Anatomy of a Linux DNS Lookup – Part II

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 determined that most programs reference `/etc/resolv.conf` along the way to figuring out which DNS server to look up.

That stuff was more general linux behaviour **(*)** but here we move firmly into distribution-specific territory. I use ubuntu, but a lot of this will overlap with Debian and even CentOS-based distributions, and also differ from earlier or later Ubuntu versions.

(*) in fact, it's subject to a POSIX standard, so
is not limited to Linux (I learned this from
a fantastic comment (https://zwischenzugs.com/2018/06/08/anatomy-of-a-linux-dns-lookup-part-i/#comment-2312) on the previous post)
In other words: your host is more likely to differ in its behaviour in specifics from here.

In Part II I'll cover how `resolv.conf` can get updated, what happens when `systemctl restart networking` is run, and how `dhclient` gets involved.

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 III (https://zwischenzugs.com/2018/07/06/anatomy-of-a-linux-dns-lookup-part-iii/)

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) Updating /etc/resolv.conf by hand

We know that `/etc/resolv.conf` is (highly likely to be) referenced, so surely you can just add a nameserver to that file, and then your host will use that nameserver in addition to the others, right?

If you try that:

```
$ echo nameserver 10.10.10.10 >> /etc/resolv.conf
```

it all looks good:

```
# 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 10.0.2.3
search home
nameserver 10.10.10.10
```

until the network is restarted:

```
$ systemctl restart networking
$ 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 10.0.2.3
search home
```

our `10.10.10.10` nameserver has gone!

This is where those comments we ignored in Part I come in…

2) resolvconf

You see the phrase `generated by resolvconf` in the `/etc/resolv.conf` file above? This is our clue.

If you dig into what `systemctl restart networking` does, among many other things, it ends up calling a script: `/etc/network/if-up.d/000resolvconf` . Within this script is a call to `resolvconf` :

```
/sbin/resolvconf -a "${IFACE}.${ADDRFAM}"
```

A little digging through the man pages reveals that the `-a` flag allows us to:

```
Add or overwrite the record IFACE.PROG then run the update scripts
if updating is enabled.
```

So maybe we can call this directly to add a nameserver:

```
echo 'nameserver 10.10.10.10' | /sbin/resolvconf -a enp0s8.inet
```

Turns out we can!

```
$ cat /etc/resolv.conf  | grep nameserver
nameserver 10.0.2.3
nameserver 10.10.10.10
```

So we're done now, right? This is how `/etc/resolv.conf` gets updated? Calling `resolvconf` adds it to a database somewhere, and then updates (if configured, whatever that means) the `resolv.conf` file

No.

```
$ systemctl restart networking
root@linuxdns1:/etc# 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 10.0.2.3
search home
```

Argh! It's gone again.

So `systemctl restart networking` does more than just run `resolvconf` . It must be getting the nameserver information from somewhere else. Where?

3) ifup/ifdown

Digging further into what `systemctl restart networking` does tells us a couple of things:

```
cat /lib/systemd/system/networking.service
[...]
[Service]
Type=oneshot
EnvironmentFile=-/etc/default/networking
ExecStartPre=-/bin/sh -c '[ "$CONFIGURE_INTERFACES" != "no" ] && [ -n "$(ifquery --read-environment --list --exclude=lo)" ] && udevadm settle'
ExecStart=/sbin/ifup -a --read-environment
ExecStop=/sbin/ifdown -a --read-environment --exclude=lo
[...]
```

First, the networking 'service' restart is actually a 'oneshot' script that runs these commands:

```
/sbin/ifdown -a --read-environment --exclude=lo
/bin/sh -c '[ "$CONFIGURE_INTERFACES" != "no" ] && [ -n "$(ifquery --read-environment --list --exclude=lo)" ] && udevadm settle'
/sbin/ifup -a --read-environment
```

The first line with `ifdown` brings down all the network interfaces (but excludes the local interface). **(*)**

(*) I'm unclear why this doesn't boot me out of my
vagrant session in my example code (anyone know?).

