Diving into the NTP Pool

Giovane C. M. Moura, Marco Davids, Caspar Schutijser, Cristian Hesselman
SIDN Labs, Arnhem, The Netherlands
https://sidnlabs.nl/en
Technical Report #2021-22
Oct. 29th, 2021

Abstract

Time is of essence also on the Internet: core protocols and services correct functioning depend upon the ability to have synchronized clocks. The default protocol to synchronize clocks on the Internet is the Network Time Protocol (NTP). Among time service providers, we find the NTPPool project, a volunteer-based project which facilitates the matching between users/devices needing synchronizing their clocks and time servers that can provide this service for free (also donated by other volunteers). Daily, millions of users, Android mobile phones, Linux servers and a large array of devices depend upon the NTPPool project to find suitable, volunteer and free NTP servers that they can use to synchronize their clocks, which they do via using the Internet Domain Name System Protocol (DNS). The NTPPool project has been continuously providing their services for over 18 years, and yet there is not much scrutiny into what servers get to service what clients – and potential implications. In this paper, we dive into the NTPPool, and reveal, with active measurements and emulation experiments, how it matches clients to servers. We find multiple cases of single point of failures, in which the entire population of 22/35 (IPv4/IPv6) countries depend upon a single time provider (Cloudflare), including populous countries as Nigeria, Pakistan, and Israel. Finally, we make recommendations on how their services can be improved.

1 Introduction

There is a series of core Internet applications, services and protocols that can be compromised or impaired simply by tampering with their hosting system's clocks. TLS [12], DNSSEC signatures [3], DNS caches [30], RPKI [9], Kerberos [31] and even Bitcoin are among the many that fundamentally depend on correct time information [22, 26, 50].

The Network Time Protocol (NTP) [26] is the Internet's default protocol for device's clock synchronization. NTP *servers* are synchronized with reference clocks (*e.g.,* atomic clocks, and GPS/Galileo) and provide correct time information to *clients/secondary time servers*.

Clients, in turn, comprise a very large diverse group: servers, IoT devices, mobiles phones, among others. Similarly to DNS, NTP clients can choose from various *time providers*. Vendors like Apple [2], Google [16], Cloudflare [10], and Microsoft [25] all run their own time services. Just like dynamic configuration of DNS servers, the DHCP [13] protocol also enables dynamically setting NTP servers [1, 15]. However, there is no evidence of the widespread usage of this feature. If clients are not configured with NTP servers via DHCP, they then have to resort to their hardcoded servers, regardless of what the network they are connected to.

There is a large number of public NTP servers on the Internet, and the NTPPool project [38] was proposed to simplify the access to volunteer and verified NTP servers [50]. It lists 4376 active NTP severs – 2918 IPv4 and 1458 IPv6 (June 30th, 2021) [32] and it is the default time provider for Linux- and Android-based system, as well as various vendors, such as Linksys, FritzBox, Asus, Zyxel and Sonos [32]. Given such large user base, one could easily see the NTPPool as one of the *core* services on the Internet, although it is typically overlooked.

Despite its importance, there is not a large body of research, documentation or publicly scrutiny on *how* the NTPPool works, and how the NTP servers it lists are shared and distributed among their larger client population. That is the goal of this paper: characterizing how the NTPPool distributes the NTP servers to its clients and how the clients "see" the NTPPool, which ultimately determines who gets to synchronize what users' clocks.

We start by providing background (§2) information about the NTPPool. At its core, the NTPPool uses DNS [27] to redirect users to NTP servers. It employs its own customized authoritative server for this purpose: GeoDNS [8], an open-source, authoritative DNS server [17]. NTP clients must first resolve pre-configured domain names, which are then responded by the NTPPool authoritative servers, which, in turn, provide up to 4 different NTP servers IP addresses from the NTPPool per DNS query.

How GeoDNS chooses these addresses among the 4.3k available is not currently clear, and there is currently no study that aims at scrutinizing the client's view from the NTPPool. The NTPPool home pages simply states that "the NTP Pool DNS server uses GeoIP data from MaxMind to help choose an NTP server from the Pool." [38]. (GeoIP data refers to the mappings between an IP address and the geographical location of the host using it [44]).

In this context, our first contribution of this paper is to demonstrate, that, in the wild, clients see a very dissimilar number of NTP servers (§3). Drawing upon roughly 9k vantage points from the Internet (RIPE Atlas probes [46, 47]), we show that the NTPPool can be rather unfair: 10% of our vantage points were served only 12 and 5 NTP servers by GeoDNS (IPv4 and IPv6 addresses, respectively), despite the pool having thousands of servers, regardless of the number of DNS queries we send to the authoritative servers of the NTPPool. Worse, 2.5% of clients see only up to 2 IPv4 NTP servers, and 5.4% of clients see up to 2 IPv6 NTP servers.

Our second contribution is to show in a controlled environment how GeoDNS maps clients to NTP servers (§4) – which explains the variation in the number of NTP servers each client sees. We show that a combination of a strict matching criteria – the matching between the client's IP address geographical location (country level) with geographical DNS sub-zones hosted by GeoDNS (DNS zones [27] contains the set of NTP servers available for each country and continent in the NTPPool).

We *reverse engineer* the NTPPool DNS zones, and validate them. We then are able to show that many of them are rather small: 32% and 48% of the sub-zones (IPv4 and IPv6) in the NTPPool have up to 3 NTP servers only. The consequences of that, however, can be very serious: all clients from the countries that the sub-zones refer to are presented only with these IP address, no matter how many DNS queries are sent. We show that *all* clients from 13 countries (IPv4) and 42 countries (IPv6) are served by a single provider (Cloudflare), which creates a single point of failure and leads towards more centralization and consolidation on the Internet [4–6, 20, 21, 28, 51, 53].

Finally, our last contribution is to *emulate* clients from all countries worldwide (247 in total, §5), based on the DNS zones file we have reverse engineered. We show a large variation in how often NTP servers within a country zone are served to the clients. We show how larger zones tend to follow Pareto's distribution: 20% of the servers are present in 80% of the responses, and how many clients could potentially benefit if their countries sub-zones (having fewer than 2 NTP servers) were removed from the NTPPool, which would cause them to be mapped to their larger continent zones.

Overall, our works is the first to bring to light how NTPPool operates and how that influences what clients see, and we point places where it can be improved to reduce the dependence on a single or few providers (§6). We will release all datasets upon acceptance.

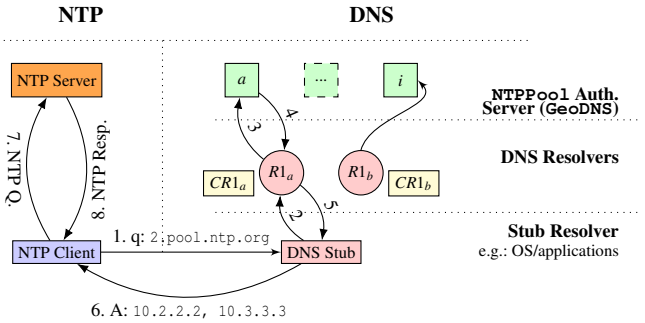

Figure 1: Relationship between NTP and DNS and its different parts: NTP client (blue), NTP Server (orange), DNS resolvers (red) with their caches (yellow), and the authoritative DNS servers (green).

2 The NTPPool Project

The NTPPool project is a volunteer-based network of NTP servers [26] that are made available via DNS, under the pool.ntp.org zone and its subdomains. The idea originated in 2003 as a solution to reduce the abuse of publicly available NTP servers [57]. Instead of having a large list with public NTP servers (which individually could more easily become overload), the NTPPool project proposed to "load balance" the NTP traffic to servers using DNS.

To illustrate how this works, consider an NTP client (Figure 1) that wants to synchronize its clock with the NTPPool. In the first step, the client would ask the DNS software on its computer for the IP addresses of the zone in question, using a DNS query. (This is done by DNS stubs, which, in turn, forward the query to recursive resolvers – step 2 – which are capable to perform the recursive resolution tasks). If the resolver does not know the answer from the cache, it will contact one of the NTPPool authoritative servers [17] (step 3, that run GeoDNS), which are the servers that can provide answers about the zone in question.

Upon receiving a query, the authoritative servers ran by the NTPPool return a permuted list containing n IP addresses to the resolver (step 4), which is a subset of the available servers [32]. The resolver, in turn, will it respond the DNS stub (step 5) with these addresses, which finally passes it to the NTP client (step 6). Then, the client can contact any of the IP addresses provided and send a NTP query to it (step 7), allowing it ultimately to receive a response (step 8).

The NTP protocol, latency and jitter: the NTP protocol is designed to mitigate the effects of changes in latency (jitter). NTP clients measure several latency-related metrics (§10 in [26]). It measures one-way delay between server and client and use it when calculate the *time offset* between the server

(reference clock) and client. The offset, in turn, is the value used to adjust the client's clock. Shorter and more symmetric round-trip times (RTT) tend to lead to more accurate offsets. (This seems to be one of the NTPPool's motivating factors to develop GeoDNS, to be able to serve clients with servers likely geographically closer and with lower, more stable RTTs).

NTP clients use the clock filter algorithm to mitigate jitter effects when determining the best time *offset* to synchronize the local clock and also to exclude varying jitter servers from the list of available servers (in the case of the NTPPool, it is provided by DNS).

The NTPPool project estimates that there are 5–15M clients [40][1], but the real number is hard to know given that DNS heavily relies on caching [30] (*CRn* nodes in Figure 1), and most actual clients are behind resolvers that may cache answers (the DNS records have a 150 s TTL (2.5min), which is the upper limit for caching), and many IPv4 clients are behind Network Address Translation (NAT) devices. In the case of the NTPPool, having low TTLs reduces the consequences of clients receiving only high jitter NTP servers; they can retrieve a set of new NTP servers from GeoDNS (if available in the zone – §4) after the TTL expires.

DNS zones and sub-zones: authoritative DNS servers use *zone files* [27], which stores DNS *records* and are loaded into memory by a DNS authoritative server. In the case of NTPPool authoritative server, it is known that it has two main types of zones: (i) geographical zones, which cover five continents and multiple countries and are public listed [36] and (ii) vendor zones [40], which include Android' s dedicated subzone (android.pool.ntp.org) and many others vendors are not public listed. It is not clear, however, how GeoDNS uses these zones to choose what portion of the 4k NTP server the clients see. Software vendors are encouraged to obtain their own sub-zone from NTPPool and not use the default name servers. For example, the Debian Linux distribution uses debian.pool.ntp.org. This allows the NTPPool operators to easily mitigate unintentional excess traffic from specific vendors and brands by simply managing the zone used by their devices [40].

Joining and using the pool: Any user can join the pool of servers, providing the servers have static IP addresses and use static NTP upstream servers. The NTPPool recommends users to use the [0-3].pool.ntp.org zones to obtaining NTP servers IP addresses [34]. **[RF: SM-1]** Users, however, are free to use country or continent zones directly if they wish.

