New (and Old) Kid on the block: GeoDNS Authoritative Server (abstract)

Giovane C. M. Moura (1)  Caspar Schutijser (1)  Marco Davids (1)  Cristian Hesselman (1,2)
1: SIDN Labs; 2: University of Twente

Authoritative DNS servers are those that know the contents of a DNS zone from memory [4]. They are the ones that provide recursive resolvers with answers, which ultimately is used to answer clients and applications. The DNS community is very familiar with open-source Authoritative DNS servers [4]: BIND [5], NSD [11], KnotDNS [3], and PowerDNS [16] are just few of them.

Lesser known, however, is GeoDNS [2], an open-source authoritative DNS server developed by the NTPPool [13] operators and that is used in production tailored to their specific needs. It is written in Go and it has been used in production for many years for the NTPPool authoritative zones. It has been maintained on GitHub for the last eight years.

We recently released a technical report on how GeoDNS works [10] and their implications for the NTPPool users at large. In this paper, we intend to introduce GeoDNS to a larger, DNS operations and researchers audience.

The NTPPool project: there is a large number of public NTP servers on the Internet, and the NTPPool project [13] was created to simplify the access to volunteer and verified NTP servers [17]. It lists 4376 active NTP servers – 2918 IPv4/1458 IPv6 (June 30th, 2021) [12] and it is the default time provider for Linux- and Android-based system, as well as various vendors, such as Linksys, FritzBox, Asus and Sonos [12].

GeoDNS: why not using other DNS software? The NTPPool project uses DNS to manage the pool of NTP servers they have, with two major requirements that are out of scope for major authoritative servers: geolocation-based routing and load-balancing. Next we cover both of them.

1. Geolocation-based routing: the NTPPool project lists NTP servers from all over the world. However, NTP clients are very sensitive to jitter [9], and the smaller the jitter, the more precise the clock synchronization. To minimize delay and, indirectly, reducing jitter, given that synchronization errors increase in proportion to the delay – §3 in [9], GeoDNS strategy is to match DNS clients IP geolocation data with NTP servers geolocation data. For example, DNS resolvers in Sweden should preferably be served by NTP servers located in Sweden. In this way, assuming that the clients are also located in Sweden (resolvers and clients can be in different countries), this heuristic should minimize latency and jitter. For that, it relies on Maxmind’s IP2location database [8].

2. Load-balancing: NTPPool volunteers can set how much bandwidth they want to donate when adding their NTP servers to the NTPPool project. To manage that, GeoDNS DNS supports record weights, which are values that determine how often a DNS records should be included in a response. For example, consider the example DNS zone file in Listing 1 (GeoDNS uses JSON instead of traditional DNS formats): in this example, we see that Turkey’s subzone (tr.pool.ntp.org) has, as its A record, two servers. The first one has a weight of 100, which means it will show up 100× more frequently in responses than the second server which has a weight of 1.

```
{
  "tr": { # sub-zone: tr.pool.ntp.org
    "a": [
      ["77.243.184.65", "100" # weight],
      ["212.175.18.126", "1" # weight],
    ]
  }
}
```

Listing 1: GeoDNS demo zone file for pool.ntp.org.

Supported (and not) record types: GeoDNS does not support DNSSEC [1], but it supports NS, A, AAAA, CNAME, MX, TXT, SPF and two record types used internally by GeoDNS. It also supports EDNS Client Subnet (ECS), which can be used for client geolocation. It supports both UDP and TCP.

GeoDNS open-source alternatives: Both PowerDNS [15] and BIND [7] have their own GeoIP implementations. However, both were developed in 2014 [6, 14], whereas GeoDNS is at least from 2013.

Research applications: GeoDNS seems to be the first open-source authoritative that supports GeoIP. Researchers can benefit from its simplicity and being relatively lightweight compared to other alternatives: zone files are written in JSON (easier integration with Python), and it automatically reloads when a zone file is re-written.

We still need to evaluate its performance, but we can say it has survived the test of time on the NTPPool, running since 2013.
REFERENCES