The second line makes sure the system has done all it needs to do regarding the bringing of network interfaces down before going ahead and bringing them all back up with `ifup` in the third line. So the second thing we learn is that `ifup` and `ifdown` are what the networking service 'actually' runs.

The `--read-environment` flag is undocumented, and is there so that `systemctl` can play nice with it. A lot of people hate `systemctl` for this kind of thing.

Great. So what does `ifup` (and its twin, `ifdown`) do? To cut another long story short, it runs all the scripts in `etc/network/if-pre-up.d/` and `/etc/network/if-up.d/` . These in turn might run other scripts, and so on.

One of the things it does (and I'm still not quite sure how – maybe `udev` is involved?) `dhclient` gets run.

4) dhclient

`dhclient` is a program that interacts with DHCP servers to negotiate the details of what IP address the network interface specified should use. It also can receive a DNS nameserver to use, which then gets placed in the `/etc/resolv.conf` .

Let's cut to the chase and simulate what it does, but just on the `enp0s3` interface on my example VM, having first removed the nameserver from the `/etc/resolv.conf` file:

```
$ sed -i '/nameserver.*/d' /run/resolvconf/resolv.conf
$ cat /etc/resolv.conf | grep nameserver
$ dhclient -r enp0s3 && dhclient -v enp0s3
Killed old client process
Internet Systems Consortium DHCP Client 4.3.3
Copyright 2004-2015 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Listening on LPF/enp0s8/08:00:27:1c:85:19
Sending on   LPF/enp0s8/08:00:27:1c:85:19
Sending on   Socket/fallback
DHCPDISCOVER on enp0s8 to 255.255.255.255 port 67 interval 3 (xid=0xf2f2513e)
DHCPREQUEST of 172.28.128.3 on enp0s8 to 255.255.255.255 port 67 (xid=0x3e51f2f2)
DHCPOFFER of 172.28.128.3 from 172.28.128.2
DHCPACK of 172.28.128.3 from 172.28.128.2
bound to 172.28.128.3 -- renewal in 519 seconds.

$ cat /etc/resolv.conf | grep nameserver
nameserver 10.0.2.3
```

So that's where the nameserver comes from…

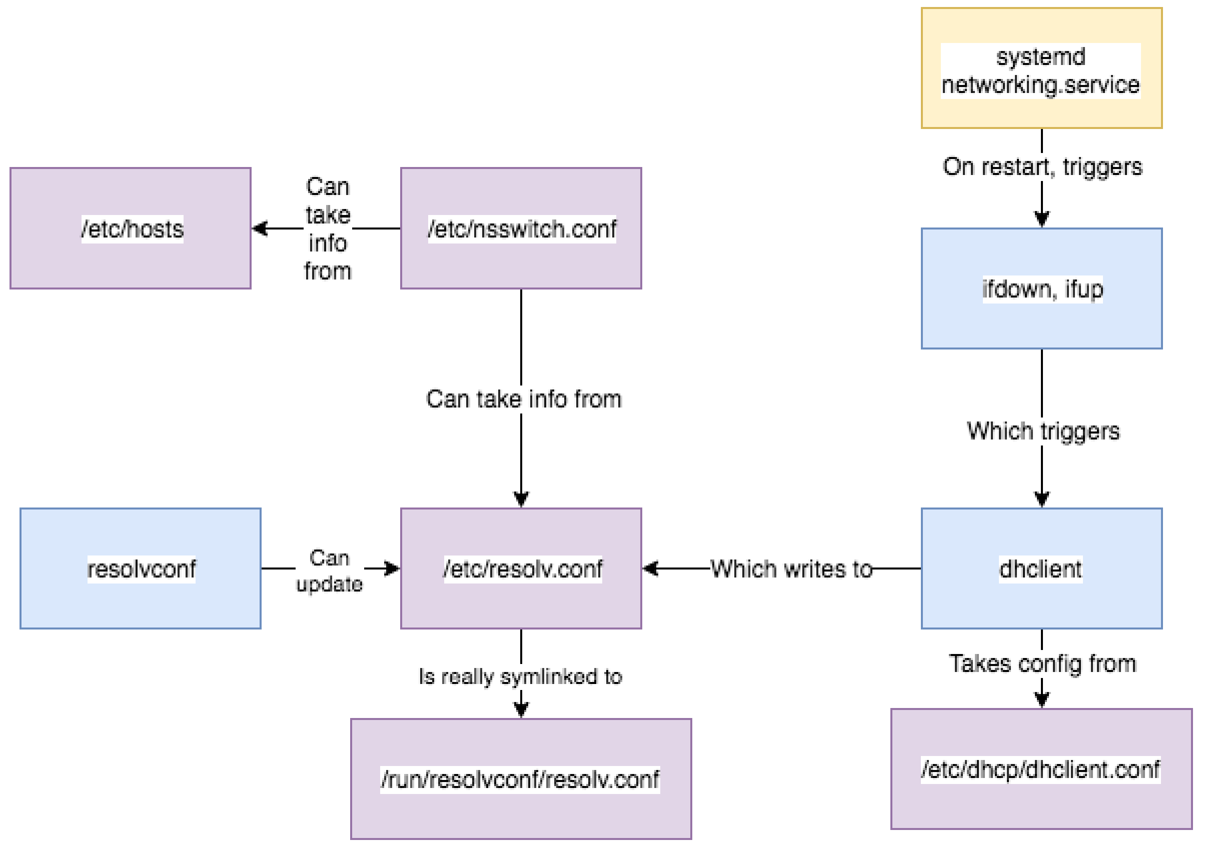
But hang on a sec – what's that `/run/resolvconf/resolv.conf` doing there, when it should be `/etc/resolv.conf` ?