NTPPool authoritative DNS server infrastructure: in Appendix A, we include an analysis on the authoritative DNS servers that are responsible for the NTPPool zone and subzones. These are also provided by volunteers, and currently the NTPPool authoritative DNS infrastructure is hosted in 17 and 12 different ASes (IPv4 and IPv6, respectively), which is more diverse than most top-level domains (TLDs), such as .com and .it.

[1] **[RF: SM-2]** Private feedback from NTPPool operators is that these public numbers is far lower than current traffic: a single NTP server in the the U.S. received more than 33M IPv4 clients in a 25h period.

3 The clients' pool view

The NTPPool operators self-report that 4.3k IP addresses are served in the pool [32]. We are interested in how much of it individual clients are able to "see", in the wild.

To do that, we employ ∼10k vantage points (VPs) on the Internet, on more than 3082 Autonomous Systems and 166 countries, which are like real-world clients in these networks. These VPs are RIPE Atlas probes, which are are hardware devices or virtual machines (VMs) that can be remotely programmed to carry out active Internet measurements. We configure ∼10k Atlas probes to query directly one of the authoritative servers of the NTPPool (b.ntpns.org over IPv4 – 185.120.22.23, so we bypass DNS resolvers – Figure 1 – and avoid hitting the resolver's cache). In this way, we are able to retrieve fresh NTPPool addresses every new query. The probes are configured to send queries every 5min – a safe limit that does not overload RIPE Atlas and neither the NTPPool authoritative servers. than 4k NTP servers), while other clients see a far more diverse distribution of servers.

Table 1 shows the experiments details. In the first (EnumV4), we configure Atlas probes to query for IPv4 NTP servers (defined as A records in DNS [27]). The second experiment (EnumV6), probes query for IPv6 servers (AAAA record types [56]). For both experiments, we see ∼ 9.2k active probes, having 9.1k received valid responses (some probes are blocked/hijacked or contain wrong answers [29], which we disregard). These 9.1k clients provide us with a view from from ∼ 3k Autonomous Systems (ASes), totaling ∼ 25M responses per experiment.

In these experiments, each Atlas probe (analogous to a client) sends a median 275 queries and it receives 4 addresses per response (Table 1). This, theoretically, would allow each probe to retrieve up to 1100 unique addresses from the NTPPool, if the process would be completely random and no address would be served more than once per probe.

We analyze the datasets to see how much of these 1100 addresses each probe sees. Figure 2 shows the CDF (Cumulative Distribution Function) of the number of unique IPs retrieved by each probe. We see that most probes see fewer than 200 unique IPs, for both measurements, which is fewer than 20% of the maximum theoretical value of 1100 addresses. We see that 10% of the clients see up to 12 NTP servers (EnumV4) and 5 NTP servers (EnumV6). Given that there are fewer IPv6 NTP servers, we see the CDF shifted to the left.

These two experiments with ∼ 9k VPs show us that the NTPPool NTP servers distribution is rather uneven among clients in the wild. They motivate us to investigate the reasons *why* this happens, which we cover next.

Measurement	EnumV4	EnumV6	ArgV4	ArgV4-Emul
Target	185.120.22.23		185.120.22.23	54.93.163.251
QNAME	2.pool.ntp.org.		3.ar.pool.ntp.org	wilson.ants
QType	A	AAAA	A	A
Date	2021-08-2[6-7]	2021-08-3[0-1]	2021-08-02	2021-08-06
Interval	5min	5min	10min	10min
Duration	24h	24h	2h	2h
Probes	9260	9272	9219	9229
valid resp.	9113	9127	9068	9052
no resp.	147	145	783	382
ASes	3116	3133	3127	3128
valid resp.	3082	3095	3080	3067
no resp.	156	148	474	262
Countries	166	168	1	1
Responses	2534199	2583318	107031	110292
Valid Responses	2469211	2535981	104331	107793
invalid/empty	64988	47337	2700	2499
#NTPPool Addresses	3056	1479	1481	8
per response (median)	4	4	2	2
per response (1st quartile)	4	4	2	2
per response (3rd quartile)	4	4	2	2
Queries per probe (median)	275	275	11.6	11.9

Table 1: NTPPool RIPE Atlas experiments. Datasets: [45]

.

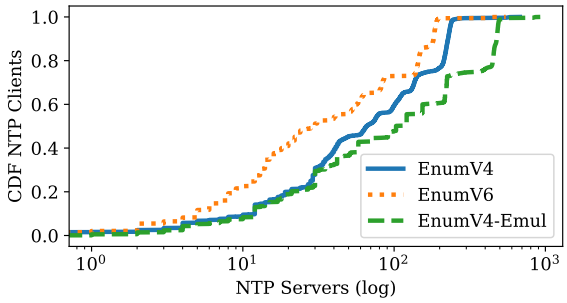

Figure 2: CDF of Pool IPs per Probe

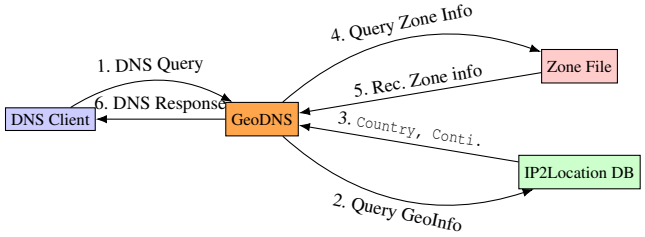

Figure 3: DNS client and GeoDNS relationship.

4 How GeoDNS works

We set to explain the large variation in client's pool visibility observed in §3. We study the GeoDNS server behavior by running it ourselves, and then carrying out active measurements against it. Our goal is to understand how it decides what each client sees what NTP servers. (The GeoDNS documentation does not provide such level of detailed information.)

GeoDNS behavior in a nutshell: we determined how GeoDNS decides what clients get to see what subset of NTP serve, which we summarize next. To operate, GeoDNS requires two input files: a DNS zone file, which contains the list of domains and records (NTP servers) it will use to serve its clients, and Maxmind's IP geolocation database [23], which provide geographical information associated with IP addresses on the Internet. The process starts by a client sending a DNS query (such as pool.ntp.org) to GeoDNS (step 1 in Figure 3). GeoDNS, in turn, will look up the IP geolocation related to client's IP address (step 2), and receive the *country* and *continent* associated with the client's IP address (step 3).

Then GeoDNS will attempt to find in its zone files if there is a country sub-zone that matches the client's country of origin. If there is no country zone (or the country zone is empty), then GeoDNS proceeds to a less specific sub-zones – first the continent and the global zone (@).

For example, consider a log sample from GeoDNS in Listing 1. In this log, an Israel-based client (RemoteAddr field) can be mapped to three sub-zones (Target: il, asia, @), but it is ultimately mapped to Israel's country zone (il) – steps 4 and 5 in Figure 3.

```
{  "Time": 1626941639825507800,
  "Origin": "zweig.welt",
  "Name": "zweig.welt.",
  "Qtype": 1,
```

```
  "Rcode": 0,
  "Answers": 2,
  "Targets": [
    "il",
    "asia",
    "@"
  ],
  "LabelName": "il",
  "RemoteAddr": "132.64.6.1",
  "ClientAddr": "132.64.6.1/32",
  "HasECS": false}
```

Listing 1: Sample Log from GeoDNS

In this way, GeoDNS iteratively attempts to find a zone which servers are more likely to be geographically closer to the clients, which hopefully can have shorter and more stable RTT, which results in better offset calculations (§2). Ultimately, GeoDNS sends the clients the A/AAAA records associated with NTP servers (step 6), and thus the NTP client can proceed to evaluate these servers and choose the most suitable ones.

To figure out this behavior, we had two carry out two main tasks. First, we reverse engineer the NTPPool zone file (§4.1), and then carry out an experiment using GeoDNS (§5). We describe both next.

4.1 Reverse engineering NTPPool zone files

We strive for having a zone file as similar as possible to the one ran by NTPPool project. Given that these zone files are not publicly available, we resort to reverse engineer them.

Traditional authoritative DNS server use standardized text zone file formats [27], but GeoDNS uses JSON zone files instead [8]. We bootstrap the process by using a sample zone file provided by the GeoDNS developers [7], which we partially show in Listing 2. The GeoDNS zone file has multiple DNS sub-zones, like Turkey's tr (tr.ntpool.org). Each sub-zone has a list of IPv4 and IPv6 addresses (A and AAAA records), which lists NTP servers available to that particular country – and clients from these countries will see the A/AAAA records showed in this sub-zones. Each A/AAAA records is followed by a *weight*, which is a non-standard DNS feature used by GeoDNS to sort the frequency in which records should be returned to clients, a method that allow NTPPool volunteers to set indirectly the amount of traffic they want to receive at their NTP servers (we investigate it in §4.3).

```
{   "ttl" : 390, #DNS TTL
   "serial" : 1345449135, # DNS Zone file serial number
   "data" : {
      "" : {  "" indicates the "global" pool
         "ns": [ #authoritative DNS servers
             "a.ntpns.org",
             "b.ntpns.org",
             "x.example.com"
         ],
         "a" : [ #IPv4 addresses of all NTP servers in
    the global zone
            [
               "203.17.251.1", #IPv4
               "1000" #weight
            ],
            ...
         ]
      },
   "tr" : { #sub-zone: tr.ntpool.org
        "a" : [
           [
              "77.243.184.65",
              "1000000"
           ],
           [
              "212.175.18.126",
              "100000"
           ],
    ...}
```

Listing 2: GeoDNS demo zone file for pool.ntp.org. # marks our comments.

	Rev. Engineered	Self-reported
Zones - IPv4	126	257
non empty	125	116
Zones - IPv6	126	257
non empty	112	110

Table 2: NTPPool zone files results

To determine what zones exist, and what servers are in each zone, we combined (i) all the NTP servers we received in the DNS responses from the measurements EnumV4 and EnumV6 and (§3) and (ii) a NTPPool website service, which provides a web page per NTP server it has in its zone. Each page lists what *sub-zones* the NTP server is allocated, as well its monitoring score. For example, the server 95.217.188.206 is allocated to the global (@), Europe, and Finland's fi zones [39].

We use this facility and scrape the NTPPool website using the list of IP addresses we obtained in our DNS responses from §3 (EnumV4 and EnumV6), on the same day we crawled them. Each server has a *score*, which is provided by the NTPPool monitoring system, and only servers with score higher than 10 are assigned to zones. We consider all server is regardless of their score, given we found experimentally that *some* servers with low score (<10) were seeing in responses from the NTPPool servers – which may be to the fact that scores can dynamically change in the course of the measurements.

Table 2 shows the results of our attempt to reverse engineer the zone files. The 3056 NTP servers from §3 are associated with 126 sub-zones (A records), and the 1479 IPv6 are associated with 112 non empty zones. (We found no vendor zones, as the debian and android zones – see §4.4.1).

Figure 4 shows the CDF of the NTPPool zone sizes that we have reverse engineered. We see a large variety in terms of size: we see that 32% of the zones have up to 3 IPv4 NTP servers, and 48% of the zones have up to 3 IPv6 NTP servers. That is a rather uneven distribution of servers among all zones, we see that fewer than 9% of the zones (IPv4) have more than 100 NTP servers, and fewer than 7% for IPv6.

