Survey of the DNS servers in the fediverse

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

First publication of this article on 18 November 2022

https://www.bortzmeyer.org/nameservers-fediverse.html

The fediverse is supposed to be decentralized. But the fact that the network has a decentralized architecture does not mean that it is perfectly decentralized in practice. We survey here the DNS authoritative servers for the domains used by the fediverse instances. Unfortunately, yes, they are too concentrated.

Mostly because of the incredibly childish behaviour of Elon Musk, the current wave of migration from Twitter to the fediverse is much larger than the previous ones. As a result, there is a renewed interest in the fediverse, a system which is far from recent. The fediverse is made of various **instances**, each one being independently administered, both from a technical point of view, and from a political one. Nothing new or extraordinary, this is how the Internet was architectured, and how its services worked, before a few centralized US-based giants distorted the vision of many people, making them believe that centralization is normal. Each fediverse instance has a **domain name** and this domain name is hosted on a set of **authoritative name servers**, which will reply to the DNS requests of the clients, the DNS resolvers. As an example, the instance I use, `mastodon.gougere.fr` is in the domain `gougere.fr`, whose set of authoritative name servers is currently composed of three servers <https://dns.bortzmeyer.org/gougere.fr/NS>. Unix users can for instance display them with dig :

```
% dig +short NS gougere.fr
nsb.bookmyname.com.
nsa.bookmyname.com.
nsc.bookmyname.com.
```

Note you can also perform DNS queries on the fediverse itself, as explained here <https://www.bortzmeyer.org/fediverse-bot.html> (see the current result <https://botsin.space/@DNSresolver/109366375952923612>).

A domain "works", for the users, only if the authoritative name servers reply, and reply properly. This makes them critical infrastructure for an instance. If your DNS authoritative servers fail, or lie, the instance is unusable, even if the Pleroma <https://pleroma.social/>, Mastodon or other server still works fine. So, what are the DNS servers used for this critical role in the current fediverse ?

To get data, I started from a list of current instances. I got one from `instances.social`. Note that, on a decentralized network like the fediverse, you cannot get an official and complete list of all instances. All lists are imperfect. But I have to start from somewhere (other possibilities : `http://demo.fedilist.com/instance` or the public list of peers of a big instance such as `https://mastodon.social/api/v1/instance/peers`). So, I created an authentication token on `instances.social`, and retrieved a list with curl :

```
% curl -s -H "Authorization: Bearer MY-TOKEN" https://instances.social/api/1.0/instances/list\?count=10000\&
```

I included only live instances (`include_down=false`) but, still, some instance names already were missing from the DNS.

The resulting list was in the JSON format and processed by two Python scripts (see the code later). The first one queried the DNS (with the excellent dnspython `<https://www.dnspython.org/>` library) to get information such as the list of authoritative name servers and their IP addresses. The second script processed the JSON data of the first to output meaningful statistics. Of course, like with any quantitative survey, the hard problem is often **which** questions to ask, rather than the answers.

Let's start with the actual numbers :

```
%./nameservers-fediverse-analyze.py
There are 1596 instances
```

Remember that the list is incomplete (no one knows how many instances really live). Some of these instances are grouped into the same registered domain (the domain you bought from a registry, may be through a registrar). I choosed to group the instance names in the DNS zones (a set of contiguous names hosted on the same set of name servers). For instance, `medievalist.masto.host` and `piano.masto.host` are in the same zone, `masto.host`, they have the same servers. There are a bit less zones than instance names :

```
There are 1596 instances in 1540 zones. The largest zone, masto.host., encompasses 10 instances
```

Except `masto.host`, there are few fediverse hosters with such subdomains, so no sign of concentration/centralisation here. There is no giant fediverse hoster.

Now, the name servers themselves :

```
There are 2062 nameservers. The largest one, dns2.registrar-servers.com., hosts 85 instances.
```

Some name servers are more common, such as this `dns2.registrar-servers.com`, used by a big DNS hoster. But it is still only for a small minority of instances.

But wait, name servers come in groups, called sets. The instance administrator, except if he or she is a DNS fan, typically does not cherry-pick his or her name servers, they use a set indicated by the DNS hoster. Let's study sets instead :

```
There are 951 nameserver sets. The largest one, dns1.registrar-servers.com.;dns2.registrar-servers.com., hos
```

There are less sets than instances (a sign of a small centralisation) but not by a large margin. The biggest set is not so big but we note that the top tenth of the sets hosts more than a third of the instances. There are not so many DNS providers.

But there is a catch. Some DNS hosters use a lot of names so you may think there are many various sets but they actually depend on one company, a bad thing for decentralisation. The typical example is Cloudflare, with a lot of cool names depending on the domain they host (`eleanor.ns.cloudflare.com`, `sue.ns.cloudflare.com`, etc). So, we have to group the name servers by company. One of the simplest ways is to find the registered domain of the name servers' names. To do so, we use the Public Suffix List `<https://publicsuffix.org/>`, which is not authoritative and not perfect but sufficient for us. It will put all the Cloudflare names into one bucket, the registered domain `cloudflare.com` :

```
There are 667 nameserver's domains. The largest one, cloudflare.com, hosts 366 instances. The top 10 % hosts 132
```

This time, we have a clear sign of centralisation. A lot of seemingly independent instances have a shared provider, Cloudflare. If Cloudflare breaks, or does evil things, it will affect many instances. And the top 10 % of hosters serves the overwhelming majority of fediverse instances. For your information, the biggest ones are :
— `cloudflare.com` : 366
— `registrar-servers.com` : 91
— `gandi.net` : 90
— `domaincontrol.com` : 55
— `googledomains.com` : 49
— `digitalocean.com` : 48
— `linode.com` : 39
— `inwx.de` : 23
— `inwx.eu` : 23
(It can be noticed that, unlike the fediverse itself, which uses a lot of "new TLDs" such as `.social`, the name servers are faithful to the old TLDs.) Playing with names, as seen in the Cloudflare case, complicated things. We have also the case of AWS which does not appear on the above list but they should ; they use a lot of registered domains (`awsdns-46.co.uk`, `awsdns-20.com`, etc) so they appear as many different hosters. Also, about Cloudflare, remember that this survey is **only** about the DNS : I did not try to see where the server instance is hosted. I repeat : just DNS, no HTTP was used or considered here. (One also could note that some authoritative name servers are installed at one big machine hoster, which gives Cloudflare an even more prominent place `<https://infosec.exchange/@mnordhoff/109368079955659889>`.)

So, as you can see, but this is hardly a surprise, the fediverse is not perfect and there are signs of centralisation, at least as far as DNS is concerned.

The two programs used for this article are :
— `nameservers-fediverse-gather.py` : it gathers information from the DNS and produces a JSON file storing, for each instance, its DNS zone, the name servers and their IP addresses (which were not used for this article). To find the DNS zone, it just climbs the DNS tree until it founds name server records.
— `nameservers-fediverse-analyze.py` : it reads the JSON file produced by the first program and displays statistics.

Among the weaknesses of this analysis, you'll note that our instances are considered equal. But there are very small and very big instances. It could be useful to tweak results depending on the size of the instance. `instances.social` gives us this information but it has to be handled with care, it is just a declaration from the instance (and many accounts may be inactive, anyway).