BigTime: Characterizing Large Time Service Providers

Technical Report SIDN Labs 2025-12-01

Pascal Huppert
University of Twente and University of Münster
Enschede and Münster, The Netherlands / Germany
pascal.huppert@utwente.nl

Giovane C. M. Moura
SIDN Labs and TU Delft
The Netherlands
giovane.moura@sidn.nl

Marco Davids
SIDN Labs
The Netherlands
marco.davids@sidn.nl

Pieter-Tjerk de Boer
University of Twente
The Netherlands
p.t.deboer@utwente.nl

Ralph Holz
University of Münster and University of Twente
Germany / The Netherlands
ralph.holz@uni-muenster.de

Rein Fernhout
University of Twente
The Netherlands
r.p.j.fernhout@student.utwente.nl

Georgios Smaragdakis
TU Delft
The Netherlands
g.smaragdakis@tudelft.nl

Cristian Hesselman
SIDN Labs and University of Twente
The Netherlands
cristian.hesselman@sidn.nl

Abstract

Clock synchronization is crucial on the Internet, as core applications and protocols depend on it for proper functioning, including DNS caches, RPKI, and TLS. The Network Time Protocol (NTP) is the default protocol for clock synchronization on the Internet. For decades, free time services on the Internet have been provided by national institutes such as NIST and by the NTP Pool, which is a volunteer-based initiative. More recently, large cloud providers and OS vendors such as Microsoft, Apple, and Google have begun operating their own time services and setting their billions of systems/devices to use their NTP servers by default. This creates a level of service centralization around these providers, and yet there has been little scrutiny of these services and their characteristics. In this paper, we characterize seven large time service providers in terms of replication architecture, client mapping, accuracy and service provided. We found a large variation in service architecture, how they map clients to servers, and in their accuracy and (security) feature support. This includes violations of best practices, deviations from the correct time of up to 50 ms, the use of outdated NTP versions, and a lack of support for NTS, RPKI, and DNSSEC.

1 Introduction

Synchronized clocks are essential for modern societies, being required for financial transactions, power grid operations, and telecommunication networks [29]. On the Internet, core services and applications depend on clock synchronization to verify validity [21, 41, 50, 77], such as TLS [72], DNSSEC signatures [4], DNS caches [59] and RPKI [12]. Various services and applications can fail when wrong time information is served to clients, as in the 2012 US Naval Observatory (USNO) event, where their time services published wrong time information (off by 10 years), causing many network routers and Active Directory servers to experience outages [7, 45].

The Network Time Protocol (NTP) [50] is the Internet's default protocol for clock synchronization. NTP clients request time information from NTP servers, which, in turn, are synchronized with out-of-band high precision references, such as atomic clocks or GNSS like GPS and Galileo [50]. There have been many public NTP services on the Internet for decades, including NIST [61] in the US, NPL [38] in the UK, PTB [10] in Germany and the NTP Pool [64].

More recently, device vendors, such as Apple, Microsoft, Google, and Ubuntu started to provide their own time services [56] and, more importantly for our study, *setting them as default in their device*, sometimes even hardcoding them – for instance, Apple's iOS cannot be reconfigured [39]. Other cloud and content providers have also started their own time services, including Cloudflare [16], AWS [79], and Meta [66].

The implications of these settings are significant: billions of devices are configured to *use only* their vendor-provided time servers, regardless of network or geographic location. As shown in Table 1, there are at least 6.6B active devices

Provider	Domain Name	User Base
Microsoft	`time.windows.com`	1.4B [46]
Apple	`time,time-[macos,euro,ios].apple.com`	2.2B [3]
Google	`time.android.com`, `time.google.com`	3.0B [9]
Ubuntu	`ntp.ubuntu.com`	Unclear
AWS	`time.aws.com`	-
Cloudflare	`time.cloudflare.com`	-
Meta	`time,time[1-5].facebook.com`	-

Table 1: Evaluated time service providers. Providers highlighted are enabled by default in their respective operating systems (OSes).

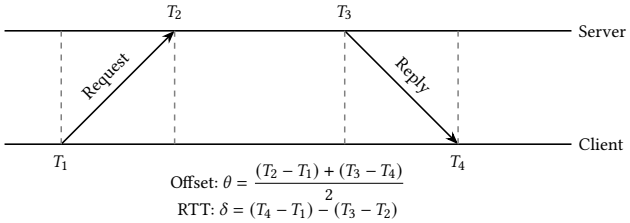


Figure 1: Timestamps used in NTP offset calculations

from Apple (laptops, smartphones and watches), Microsoft (servers, laptops, and smart devices), and Google (Android) configured by default to use their vendor's time servers.

Compared to DNS, where clients typically change resolvers when moving across networks, NTP settings typically remain constant (although DHCP [1] allows operators to configure provide NTP servers, clients may simply ignore it). Therefore, NTP centralization is *far more pronounced* than in other services, such as the DNS [55]: a small number of vendors effectively serve as the time authorities for billions of devices (Table 1). This makes the study of these time services even more pressing.

While a previous study [56] has examined the NTP Pool, which is widely used among Linux distributions, to date there have been no studies scrutinizing the time services provided by these other vendors and cloud/content providers – how their time services are set up, how they map clients to servers, and how accurate they are.

Therefore, in this paper we characterize the time service networks of large commercial vendors and cloud operators – "providers" hereafter (Table 1). We investigate their architecture – how they replicate their services to serve their billions of clients (§3) – and compare this to a well-documented replicated service: the Root DNS system [76]. We investigate the criteria used by each time service provider to map clients to specific NTP servers (§4). Finally, we evaluate the accuracy provided by each time provider and their services characteristics (§5).

We make the following contributions:

- We show a large diversity in terms of replication: out of seven providers, three use IP anycast [67], whereas four use multiple (up to 90) IP addresses. For the unicast-based providers, we find that client geolocation is the primary method to map clients to server, just as for the NTP Pool.
- We show how Microsoft's time service served half of our 9k vantage points (VPs) with a single NTP server, violating NTP best practices [71]
- We measure the accuracy of the providers on two GPS-synced VPs and show they deliver accurate time. However, we found that three of Microsoft's servers were out-of-sync during our measurements, by up to 50ms.
- We show the diversity of NTP service provided, in terms of stratum of the servers, NTP version, and IPv6 support. Only Cloudflare deploys the latest standards: Network Time Security (NTS) [23], RPKI, and DNSSEC, and supports IPv6. Microsoft is the only provider not supporting IPv6 yet, and provides NTP version 3 only (all others provide NTPv4).

2 Background

NTP is designed to synchronize the clocks of computer systems over variable-latency networks [50]. It provides an accuracy of milliseconds of Coordinated Universal Time (UTC). It uses a hierarchical system of time sources and servers ordered into levels by *stratum*, which indicates the distance from the reference clock. Stratum 1 servers are directly connected to a time source, for instance an appliance that uses Global Navigation Satellite Systems (GNSS) like GPS and Galileo, and stratum 2 severs are synchronized with stratum 1 servers.

Client-Server synchronization: NTP clients synchronize their clocks by exchanging mutually time-stamped requests and responses with NTP servers. The synchronization process involves four timestamps measured at both client and server side, as shown in Figure 1: the origin timestamp T_1 (client's time when the request leaves the client), the receive timestamp T_2 (server's time when the request arrives at the server), the transmit timestamp T_3 (server's time when the response leaves the server), and the destination timestamp T_4 (client's time when the response arrives back at the client). These times are used to compute the offset θ_{NTP}, which indicates the difference between server and client clock. Clients, in turn, *may* update their clock by the calculated offset, synchronizing it with the NTP server in this way.

Protocols: The Simple Network Time Protocol (SNTP) [49] is a simplified version of the Network Time Protocol (NTP) designed for less powerful computers and applications where accuracy of clock synchronization is less critical. SNTP uses

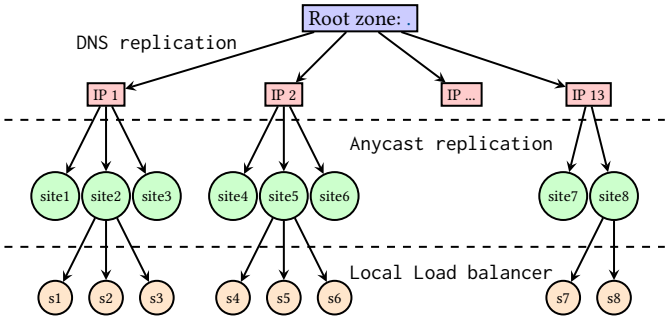


Figure 2: Service replication in the Root DNS system.

the same packet format and general process as NTP, and differs mainly in its client implementation. A recent non-peer-reviewed study [36] has shown how clients of all major OSes (Windows Server, macOS and Ubuntu up to Oct. 2025) use only SNTP-based clients which are vulnerable to time-shift attacks.

Network Time Security (NTS) [23] is an extension to NTP designed to provide cryptographic security for time synchronization. It uses TLS to provide ensures authenticity, integrity and partial confidentiality of time-stamped messages exchanged between clients and servers, protecting against various attacks such as replays and man-in-the-middle attacks. Of the time service providers from Table 1, only Cloudflare and Ubuntu currently support NTS. (Ubuntu has started support in Oct. 2025 [44]).

3 Time Services Architecture

The commercial time services we evaluate (Table 1) serve billions of devices daily, which is expected to be highly replicated. In this section, we carry out measurements to determine their replicaton choices. Before we do that, we summarize the replication strategies used by two services: NTP Pool [64] and the Root DNS server system [76].

3.1 Examples of highly replicated services

NTP Pool: The NTP Pool uses DNS-level replication to serve its clients. It has 3,176 IPv4 NTP servers (as of 2025-11-04), all associated with the `pool.ntp.org` domain name. Clients are mapped to a subset of these servers based on their geographical location [56]. Therefore, the NTP Pool uses DNS-level replication, because multiple IP addresses are associated with the zone.

The Root DNS system: The Root DNS system [76] comprises 13 IP addresses with 1,959 servers (as of 2025-11-04), referred to as instances. The root servers deploy three levels of replication (Figure 2): DNS-level, IP Anycast-level [67], and site-level (local load balancing).

	Domains	IPv4	IPv6	ASes v4	ASes v6
Microsoft	1	12	0	1	0
Apple	4	53	48	2	2
Google	2	4	4	1	1
Ubuntu	1	4	3	1	1
Amazon	1	90	90	2	2
Cloudflare	1	2	2	1	1
Meta	1	5	5	1	1

Table 2: DNS-level replication of time service providers.

At DNS-level, the system's only domain name (the dot .) is associated with 13 IPv4 and 13 IPv6 addresses. Each of these IP addresses is further replicated with IP anycast, which allows the same routing prefix to be announced from multiple locations on the globe, with each location referred to as an anycast site (site1–site3 in Figure 2). For instance, F-Root [17] has 360 anycast sites as of 2025-05-01. Each of these anycast sites may employ further, local replication using load balancers (`s1–s3` in Figure 2). For example, K-Root [14] uses three instances for its Amsterdam site.

3.2 DNS and Anycast Replication

We evaluate how the time service providers (Table 1) compare to these well-studied examples.

3.2.1 DNS-level replication. We first evaluate if the provider deploy DNS-level replication. To determine whether they rely on the client's geographical location when mapping users to servers as the NTP pool does, we configure 500 RIPE Atlas probes [74, 75] as vantage points to send one DNS query per 10 minutes, for 8h (datasets: [73]). We configure each vantage point to query one of the authoritative DNS servers of the time service directly[1]. This way, we bypass the recursive resolvers' caches and retrieve fresh responses with each new query.

For each provider, we then compute the number of unique IP addresses they return over the course of the measurement. Table 2 summarizes the results. We notice clear differences: AWS has 90 IPv4 servers, followed by Apple (53) and Microsoft (12). Google, despite likely having the largest client population, has only 4 IPv4 address.