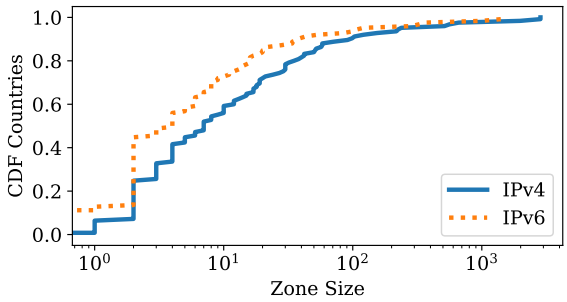

Figure 4: CDF NTPPool countries and continent zone sizes

	# Zones	VPs
$NTP_{Atlas} = NTP_{Emul}$	93	2265
$NTP_{Atlas} < NTP_{Emul}$	66	7282
$NTP_{Atlas} > NTP_{Emul}$	12	47

Table 3: Validation results per zone

Cat	Zones	VPs
CC zone =1	cm, gg, re	34
Rest	es, ie, ir, it, lt, ru south-america, tr, ua	13

Table 4: Atlas VPs that see more NTP servers than what is available in their respective country sub-zones.

4.2 Validation

To validate if our reverse engineered zone files can be used to *predict* responses from real clients, we run it on an instance of GeoDNS and reproduce the incoming queries from the experiment EnumV4.

We set up GeoDNS (v. 3.0.2) on a Debian Linux server. Just like the NTPPool authoritative DNS servers, we configure our server to respond with 4 DNS records per query. We configure it with MaxMind's GeoLite2 country IP2location database [23] from 2021-08-24, and use the zone file from §4.1. We assign all servers with the same weight (1000), so we eliminate the influence of the weights in the query distribution. We create a fictitious zone (basakatu.eus) that mimics pool.ntp.org.

Client Setup: The next step consists of querying our GeoDNS server to see how IP addresses are mapped. We repeat the same number of queries and order from the experiment EnumV4, which we do by employing IP spoofing (forged source IP addresses [14]) using a customized Python script. To avoid that spoofed responses are forwarded to the Internet, we run our client on a server without Internet connectivity. We send one query per IP address for the A records of basakatu.eus, on 2021-08-30.

Collected datasets: we collect network traffic at our GeoDNS server (*i.e.,* pcap files), as well as GeoDNS's logs files (sample in Listing 1), which lists the metadata associated with each DNS query and response.

Results: For each IP address in both EnumV4 and EnumV4-emul, we extract *all* unique A records, which shows how much of the NTPPool each client has seen. Then, we compare the Atlas and Emulation results, for each client. Table 3 shows the results. We see that 2.2k VPs see exactly the same NTP servers set, in both Atlas and Emulation experiments – which is all servers that belong to the zones they are mapped to (93 zones in total).

We also see 7.2 VPs clients from 66 zones see *fewer* NTP servers than our emulation. We speculate this can be due to the use of uniform weights (1000) in our emulation experiment, whereas in reality we could expect more diversity in weights within a zone (see §4.3 for more).

The last category is the more concerning one: that Atlas clients 47 clients see more NTP servers than our emulation experiments – and they are mapped to 12 sub-zones. If GeoDNS bounds a client to a country or continent zone, having Atlas seeing more records that are in the mapped zone suggests that GeoDNS mapping process may not be as we expected.

We investigate these clients further and classify them intro 3 categories (Table 4). The first category, we see 3 zones and 34 Atlas probes that see more NTP servers because their country zones have only one (CC zone=1).

We investigate the probes from these three countries (CC Zone=1) and illustrate here an Atlas probe located located at the Bailiwick of Guernsey, an island located in Europe (gg code). Our Atlas measurements show that this gg probe sees, in total, 21 unique NTP servers during the monitoring period– far fewer than europe zone size (1980 IPv4 servers). This probe receives initially a single NTP server in the DNS responses – 51.255.142.175 – which is a NTP server from Guernsey (we include the responses sequence in Appendix B). From just before Aug. 26th, 9PM, this Atlas probes receives 20 different NTP servers in 4 subsequent queries, spanning over 20 minutes – all servers belonging to the europe zone. It seems like this client was mapped during this period to europe zone and not gg zone. After these four queries, however, the same Atlas probe start to receive a single NTP server in the DNS responses – the same 51.255.142.175. Figure 5 shows the time series of number of A Records per DNS response.

So what could explain these unusual responses? Our Atlas measurements do not allow us to tell what happened to this server, but fortunately the NTPPool page about this particular NTP server provides a history of its *scores*,as measured by the NTPPool monitoring system – in a form of a csv file [37]) – a measurement that is carried independently from ours.

We correlate our RIPE Atlas results with the NTPPool's server score logs show the scores time series also in Figure 5. We see that this particular server scored dropped below 10 – the minimum value for it to be used in the NTPPool zones –

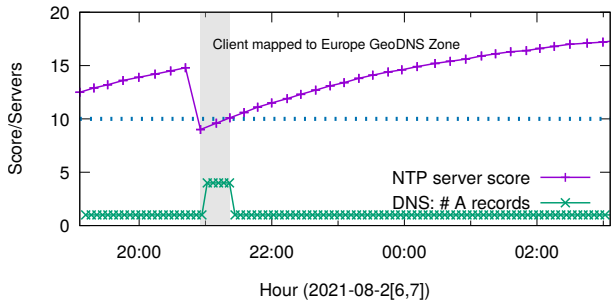

Figure 5: Time series for NTP server 51.255.142.175 and number of responses seen from a client located in Guernsey: low scores lead to NTPPool eviction and fallback to continent zone.

between 20:55:32 and 21:22:04 (Aug. 26th, 2021, GMT). We show this in the gray area. (we show the NTPPool monitoring logs also in Appendix B). We also see that during this interval, the Atlas probe located in Guernsey receives 4 NTP servers per response, completely *coinciding* with the time that this NTP server had poor scores (gray area). Given these low scores, we can infer that this NTP server was likely evicted from the gg zone. Since this was the *only* server in the zone, then GeoDNS mapped this Atlas probe to the Continent zone (europe) Once the NTP server score surpasses 10 again, after 21:22, we see the client again receiving 1 NTP server in the DNS response, likely from the NTP server joining the gg zone again. The servers from cm and re show similar score issues.

This shows the dynamic behavior of the zones in the NTPPool: poor performing servers seem to be automatically removed from the NTPPool of severs (and automatically included as well). In our emulation experiment, however, we do not account for this dynamic behavior, as we generate a single snapshot of the DNS zone files, while the NTPPool uses dynamic zone updates to constantly update its zones.

The remaining of the probes (Rest in Table 4) comprise probes belonging to multiple countries. We found that these probes are being mapped to another continent/country zones outside of their own, either by an error in the IP2 location mapping or DNS hijacking, in which the original DNS query to the NTPPool authoritative servers is intercepted on the way – and other studies have shown that this occurs with RIPE Atlas probes [29]. Due to ethical implications, we do not explore this further here.

Overall, we can conclude that our reverse engineered zone file can be used to emulate the original NTPPool authoritative DNS servers. We cannot yet determine the *frequency* each NTP server is seen in the response, which we analyze next.

		ArgV4		ArgV4-Emul	
IP	**ASN**	**Counts**	**Ratio**	**Counts**	**Ratio**
162.159.200.1	13335-Cloudflare	37580	18.3%	37504	17.7%
168.96.251.227	3597-InnovaRed	31142	15.2%	31763	15.1%
170.210.222.10	4270-Red de Inter.	28599	13.9%	29288	13.9%
168.96.251.226	3597-InnovaRed	25707	12.5%	26737	12.7%
181.93.10.58	7303-TelecomArg	24878	12.1%	25836	12.2%
168.96.251.195	3597-InnovaRed	24731	12.1%	25812	12.2%
168.96.251.197	3597-InnovaRed	17223	8.4%	18288	8.7%
162.159.200.123	13335-Cloudflare	14838	7.2%	15832	7.5%

Table 5: NTP Servers occurrence for ArgV4 and ArvgV4-Emul experiments

4.3 Estimating NTP servers weights

Each A/AAAA record in GeoDNS zone files have a *weight* associated with it. While this is not a standard feature in DNS, this is used by GeoDNS to load-balance NTP traffic – volunteers can specify how much bandwidth (which is later translated to weights) they want to server. GeoDNS will return records with high weight values more often than others. In Listing 2 example, the server 203.17.251.1 is likely to appear 100x more often in responses than 149.255.99.71 (calculated by the ratio between their weights).

Now that we have shown to GeoDNS maps clients to NTP servers, we can evaluate how each NTP server weight determines the distribution of NTP servers among clients when weight is considered. To do that, we first carry out an experiment in the wild to determine the *relative weights* of each server. Instead of querying the NTPPool directly – as in §3, we query the authoritative server of one of its country sub-zones – Argentina's ar. We choose it because it has only 8 active IPv4 NTP servers (in 2021-08-02 [33]), so it reduces the number of necessary queries to evaluate the weight's influence.

By querying Argentina's sub-zone directly (3.ar.pool.ntp.org.), we *bypass* GeoDNS's geolocation steps (steps 2 and 3 in Figure 3), obtaining records only listed in the ar sub-zone. (we confirm this behavior experimentally by running a test locally). As shown in Table 1 (dataset ArgV4), we send 107k queries from 9.2k Atlas probes, towards one of ar authoritative servers. Each valid response received only *two* A records, and in total we have seen 8 distinct A records associated with NTP servers under the Argentina's zone, as reported also in [33].

Table 5 shows the results. We see that each server receives from 7.2 to 18.3% of all queries – so, in the case of Argentina's sub-zone, the popular NTP service may appear at least twice as more often the less popular server. We use these results as a baseline.

To determine if the count values obtained in Table 5 represent NTP servers weights in the DNS zone – so we can reproduce the response results from the real world in a controlled environment, we create a test zone using the A records from Table 5 and, as weights, we simply use the counts value

(we show the final zone file in Appendix C). We configure GeoDNS with this zone file on a AWS EC2 Frankfurt Ubuntu VM and use $\sim$ 10k Atlas probes to query this zone, as shown in Table 1 (dataset ArgV4-Enum), and use the same parameters (frequency, duration) in the ArgV4 experiment.

We obtain 110k responses from 3128 ASes, very similar to ArgV4 experiment, as shown in Table 5. We then compute the occurrence of each IP address from our demo zone in the Atlas responses and find that the query distribution per IP is *very similar* to the *original* experiment using the production servers of the NTPPool. This, in turn, allow us to conclude we can reverse engineer the pool weights and emulate the influence of each IP address in the entire zone by simply observing the frequency in the responses.

4.3.1 Inferring the weights of all sub-zones

Next, we proceed to reverse engineer the weights of all sub-zones from the NTPPool zone file, based on the validation from the previous section. We start with the reverse engineered zone file (§4.1) and set all A/AAA weights to 0. Then, for each DNS response from EnumV4 and EnumV6 (§3), we increment an IP address' weight by one for the sub-zone in which the IP address was mapped to (LabelName in Listing 1) – information that we obtained from our emulation experiment (§5). We repeat this process for all EnumV4 and EnumV6 responses, and ultimately generate a zone file with the record's weights populated.