Well, it turns out that `/etc/resolv.conf` isn't always 'just' a file.

On my VM, it's a symlink to the 'real' file stored in `/run/resolvconf` . This is a clue that the file is constructed at run time, and one of the reasons we're told not to edit the file directly.

If the `sed` command above were to be run on the `/etc/resolv.conf` file directly then the behaviour above would be different and a warning thrown about `/etc/resolv.conf` not being a symlink (`sed -i` doesn't handle symlinks cleverly – it just creates a fresh file).

`dhclient` offers the capability to override the DNS server given to you by DHCP if you dig a bit deeper into the `supersede` setting in `/etc/dhcp/dhclient.conf` …



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

End of Part II

That's the end of Part II. Believe it or not that was a somewhat simplified version of what goes on, but I tried to keep it to the important and 'useful to know' stuff so you wouldn't fall asleep. Most of that detail is around the twists and turns of the scripts that actually get run.

And we're still not done yet. Part III will look at even more layers on top of these.

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`

Published by zwischenzugs

View all posts by zwischenzugs

33 thoughts on “Anatomy of a Linux DNS Lookup – Part II”

Pingback: Recommended Read: Anatomy of a Linux DNS Lookup – Part II | thechrisshort
Nick Huanca says:
June 19, 2018 at 6:29 pm
Is there something like this article for how dns Is traversed in Kubernetes? KubeDNS vs CoreDNS etc?

↪ Reply
zwischenzugs says:
June 19, 2018 at 6:35 pm
There’s another article I wrote a while ago on DNS in OpenShift that might be relevant while you wait…

↪ Reply
Pingback: New top story on Hacker News: Anatomy of a DNS Lookup – Part II – Tech + Hckr News
Pingback: New top story on Hacker News: Anatomy of a DNS Lookup – Part II – World Best News
Pingback: New top story on Hacker News: Anatomy of a DNS Lookup – Part II – News about world
Pingback: Anatomy of a DNS Lookup – Part II
Pingback: New top story on Hacker News: Anatomy of a DNS Lookup – Part II – Latest news
Pingback: New top story on Hacker News: Anatomy of a DNS Lookup – Part II | Do Mithay Bol
Anonymous says:
June 19, 2018 at 8:33 pm
cat | grep? you can just grep, man! Nice write up!