3.2.2 Anycast replication. To determine if the IP addresses obtained from our Atlas measurements are deployed using

[1]For Microsoft and Apple, we do not configure the VPs to query the domains shown in Table 1 but the *actual* domains used. Microsoft's `time.windows.com` domain name redirects to `twc.trafficmanager.net` using a CNAME record. Apple's domains redirects to two domains: `[time,time-osx].g.aaplimg.com`. However, we find that both domain names point to the same NTP servers. Hence, we just report on `time.g.aaplimg.com` and include `time-osx.g.aaplimg.com` in Appendix B and Appendix C

Provider	Prefixes	Anycast	Sites (IPv4)	Sites (IPv6)
Microsoft	11/0	No	–	–
Apple	53/48	No	–	–
Google	1/1	Yes	41	No data
Ubuntu	4/3	No	–	–
AWS	46/9	No	–	–
Cloudflare	1/1	Yes	63	47
Meta	5/5	Yes	8	11

Table 3: Anycast replication of time service providers.

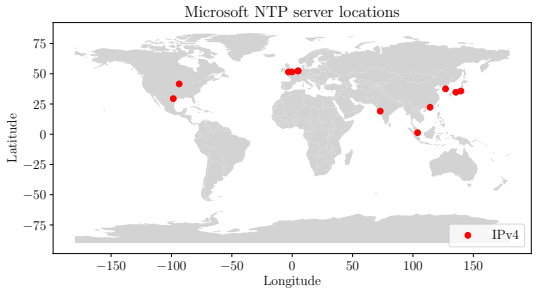
anycast §3.2, we check them against IPInfo [32], the BGP-Tools anycast prefix list [6], and the MAnycastR Anycast census [82, 83]. We consider an IP as anycast if it is listed in one or more of those data sources.

Table 3 summarizes the results. We see that Google, Cloudflare, and Meta deploy IP anycast for their time services. For each network prefix, we report the maximum number of anycast sites as reported in March 2025 by the Manycast2 data [81], which we show in Table 3. For instance, we see that Google has 41 sites, far more than IP addresses. Note that these are lower bounds, as measuring anycast deployments requires a large number of vantage points and depends on BGP routing. For Meta, the small number of sites is likely accurate, but for larger numbers, the anycast census reports significantly smaller numbers, meaning that Google and Cloudflare might have multiple hundreds of sites [26]. Statistics about Google's network [25] suggest actual numbers may be up to 200 sites, while Cloudflare claims to have 330 locations [16].

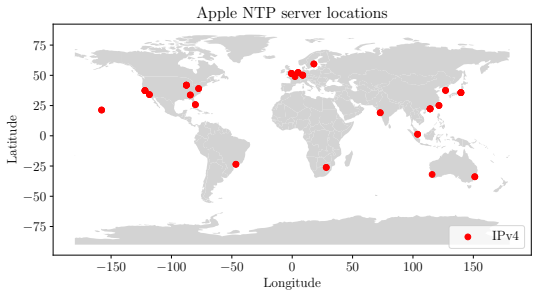
Unicast services: The remaining providers (Microsoft, Apple, AWS, and Ubuntu) all use IP unicast services. Among these, Ubuntu uses only 4 IP addresses for the entire client population, whereas Microsoft and Apple, with more than 1B each, have 12 and 53, respectively.

Given Apple and Microsoft serve billions of devices, we expect them to have locations globally worldwide. We show the geographical location of their services and AWS in Figure 3 and for Ubuntu in Appendix B. We see Microsoft has no servers in the southern hemisphere, and Microsoft is the only provider which does support IPv6.

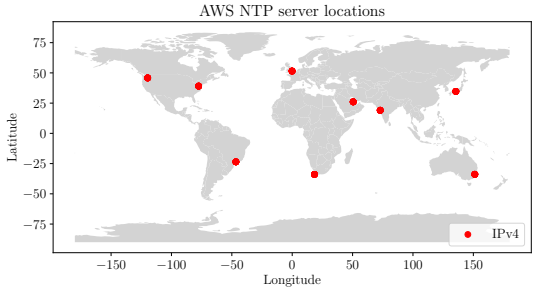
Verifying unicast providers size: Table 2 to determine the number of servers for the unicast providers (Microsoft, Apple, and AWS), we carry out additional measurements from a larger set of VPs: 9k Atlas Probes from 170 countries and 3k ASes for each time provider. Table 4 summarizes the results. We send 640k DNS queries per provider. After data sanitation (excluding IP addresses that do not belong to the time service provider's own Autonomous System (AS) and likely are received due to interception of client DNS queries [57, 62]), we identify that Microsoft, Apple, and AWS return 12, 53, and 90 IPv4 addresses, Appendix C shows how often this happens for each NTP server.



(a) Microsoft. Dataset: Microsoft-map [73]



(b) Apple. Dataset: Apple-map1 [73]



(c) AWS. Dataset: AWS-map1 [73]

Figure 3: Unicast NTP servers geolocation.

3.3 Load Balancers

We analyze if time service providers also deploy local load balancers and multiple servers in the same location, as the Root DNS does (s1–s3 in Figure 2). The motivation for this investigation is the fact the Microsoft has only 12 IP addresses to serve its billions of clients – which we would think they would consider another layer of replication.

There are no known public reports of NTP services using load balancers—the NTP pool community also agrees on it being *bad practice* [35]. The reason is that it is difficult to guarantee that a client will always be forwarded to the same

	Microsoft	Apple	AWS
Domain	twc. traffic-manager.net	time.g. aaplimg.com	time.aws.com
VPs	9,010	8,994	8,939
Countries	170	170	171
ASes	3,094	3,094	3,081
DNS Queries	642,617	642,042	640,253
NTP Servers	12	53	90
ASes	1 (AS8075)	2 (AS6185 & AS714)	2 (AS14618 & AS16509)
Dataset	Microsoft-map	Apple-map-1	AWS-map
Auth server	13.107.222.240	17.253.206.8	205.251.193.84

Table 4: Atlas measurements. Datasets: [73]

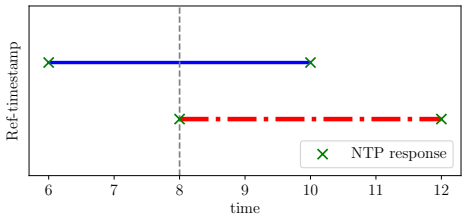


Figure 4: Load balancer detection toy example.

back-end server, which may compromise the client abilitity to evaluate the server's accuracy.

Key idea: We use the reference timestamp in the NTP header as a *transient identifier* to identify a particular NTP server at a time t. The reference timestamp field describes the "time when the system clock was last set or corrected" [50]. Our assumption is that servers behind a common load balancer would have differing reference timestamps, which allows us to identify them individually.

Figure 4 shows a toy example with four NTP responses from the same IP address (green crosses). Response times (transmit timestamp) are on the x-axis. The reference timestamps of each response are shown on the y-axis.

For each response, we extract its transmit time [50] ($T3$ in Figure 1), which is the server time when the response was sent, and show it on the x axis, and reference time for the y-axis. For each reference timestamp t_r we see, if the first time we see it is t_1 and the last time is t_2, then all times t with $t_1 \leq t \leq t_2$ should have reference time t_r. This is indicated by the interpolated lines in Figure 4. If we find that two lines overlap at any x axis values (client's query time), we can conclude that the responses cannot stem from the same server, and therefore there must be more than one server behind the IP address.

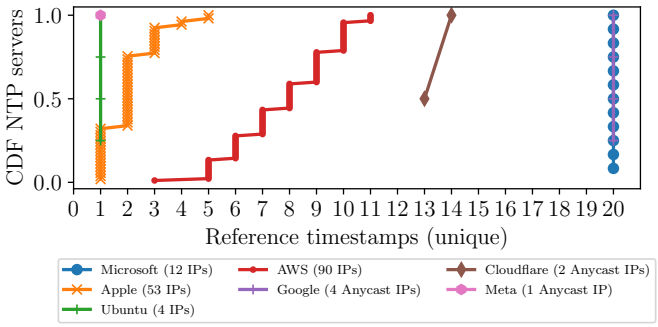


Figure 5: Number of different reference timestamps seen per server, out of 20 queries. Measured from a single VP (IPv4).

Limitations: Our methodology only works if (i) NTP servers "reuse" reference timestamps – if they send the same reference timestamps for a period at least twice as long our probing interval, (ii) servers behind a load balancer do not have identical reference timestamps and (iii) the local load balancer does not tamper with the reference timestamps. In case of anycast servers, the routes also need to remain unchanged during the measurement period.

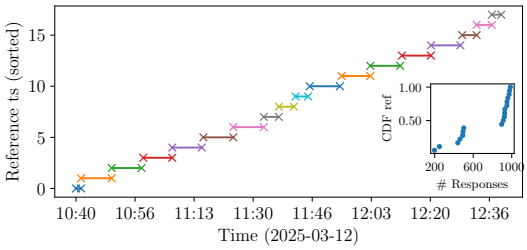
Filtering: We first filter for IP addresses that return the same reference timestamp for multiple NTP queries. To this end, we configure a single VP in Europe to send 20 queries to each IP address in Table 2, at an interval of 200 ms. We deem this interval short enough to capture frequently changing timestamps and long enough to avoid stressing servers. Each provider is measured for less than 10 s.

For each time provider, we then compute the number of unique reference timestamps observed[2]. Figure 5 shows the results. Microsoft and Google return 20 different reference timestamps for our 20 queries, and therefore does not meet our conditions for load-balancing detection. Ubuntu and Meta both return a single reference timestamp. For the other providers, we see some reuse of reference timestamps. Apple shows some degrees of reuse. AWS does as well, but to a lesser extent.[3]

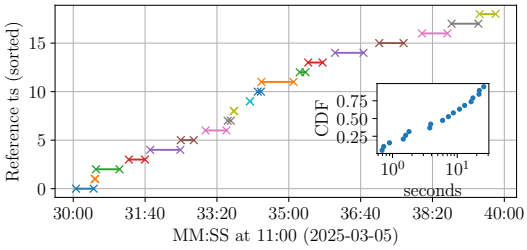
Ubuntu: As Ubuntu uses only four unicast IPv4 and three unicast IPv6 addresses, they seem a likely candidate to employ load balancing.

[2]For anycast-based provider, we can only reach a single site with a single VP—to reach more or all sites, we would need to run a setup with a large number of globally distributed VPs that can query often. Atlas VPs are not designed for such high-frequency queries. Assessment of all anycast sites of the providers hence remains future work.

[3]We found that one of the 53 IP addresses from Apple always returned a very old reference timestamp (243 days), which we reported to Apple, who then fixed it. See dataset apple-wrong-refts in [73]



(a) Ubuntu. Dataset: ubuntu-local-server in [73]



(b) Apple. Dataset: apple-local-servers in [73]

Figure 6: Reference Timestamps duration.

We therefore choose one server from Ubuntu and Apple to test if they deploy load balancers. We configure 111 Atlas anchors as vantage points. Anchors are more robust Atlas hardware or VMs. We send one NTP query per minute to one IP addresss of Ubuntu, for a period of two hours (dataset ubuntu-local-server in [73]). In total, we receive 13,220 NTP responses. We extract all 18 unique reference timestamps and reconstruct the period when they are actively used by interpolating the first and last transmission timestamp associated with them. We show the result in Figure 6a. Each reference timestamp is depicted on the y-axis, a line connects the first and last transmission. We see that each reference timestamp was used in 200-1000 responses. However, we find no overlapping timestamps for Ubuntu, so we can conclude that they have no load balancers in place. The same can be said for Apple (Figure 6b), which we also measured in the same way and found no evidence of load balancer for this particular server.