Figure 6 shows the CDF for the A records weights for 4 zones. We choose these zones given they have different sizes, to show the contribution of each server in terms of how often they appear in the responses seen from the real world measurements form EnumV4. On the *y* axis, we show the CDF of NTP servers in the zone (each zone size is in its legend). On the *x* axis, we show the cumulative normalized weight of the servers, *i.e.,*the cumulative individual weight divided by the sum of all weights in the zone. We see that for Germany (de) and the United States (us) zones, both with more than 500 servers, we see that a Pareto like distribution: 78% of the servers account for only 20% of all occurrences (scores); and 22% of the servers account for 80% scores. For smaller zones, such as Japan (jp) and Taiwan (tw), we see a more homogeneous distribution, show by more straight lines.

This has to do with the nature of the servers in their zone and how much each volunteer is willing to share. For Germany's zone, the smallest weight for a NTP server is one, meaning it appeared only once in all 1.2M times records from the de zone were returned. In contrast, the server that appeared the most did it for 30k times (2.3% of all times), which in this case is one of Cloudflare's NTP server (162.159.200.1).

We show in Figure 7 the top 50 zone for A records, in terms of zone size (we include the country code mapping to country in Appendix E, and the we show the AAAA figure in Appendix D). For each zone, we compute its server weight

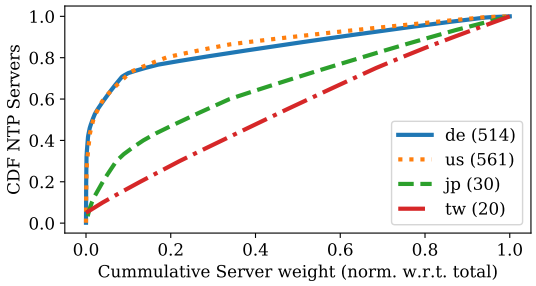

Figure 6: Cumulative weights and sub-zones - A records. Numbers is parentheses shows number of A records in the zone.

distribution. On the left *y* axis show the normalized weights distribution (w.r.t. the total weight) for all records within each zone, while on the right *y* axis we show the number of records in the sub-zone we reversed engineering.

Variations within a zone: to determine how much variation exist in each zone, we compute its interquartile range (IQR)– which is show by the height of the box in Figure 7. The larger the box, the larger the weight distribution variation. For A records, we see zones (such as Germany and United States' zones – de and us) with a large variation, while others far more concentrated (Portugal's jp and India's tw).(We show their respective CDF in Figure 6).

The reality for the NTPPool, however, is that the diversity that clients see from a zone is not necessarily linearly correlated with the size of the pool: a small fraction of the servers may dominate a zone, which is a direct result by how much each volunteer is able to donate as weights.

4.4 Vendor zones and IPv6 clients

4.4.1 Vendor zones

The NTPPool operators encourage vendors to ask for their own DNS sub-zones [40]. However, we did not find any vendor zones while reverse engineering in NTPPool zone files (they are not publicly disclosed, and server's report pages do not list them).

This could be due to two things: either vendor zones are kept separated from the rest, or they use the same geographical codes from the rest of the NTPPool. To determine that, we carry out experiments with RIPE Atlas, asking 32 probes located in Argentina to query for the A record of the Android vendor zone (2.android.pool.ntp.org). We analyzed the A records returned to these responses (dataset ArgV4-Android in [45]), and found only 7 distinct IP addresses,as shown in Table 6, all of them belonging to the ar geographical zone. On the same day (2021-08-23), there were only 7 servers active in the ar zone [33] .

Therefore, we can conclude that the vendor zones seem to be a *replica* of the geographical zones – only that they give the

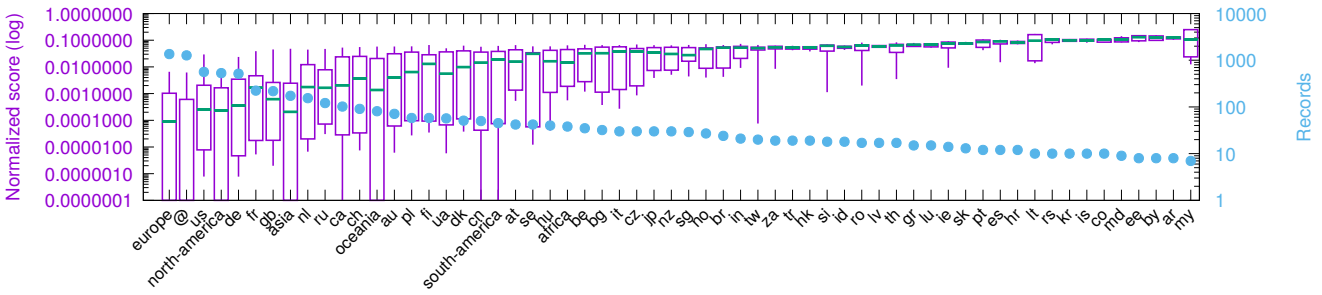

Figure 7: A records: Top 50 zones records weights distribution . Whiskers show *min* and *max* weight values, boxes shows IQRs, median shown as a green line.

NTP server	ArgV4-Android	ArgV6
162.159.200.1	748	182
162.159.200.123	748	182
168.96.251.195	747	181
168.96.251.227	379	52
168.96.251.197	195	61
168.96.251.226	121	63
170.210.222.10	54	7

Table 6: Query distribution for ArvgV4-Android and ArgV6 experiments. Datasets: [45].

ability to the NTPPool operators to remove them in case of a vendor specific error. As such, clients using vendor sub-zones are still bound by the geographical zones and their limits. As such, by reverse engineering only the geographical zones, we are likely to be covering most of the vendor zones given they make use ultimately of their geographical zones.

4.4.2 IPv6 clients

Clients can send queries over IPv4 and IPv6 to the NTPPool authoritative servers, and they can be used to retrieve both A or AAAA records (we found 12 responsible authoritative servers for the NTPPool over IPv6 – Appendix A).

To determine if IPv6 clients have a different view from the NTPPool, we configure 12 RIPE Atlas probes to send queries over IPv6 from Argentina to the NTPPool authoritative servers, as experiments EnumV4 and EnumV6. Our goal is to determine if they would be also mapped to the Argentina's ar sub-zone, or if they would fall back to the continent or global zone.

Table 6 shows the results (Argv6 column and dataset). We see that IPv6 clients geolocated in Argentina, when asking for A records of 2.pool.ntp.org, are also mapped to the ar sub-zone. We confirm that by manually checking the IP address against Maxmind's gelocation database. We can therefore conclude that GeoDNS uses the same mapping process for IPv4 and IPv6 clients.

5 Emulating queries from all countries

The results from §4 showed us that *countries* (and continents) is what GeoDNS uses to choose what NTP servers what clients see, and that NTP server *weights* impact query distribution within a zone. As such, in this section, we are interested in determining the view from *all countries* from the pool (§3 provides 166 countries). Besides that, by running our experiments, we are not bound by query limits and do not generate traffic to the NTPPool authoritative servers.

5.1 Setup

Server side: the first step consists in providing GeoDNS with a zone file representative of the NTPPool. We use the zone file generated in §4, including the weights associated with each A/AAAA record (§4.3), but we slight change: all A/AAAA having a weight equal to 0 we set their weights to 1. Weights equal to zero mean that these A/AAAA records were reported in the NTPPool website but were not captured by our experiments from §3 using RIPE Atlas, either because they had low weights or because these servers were inactive, causing them to be removed from the NTPPool zone.

To compensate for that uncertainty, we assign a weight equal to 1 to all A/AAAA records that had 0 weight in our reverse engineered zone. In this way, these records also have (albeit small) chance of being served in the responses provided by GeoDNS. We use the setup from §4.2 to run GeoDNS server.

Client side: Given that GeoDNS uses the country (or continent) associated with an IP address to decide what clients see what NTP servers (§4), we can use a single IP address per country to determine the visibility of a client and its country. We then compile a *source list*, which consists of one IPv4 address per country that we obtain by querying Maxmind's GeoLite2 database (IPv6 addresses mapped to the same country would have similar responses, as we shown in §4.4.2).

Experiment: we configure our client to send 10k per country, for both A and AAAA records each. We choose 10k queries because it is larger than any sub-zone we have seen

	CCv4	CCv6
Qname	idiazabal.kaas	
Qtype	A	AAAA
VPs/Countries	247	247
Responses	2.47M	2.47M
Records per resp		
4	2.06M	1.98M
3	1.00M	0.06M
2	0.23M	0.40M
1	0.08M	0.03M

Table 7: Country Code Experiments

(the largest zone is the global zone A records, with 2834 addresses) and likely not large enough to go through the list of all IPs within a zone, given the role of server weights. Table 7 shows the experiments. We send 2.47M queries per experiment and collect both traffic and GeoDNS log files.

5.2 Results

For each obtained response from the pcap files, we extract the A/AAAA records, as well as the IP address of the simulated client (1 per country). Given we send 10k queries/IP per record type (A or AAAA), we them com pile a list of all NTP servers each client IP has seen. Then, we extract from the GeoDNS log files what sub-zone each client IP is mapped (LabelName in Listing 1).

For each IP address/country, we compute the sub-zone visibility, which is the total number of unique NTP servers seen by the address/country divided by the entire sub-zone size. For example, if an IP is mapped to the us zone (size=561), and this IP retrieves 452 unique records from the us zone, we say its visibility ratio is $452/560 = .80$, or 80%.

Figure 8 shows how each geographical zone ranks, which we order by zone size, for A records (we show the AAAA graph in Appendix D). On the left y axis, we show the number of unique NTP servers that the client retrieved from our servers with 10k queries. On the right y axis, we show the zone visibility for the country (how many NTP severs were retrieved from the total zone size). On the x axis we show countries/continents.

We start with Figure 8, which shows the results for the measurement CCv4 (A Records). We divide the countries form this figure into several groups. First, the salmon area on the left shows that there are 41 countries that can only see up to 3 NTP severs (A records), in all continents/areas: Israel in the middle east, Nigeria in Africa, Philippines in Southeast Asia, and Mozambique in Africa. For AAAA records, we see in Figure 12 that there are 48 countries. These are the most concerning countries, and we reported these finds to the NTPPool operators. We show the full list on Appendix E. The conclusion is that despite of thousands of servers being available to the NTPPool, users from these countries can only receive time information from up to three servers.

Next category we have countries highlight in grey areas; these are countries that *do not* have a country sub-zone in the pool (or have empty sub-zones), and wind up being mapped to the continent or the global zone. These include Bolivia in South America, Ghana in Africa, Fiji Islands in Oceania, and Yemen in Asia. As can be seen in the figure, these countries are able to reach a large number of NTP servers (min 38 for A records from Africa zone, 13 for South America AAAA records). These countries are better off in server diversity that the countries with few ($<= 3$ NTP servers) that the 40+ countries on the left that have their own zones. (Notice, however, that countries mapped to the Europe zone (EU) have only been able to see 50-60% of the entire EU zone, so given more queries, they would be able to reach even more severs).