↪ Reply
Omachonu Ogali says:
June 20, 2018 at 12:07 am
You’re not “booted” because the commands happen in rapid succession — fast enough that your TCP connection remained intact (aside from one retransmitted packet).

If you were to put a ‘sleep 20’ in there, you’d likely get booted.

↪ Reply
Dave Anderson (@davecanderson) says:
June 20, 2018 at 1:35 am
“The first line with ifup brings down all the network interfaces” shouldn’t this be ifdown?

↪ Reply
Pingback: New top story on Hacker News: Anatomy of a DNS Lookup – Part II – ÇlusterAssets Inc.,
Dave says:
June 22, 2018 at 7:52 am
You should write about local dns caching in part 3. It can be rather weird with timeouts, multiple retires, dual requests due to ipv4+6 etc.

↪ Reply

zwischenzugs says:
June 22, 2018 at 7:54 am
Thanks – do you have useful resources on that?

↪ Reply

Pingback: Anatomy of a Linux DNS Lookup – Part I – zwischenzugs
phocean (@_phocean) says:
July 4, 2018 at 8:48 pm
Have you checked Ubuntu 18.04, for instance?
Now enters systemd-resolved to add even more to this complexity…
It is really a hell to get it to work fine with VPN, and to debug…

↪ Reply

zwischenzugs says:
July 4, 2018 at 9:12 pm
Dealing with an 18.04 bug that spiked cpu because systemd-resolved was fighting dnsmasq was the trigger for me to actually unpick this mess!

↪ Reply

Francisco says:
July 12, 2018 at 5:29 pm
I also had a hard time trying to debug an issue with systemd-resolved… complexity and subtle bugs, a real pain for something as critical as name resolution.

↪ Reply

phocean (@_phocean) says:
July 4, 2018 at 8:48 pm
And, great article by the way. Thanks !

↪ Reply

Pingback: Anatomy of a Linux DNS Lookup – Part III – zwischenzugs
Pingback: Weekly DevOps News #1 - Codeogre
Pingback: Anatomy of a Linux DNS Lookup – Part IV – zwischenzugs
Pingback: DNS lookup tutorial (2) | 0ddn1x: tricks with *nix
test says:
August 23, 2018 at 8:30 am
once upon a time, dnsmasq plays a role in generating resolv.conf.

↪ Reply

Pingback: Anatomy of a Linux DNS Lookup – Part V – Two Debug Nightmares – zwischenzugs
Pingback: Linux DNS 查询剖析（第二部分） – 前端开发，JQUERY特效，全栈开发，vue开发
Youssef says:
March 27, 2019 at 3:08 pm
Hi Ian,

Thanks for your articles.
How did you find that "systemctl restart networking does, among many other things, it ends up calling a script:
/etc/network/if-up.d/000resolvconf" ?

Tried strace but only had either WSL (no no systemd) or Linux 18.04 which seems a little bit different.

Many thanks

↪ Reply

zwischenzugs says:
March 27, 2019 at 3:18 pm
strace won’t help you – I think I did a system-wide check of which files were opened while the command was running using sysdig I think

↪ Reply

Youssef says:
March 27, 2019 at 8:17 pm
Thanks Ian, I will have a check.

chirlo says:
May 15, 2019 at 12:13 pm
Unrelated, but I see it all too often: These two are equivalent:

cat /etc/resolv.conf | grep nameserver

grep nameserver /etc/resolv.conf #no need to cut and pipe

Cheers

↪ Reply

Kai says:
February 12, 2021 at 11:44 am
^^ “So systemctl restart networking does more than just run resolvconf. It must be getting the nameserver information from somewhere else. Where?” ^^
On Debian Systems take a look at /etc/network/interfaces for local network configuration.
Either a “nameserver” statement or “dhpc” advise if-up which nameserver to put in resolv.conf.

↪ Reply

Pingback: Anatomy of a Linux DNS Lookup – A2Z Facts