3.4 Discussion

Our results show a large diversity in the ways time providers replicate their services, which we summarize in Table 5. The least replicated service was Ubuntu's at just 4 sites globally. The most replicated at DNS level is AWS with 90 unicast IP addresses, at anycast level it is Cloudflare and Google with hundreds of sites, likely. Among the vendors with billions of clients, Google relies more on anycast replication, while Microsoft and Apple opt for unicast combined with DNS replication, similarly to how the NTP Pool operates. These are different design choices to distribute traffic,.

Provider	DNS Replication?	Anycast?	Load balancer?
Microsoft	Yes	No	–
Apple	Yes	No	No*
Google	Yes	Yes	–
Ubuntu	Yes	No	No*
Cloudflare	Yes	Yes	–
Meta	Yes	Yes	–
AWS	Yes	No	–

Table 5: Server replication for time providers

For an operator, a service replicated only with DNS allows for more flexible load distribution management (§4.3). With anycast, an operator has less control on how to shift traffic among sites: it may use BPG communities or route prepend in the hope to shift traffic [57, 78], but route decisions are ultimately placed on the client's network side. Which type of replication to use is the operator's choice to make.

4 Client/Server Mapping

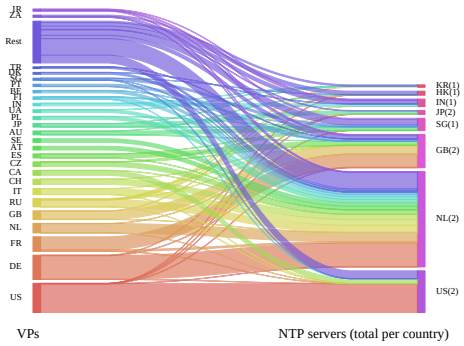
4.1 Mapping Criteria

Next we set out to determine how each time service maps clients to their available servers. As a comparison, the NTP Pool maps based on their geolocation: clients are mapped either to servers in the same country, or the same continent in case of no servers being run in the country [56]. (The pool operators state they employ this criteria to avoid risks of packet loss and issues that asymmetric routing can cause [5]).

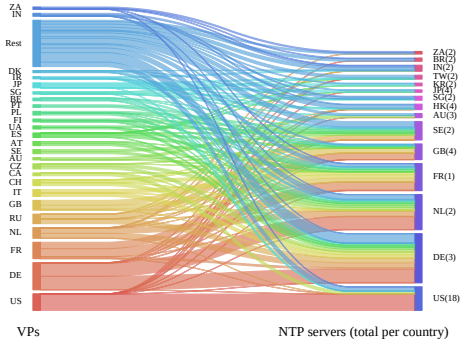
We investigate the mapping criteria used by Microsoft, Apple, and AWS to map their clients to their NTP servers (Figure 3). We use the Atlas measurements from Table 4. For each Atlas VP, we compare its geographical location (from probe metadata [15]) to the geolocation of their designated NTP servers. (We do not evaluate Anycast providers given the mapping is done by BGP; and Ubuntu, given it returns all four NTP server addresses to all clients).

Figure 7 shows the results. The left side of the Sankey diagrams contains the countries of the Atlas VPs (which represent real clients) and the right side shows the NTP servers' countries. Analyzing these figures, we can see that these providers seem to use the client's geolocation to assign them to specific NTP servers.

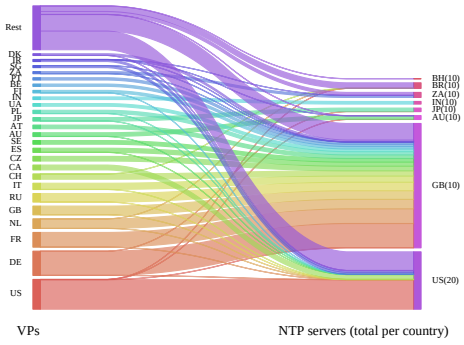
In-country mappings: For clients located in countries where the time providers operate a server, we see that, just like the NTP Pool, they are mapped to in-country servers. For example, Microsoft maps 1,297 out of 1,313 US-based VPs (98.7%) to US-based servers, AWS mapping 1,305 out of 1,311 (99.5%). Apple, in turn, seems to do the same for the US-based VPs, but not for Europe: European VPs are distributed among servers



(a) Microsoft. Dataset: Microsoft-map [73]



(b) Apple. Dataset: Apple-map-1 [73]



(c) AWS. Dataset: AWS-map-1 [73]

Figure 7: Client-server mapping: sankey diagrams of Atlas Probe VPs and NTP Servers. We show only countries with more than 70 VPs, the remaining grouped as *Rest*

in multiple countries in the region – for instances, we see 1,036 German VPs being mapped to both Germany (DE) and the Netherlands (NL). Apple seems to group all of Western Europe together in these mappings (Figure 7b). Considering that the area of Western Europe is 1/8 of the continental US area, this seems reasonable geographically.

Out-of-country mappings: For clients without NTP servers in their country, we see a more diverse mapping. Microsoft behaves similar to the NTP Pool, mapping clients to their continent in case of no in-country servers. We see that German VPs are mapped mostly to servers in the Netherlands. Russia, which spans Europe and Asia, is mapped to both continents. As we showed in Figure 3a, Microsoft has no servers in South America and Africa. Most of African VPs are mapped to Europe, while most of South American VPs are mapped to the US servers. It is worth noting that Microsoft time services domains point to a domain name associated with Azure's DNS-based traffic distribution (`twc.trafficmanager.net`), which supports geolocation-based mapping [47].

Multi-continent mappings: Apple maps clients from EU countries with no NTP servers to mostly EU countries (ES VPs are mapped to FR, GB, and DE). It has NTP servers in Africa (South Africa), but maps African VPs to virtually all continents: South African VPs are mapped both to South Africa (91), Brazil (53), and Asia (38). South American VPs are mostly mapped to both Brazil (same continent) and North America. Iranian VPs (77), are mapped to both India and Sweden – so two different continents. This contrasts with the NTP Pool, which map all clients from a specific country to the same single continent, in case there are no server in the client's country.

For AWS, most of EU-country probes are mapped to servers in Great Britain (GB) – the only European NTP server location for AWS (Figure 3c), despite AWS having presence in other EU countries. South American VPs are mapped to Brazil's location (23 from Argentina, 59 from Chile), and African VPS are mapped to South Africa's servers. AWS differs from the NTP Pool in regards to Asia (and the Middle East): despite having server in Bahrain, India, and Japan, it maps many Asian countries to North America, including Singapore, Indonesia, Iran and China. This also differs from the NTP Pool's mapping.

Takeway: Amazon and Apple, the unicast-based service providers with dozens of sites, use the client's geographical location to map them to "nearby" servers, similar to the NTP Pool. We see some improvements that differ from the NTP Pool (handling Europe as a continent instead of individual countries and mapping clients to multiple continents). However, it is interesting to observe that both the NTP Pool and the large commercial providers in essence perform a similar mapping: the client's geographical location is the primary criteria.

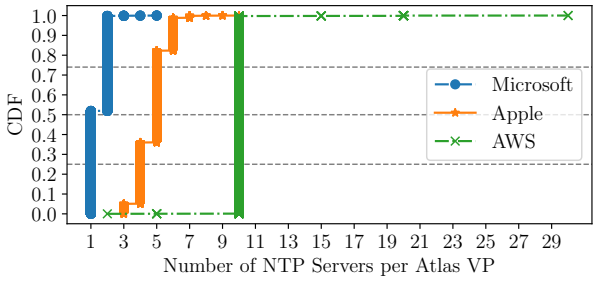


Figure 8: Number of NTP servers Atlas VPs are served (measurement from Table 4).

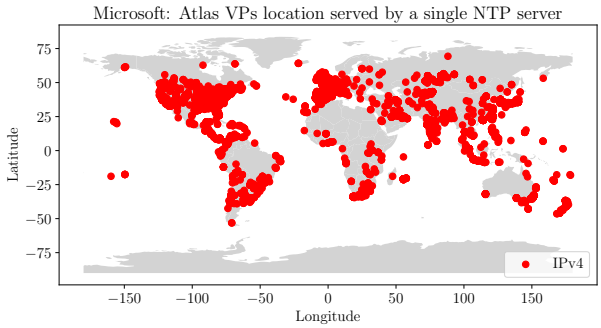


Figure 9: Geolocation of Atlas VPs served only a single time server from Microsoft's time services.

4.2 Number of servers per client

BCP223/RFC8633 (§7 in [71]) discusses the NTP best current practices and expressly states that each client should use multiple time servers. We evaluate whether providers serve all clients with more than one time server.

All anycast providers return multiple IP addresses (Table 2) to each client, and so does Ubuntu (unicast). Therefore, only Microsoft, AWS, and Apple are left to be assessed, which have from 12 to 90 NTP server addresses (IPv4).

We analyze the datasets from Table 4, which we drew from more than 9k Atlas VPs, to determine how many servers their clients are served, from all the available servers. For each VP, we compute a list of all NTP server's IP addresses it has been served in the DNS response (each probe sent 72 queries over a 12-hour period, one each 10 minutes, bypassing caching resolvers by querying the authoritative servers directly).

Figure 8 shows the results for these providers. AWS serves each client with 10 addresses, whereas Apple serves most with five addresses.

Microsoft, however, serves half of our VPs with a single time source, violating RFC8633. This reduces reliability, given those clients will rely on a single source of time. Considering that Windows has 1.4B active installations, the possibility of so many installations depending on just one source is quite alarming.

To determine if these VPs are bound to specific regions, we show in Figure 9 the geolocation of VPs served by a single IP address. We see that they are distributed all over the world. We notified Microsoft on 2025-05-02 about the issue and have not heard back.

4.3 Load balancing clients

Given the large levels of replication of the providers, how do they distribute clients among their available servers? Anycast providers leave this decision to BGP, whereas unicast providers rely on the DNS to do it. We next evaluate the unicast providers to analyze how they load balance clients.

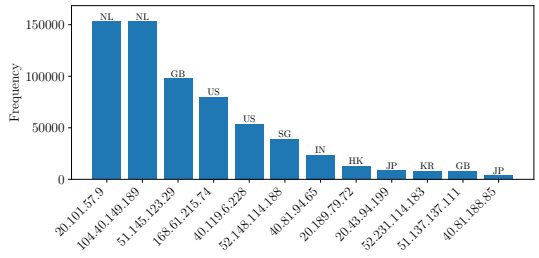
For each provider, we compute the number of times each server was included in a DNS response, for the datasets of Table 4, which we show in Figure 10.

Microsoft: Figure 10a shows the client distribution for Microsoft's servers. We show the NTP server's geolocation above each bar. We see that the first two servers are returned in the same frequency – both located in the Netherlands. Even if Microsoft has two other servers in Europe (in Great Britain), the Netherlands-based ones are returned more often – German probes could also have been mapped to GB NTP servers, for instance. With regard to the US, we see that one server is returned more often then the other. In conclusion, the client distribution per server demonstrate that Microsoft prioritize some servers in relation to others, using DNS for load balancing, at least from our VPs set.

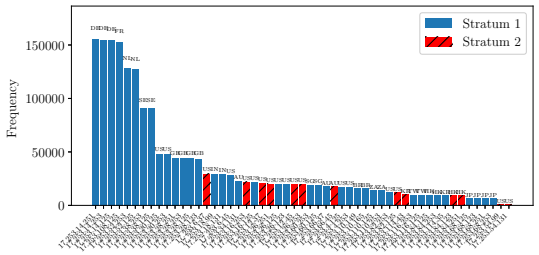
Apple: Apple operates 53 unicast NTP servers. We show the distribution of all of them in Figure 10b (we include figures for the other Apple domain in Appendix B). First, we see that there is a clear pattern in load distribution. Given that most US-based servers are assigned to US VPs (Figure 7b), we analyze how often the servers are returned to these VPs. We see a clear pattern in the frequency of NTP servers: some appear 4× more often than others (48k vs 12k). We see a similar ratio for EU-based servers (155k vs 43k, figure in Appendix B). We look at the stratum of these servers and see that 4 US-based servers are stratum 2 (shown in red in Figure 10b), and they are among the ones returned less often.

