists:

```
$ ps -ef | grep dnsmasq
$
```

Although it's configured to be used, confusingly it's not actually installed! So you're going to install it.

Before you install it though, let's check the state of `/etc/resolv.conf` .

```
$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 10.0.2.2
search home
```

It's not been changed by NetworkManager.

If `dnsmasq` is installed:

```
$ apt-get install -y dnsmasq
```

Then `dnsmasq` is up and running:

```
$ ps -ef | grep dnsmasq
dnsmasq  15286     1  0 16:54 ?        00:00:00 /usr/sbin/dnsmasq -x /var/run/dnsmasq/dnsmasq.pid -u dnsmasq -r /var/run/dnsmasq/resolv.conf -7 /etc/dnsmasq.d,.dpkg-dist
```

And `/etc/resolv.conf` has changed again!

```
root@linuxdns1:~# cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 127.0.0.1
search home
```

And `netstat` shows `dnsmasq` is serving on all interfaces at port 53:

```
$ netstat -nlp4
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address     Foreign Address State   PID/Program name
tcp        0      0 127.0.0.1:53      0.0.0.0:*       LISTEN  15286/dnsmasq
tcp        0      0 10.0.2.15:53      0.0.0.0:*       LISTEN  15286/dnsmasq
tcp        0      0 172.28.128.11:53  0.0.0.0:*       LISTEN  15286/dnsmasq
tcp        0      0 0.0.0.0:22        0.0.0.0:*       LISTEN  1237/sshd
udp        0      0 127.0.0.1:53      0.0.0.0:*               15286/dnsmasq
udp        0      0 10.0.2.15:53      0.0.0.0:*               15286/dnsmasq
udp        0      0 172.28.128.11:53  0.0.0.0:*               15286/dnsmasq
udp        0      0 0.0.0.0:68        0.0.0.0:*               10758/dhclient
udp        0      0 0.0.0.0:68        0.0.0.0:*               10530/dhclient
udp        0      0 0.0.0.0:68        0.0.0.0:*               10185/dhclient
```

3) Unpicking dnsmasq

Now we are in a situation where all DNS queries are going to `127.0.0.1:53` and from there what happens?

We can get a clue from looking again at the `/var/run` folder. The `resolv.conf` in `resolvconf` has been changed to point to where `dnsmasq` is being served:

```
$ cat /var/run/resolvconf/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 127.0.0.1
search home
```

while there's a new `dnsmasq` folder with its own `resolv.conf` .

```
$ cat /run/dnsmasq/resolv.conf
nameserver 10.0.2.2
```

which has the nameserver given to us by `DHCP`.

We can reason about this without looking too deeply, but what if we really want to know what's going on?

4) Debugging Dnsmasq

Frequently I've found myself wondering what dnsmasq's state is. Fortunately, you can get a good amount of information out of it if you set change this line in `/etc/dnsmasq.conf`:

```
#log-queries
```

to:

```
log-queries
```

and restart `dnsmasq`

Now, if you do a simple:

```
$ ping -c1 bbc.co.uk
```

you will see something like this in `/var/log/syslog` (the `[...]` indicates that the line's start is the same as the previous one):

```
Jul  3 19:56:07 ubuntu-xenial dnsmasq[15372]: query[A] bbc.co.uk from 127.0.0.1
[...] forwarded bbc.co.uk to 10.0.2.2
[...] reply bbc.co.uk is 151.101.192.81
[...] reply bbc.co.uk is 151.101.0.81
[...] reply bbc.co.uk is 151.101.64.81
[...] reply bbc.co.uk is 151.101.128.81
[...] query[PTR] 81.192.101.151.in-addr.arpa from 127.0.0.1
[...] forwarded 81.192.101.151.in-addr.arpa to 10.0.2.2
[...] reply 151.101.192.81 is NXDOMAIN
```

which shows what `dnsmasq` received, where the query was forwarded to, and what reply was received.