We see 7 and 8 countries are mapped to the global (@) zone, for A and AAAA records, respectively. These include South Sudan (ss), Bouvet Island (.bv), and Antarctica (aq). We found that South Sudan, for example, has no continent data in its target logs (Listing 1) – only country and global. Bouvet Island, in turn, has Antarctica as continent, which is an empty zone.

The final group is the areas left in white, which show countries that have their own zone with the NTPPool. The left most white area cover countries with 4–32 servers (A records), and 4–13 (AAAA records). Note that our 10k query per countries led to complete visibility of these zones. They include countries such as Greece, Ireland (in Europe), Thailand, South Africa, and Brazil.

The better off countries are the ones listed between Asia and North America: they have their own zones in the pool. For A records, we have the U.S. and Germany with 561 and 514 records in their zones, respectively. For AAAA records, we have Germany with 362 records.

5.2.1 Time providers per country

Next we determine the number of time providers per country. To do that, we count, per country, the number of Autonomous Systems (ASes) that the NTP servers in their zones belong to. Figure 9 shows the CDF. We see that there are 13 countries that have a single time provider for IPv4, 13 countries with no IPv6 provider, and 42 countries with a single IPv6 time provider. We see that only $\sim$ 40% of the countries have more than 10 IPv4 time providers, and 22% of countries have more than IPv6 providers. We include the full list in Appendix E.

Cloudflare's participation: Cloudflare's NTP service is also offered freely on the NTPPool project. Given that Cloudflare operates a large CDN using anycast, with a large number of sites, it could afford to share many more resources than, for example, regular users.

Given that having a single party with very high weight in the NTPPool DNS zone, it could skew the distribution of

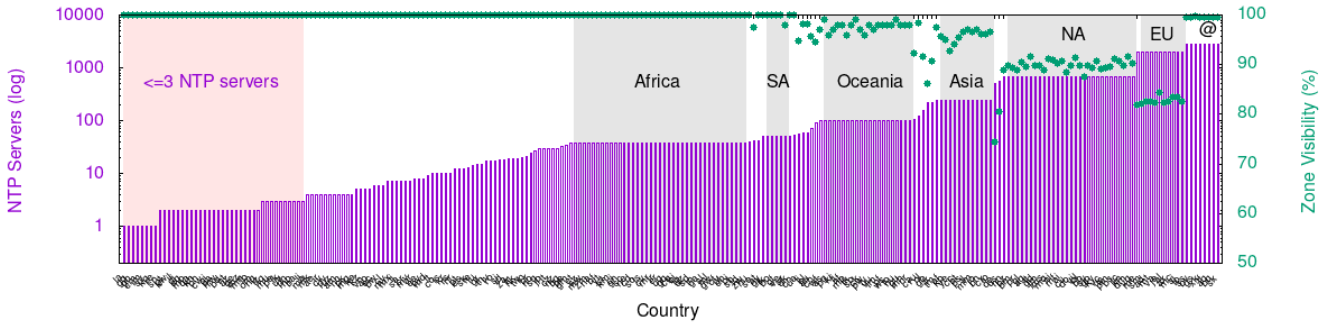

Figure 8: CCv4 :Countries visibility of the NTPPool.

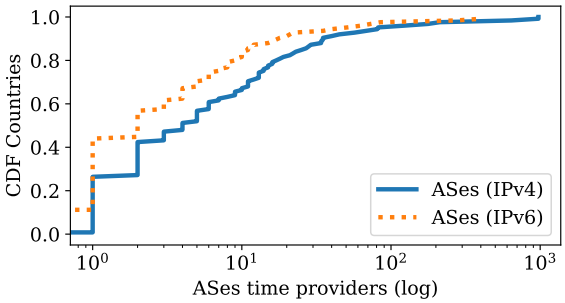

Figure 9: CDF time providers per country

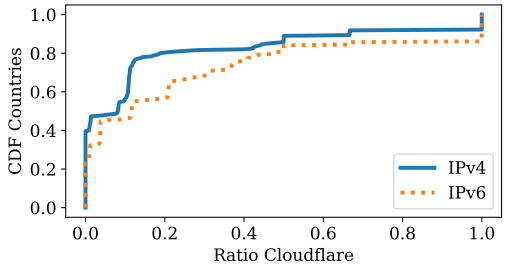

Figure 10: CDF Cloudflare's participation per country

A	AAAA
az,bh,cw,dj,eg, gt,ht,il,kh,kw lb,mn,mo,mz,ng, om,pa,qa,rw,sn pk,qa,rw,sn	ae,ao,az,bh,co,cw,dj eg,gt,hr,ht,il,iq,kw,lb lk,mg,mn,mo,mu,mv,mz ng,np,om,pa,pe,ph pk,py,qa,rw,sn,tz,vn

Table 8: List of Countries that fully depend on Cloudflare for NTPPool time service.

servers towards a single provide. Therefore, we calculate, for each country, the percentage of all records returned that belonged to Cloudflare NTP services, in order to determine if Cloudflare dominates parts of the traffic.

Figure 10 shows the CDF of countries and Cloudflare's participation in the responses (we analyze 247 countries in total). First, we see a spike on x=1, which corresponds to countries that solely rely on Cloudflare as time provider. We found that 24 and 36 countries, for A and AAAA records, respectively, only rely on Cloudflare, as shown in Table 8. The include populous countries such as Pakistan, Nigeria, and are distributed all over the world. It is also interesting to notice that some countries are only present on the AAAA list, which means that they depend on Cloudflare completely only for AAAA records.

To confirm if these sub zones really have only Cloudflare as solely NTP provider, we send DNS queries *each sub-zone directly*, *i.e.,* querying the A records o of `$zone.pool.ntp.org`, where each `$zone` is the 23 sub-zones mentioned. We confirm that all of them have only Cloudflare as IPv4 NTP provider.

On the other extreme, we see that many countries *are not* served by Cloudflare (spike at $x = 0$ in Figure 10): 98 countries for A records, 57 for AAAA). These are countries that have their own DNS sub-zones in the pool (so they do not have to fall back to the continent's zone) and do not have Cloudflare in their sub-zones – including Austria `at` and Russia (`ru`).

5.2.2 Internet Users per NTP server

Next, we take the results per countries from the previous subsection and calculate the number of Internet users per each NTP server that their mapped zone has. We use Internet user data obtained from the `Wikipedia` entry on Internet users, which draws from various data sources [58].

Table 9 shows the results. At the top of the list, we see Nigeria, with 68.1 million users per NTP server provided by the NTPPool (we compute IPv4 and IPv6 results apart). All countries in this talbe have at least 6.7M users NTP server. In contrast, there are 103 and 184k Internet users per NTP servers (IPv4 and IPv6, respectively) for clients based in The Netherlands. Even in the countries from Table 9 may not necessarily

	A			AAAA		
	CC	Servers	Ratio	CC	Servers	Ratio
1	ng	2	68.1	ng	2	68.1
2	in	21	37.1	in	17	45.8
3	eg	2	27.3	ir	2	39.1
4	ph	3	24.3	ph	2	36.5
5	ir	4	19.52	cn	28	35.3
6	cn	52	19.0	vn	2	34.1
7	vn	5	13.6	eg	2	27.3
8	id	18	11.8	id	9	23.5
9	mx	7	12.7	br	8	20.0
10	br	24	6.7	co	2	15.2

Table 9: Top 10 Countries with highest number of Internet Users per NTP server from the NTPPool (in million)

.

be those having fewer servers, they all share a common need: more NTP servers in their sub-zone are needed.

6 Discussion and recommendations

The NTPPool project has been providing an invaluable service for millions of users, servers, and devices all over the world, for over 18 years (since Jan. 2003). Their operators and developers – all volunteers – deserve to be acknowledged for the continuous services they have been providing.

Our results and findings from this research can be used by the NTPPool operators to improve even further their services. In specific, we make the following recommendations:

Mitigate single point-of-failures: we found 20 and 35 countries (IPv4 and IPv6) that have a single time provider: Cloudflare (§5). This creates a single point of failure, meaning that *all users* from these countries were experience issues if there are issues with Cloudflare. As we have shown in §4.2, the NTPPool monitoring system is very fast to remove poorly behavior servers from the pool; however, they use a single vantage point in the U.S., and this may not capture the experience of users in these affected countries, measured by the networking conditions between clients and servers.

Another way to mitigate single points of failure would be to force GeoDNS to use the continent zones for clients whose country's zones are smaller than a threshold, such as 10 NTP servers. That would introduce more time provider resiliency, although it may provider the clients with servers with higher latency and jitter. However, NTP clients can sort out the best servers for them, latency and jitter wise, with using their built-in metrics. (**[RF: SM-1]**A per-client solution is also possible, by having each client to use either continent or the global zone instead of the recommended `[0-3].pool.ntp.org` [34] or their vendor zones. However, this approach requires configuring each individual client, which is not scalable, and using NTP server from other countries **[RF: SM-3]** and may increase the chances of NTP traffic being filtered [54]).

We showed in §5 that for a country it is actually better, diversity wise, to *not have* a dedicated zone in GeoDNS and, instead, use the continent zone, if the country zone was suppose to have few servers. We see these countries in the gray areas in Figure 8. Continent zones tend to be far larger than most country zones, so that creates a pool of diversity larger than having few servers in a country zone

Add more servers in countries with few: we hope also that the results presented here can be use by policy makers and national Internet agencies to increase the deployment of public NTP servers in countries where is most needed.

7 Related work

A previous study to ours on NTP servers [50] has also crawled the NTPPool authoritative servers to enumerate them (similar to what we did in §3). They used a single vantage point whereas we used 9k with RIPE Atlas (§3). Whereas they use their datasets to analyze the distribution of time providers, we use it to reverse engineer the NTPPool zones and scrutinize GeoDNS behavior, which allowed us to determine how it works and ultimately emulating queries from all countries, which covers all users everywhere.

While not directly related to ours, several other studies covered either the NTP protocol vulnerabilities [22], how NTP clients can be vulnerable to malicious time servers [43] or how NTP servers can be used in distributed denial-of-service (DDoS) amplification attacks [11] (which spoofed queries are sent with the source address of the target, which then receives unsolicited traffic), or off-path attacks using DNS cache poisoning [19]. While related to ours, they do not focus on the NTPPool itself, as we do.

With regards NTP traffic characterization, a previous study has characterized traffic at the NIST's NTP servers [52] or running several NTP servers that are part of the NTPPool [50]. Our study, in turns, solely analyzes DNS traffic towards the NTPPool DNS servers (or emulated), which then is used to point clients to NTP servers.

8 Conclusions and future work

The NTPPool provides a core service to millions of users and devices worldwide. We have scrutinized its DNS authoritative service, which is responsible to decide *who* gets to see what servers. Despite having $\sim$ 4k NTP servers in the NTPPool, we see that most countries (60%) are served by 10 time providers only. Worse, many countries – including Nigeria, Israel and Pakistan – have a single provider – which in the case of Nigeria is about four times the population of California. This, however, is due to the way GeoDNS works, by confiding clients to country sub-zones.