AWS: AWS has the most clear-cut distribution (Figure 10c). We see a step-like graph, where each level has 10 severs (we show country codes of only 5 for each group, for readability), and they are returned at same rates. So they use the DNS to redirect clients to specific locations, but clients are served by all NTP servers in that location. Interestingly, despite AWS having two US locations (Virginia and Oregon), the servers in Virginia appear 90× more often than Oregon-based ones.

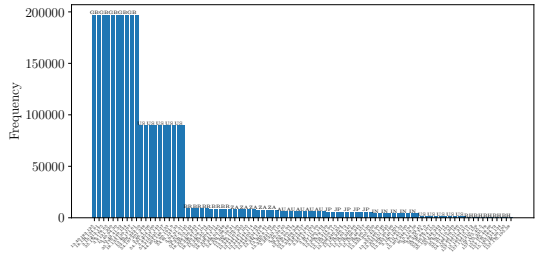
Takeaway: Load balancing varies among providers, depending on their server locations and number of servers. For



(a) Microsoft



(b) Apple (`time.g.aaplimg.com`)



(c) AWS

Figure 10: Bar Charts of number of responses for each NTP server for each time provider.

our 9k VP set, we see preference towards some servers in relation to others. Anycast-based providers have to rely on BGP to load balance traffic for them.

5 Accuracy and NTP features

Now that we understand the architecture of these time services, we turn to the question of the quality of their service: *How accurate are they*?

Whereas previous sections we relied on DNS measurements, in this section we carry out NTP measurements from two VPs synchronized with reference sources (§5.1). We compare the responses to our reference clock to determine their accuracy (§5.2), detect cases of out-of-sync servers (§5.3) and explore the characteristics of services they provide (§5.4).

VP	ASN	Time Source	Method
SE-AWS	16509	GNSS, atomic [2]	PHC [2, 70]
NL-SIDN	1140	GNSS, radio, atomic [80]	Linux PTP [70]

Table 6: Accuracy Experiment Vantage Points

5.1 Experiment setup

To evaluate the accuracy of time services providers, we have to use VPs which are directly connected to reference sources, having "ground truth" time information. Therefore, we cannot rely on Atlas probes as vantage points, given they have various different methods of updating their clocks [28] but are not directly connected to reference sources.

Therefore, we set up our own VPs, as shown in Table 6, which are connected directly to reference sources, and run Ubuntu Linux and `chrony` [19] as NTP client on both, which is a state-of-the-art NTP implementation [36].

VP locations: Both VPs rely on sources that employ both GNSS and atomic clocks. The first VP is located at AWS Sweden (SE-AWS), and leverages the AWS time service [2]. To prevent bias from the AWS-based location (given AWS is also a time provider), we set up a second VP (NL-SIDN) in the Netherlands.

Time sources: We connect NL-SIDN to a Multi Reference Source (MRS [33]), which takes time input from multiple time sources, in our case GNSS (GPS and Galileo), Radio (DCF77 [11]), and a rubidium atomic oscillator. Both VPs are synchronized using the Precision Time Protocol (PTP), which is typically at least three orders of magnitude more accurate than NTP [31]. For SE-AWS, we configure `chrony` to use the PTP Hardware Clock (PHC) [70], whereas for NL-SIDN we configure it with Linux PTP [22].

Measurements: We use a custom python script to send one NTPv4 query (mode 3, client) to each IP address associated with the time services (Table 2), every 5 minutes for 24h (May 7th 2025 for SE-AWS VP and October 29th 2025 for NL-SIDN VP). In total, each VP sends 288 NTP queries per IP address over the 24h period. We capture the traffic using `tcpdump` and analyze the traces to compute the offset between our VP clocks and the provider's clocks. We will publicly release the measurement data upon acceptance.

Limitations: Our experiment has two limitations. First, our VPs can reach only a single site (location) of anycast providers (Google, Cloudflare, and Meta) , assuming routing does not change during the measurements [86]. Therefore, we can only partially evaluate these services' accuracy. (For non-anycast services, we measure all locations by measuring all IP addresses).

Second, while not an inherent limitation of our experiments, NTP assumes symmetric network paths [50]. However, most Internet paths are *asymmetric* [85], often exhibiting different one-way delays ($T_2 - T_1 \neq T_4 - T_3$ in Figure 1). Since the NTP clock offset computation (Figure 1) relies on an equal-delay assumption by averaging the timestamps, persistent delay asymmetries result in systematic estimation errors in the reported offsets [54, 63]. We reduce the impact of this by using two vantage points and computing the maximal error in §5.3.

5.2 Accuracy results

Table 7 summarizes our results for the vantage point SE-AWS. We include the same details for NL-SIDN in Appendix C and discuss the differences in this section.

We evaluate *accuracy* by measuring the clock offset between our VPs and the time service providers (Figure 1). An offset of zero indicates perfect synchronization between the providers' clocks and our own ("ground truth", to within the uncertainty introduced by propagation delays).

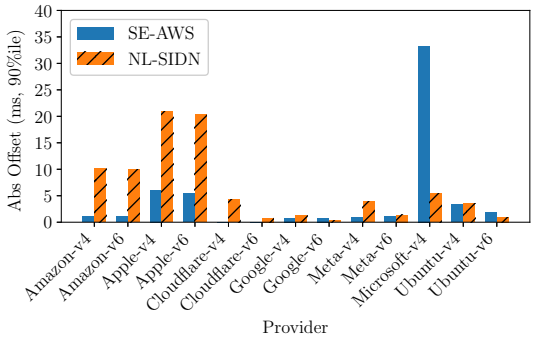
In Figure 11a, we show 90th-percentile absolute offset for each provider and both VPs (using absolute values prevents positive and negative values from evening each other out). We see that from our SE-AWS VP, all offsets' 90%iles are under 21ms, which can be considered good for most applications, except for Microsoft (33.5ms). (We include a figure of the mean offsets in Appendix B, which shows the same pattern).

Accuracy per VP: Comparing VPs, we see that SE-AWS consistently has offsets closer to 0 than NL-SIDN (except for Ubuntu over IPv6), despite having mostly larger average RTTs for the non-anycast providers (Figure 11b). This might be due to better network conditions at the SE-AWS vantage point, especially for the anycasted providers.

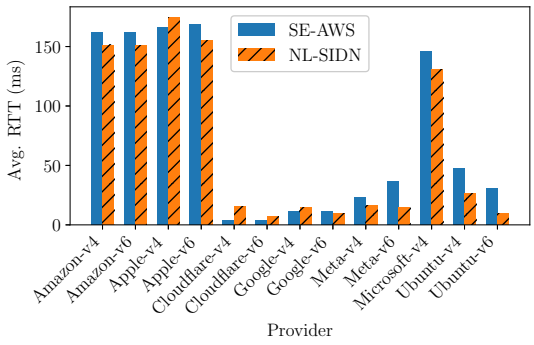
Anycast: We see for both VPs how anycast providers (Google, Cloudflare and Meta) have a lower RTT, which is due to us only reaching a single site, whereas we measured all locations, globally distributed, for the others (§3.2).

Offsets distribution per provider: Figure 12 shows the offset distribution for each VP. We see first a clear difference between the same providers for each VP – NL-SIDN VP has a more dispersed offset for the same providers – including anycast providers, even though it has a lower RTT to many providers.

Microsoft: We zoom in on Microsoft's servers for both VPs (Figure 12c). We see that for most servers, the NL-SIDN VP has a distribution that is more compact and close to 0, except for two servers (one in JP and other in HK). For our first vantage point (SE-AWS), Microsoft's time servers had predominantly negative offsets, meaning that the server's clock is lagging behind our reference clock.



(a) 90th-percentile absolute offset



(b) Average RTT

Figure 11: Accuracy and RTT results per provider.

The significant differences for Microsoft are likely due to the different measurement times, while the large offsets for two of the servers might be due to large variances in network delays to those specific servers in Asia.

RTT and accuracy: Mathematically, lower RTT improves worst-case accuracy of the offset (Appendix E). This is also the reason behind geo-based mappings in the NTP Pool. Our results, however, show that it depends on the VP. SE-AWS had a weak Pearson correlation between absolute offset and RTT (0.38), whereas NL-SIDN had a strong one (0.75). Therefore, we conclude that this assumption does not always hold, and the VP location also plays a role. This makes intuitive sense: inaccuracies are not caused by RTT, but by asymmetric network latency. We see that despite having 89 IPv4 addresses, 90%ile offset for AWS is under 2ms for SE-AWS (likely due to bias given both VP and provider are the same company), but not for NL-SIDN: it delivered the worst offsets (??).

Stratum and accuracy: From SE-AWS, Cloudflare's stratum 3 servers provide more accurate results than Google's stratum 1 (Table 7, Figure 12). This shows that a lower stratum does not always lead to more accurate offsets.

Provider	IPv	#IP Addr.	RPKI	DNS-SEC	#Req.	#Resp.	No Resp. #	No Resp. %	Offset mean	Abs. Offset 90th %ile	RTT mean	NTP version	NTS	Stratum
Amazon	v4	89	Yes	No	25 632	25 608	24	0.09%	0.05 ms	1.03 ms	162.05 ms	v4	No	4
	v6	89	Yes	No	25 633	25 613	20	0.08%	0.04 ms	1.04 ms	162.08 ms	v4	No	4
Apple	v4	51	No	No	14 688	14 058	630	4.29%	0.50 ms	6.12 ms	166.54 ms	v4	No	1/2 (78%/22%)
	v6	46	No	No	13 248	12 253	995	7.51%	0.80 ms	5.44 ms	168.75 ms	v4	No	1/2 (77%/23%)
Cloudflare	v4	2	Yes	Yes	576	576	0	0.00%	-0.01 ms	0.06 ms	3.97 ms	v4	Yes	3
	v6	2	Yes	Yes	576	576	0	0.00%	0.01 ms	0.05 ms	3.95 ms	v4	Yes	3
Google	v4	4	Yes	No	1 152	1 152	0	0.00%	0.02 ms	0.79 ms	11.26 ms	v4	No	1
	v6	4	Yes	No	1 152	1 152	0	0.00%	-0.02 ms	0.79 ms	11.12 ms	v4	No	1
Meta	v4	5	Yes	No	1 440	1 440	0	0.00%	0.04 ms	0.90 ms	23.02 ms	v4	No	1
	v6	5	Yes	No	1 440	1 440	0	0.00%	-0.04 ms	1.07 ms	36.24 ms	v4	No	1
Microsoft	v4	12	Yes	No	3 456	3 450	6	0.17%	-13.75 ms	33.25 ms	145.65 ms	v3	No	3
Ubuntu	v4	4	No	No	1 152	1 151	1	0.09%	0.34 ms	3.30 ms	47.45 ms	v4	Yes	2
	v6	3	No	No	864	864	0	0.00%	1.38 ms	1.85 ms	31.01 ms	v4	Yes	2

Table 7: SE-AWS measurement details. Anycasted services are highlighted in orange.

5.3 Detecting out-of-sync servers

Given that our VPs maintain ground-truth time, the offsets we observe are errors. Such errors arise for two reasons: variable network delay or server desynchronization on the provider's side. Because the measured offset combines the server's true offset with network-delay-induced error, an offset $\neq 0$ does not necessarily indicate that the server's clock is incorrect. Moreover, without prior clock synchronization, it is generally not possible to distinguish between delay effects and an actual clock error.