If the query is returned from the cache (or, more exactly, the local 'time-to-live' for the query has not expired), then it looks like this in the logs:

```
[...] query[A] bbc.co.uk from 127.0.0.1
[...] cached bbc.co.uk is 151.101.64.81
[...] cached bbc.co.uk is 151.101.128.81
[...] cached bbc.co.uk is 151.101.192.81
[...] cached bbc.co.uk is 151.101.0.81
[...] query[PTR] 81.64.101.151.in-addr.arpa from 127.0.0.1
```

and if you ever want to know what's in your cache, you can provoke dnsmasq into sending it to the same log file by sending the `USR1` signal to the dnsmasq process id:

```
$ kill -SIGUSR1 <(cat /run/dnsmasq/dnsmasq.pid)
```

and the output of the dump looks like this:

```
Jul  3 15:08:08 ubuntu-xenial dnsmasq[15697]: time 1530630488
[...] cache size 150, 0/5 cache insertions re-used unexpired cache entries.
[...] queries forwarded 2, queries answered locally 0
[...] queries for authoritative zones 0
[...] server 10.0.2.2#53: queries sent 2, retried or failed 0
[...] Host             Address         Flags      Expires
[...] linuxdns1        172.28.128.8    4FRI   H
[...] ip6-localhost    ::1             6FRI   H
[...] ip6-allhosts     ff02::3         6FRI   H
[...] ip6-localnet     fe00::          6FRI   H
[...] ip6-mcastprefix  ff00::          6FRI   H
[...] ip6-loopback     :               6F I   H
[...] ip6-allnodes     ff02:           6FRI   H
[...] bbc.co.uk        151.101.64.81   4F         Tue Jul  3 15:11:41 2018
[...] bbc.co.uk        151.101.192.81  4F         Tue Jul  3 15:11:41 2018
[...] bbc.co.uk        151.101.0.81    4F         Tue Jul  3 15:11:41 2018

[...] bbc.co.uk        151.101.128.81  4F         Tue Jul  3 15:11:41 2018
[...]                  151.101.64.81   4 R  NX    Tue Jul  3 15:34:17 2018
[...] localhost        127.0.0.1       4FRI   H
[...] <Root>           19036   8   2   SF I
[...] ip6-allrouters   ff02::2         6FRI   H
```

In the above output, I believe (but don't know, and '?' indicates a relatively wild guess on my part) that:

- '4' means IPv4
- '6' means IPv6

- ‘H’ means address was read from an `/etc/hosts` file
- ‘I’ ? ‘Immortal’ DNS value? (ie no time-to-live value?)
- ‘F’ ?
- ‘R’ ?
- ‘S’?
- ‘N’?
- ‘X’

Alternatives to dnsmasq

`dnsmasq` is not the only option that can be passed to dns in NetworkManager. There’s `none` which does nothing to `/etc/resolv,conf` , `default` , which claims to ‘update `resolv.conf` to reflect currently active connections’, and `unbound` , which communicates with the `unbound` service and `dnssec-triggerd` , which is concerned with DNS security and is not covered here.

End of Part III

That’s the end of Part III, where we covered the NetworkManager service, and its `dns=dnsmasq` setting.

Let’s briefly list some of the things we’ve come across so far:

- `nsswitch`
- `/etc/hosts`
- `/etc/resolv.conf`
- `/run/resolvconf/resolv.conf`
- `systemd` and its `networking` service
- `ifup` and `ifdown`
- `dhclient`
- `resolvconf`
- `NetworkManager`
- `dnsmasq`

7 thoughts on “Anatomy of a Linux DNS Lookup – Part III”

Pingback: Weekly DevOps News #1 - Codeogre
Pingback: Anatomy of a Linux DNS Lookup – Part IV – zwischenzugs
Pingback: DNS lookup tutorial (3) | 0ddn1x: tricks with *nix
andyssq says:
August 29, 2018 at 5:49 am
For the cache flags, you may refer to the cache.c file with github link https://github.com/imp/dnsmasq/blob/master/src/cache.c.

cache->flags & F_FORWARD ? “F” : ” “,
cache->flags & F_REVERSE ? “R” : ” “,
cache->flags & F_IMMORTAL ? “I” : ” “,
cache->flags & F_DHCP ? “D” : ” “,
cache->flags & F_NEG ? “N” : ” “,
cache->flags & F_NXDOMAIN ? “X” : ” “,
cache->flags & F_HOSTS ? “H” : ” “,
cache->flags & F_DNSSECOK ? “V” : ” “);

↳ Reply
Pingback: Anatomy of a Linux DNS Lookup – Part V – Two Debug Nightmares – zwischenzugs
Pingback: Anatomy of a Linux DNS Lookup – Part I – zwischenzugs
Pingback: Anatomy of a Linux DNS Lookup – Part II – zwischenzugs

This site uses Akismet to reduce spam. Learn how your comment data is processed.