We have already shared part of these results with the NTPPool operators, and we hope it can be used to improve

their services. Overall, our study brings more transparency into who gets to tell time for clients all over the world.

As future work, we intend to investigate the quality of the services provided by the NTP servers that the NTPPool share with its clients – in terms of latency and jitter.

References

[1] S. Alexander and R. Droms. DHCP Options and BOOTP Vendor Extensions. RFC 2132, IETF, March 1997.

[2] Apple. Apple NTPService. time.apple.com, August 2021.

[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033, IETF, March 2005.

[4] J. Arkko. Centralised Architectures in Internet Infrastructure. Internet Draft, November 2019.

[5] Jari Arkko. The influence of Internet architecture on centralised versus distributed Internet services. *Journal of Cyber Policy*, 5(1):30–45, 2020.

[6] Arkko, Jari and Tramme, B. and Nottingham, M and Huitema, C and Thomson, M. and Tantsura, J. and ten Oever, N. Considerations on Internet Consolidation and the Internet Architecture. Internet Draft, July 2019.

[7] Ask Bjørn Hansen. GeoDNS sample zone file. http://tmp.askask.com/2012/08/dns/ntppool.org.json.big, August 2012.

[8] Ask Bjørn Hansen. GeoDNS servers. https://github.com/abh/geodns/, July 2021.

[9] R. Bush and R. Austein. The Resource Public Key Infrastructure (RPKI) to Router Protocol. RFC 6810, IETF, January 2013.

[10] Cloudflare. Cloudflare Time Service. https://www.cloudflare.com/time/, August 2021.

[11] Jakub Czyz, Michael Kallitsis, Manaf Gharaibeh, Christos Papadopoulos, Michael Bailey, and Manish Karir. Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks. In *Proceedings of the 2014 ACM Conference on Internet Measurement Conference*, IMC, pages 435–448. ACM, November 2014.

[12] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, IETF, August 2008.

[13] R. Droms. Dynamic Host Configuration Protocol. RFC 2131, IETF, March 1997.

[14] Toby Ehrenkranz and Jun Li. On the state of IP spoofing defense. *ACM Transactions on Internet Technology (TOIT)*, 9(2):1–29, 2009.

[15] R. Gayraud and B. Lourdelet. Network Time Protocol (NTP) Server Option for DHCPv6. RFC 5908, IETF, June 2010.

[16] Google. Google Public NTP. time.apple.com, August 2021.

[17] P. Hoffman, A. Sullivan, and K. Fujiwara. DNS Terminology. RFC 8499, IETF, November 2018.

[18] ISS. Zonemaster: pool.ntp.org Contains errors! https://zonemaster.iis.se/en/?resultid=007295f8ad2799a7, June 2021.

[19] Philipp Jeitner, Haya Shulman, and Michael Waidner. The impact of dns insecurity on time. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 266–277, 2020.

[20] Cecilia Kang and David McCabe. Lawmakers, united in their ire, lash out at big tech's leaders. *New York Times*, July. 29 2020.

[21] Aqsa Kashaf, Vyas Sekar, and Yuvraj Agarwal. Analyzing Third Party Service Dependencies in Modern Web Services: Have We Learned from the Mirai-Dyn Incident? In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 634–647, New York, NY, USA, 2020. Association for Computing Machinery.

[22] Aanchal Malhotra, Isaac E Cohen, Erik Brakke, and Sharon Goldberg. Attacking the Network Time Protocol. In *Proceedings of the 23rd Network and Distributed System Security Symposium (NDSS 2016)*, 2016.

[23] Maxmind. Maxmind, 2021.

[24] D. McPherson, D. Oran, D. Thaler, and E. Osterweil. Architectural Considerations of IP Anycast. RFC 7094, IETF, January 2014.

[25] Microsoft. Microsoft NTPService. time.windows.com, August 2021.

[26] D. Mills, J. Martin, J. Burbank, and W. Kasch. Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905, IETF, June 2010.

[27] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034, IETF, November 1987.

[28] Giovane C. M. Moura, Sebastian Castro, Wes Hardaker, Maarten Wullink, and Cristian Hesselman. Clouding up the internet: How centralized is dns traffic becoming? In

Proceedings of the ACM Internet Measurement Conference, IMC '20, page 42–49, New York, NY, USA, 2020. Association for Computing Machinery.

[29] Giovane C. M. Moura, Ricardo de O. Schmidt, John Heidemann, Wouter B. de Vries, Moritz Müller, Lan Wei, and Christian Hesselman. Anycast vs. DDoS: Evaluating the November 2015 root DNS event. In *Proceedings of the ACM Internet Measurement Conference*, pages 255–270, Santa Monica, California, USA, November 2016. ACM.

[30] Giovane C. M. Moura, John Heidemann, Ricardo de O. Schmidt, and Wes Hardaker. Cache me if you can: Effects of DNS Time-to-Live. In *Proceedings of the ACM Internet Measurement Conference*, pages 101–115, Amsterdam, the Netherlands, October 2019. ACM.

[31] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120, IETF, July 2005.

[32] NTP Pool. All Pool Servers. https://www.ntppool.org/zone, 6 2021.

[33] NTP Pool. Argentina — ar.pool.ntp.org. https://www.ntppool.org/zone/ar, June 2021.

[34] NTP Pool. How do I use pool.ntp.org? https://www.ntppool.org/en/use.html, June 2021.

[35] NTP Pool. NTP Pool DNS servers. https://www.pool.ntp.org/dns-server.html., July 2021.

[36] NTP Pool. NTP servers in Global, pool.ntp.org. https://www.ntppool.org/zone/@, June 2021.

[37] NTP Pool. pool.ntp.org: statistics for 51.255.142.175 . https://www.ntppool.org/scores/51.255.142.175/, August 2021.

[38] NTP Pool. pool.ntp.org: the internet cluster of ntp servers. https://www.ntppool.org/en/, June 2021.

[39] NTP Pool. poo.ntp.org: Statistics for 95.217.188.206. https://www.ntppool.org/scores/95.217.188.206, 6 2021.

[40] NTP Pool. The NTP Pool for vendors. https://www.ntppool.org/en/vendors.html, June 2021.

[41] NTP Pool Community. The time has come: we must enable IPv6 entirely. https://community.ntppool.org/t/dns-configuration-errors-issues-with-pool-ntp-org/2039, June 2021.

[42] C. Partridge, T. Mendez, and W. Milliken. Host Anycasting Service. RFC 1546, IETF, November 1993.

[43] Yarin Perry, Neta Rozen-Schiff, and Michael Schapira. A Devil of a Time: How Vulnerable is NTP to Malicious Timeservers? In *Proceedings of the 28th Network and Distributed System Security Symposium (NDSS 2021)*, 2021.

[44] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. IP Geolocation Databases: Unreliable? *SIGCOMM Comput. Commun. Rev.*, 41(2):53–56, April 2011.

[45] RIPE NCC. RIPE Atlas measurement IDS. https://atlas.ripe.net/measurements/ID, June 2021. , where ID is the experiment ID: EnumV4: 32025718, EnumV6: 32058440, ArgV4: 31789516, ArgV4-Emul:31830680, ArgV4-Android: 31992051, DE-Android:31970486, ArgV6:32001506.

[46] RIPE NCC Staff. RIPE Atlas: A Global Internet Measurement Network. *Internet Protocol Journal (IPJ)*, 18(3):2–26, Sep 2015.

[47] RIPE Network Coordination Centre. RIPE Atlas. https://atlas.ripe.net, 2020.

[48] Root Server Operators. Root DNS, May 2020. http://root-servers.org/.

[49] Root Zone file. Root, February 2020. http://www.internic.net/domain/root.zone.

[50] Teemu Rytilahti, Dennis Tatang, Janosch Köpper, and Thorsten Holz. Masters of Time: An Overview of the NTP Ecosystem. In *2018 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 122–136, 2018.

[51] Bruce Schneier. Censorship in the Age of Large Cloud Providers, 2018.

[52] Jeff A Sherman and Judah Levine. Usage analysis of the nist internet time service. *Journal of Research of the National Institute of Standards and Technology*, 121:33, 2016.

[53] Internet Society. Consolidation in the Internet Economy, 2019.

[54] Steven Sommars. NTP Filtering (Delay & Blockage) in the Internet. https://weberblog.net/ntp-filtering-delay-blockage-in-the-internet/, September 2020.

[55] Raffaele Sommese, Leandro Bertholdo, Gautam Akiwate, Mattijs Jonker, van Rijswijk-Deij, Roland, Alberto Dainotti, KC Claffy, and Anna Sperotto. MAnycast2—using anycast to measure anycast. In *Proceedings of the ACM Internet Measurement Conference*, Pittsburgh, PA, USA, October 2020. ACM.

Server	a	b	c	d	e	f	g	h	i
# IPv4	6	2	3	4	3	3	1	1	1
# IPv6	3	1	1	3	3	2	0	1	1

Table 10: DNS infrastructure of `[a-i].ntps.org` (2021-06-29).

	IPv4	IPv6	ASes(IPv6)	ASes(IPv6)
Unique	24	15	20	14
Resp.	21	12	17	12
Not Resp.	3	2	3	2

Table 11: Agregated data from `NTPPool` authoritative servers (2021-06-30) [18].

[56] S. Thomson, C. Huitema, V. Ksinant, and M. Souissi. DNS Extensions to Support IP Version 6. RFC 3596, IETF, October 2003.

[57] Adrian von Bidder. ntp DNS round robin experiment. https://groups.google.com/g/comp.protocols.time.ntp/c/cShrN7imCJ0, January 2003.

[58] Wikiepdia. List of countries by number of Internet users. https://en.wikipedia.org/wiki/List_of_countries_by_number_of_Internet_users, 08 2021.

A `NTPPool` DNS infrastructure

NTP Servers under the `NTPPool` are available under `*.pool.ntp.org` and `pool.ntp.org`. The zone `pool.ntp.org` has, in turn, `[a-i].ntps.org` as authoritative servers (Figure 1). The `NTPPool` also uses volunteer authoritative DNS servers [35].These are the servers that are ultimately responsible for answering resolver queries about the `NTPPool`, which in turn allows clients to synchronize their clocks. Given that, it is vital that the `NTPPool` DNS infrastructure is highly redundant, which we investigate next.

To do that, we query directly one of their parent's authoritative servers (`anyns.pch.net`, *i.e.,* one of the authoritative servers for `[a-i].ntps.org`) over the IPv4 address for the A and AAAA records of `[a-i].ntps.org`. Table 10 shows the results. We see that in total, there are 24 A records and 15 AAAA records associated with the `[a-i].ntps.org` domains.

Taking as whole, Table 11 shows the number of unique addresses for all authoritative servers of the `NTPPool`. We see that 3 IPv4/ASes and 2 IPv6/ASes are not responsive (we keep a snapshot of the measurement here [18]). We have reported this to the operators (2021-06-30) [41], but it has not been fixed by time of this writing (2021-07-07).