However, the contribution of network delay to the offset is bounded by the RTT. The offset must fall within $\pm RTT/2$ [51] (Appendix E). If the measured offset falls within this delay-dependent bound, then network latency is sufficient to explain the deviation, and the server's clock may still be correct. However, offsets that exceed this bound cannot be explained by latency alone and therefore imply a genuine clock error.

In our measurements, almost all offsets fall within the delay-derived bound, indicating that network latency dominates in the measurement. However, 199 responses from Microsoft servers violate this, meaning the server's clocks are out-of-sync. Figure 13 shows the offsets received during our measurement. We provide further statistics about the three affected servers in Appendix C (all of which are in Western Europe). The colored area marks the offset bounds. Square points are offsets that fall outside the bounds, indicating that the server's clock is incorrect (the offset cannot be explained by latency alone). The shape of the curve suggests that these servers have synchronized to wrong offsets, and slowly approach 0 again.

5.4 Time services profiling

Next we determine other characteristics of the NTP services providers based on the responses we collected.

NTP version and IPv6 support: As shown in Table 7, all services deploy NTPv4 – except for Microsoft, which deploys NPTv3, a standard from 1992 [48]. Moreover, Microsoft is the only provider which does not support IPv6.

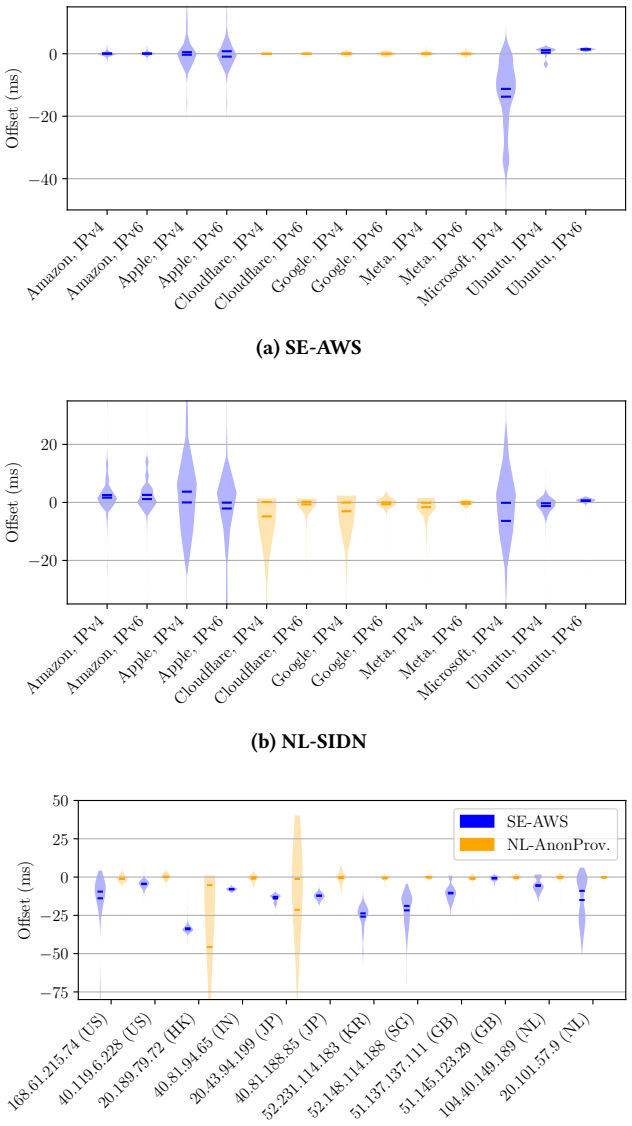
DNSSEC and RPKI support: Domain Name System Security Extensions (DNSSEC [4]) adds cryptographic signatures to DNS data to ensure its authenticity and integrity, protecting against spoofing and cache poisoning, which can protect NTP clients and others from man-in-the-middle attacks. It is used by DNS resolvers. Resource Public Key Infrastructure (RPKI [40]) provides cryptographic signatures that determine if an IP prefix can be be announced by any given AS, which help to prevent BGP hijacks from false route announcements. Table 7 shows Apple and Ubuntu do not use RPKI, while DNSSEC is only used by Cloudflare.

NTS support: As of the writing of this paper, only Cloudflare and Ubuntu support NTS – Ubuntu since Oct. 2025 [44].

Stratum: The time service providers use a variety of strata (Table 7), from 1 to 4. Only Google, Meta and Apple (80% of servers) are stratum 1, although we could not measure all anycast sites of the first two. The remaining are all stratum 2 to 4.

Reference IDs: NTP responses including the reference ID field, describing their source of time. We classify the reference IDs obtained and show them in Figure 20.

We find that 5 of the 7 providers return reference ID values that do not allow the users to figure out the upstream source. Google, for instance, returns "GOOG", Meta returns Airport codes (LHR6), while Cloudflare, AWS, Microsoft return private IP addresses. Only Apple and Ubuntu return references that provide information about their source. However, Meta has published information online about their time services, including the time source: GNSS [66]. Apple's stratum 1 servers all return descriptive strings, the majority returning `GPSs` (for: GPS via Shared Memory), and few returning



(a) SE-AWS

(b) NL-SIDN

(c) Microsoft servers on both VPs

Figure 12: Violin plots of offset distributions. Positive offset means that the polled server's clock is ahead of our reference clock.

SHM or MRS (for Shared Memory and likely Multi-reference Source), the two of which are generic technologies and do not describe a specific time source [33]. Apple's stratum 2 servers return IP addresses of Apple stratum 1 servers.

Ubuntu, in turn, provides the IP addresses of their upstream sources. One of Ubuntu's servers is synchronized with one of Apple's stratum 1 servers, thus creating a dependency between both services. Another Ubuntu server synchronized with NIST's server at the time.

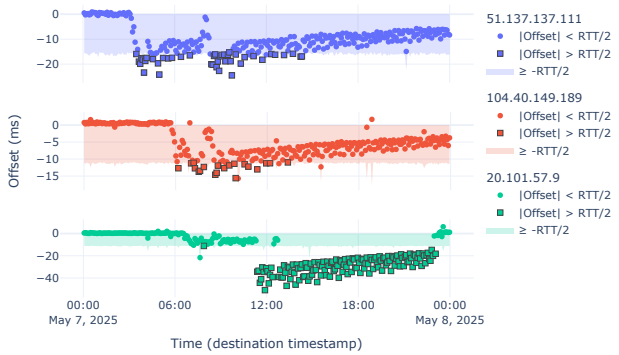


Figure 13: Offsets of out-of-sync Microsoft servers from SE-AWS. Points with |Offset| > RTT/2 indicate the server is out-of-sync.

Type	String (Stratum 1)				Address (Stratum 2)	
Kind / Organization	Non-descript ID	GPSs	SHM	MRS	Private/v6	Global
Amazon	0.0%	0.0%	0.0%	0.0%	**100.0%**	0.0%
Apple	0.0%	**69.2%**	**4.4%**	**3.6%**	**1.2%**	**21.6%**
Cloudflare	0.0%	0.0%	0.0%	0.0%	**100.0%**	0.0%
Google	**100.0%**	0.0%	0.0%	0.0%	0.0%	0.0%
Meta	**100.0%**	0.0%	0.0%	0.0%	0.0%	0.0%
Microsoft	0.0%	0.0%	0.0%	0.0%	**100.0%**	0.0%
Ubuntu	0.0%	0.0%	0.0%	0.0%	0.0%	**100.0%**
Total (7 providers)	**28.6%**	**9.9%**	**0.6%**	**0.5%**	**43.0%**	**17.4%**

Table 8: Reported reference IDs by each provider

Server software fingerprinting: We attempt to fingerprint the software used by each provider by analyzing the NTP version field returned in the responses. First, we run the NTP server software shown in Table 9 varying the version field in the NTP request, and we observe the responses in order to obtain fingerprints of common server software.

Then, we send NTP queries to each service provide varying the NTP version field. Table 10 shows the results. Our results show that both Apple (in part) and Google may be using ntpd or openntpd. Amazon, Cloudflare, Meta and Ubuntu use software that behaves similar to Chrony and ntpsec, and there are public reports of Meta using Chrony [66]. We could not find matches for Microsoft and part of Apple's servers.

6 Discussion and recommendations

Despite providing services for billions of devices daily, so far the large time service providers have escaped the level of scrutiny that the NTP Pool has faced. We fill the void in

		NTP Version								
Daemon	Version	v0	v1	v2	v3	v4	v5	v6	v7	Q.
chronyd [19]	4.7	-	v1	v2	v3	v4	-	-	-	Response
ntpd [60]	4.2.8	-	v1	v2	v3	v4	v5	v6	v7	
ntpd-rs [69]	1.6.0	-	-	-	v3	v4	-	-	-	
ntpsec [65]	1.2.4	-	v1	v2	v3	v4	-	-	-	
openntpd [27]	6.8p1	-	v1	v2	v3	v4	v5	v6	v7	

Table 9: NTP version of queries and responses.

	NTP Version								
Provider	v0	v1	v2	v3	v4	v5	v6	v7	Query
Apple	v0	v1	v2	v3	v4	v4	v4	v4	Response
	-	v1	v2	v3	v4	v5	v6	v7	
Amazon	-	v1	v2	v3	v4	-	-	-	
Cloudflare	-	v1	v2	v3	v4	-	-	-	
Google	-	v1	v2	v3	v4	v5	v6	v7	
Meta	-	v1	v2	v3	v4	-	-	-	
Microsoft	-	v3	v3	v3	v3	-	-	-	
Ubuntu	-	v1	v2	v3	v4	-	-	-	

Table 10: NTP version responses per provider

this paper. This is more pressing given most clients never change their NTP settings, and therefore billions of devices are served by less than a dozen of time services.

Architecture: we show a large variety in deployment architecture of time service providers – the most notable difference being the use of IP anycast (3 of the 7 providers we evaluate employ anycast). The choice depends on the operator's preference – unicast servers combined with DNS replication give a more fine-grained control over load balancing clients.

Client mapping: previous studies have pointed out issues with the strict client/server mapping of the NTP Pool [56] – a client must be mapped to a server in the same country or continent. The non-anycast providers we evaluate do not exhibit these shortcomings: Apple, Microsoft, and AWS use more flexible methods to do such mappings. We show how Microsoft violates NTP best practices, serving half of our VPs with a single time server.

Accuracy: we show that all services provide good accuracy, except for Microsoft's, three of which were out-of-sync during part of our measurements. Considering that 50% of our VPs are served by a single server, these Microsoft clients would have no additional time servers available to notice incorrect offsets. This also demonstrates the need for longer measurements from more VPs with reference clocks for complete accuracy studies.

Recommendations: We recommend that all service providers implement NTS, DNSSEC, and RPKI to mitigate a broad range of attacks. At the time of writing, only Cloudflare supports NTS, DNSSEC, and RPKI in their time services. Alongside, these vendors must deploy robust client software that supports NTS by default, to prevent man-in-the-middle attacks.

We recommend Microsoft to adhere to the NTP best practices [71] by serving clients with multiple NTP servers; currently, roughly 50% of our vantage points that use Microsoft are served by a single time source.

Our measurements show that even large time providers can be out-of-sync. We therefore recommend that Microsoft and other non-stratum-1 providers deploy services that derive time from diverse reference sources, include non-GNSS inputs, and implement holdover mechanisms to maintain stability during outages.

7 Related Work

The closest study to ours investigates the NTP Pool architecture, client/server mapping [56], and accuracy from a subset of RIPE Atlas VPs. Different than this, we analyze seven providers and carry out accuracy experiments from two VPs synced with reference sources.

NTP Pool: The NTP Pool has been studied extensively [37, 56, 68, 77], but so far the same analysis has not been done for other large time service providers, despite them being the default providers for billions of devices. We cover them in this paper.

IP-wide scans: multiple have focussed on mapping NTP servers in the entire IP space [30, 52, 53, 77]. Considering the user base of the largest time providers, we argue that it is more important to focus on scrutinizing these – which include those in this paper and the NTP Pool, as previously shown [56].

Accuracy: A previous study has used GPS-synced VPs to measure accuracy of NTP servers [13]. However, they focus on different servers and mostly on accuracy.

NTP security: Protocol vulnerabilities have also been investigated [8, 41–43, 84], demonstrating how NTP clients can be susceptible to malicious time servers [21, 24] or to attacks by third parties [41], often focussing on broadcast-mode availability [41, 84]. Moreover, usage of NTP for amplification of distributed denial-of-service (DDoS) attacks [20] has been covered, as well as off-path attacks using DNS cache poisoning or BGP hijacking [34, 41].

8 Conclusion

NTP is a fundamental but often overlooked service on the Internet. Compared to the DNS, NTP services are significantly more centralized, as most devices use default (and sometimes hardcoded) configuration settings. We focussed on the time services of major OS and device vendors (including Windows, Ubuntu Linux, Android, macOS/iOS).

Our results show a large diversity in terms of architecture and services offered, and a case where three time servers from one provider (Microsoft) were out-of-sync. Our hope is that this study brings attention to a rather overlooked aspect

of the NTP ecosystem, and can motivate vendors to improve their services by deploying servers that derive time from multiple reference sources, and that support NTS by default, as well DNSSEC and RPKI.

References

[1] S. Alexander and R. Droms. 1997. *DHCP Options and BOOTP Vendor Extensions.* RFC 2132. IETF. http://tools.ietf.org/rfc/rfc2132.txt

[2] Amazon Web Services. 2024. *Precision clock and time synchronization on your EC2 instance.* https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/set-time.html Accessed: 2025-10-27.

[3] Apple Inc. 2024. Apple Reports First Quarter Results. https://www.apple.com/newsroom/2024/02/apple-reports-first-quarter-results/. Accessed: 2025-02-18.

[4] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. 2005. *DNS Security Introduction and Requirements.* RFC 4033. IETF. http://tools.ietf.org/rfc/rfc4033.txt

[5] Ask Bjørn Hansen. 2023. Minor New Features on the website. https://community.ntppool.org/t/minor-new-features-on-the-website/2947/8.

[6] bgptools. 2025. anycast-prefixes. https://github.com/bgptools/anycast-prefixes Accessed: 2025-02-20.

[7] Leo Bicknell. 2012. NTP Issues Today. https://mailman.nanog.org/pipermail/nanog/2012-November/053449.html.

[8] M. Bishop. 1990. A security analysis of the NTP protocol version 2. In *[1990] Proceedings of the Sixth Annual Computer Security Applications Conference.* 20–29. https://doi.org/10.1109/CSAC.1990.143746

[9] Russel Brandon. 2021. *Google announces that Android has over 3 billion active devices.* https://www.theverge.com/2021/5/18/22440813/android-devices-active-number-smartphones-google-2021 Accessed: 2025-02-18.

[10] Physikalisch-Technische Bundesanstalt. 2025. *Questions about Time.* https://www.ptb.de/cms/en/ptb/fachabteilungen/abt4/fb-44/fragenzurzeit/fragenzurzeit07.html Accessed: 2025-05-01.

[11] Physikalisch-Technische Bundesanstalt. n.d.. DCF77. https://www.ptb.de/cms/en/ptb/fachabteilungen/abt4/fb-44/ag-442/dissemination-of-legal-time/dcf77.html. Accessed: 2025-11-06.

[12] R. Bush and R. Austein. 2013. *The Resource Public Key Infrastructure (RPKI) to Router Protocol.* RFC 6810. IETF. http://tools.ietf.org/rfc/rfc6810.txt

[13] Yi Cao and Darryl Veitch. 2020. Toward Trusted Time: Remote Server Vetting and the Misfiring Heart of Internet Timing. *IEEE/ACM Transactions on Networking* 28, 2 (2020), 944–956. https://doi.org/10.1109/TNET.2020.2977024

[14] RIPE Network Coordination Centre. 2025. *K-Root.* https://www.ripe.net/analyse/dns/k-root/ Accessed: 2025-03-13.

[15] RIPE Network Coordination Centre. 2025. *RIPE Atlas Probe Archive.* https://ftp.ripe.net/ripe/atlas/probes/archive/ Accessed: 2025-02-12.

[16] Cloudflare. 2024. Cloudflare Time Service. https://www.cloudflare.com/time/.

[17] Internet Systems Consortium. 2025. *F-Root.* https://www.isc.org/f-root/ Accessed: 2025-03-13.

[18] C. Contavalli, W. van der Gaast, D. Lawrence, and W. Kumari. 2016. *Client Subnet in DNS Queries.* RFC 7871. IETF. http://tools.ietf.org/rfc/rfc7871.txt

[19] Richard Curnow and Miroslav Lichvar. 2024. *chrony - Network Time Protocol (NTP) Implementation.* https://chrony-project.org/ Accessed: 2025-03-04.

[20] Jakub Czyz, Michael Kallitsis, Manaf Gharaibeh, Christos Papadopoulos, Michael Bailey, and Manish Karir. 2014. Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks. In *Proceedings of the 2014 ACM Conference on Internet Measurement Conference* (Vancouver, BC, Canada) *(IMC)*. ACM, 435–448. https://doi.org/10.1145/2663716.2663717

[21] Omer Deutsch, Neta Rozen Schiff, Danny Dolev, and Michael Schapira. 2018. Preventing (Network) Time Travel with Chronos.. In *NDSS*.

[22] Linux PTP development team. 2025. *LinuxPTP – Precision Time Protocol (PTP) implementation for Linux.* https://linuxptp.sourceforge.net/ Accessed: 2025-11-05.

[23] D. Franke, D. Sibold, K. Teichel, M. Dansarie, and R. Sundblad. 2020. *Network Time Security for the Network Time Protocol.* RFC 8915. IETF. http://tools.ietf.org/rfc/rfc8915.txt

[24] Tweede Golf. 2024. Manipulating time through (S)NTP. https://tweedegolf.nl/en/blog/142/manipulating-time-through-sntp Accessed: 2025-11-18.

[25] Google. 2025. Network edge locations. https://cloud.google.com/about/locations/. [Accessed 15-05-2025].

[26] Remi Hendriks, Matthew Luckie, Mattijs Jonker, Raffaele Sommese, and Roland van Rijswijk-Deij. 2025. MAnycast Reloaded: a Tool for an Open, Fast, Responsible and Efficient Daily Anycast Census. arXiv:2503.20554 [cs.NI] https://arxiv.org/abs/2503.20554

[27] Henning Brauer. 2022. *OpenNTPd.* https://www.openntpd.org Accessed: 2025-05-13.

[28] Philip Homburg. 2015. NTP Measurements with RIPE Atlas. https://labs.ripe.net/author/philip_homburg/ntp-measurements-with-ripe-atlas/.

[29] Nate Hopper. 2022. The Thorny Problem of Keeping the Internet's Time. *The New Yorker* (Sept. 30 2022). https://www.newyorker.com/tech/annals-of-technology/the-thorny-problem-of-keeping-the-internets-time

[30] Zhentian Huang, Shuai Wang, Li Chen, Dan Li, and Yilun Liu. 2025. Measuring the Time Source Vulnerabilities in the NTP Ecosystem. In *Proceedings of the 2025 ACM Internet Measurement Conference* (USA) *(IMC '25')*. Association for Computing Machinery, New York, NY, USA, 16 pages.

[31] IEEE. 2020. IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2019 (Revision ofIEEE Std 1588-2008)* (2020), 1–499. https://doi.org/10.1109/IEEESTD.2020.9120376

[32] IPinfo.io. 2025. *IPinfo - Trusted IP Data Provider.* https://ipinfo.io Accessed: 2025-02-12.

[33] Andreja Jarc. 2017. *Multi-Reference Clock.* https://blog.meinbergglobal.com/2017/11/16/multi-reference-clock/ Accessed: 2025-11-10.

[34] Philipp Jeitner, Haya Shulman, and Michael Waidner. 2020. The Impact of DNS Insecurity on Time. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 266–277. https://doi.org/10.1109/DSN48063.2020.00043

[35] Simon Kepp. 2025. *Does Load-Balancing Make Sense for NTP?* https://community.ntppool.org/t/does-load-balancing-make-sense-for-ntp/1751 Accessed: 2025-03-13.

[36] S. Konjerla, G. C. M. Moura, G. Smaragdakis, and T. Dittrich. 2025. *Are NTP Clients Always Right? Evaluating NTP Clients under Normal and Attack Scenarios.* Technical Report SIDN Labs Technical Report 2025-10-16. SIDN Labs and TU Delft, Arnhem, The Netherlands. https://www.sidnlabs.nl/downloads/7M1bz1otGu0D7hqr0hRT2c/3bc09536c3bc87f63b80d66944abc1d9/ntp-clients-tech_report-20251016.pdf Updated October 30, 2025.

[37] Jonghoon Kwon, Jeonggyu Song, Junbeom Hur, and Adrian Perrig. 2023. Did the Shark Eat the Watchdog in the NTP Pool? Deceiving the NTP Pool's Monitoring System. In *30th USENIX Security Symposium.* https://www.usenix.org/conference/usenixsecurity23/presentation/kwon

[38] National Physical Laboratory. 2025. *Internet Time Service.* https://www.npl.co.uk/products-services/time-frequency/internet-time Accessed: 2025-05-01.

[39] leeand00. 2019. *Is it true that iOS devices use a hard coded* `time.apple.com` *DNS for an NTP server?* https://apple.stackexchange.com/questions/364765/is-it-true-that-ios-devices-use-a-hard-coded-time-apple-com-dns-for-an-ntp-serve "Asked 6 years, 3 months ago" on Ask Different (Stack Exchange).

[40] M. Lepinski and S. Kent. 2012. *An Infrastructure to Support Secure Internet Routing.* RFC 6480. IETF. http://tools.ietf.org/rfc/rfc6480.txt

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

[42] Aanchal Malhotra and Sharon Goldberg. 2016. Attacking NTP's Authenticated Broadcast Mode. *SIGCOMM Comput. Commun. Rev.* 46, 2 (may 2016), 12–17.

[43] Aanchal Malhotra, Matthew Van Gundy, Mayank Varia, Haydn Kennedy, Jonathan Gardner, and Sharon Goldberg. 2017. The Security of NTP's Datagram Protocol. In *Financial Cryptography and Data Security: 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers 21.* Springer, 405–423.

[44] Lukas Märdian. 2025. PSA: Installing chrony by default, to enable Network Time Security (NTS). Ubuntu-devel mailing list. https://lists.ubuntu.com/archives/ubuntu-devel/2025-May/043355.html Message posted on May 22, 2025.

[45] Mark Morowczynski. 2012. Did Your Active Directory Domain Time Just Jump To The Year 2000? https://techcommunity.microsoft.com/t5/core-infrastructure-and-security/did-your-active-directory-domain-time-just-jump-to-the-year-2000/ba-p/255873.

[46] Microsoft. 2022. Windows Devices by the Numbers. https://web.archive.org/web/20220419050729/https://news.microsoft.com/bythenumbers/en/windowsdevices. Accessed: 2025-02-18.

[47] Microsoft. 2025. *Traffic Manager Geographic Traffic Routing Method.* https://learn.microsoft.com/en-us/azure/traffic-manager/traffic-manager-faqs#traffic-manager-geographic-traffic-routing-method Accessed: 2025-05-02.