Another layer of redundancy can be added by using IP anycast [24, 42], which is heavily used in the Root DNS zone and multiple other zones as well. We use IPv4 anycast census the 2021 dataset from [55] and match the A records used by the authoritative servers for the `NTPPool`. We see that none of them use IP anycast according to the 2021 census.

Regardless of these errors and lack of anycast, we can conclude that the AS and IP diversity of the addresses used by the parent DNS servers of the pool is highly diverse (Table 11) – likely to the diversity in volunteering organizations willing provide the service. For comparison, the Top-level domains (TLDs) from Root DNS zone [48, 49] have, on average, 2.7 ASes for IPv4 and IPv6 addresses (2021-07-05).

B Dynamic behavior from the `NTPPool` authoritative servers

Listing 3 shows the RIPE Atlas logs from the Guernsey Probe, obtained from measurement EnumV4 [45].

```
#timestamp,AtlasProbeID,Reponse
1630011101,17580,|51.255.142.175
1630011405,17580,|51.255.142.175
1630011696,17580,|37.221.193.210|149.156.70.60|
                 185.57.191.229|94.16.114.254
1630012009,17580,|213.239.234.28|194.58.204.148|
                 95.215.175.2|54.36.152.158
1630012309,17580,|78.36.18.184|138.201.16.225
                 62.116.130.3|212.83.158.83
1630012598,17580,|49.12.125.53|85.199.214.100|
                 85.236.36.4|178.62.250.107
1630012903,17580,|217.114.59.3|217.114.59.66|
                 213.239.234.28|130.208.87.151
1630013207,17580,|51.255.142.175
1630013498,17580,|51.255.142.175
```

Listing 3: Dynamic behavior observed with `GeoDNS` in the wild. .

Listing 4 shows the `NTPPool` score with respect the 51.255.142.175 server, measured from the monitoring system from the `NTPPool`, obtained from [37].

```
ts_epoch,ts,offset,step,score,monitor_id,monitor_name,
    leap,error
1630012924,"2021-08-26 21:22:04",0.000396957,1,10.1,9,"
    San Jose, CA, US",0,
1630012924,"2021-08-26 21:22:04",0.000396957,1,10.1,,,0,
1630012190,"2021-08-26 21:09:50",0.001251177,1,9.6,9,"San
     Jose, CA, US",0,
1630012190,"2021-08-26 21:09:50",0.001251177,1,9.6,,,0,
1630011332,"2021-08-26 20:55:32",0,-5,9,9,"San Jose, CA,
    US",,"i/o timeout"
1630011332,"2021-08-26 20:55:32",0,-5,9,,,,"i/o timeout"
1630010513,"2021-08-26 20:41:53",0.002216952,1,14.8,9,"
    San Jose, CA, US",0,
1630010513,"2021-08-26 20:41:53",0.002216952,1,14.8,,,0,
1630013498,17580,|51.255.142.175
```

Listing 4: `NTPPool` score files for server 51.255.142.175 .

C Zone File ArgV4-Emul experiment

```
{
  "ttl": 390,
  "serial": 2,
  "data": {
    "": {
      "ns": [
        "ns.zeit1.org",
        "ns.zeit2.org"
      ],
      "a": [
        [
          "162.159.200.1",
          "37580"
        ],
        [
          "168.96.251.227",
          "31142"
        ],
        [
          "170.210.222.10",
          "28599"
        ],
        [
          "168.96.251.226",
          "25707"
        ],
        [
          "181.93.10.58",
          "24878"
        ],
        [
          "168.96.251.195",
          "24731"
        ],
        [
          "168.96.251.197",
          "17223"
        ],
        [
          "162.159.200.123",
          "14838"
        ]
      ]
    }
  },
  "max_hosts": 2
}
```

Listing 5: ArgV4-Emul zone file

D Extra figures

Figure 11 shows the top AAAA 50 zones and the weight distributions of the servers within each zone,

Figure 12 show the countries visibility of the `NTPPool`, for for AAAA records.

E Extra tables

Table 12 show shows the list of countries with fewer than 3 NTP IPv4 servers (A records). Table 13 the list of countries with fewer than 3 NTP IPv6 servers (AAAA),

Table 14 shows the country codes, country names, and the IP addresses we used for each country on §5.

	countryIP	zoneMapped	subZoneSize
0	la	la	1
1	gg	gg	1
2	gh	gh	1
3	cm	cm	1
4	im	im	1
5	sy	sy	1
6	kg	kg	1
7	re	re	1
8	sn	sn	2
9	gt	gt	2
10	kw	kw	2
11	il	il	2
12	dj	dj	2
13	eg	eg	2
14	mo	mo	2
15	qa	qa	2
16	kh	kh	2
17	bh	bh	2
18	cw	cw	2
19	gi	gi	2
20	pa	pa	2
21	rw	rw	2
22	bd	bd	2
23	ht	ht	2
24	az	az	2
25	ge	ge	2
26	mz	mz	2
27	mn	mn	2
28	lb	lb	2
29	om	om	2
30	ng	ng	2
31	tz	tz	3
32	iq	iq	3
33	mu	mu	3
34	pe	pe	3
35	lk	lk	3
36	uy	uy	3
37	ph	ph	3
38	np	np	3
39	ba	ba	3
40	li	li	3

Table 12: Countries with fewer than 3 NTP servers (A records)

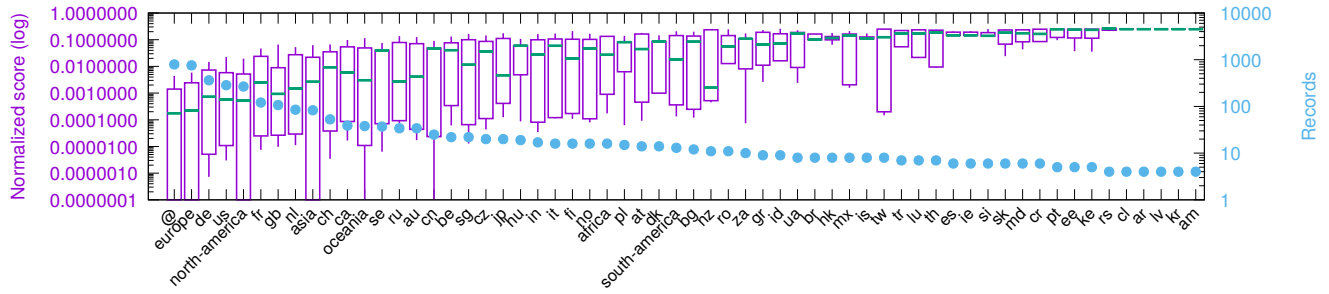

Figure 11: AAAA records: Top 50 zones records weights distribution . Whiskers show *min* and *max* weight values, boxes shows IQRs, median shown as a green line.

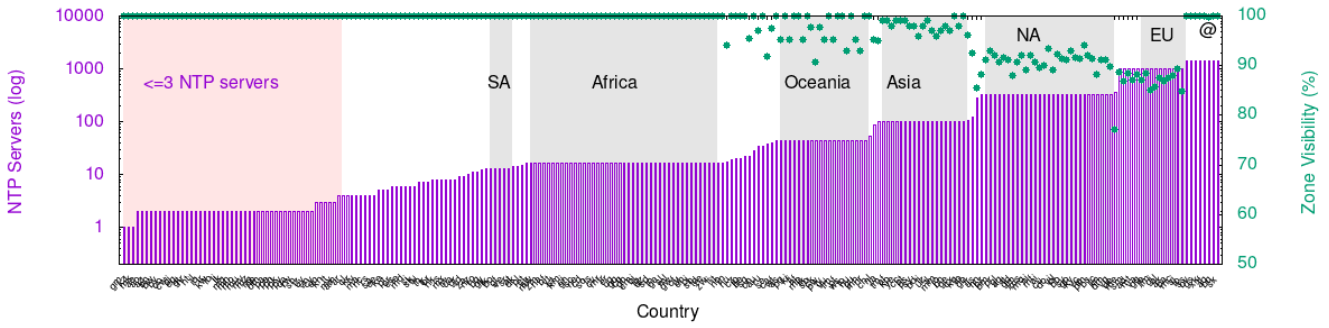

Figure 12: CCv6: Countries visibility of the `NTPPool`.

	countryIP	zoneMapped	subZoneSize
0	gm	gm	1
1	kz	kz	1
2	mk	mk	1
3	ae	ae	2
4	ao	ao	2
5	az	az	2
6	bh	bh	2
7	bw	bw	2
8	by	by	2
9	co	co	2
10	cw	cw	2
11	dj	dj	2
12	eg	eg	2
13	gt	gt	2
14	hr	hr	2
15	ht	ht	2
16	il	il	2
17	iq	iq	2
18	ir	ir	2
19	kw	kw	2
20	lb	lb	2
21	li	li	2
22	lk	lk	2
23	mg	mg	2
24	mn	mn	2
25	mo	mo	2
26	mu	mu	2
27	mv	mv	2
28	mz	mz	2
29	ng	ng	2
30	np	np	2
31	om	om	2
32	pa	pa	2
33	pe	pe	2
34	ph	ph	2
35	pk	pk	2
36	py	py	2
37	qa	qa	2
38	rw	rw	2
39	sn	sn	2
40	tz	tz	2
41	uy	uy	2
42	vn	vn	2
43	cy	cy	3
44	ec	ec	3
45	kh	kh	3
46	lt	lt	3
47	my	my	3

Table 13: Countries with fewer than 3 NTP servers (AAAA records)