[48] D. Mills. 1992. *Network Time Protocol (Version 3) Specification, Implementation and Analysis.* RFC 1305. IETF. http://tools.ietf.org/rfc/rfc1305.txt

[49] D. Mills. 1996. *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI.* RFC 2030. IETF. http://tools.ietf.org/rfc/rfc2030.txt

[50] D. Mills, J. Martin, J. Burbank, and W. Kasch. 2010. *Network Time Protocol Version 4: Protocol and Algorithms Specification.* RFC 5905. IETF. http://tools.ietf.org/rfc/rfc5905.txt

[51] David L Mills. 1991. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications* 39, 10 (1991), 1482–1493. https://doi.org/10.1109/26.103043

[52] David L. Mills, Ajit Thyagarjan, and Brian C. Huffman. 1997. Internet Timekeeping Around the Globe. https://www.eecis.udel.edu/~mills/database/papers/survey5.pdf

[53] Nelson Minar. 1999. A Survey of the NTP Network. https://www.eecis.udel.edu/~mills/database/reports/ntp-survey99-minar.pdf

[54] Faten Mkacher and Andrzej Duda. 2019. Calibrating NTP. In *2019 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS).* 1–6. https://doi.org/10.1109/ISPCS.2019.8886646

[55] Giovane C. M. Moura, Sebastian Castro, Wes Hardaker, Maarten Wullink, and Cristian Hesselman. 2020. Clouding up the Internet: How Centralized is DNS Traffic Becoming?. In *Proceedings of the ACM Internet Measurement Conference* (Virtual Event, USA) *(IMC '20).* Association for Computing Machinery, New York, NY, USA, 42–49.

[56] Giovane C. M. Moura, Marco Davids, Caspar Schutijser, Cristian Hesselman, John Heidemann, and Georgios Smaragdakis. 2024. Deep Dive into NTP Pool's Popularity and Mapping. *Proceedings of the ACM on Measurement and Analysis of Computing Systems (SIGMETRICS 2024)* 8, 1, Article 15 (feb 2024), 30 pages. https://doi.org/10.1145/3639041

[57] Giovane C. M. Moura, Ricardo de O. Schmidt, John Heidemann, Wouter B. de Vries, Moritz Müller, Lan Wei, and Christian Hesselman. 2016. Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event. In *Proceedings of the ACM Internet Measurement Conference.* https://doi.org/10.1145/2987443.2987446

[58] Giovane C. M. Moura and John Heidemann. 2023. Vulnerability Disclosure Considered Stressful. *SIGCOMM Comput. Commun. Rev.* 53, 2 (July 2023), 2–10. https://doi.org/10.1145/3610381.3610383

[59] Giovane C. M. Moura, John Heidemann, Ricardo de O. Schmidt, and Wes Hardaker. 2019. Cache Me If You Can: Effects of DNS Time-to-Live. In *Proceedings of the ACM Internet Measurement Conference.* ACM, Amsterdam, the Netherlands, 101–115. https://doi.org/10.1145/3355369.3355568

[60] Network Time Foundation. 2025. *NTPD 4.2.8p-series.* https://www.ntp.org/documentation/4.2.8-series/#building-and-installing-ntp Accessed: 2025-05-13.

[61] NIST. 2025. NIST Internet Time Service (ITS). (May. 1 2025). https://www.nist.gov/pml/time-and-frequency-division/time-distribution/internet-time-service-its

[62] Yevheniya Nosyk, Qasim Lone, Yury Zhauniarovich, Carlos H. Gañán, Emile Aben, Giovane C. M. Moura, Samaneh Tajalizadehkhoob, Andrzej Duda, and Maciej Korczyński. 2023. Intercept and Inject: DNS Response Manipulation in the Wild. In *Passive and Active Measurement,* Anna Brunstrom, Marcel Flores, and Marco Fiore (Eds.). Springer Nature Switzerland, Cham, 461–478.

[63] Andrew N. Novick and Michael A. Lombardi. 2015. Practical limitations of NTP time transfer. In *2015 Joint Conference of the IEEE International Frequency Control Symposium & the European Frequency and Time Forum.* 570–574. https://doi.org/10.1109/FCS.2015.7138909

[64] NTP Pool. 2024. pool.ntp.org: the internet cluster of ntp servers. https://www.ntppool.org/en/.

[65] NTPsec project. 2022. *NTPSec.* https://www.ntpsec.org Accessed: 2025-05-13.

[66] Oleg Obleukhov. 2020. Building a more accurate time service at Facebook scale. https://engineering.fb.com/2020/03/18/production-engineering/ntp-service/.

[67] C. Partridge, T. Mendez, and W. Milliken. 1993. *Host Anycasting Service.* RFC 1546. IETF. http://tools.ietf.org/rfc/rfc1546.txt

[68] Yarin Perry, Neta Rozen-Schiff, and Michael Schapira. 2021. 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)* (Virtual Conference).

[69] Pendulum Project. [n. d.]. *ntpd-rs: A full-featured implementation of the Network Time Protocol, including NTS support.* https://github.com/pendulum-project/ntpd-rs GitHub repository.

[70] The Linux Kernel Documentation Project. 2024. PTP hardware clock infrastructure for Linux. https://docs.kernel.org/driver-api/ptp.html. Accessed: 2025-11-05.

[71] D. Reilly, H. Stenn, and D. Sibold. 2019. *Network Time Protocol Best Current Practices.* RFC 8633. IETF. http://tools.ietf.org/rfc/rfc8633.txt

[72] E. Rescorla. 2018. *The Transport Layer Security (TLS) Protocol Version 1.3.* RFC 8446. IETF. http://tools.ietf.org/rfc/rfc8446.txt

[73] RIPE NCC. 2025. RIPE Atlas Measurements. https://atlas.ripe.net/measurements/ID. where ID is the measurement ID: Google-android: 100787434, Meta: 100787995, Meta-v6: 100930063, Ubuntu: 100788409, Apple-ZA-map: 87282270, Microsoft-map: 86743242, Apple-map1: 86743255, Apple-map2: 86743251, Ubuntu-map: 87076806, AWS-map:

87077187, AWS-map-v6: 90364265, ms-server1-detection: 87119082, ubuntu-local-server: 90982800, apple-wrong-refts: 89211992, apple-local-servers: [89183871-89183874,89183992].

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

[75] RIPE Network Coordination Centre. 2025. RIPE Atlas. https://atlas.ripe.net.

[76] Root Server Operators. 2025. Root DNS. http://root-servers.org/.

[77] Teemu Rytilahti, Dennis Tatang, Janosch Köpper, and Thorsten Holz. 2018. Masters of Time: An Overview of the NTP Ecosystem. In *2018 IEEE European Symposium on Security and Privacy (EuroS P)*. 122–136. https://doi.org/10.1109/EuroSP.2018.00017

[78] Ricardo de O. Schmidt, John Heidemann, and Jan Harm Kuipers. 2017. Anycast Latency: How Many Sites Are Enough?. In *Proceedings of the Passive and Active Measurement Workshop*. Springer, Sydney, Australia, 188–200. http://www.isi.edu/%7ejohnh/PAPERS/Schmidt17a.html

[79] Amazon Web Services. 2022. *Amazon Time Sync is now available over the internet as a public NTP service.* https://aws.amazon.com/about-aws/whats-new/2022/11/amazon-time-sync-internet-public-ntp-service/ Accessed: 2025-03-11.

[80] SIDN Labs. 2025. TimeNL. https://time.nl/index_en.html.

[81] Raffaele Sommese. 2021. MAnycast² Anycast Census Data Repository. https://github.com/ut-dacs/Anycast-Census.

[82] Raffaele Sommese, Leandro Bertholdo, Gautam Akiwate, Mattijs Jonker, van Rijswijk-Deij, Roland, Alberto Dainotti, KC Claffy, and Anna Sperotto. 2020. MAnycast2—Using Anycast to Measure Anycast. In *Proceedings of the ACM Internet Measurement Conference*. ACM, Pittsburgh, PA, USA. https://doi.org/10.1145/3419394.3423646

[83] Raffaele Sommese, Leandro Bertholdo, Gautam Akiwate, Mattijs Jonker, van Rijswijk-Deij, Roland, Alberto Dainotti, KC Claffy, and Anna Sperotto. 2025. *MAnycast2 - Using Anycast to Measure Anycast.* https://manycast2.net/ Accessed: 2025-03-06.

[84] Nikhil Tripathi and Neminath Hubballi. 2021. Preventing time synchronization in NTP broadcast mode. *Computers & Security* 102 (2021), 102135. https://doi.org/10.1016/j.cose.2020.102135

[85] Kevin Vermeulen, Ege Gurmericliler, Italo Cunha, David Choffnes, and Ethan Katz-Bassett. 2022. Internet Scale Reverse Traceroute. In *Proceedings of the 22nd ACM Internet Measurement Conference* (Nice, France) *(IMC '22)*. Association for Computing Machinery, New York, NY, USA, 694–715. https://doi.org/10.1145/3517745.3561422

[86] Lan Wei and John Heidemann. 2017. Does Anycast Hang Up On You?. In *IEEE Network Traffic Monitoring and Analysis Conference*. IEEE, Dublin, Ireland, 9. https://doi.org/10.23919/TMA.2017.8002905

A Ethics and Disclosure

Our paper has two ethical concerns: minimizing the impact of our measurements and notifying vendors of inaccuracies and issues we have found. Most of our measurements uses RIPE Atlas as a measurement platform and query at low frequency (every 10min), which cannot stress the authoritative DNS servers or NTP servers that we measure – they are provisioned to answer billions of clients. For the more high frequency measurements of §3.3, we first perform a filtering process which excludes service providers from the possibility of detecting load balancers, thus preventing unnecessary measurements. We then measure Apple and Ubuntu, 1 DNS query per minute, from a subset of 111 Atlas VPs, all Atlas anchors, which are more robust.

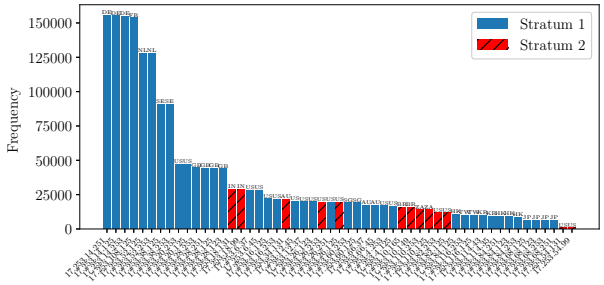


Figure 14: Bar chart of the number of responses for each NTP server for `time-osx.g.aaplimg.com`

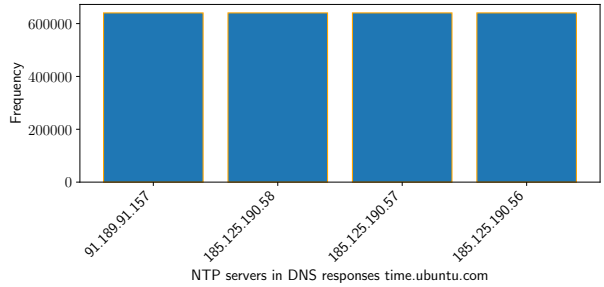


Figure 15: Bar chart of the number of responses for each NTP server for `ntp.ubuntu.com`

Secondly, we found inconsistencies in some vendors, and reported them following coordinated vulnerability disclosure guidelines [58]. We have reported to Apple that one of its time servers was sending stale reference timestamps (but precise timestamps, so no impact on clock synchronization), and they have fixed the issue. We also reported to Microsoft that their DNS servers return one 1 NTP server most clients, which violates RFC8633 [71]. We have later informed then that we will make this information publicly 45 days from the original notification. We also notified Meta that their domain names return 1 IP address only – and it would be better if it would return multiple. We have not obtained a response. We then notified them again about the public disclosure of our findings.