CC	Country Name	IP address
ad	Andorra	34.99.136.1
ae	United Arab Emirates	2.16.158.1
af	Afghanistan	27.116.56.1
ag	Antigua and Barbuda	23.132.144.1
ai	Anguilla	69.57.224.1
al	Albania	2.58.82.1
am	Armenia	2.17.249.1
ao	Angola	41.63.160.1
aq	Antarctica	146.75.179.1
ar	Argentina	2.17.202.1
as	American Samoa	38.101.164.1
at	Austria	2.16.10.1
au	Australia	1.0.0.1
aw	Ecuador	45.4.88.1
ax	Åland	79.133.0.1
az	Azerbaijan	5.10.240.1
ba	Bosnia and Herzegovina	5.43.64.1
bb	Barbados	23.236.0.1
bd	Bangladesh	5.182.185.1
be	Belgium	2.17.107.1
bf	Burkina Faso	41.78.48.1
bg	Bulgaria	2.20.45.1
bh	Bahrain	3.5.220.1
bi	Burundi	2.18.11.1
bj	Benin	41.74.0.1
bl	Saint Barthélemy	90.31.74.1
bm	Bermuda	45.42.144.1
bn	Brunei	5.182.197.1
bo	Bolivia	34.100.4.1
bq	Bonaire_ Sint Eustatius_ and Saba	143.0.32.1
br	Brazil	2.16.15.1
bs	Bahamas	23.232.250.1
bt	Bhutan	5.182.196.1
bv	Bouvet Island	46.29.219.1
bw	Botswana	41.74.48.1
by	Belarus	5.100.192.1
bz	Belize	2.56.44.1
ca	Canada	2.22.72.1
cc	Cocos [Keeling] Islands	45.42.156.1
cd	DR Congo	5.175.77.1
cf	Central African Republic	41.78.120.1
cg	Congo Republic	41.75.64.1
ch	Switzerland	2.16.12.1
ci	Ivory Coast	41.61.12.1
ck	Cook Islands	14.137.40.1
cl	Chile	2.18.236.1
cm	Cameroon	2.16.134.1
cn	China	1.0.1.1
co	Colombia	4.33.232.1
cr	Costa Rica	8.242.196.1
cu	Cuba	57.74.110.1
cv	Cabo Verde	41.74.128.1

cw	Curaçao	23.209.95.1
cx	Christmas Island	104.224.54.1
cy	Cyprus	2.56.140.1
cz	Czechia	2.16.2.1
de	Germany	2.16.6.1
dj	Djibouti	41.189.224.1
dk	Denmark	2.16.63.1
dm	Dominica	45.74.22.1
do	Dominican Republic	23.140.80.1
dz	Algeria	5.180.66.1
ec	Ecuador	45.4.88.1
ee	Estonia	2.57.220.1
eg	Egypt	2.21.128.1
eh	Western Sahara	154.150.4.1
er	Eritrea	57.82.120.1
es	Spain	1.178.224.1
et	Ethiopia	45.59.144.1
fi	Finland	2.16.144.1
fj	Fiji	14.137.38.1
fk	Falkland Islands	80.73.208.1
fm	Federated States of Micronesia	43.248.156.1
fo	Faroe Islands	37.120.252.1
fr	France	1.179.112.1
ga	Gabon	41.72.224.1
gb	United Kingdom	2.16.14.1
gd	Grenada	23.208.167.1
ge	Georgia	2.57.60.1
gf	French Guiana	45.169.164.1
gg	Guernsey	5.62.84.1
gh	Ghana	2.16.77.1
gi	Gibraltar	2.58.8.1
gl	Greenland	37.18.44.1
gm	Gambia	41.76.8.1
gn	Guinea	41.77.184.1
gp	Guadeloupe	5.187.96.1
gq	Equatorial Guinea	41.79.48.1
gr	Greece	2.16.19.1
gs	South Georgia and the South Sandwich Islands	154.65.48.1
gt	Guatemala	24.152.52.1
gu	Guam	8.3.122.1
gw	Guinea-Bissau	45.42.213.1
gy	Guyana	57.74.246.1
hk	Hong Kong	1.32.205.1
hn	Honduras	45.4.84.1
hr	Croatia	2.58.32.1
ht	Haiti	45.74.24.1
hu	Hungary	2.58.168.1
id	Indonesia	3.5.35.1
ie	Ireland	2.16.138.1
il	Israel	2.22.233.1
im	Isle of Man	5.62.80.1
in	India	1.6.0.1
io	British Indian Ocean Territory	202.44.112.1
iq	Iraq	2.56.36.1

ir	Iran	2.144.0.1
is	Iceland	5.23.64.1
it	Italy	2.16.17.1
je	Jersey	5.35.160.1
jm	Jamaica	23.156.32.1
jo	Jordan	2.17.24.1
jp	Japan	1.0.16.1
ke	Kenya	14.137.168.1
kg	Kyrgyzstan	5.57.8.1
kh	Cambodia	1.32.252.1
ki	Kiribati	57.70.164.1
km	Comoros	41.194.33.1
kn	St Kitts and Nevis	23.131.208.1
kp	North Korea	175.45.176.1
kr	South Korea	1.11.0.1
kw	Kuwait	5.104.66.1
ky	Cayman Islands	23.188.0.1
kz	Kazakhstan	2.57.96.1
la	Laos	5.253.184.1
lb	Lebanon	5.8.128.1
lc	Saint Lucia	23.189.192.1
li	Liechtenstein	5.34.248.1
lk	Sri Lanka	23.49.160.1
lr	Liberia	41.57.80.1
ls	Lesotho	23.4.87.1
lt	Lithuania	5.20.0.1
lu	Luxembourg	2.17.201.1
lv	Latvia	2.58.16.1
ly	Libya	5.63.0.1
ma	Morocco	23.232.251.1
mc	Principality of Monaco	34.99.172.1
md	Moldova	2.56.0.1
me	Montenegro	31.204.192.1
mf	Saint Martin	38.99.116.1
mg	Madagascar	41.63.128.1
mh	Marshall Islands	45.59.139.1
mk	North Macedonia	5.32.176.1
ml	Mali	41.73.96.1
mm	Myanmar	23.199.72.1
mn	Mongolia	14.0.59.1
mo	Macao	17.91.136.1
mp	Northern Mariana Islands	8.3.112.1
mq	Martinique	41.77.244.1
mr	Mauritania	41.138.128.1
ms	Montserrat	104.224.6.1
mt	Malta	2.59.128.1
mu	Mauritius	41.72.213.1
mv	Maldives	5.62.61.1
mw	Malawi	41.70.0.1
mx	Mexico	8.14.224.1
my	Malaysia	1.9.0.1
mz	Mozambique	41.76.0.1
na	Namibia	41.63.192.1
nc	New Caledonia	27.122.0.1

ne	Niger	41.78.116.1
nf	Norfolk Island	103.43.204.1
ng	Nigeria	8.35.57.1
ni	Nicaragua	45.5.216.1
nl	Netherlands	2.16.1.1
no	Norway	2.18.172.1
np	Nepal	14.137.53.1
nr	Nauru	43.230.6.1
nu	Niue	49.156.48.1
nz	New Zealand	5.181.67.1
om	Oman	5.21.0.1
pa	Panama	5.252.152.1
pe	Peru	8.24.244.1
pf	French Polynesia	43.249.176.1
pg	Papua New Guinea	14.137.32.1
ph	Philippines	1.37.0.1
pk	Pakistan	14.1.104.1
pl	Poland	2.16.172.1
pm	Saint Pierre and Miquelon	70.36.0.1
pr	Puerto Rico	12.205.64.1
ps	Palestine	2.58.132.1
pt	Portugal	2.16.65.1
pw	Palau	57.70.176.1
py	Paraguay	24.152.40.1
qa	Qatar	2.23.168.1
re	Réunion	5.57.96.1
ro	Romania	2.17.116.1
rs	Serbia	5.22.160.1
ru	Russia	1.2.9.1
rw	Rwanda	41.74.160.1
sa	Saudi Arabia	2.59.54.1
sb	Solomon Islands	14.137.34.1
sc	Seychelles	2.56.10.1
sd	Sudan	41.67.0.1
se	Sweden	2.16.66.1
sg	Singapore	1.32.128.1
sh	Saint Helena	104.224.22.1
si	Slovenia	5.32.136.1
sj	Svalbard and Jan Mayen	45.59.151.1
sk	Slovakia	2.57.64.1
sl	Sierra Leone	41.78.84.1
sm	San Marino	31.193.32.1
sn	Senegal	41.82.0.1
so	Somalia	41.78.72.1
sr	Suriname	45.74.20.1
ss	South Sudan	41.79.24.1
st	São Tomé and Príncipe	45.42.228.1
sv	El Salvador	45.5.12.1
sx	Sint Maarten	131.161.84.1
sy	Syria	5.0.0.1
sz	Eswatini	5.175.76.1
tc	Turks and Caicos Islands	63.130.249.1
td	Chad	41.74.32.1
tf	French Southern Territories	45.59.175.1

tg	Togo	41.78.136.1
th	Thailand	1.0.128.1
tj	Tajikistan	37.98.152.1
tk	Tokelau	27.96.24.1
tl	East Timor	43.254.56.1
tm	Turkmenistan	45.59.159.1
tn	Tunisia	41.62.0.1
to	Tonga	14.137.33.1
tr	Turkey	2.16.150.1
tt	Trinidad and Tobago	23.3.72.1
tv	Tuvalu	14.137.42.1
tw	Taiwan	1.32.194.1
tz	Tanzania	2.17.250.1
ua	Ukraine	2.21.89.1
ug	Uganda	2.17.248.1
us	United States	2.19.128.1
uy	Uruguay	2.18.64.1
uz	Uzbekistan	5.101.221.1
va	Vatican City	45.42.143.1
vc	Saint Vincent and the Grenadines	23.170.80.1
ve	Venezuela	8.242.232.1
vg	British Virgin Islands	2.56.144.1
vi	U.S. Virgin Islands	8.26.16.1
vn	Vietnam	1.52.0.1
vu	Vanuatu	14.137.37.1
wf	Wallis and Futuna	27.125.192.1
ws	Samoa	43.241.164.1
xk	Kosovo	185.244.25.87
ye	Yemen	5.100.160.1
yt	Mayotte	41.242.116.1
za	South Africa	2.16.140.1
zm	Zambia	41.60.0.1
zw	Zimbabwe	41.57.64.1

Table 14: List of Country codes, Country Names, and IP addresses

Table 15 shows the number of time providers per country.

country	nASesV4	nASesV6
@	974	403
ae	2	1
africa	25	11
am	3	3
ao	3	1
ar	4	2
asia	122	51
at	29	11
au	28	16
az	1	1
ba	2	0
bd	1	0
be	13	10
bg	22	8
bh	1	1

br	15	6
bw	1	1
by	5	2
ca	40	18
ch	34	20
cl	3	2
cm	1	0
cn	23	15
co	5	1
cr	4	2
cw	1	1
cy	2	2
cz	16	10
de	81	63
dj	1	1
dk	21	9
ec	2	2
ee	5	3
eg	1	1
es	6	2
europe	640	274
fi	13	6
fr	35	21
gb	68	40
ge	1	0
gg	1	0
gh	1	0
gi	1	0
gm	0	1
gr	10	6
gt	1	1
hk	12	5
hr	4	1
ht	1	1
hu	13	4
id	11	8
ie	7	4
il	1	1
im	1	0
in	11	6
iq	2	1
ir	4	2
is	5	4
it	14	9
jp	18	12
ke	2	2
kg	1	0
kh	1	2
kr	6	4
kw	1	1
kz	5	1
la	1	0
lb	1	1
li	1	1

lk	2	1
lt	7	3
lu	9	5
lv	9	3
ma	2	2
md	5	3
mg	3	1
mk	3	1
mn	1	1
mo	1	1
mu	2	1
mv	3	1
mx	6	2
my	6	2
mz	1	1
nc	2	2
ng	1	1
nl	57	35
no	16	10
north-america	212	85
np	2	1
nz	19	7
oceania	45	22
om	1	1
pa	1	1
pe	2	1
ph	2	1
pk	2	1
pl	34	7
pt	6	3
py	2	1
qa	1	1
re	1	0
ro	11	6
rs	8	2
ru	82	17
rw	1	1
sa	2	2
se	17	10
sg	15	12
si	5	1
sk	10	5
sn	1	1
south-america	27	8
sy	1	0
th	13	4
tj	2	0
tr	11	4
tw	9	5
tz	2	1
ua	35	4
us	177	73
uy	2	1
uz	2	0

vn	4	1
za	14	8

Table 15: Time providers per country