B Extra Graphs

Figure 14 shows the results from §4.3 for `time-osx.g.applimg.com`, Apple's second domain name.

Figure 15 shows that all of Ubuntu's 4 IPv4 addresses are included in every DNS response without any load balancing or mapping.

Figure 16 shows the geolocation of Ubuntu's IP addresses: they are in only 2 different locations.

Figure 17 shows the mappings of RIPE Atlas probes' countries to Apple NTP servers for `time-osx.g.aaplimg.com`. It

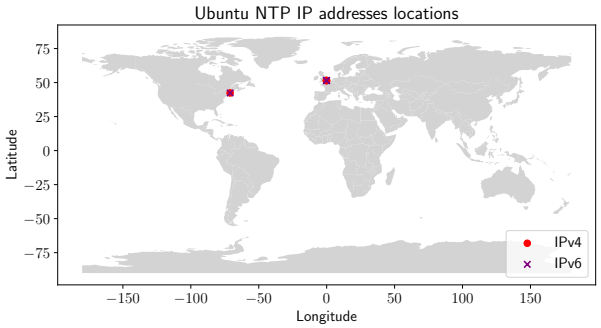


Figure 16: Geographical location of Ubuntu NTP server IP addresses

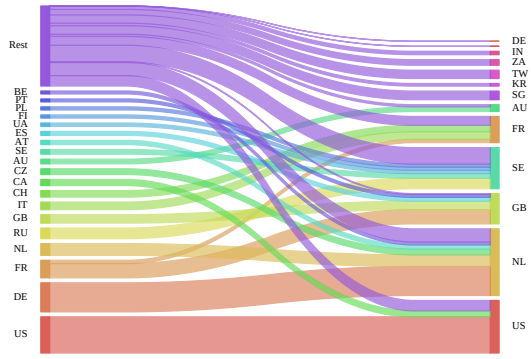


Figure 17: Sankey diagram of country of Atlas Probes VPs (left side) and Apple's IPv4 NTP servers' countries (right side) for `time-osx.g.aaplimg.com`. Dataset: Apple-map-2[73]

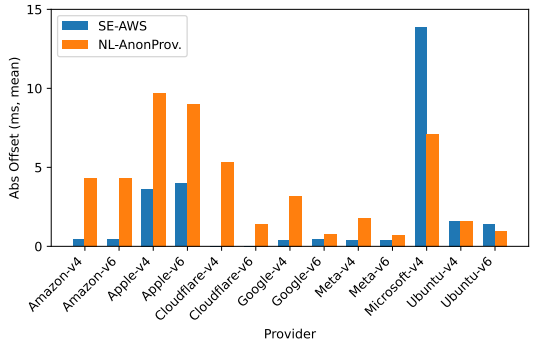
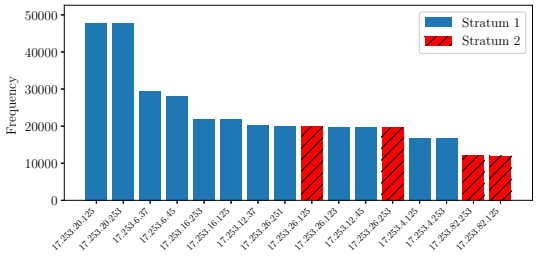
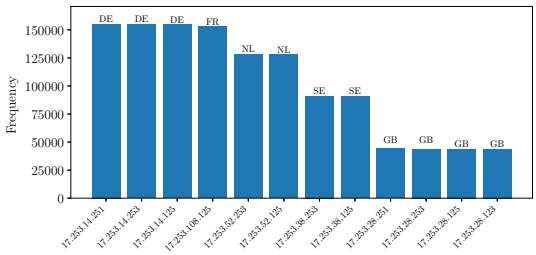


Figure 18: Average absolute offset from the NTP-measurement

follows the same mapping (and the same servers) as Figure 7b.



(a) Apple (`time.g.aaplimg.com` – US Servers)



(b) Apple (`time.g.aaplimg.com` – EU Servers)

Figure 19: Bar Charts of number of responses for each NTP server for Apple US and EU Servers

Server	104.40.149.189	20.101.57.9	51.137.137.111
# Mismatches	20	137	42
Largest Mismatch	-4.20ms	-39.08ms	-9.16ms
Mean RTT	22.31ms	22.93ms	31.57ms
Abs. Offset Q1	11.35ms	22.92ms	16.38ms
Abs. Offset Q4	13.14ms	33.74ms	18.98ms
Largest Offset	-15.61ms	-50.97ms	-24.51ms

Table 11: Statistics about out-of-sync servers (SE-AWS): Mismatch = $T_2 - T_1$ if $T_1 > T_2$ else $T_3 - T_4$ if $T_3 > T_4$ else 0, Q1 and Q4: first and last quartile

Figure 19 shows the number of responses for Apple's US and EU-based NTP servers.

C Extra Tables

Table 13 shows the accuracy experiment results for NL-SIDN.

D Apple Load Balancer Detection

D.0.1 Does Apple use local servers? Next we try one of Apple's servers, the one with more traffic in Figure 14, located in Germany. We configure 6 Atlas VPs, one per continent, to send 5 queries per minute to the Apple's NTP server. We choose Atlas anchors, which are more robust measurement VPs, and configure 5 measurements (1 per minute) to achieve

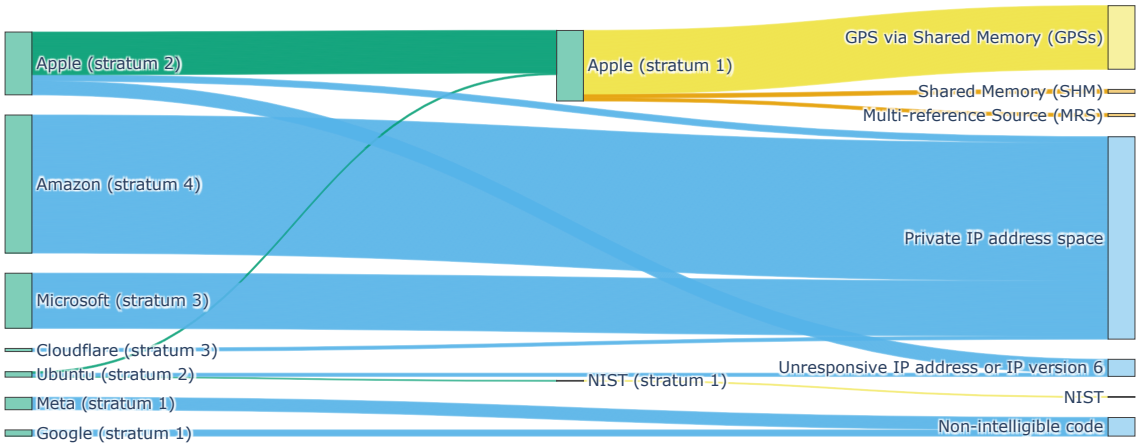


Figure 20: Sankey plot of references (time sources) used by each provider's servers (SE-AWS).

Query name	Microsoft `twc.trafficmanager.net` A	Apple1 `time.g.aaplimg.com` A	Apple2 `time-osx.g.aaplimg.com` A	AWS `time.aws.com`
Dataset	Microsoft-map	Apple-map-1	Apple-map-2	AWS-map
		Ripe Atlas		
Auth server	13.107.222.240	17.253.206.8	17.253.201.8	205.251.193.84
Total Probes	9,231	9,205	9,211	9,068
without valid response	221	211	223	129
with valid responses	9,010	8,994	8,988	8,939
only with valid responses	8,270	8,066	8,248	8,284
with valid and invalid responses	740	928	740	655
Total countries	171	171	172	172
without valid response	52	52	53	41
with valid responses	170	170	171	171
only with valid responses	167	163	164	165
with valid and invalid responses	170	170	171	171
Total ASes	3,143	3,142	3,143	3,115
without valid response	158	152	158	104
with valid responses	3,094	3,094	3,093	3,081
only with valid responses	2,865	2,874	2,921	2,905
with valid and invalid responses	487	489	394	424
DNS queries	663,637	661,834	662,165	652,105
valid response (RCODE 0)	642,617	642,042	641,436	640,253
Timeout/ network error	11,945	12,896	12,080	11,754
SERVFAIL (RCODE 2)	4,378	2,670	4337	20
REFUSED (RCODE 5)	4,697	4,210	4,312	78
malformed response	0	16	0	0
		Time Servers		
Unique NTP servers	15	57	55	93
Valid	12	53	53	90
Invalid	3	4	2	3
ASes NTP Servers	1 (AS8075)	2 (AS6185 and AS714)	2 (AS6185 and AS714)	2(AS14618 and AS16509)

Table 12: Mapping Time Service Networks. Datasets: [73]

5 queries per minute (dataset apple-local-servers in [73]), for 10 minutes.

We then combine all these datasets and extract the number of unique reference timestamps. We found 19 in total. For each of them, we extract its start and end, which we use the transmit timestamp [50] from the first and last response sent, measured at the NTP server side. We show them in Figure 6b. For each reference timestamp, we compute the difference between its last transmit time and the next reference timestamp's first transmit time – if the difference is negative, that would mean that two reference times existing at the same time, indicating more than one time server (Figure 4). Our

Provider	IPv	#IP Addr.	RPKI	DNS-SEC	#Req.	#Resp.	No Resp. #	No Resp. %	Offset mean	Abs. Offset 90th %ile	RTT mean	NTP version	NTS	Stratum
Amazon	v4	88	Yes	No	25 344	25 329	15	0.06%	2.55 ms	10.16 ms	151.36 ms	v4	No	4
	v6	88	Yes	No	25 344	25 327	17	0.07%	2.59 ms	9.92 ms	151.02 ms	v4	No	4
Apple	v4	46	No	No	13 248	12 366	882	6.66%	3.71 ms	20.90 ms	174.68 ms	v4	No	1/2 (77%/23%)
	v6	45	No	No	12 960	11 630	1330	10.26%	-2.09 ms	20.33 ms	155.06 ms	v4	No	1/2 (79%/21%)
Cloudflare	v4	2	Yes	Yes	576	576	0	0.00%	-4.82 ms	4.31 ms	15.18 ms	v4	Yes	3
	v6	2	Yes	Yes	576	576	0	0.00%	-0.68 ms	0.81 ms	6.73 ms	v4	Yes	3
Google	v4	4	Yes	No	1 152	1 152	0	0.00%	-3.05 ms	1.33 ms	14.37 ms	v4	No	1
	v6	4	Yes	No	1 152	1 152	0	0.00%	-0.62 ms	0.30 ms	9.39 ms	v4	No	1
Meta	v4	5	Yes	No	1 440	1 311	129	8.96%	-1.65 ms	3.92 ms	16.43 ms	v4	No	1
	v6	5	Yes	No	1 440	1 440	0	0.00%	-0.46 ms	1.38 ms	14.83 ms	v4	No	1
Microsoft	v4	12	Yes	No	3 456	3 303	153	4.43%	-6.37 ms	5.39 ms	130.98 ms	v3	No	3
Ubuntu	v4	4	No	No	1 152	1 146	6	0.52%	-1.24 ms	3.52 ms	26.13 ms	v4	Yes	2
	v6	3	No	No	864	863	1	0.12%	0.52 ms	0.98 ms	9.73 ms	v4	Yes	2

Table 13: Statistics from the measurement results of our second vantage point (NL-SIDN). Anycasted services are highlighted in orange.

results from this experiment show only one reference timestamp active at a time, and therefore, we could not detect the presence of multiple servers behind load balancers.

D.1 EDNS0 Support and Mapping for All Countries

The results of §4.1 were performed using Atlas VPs distributed all over the world. We performed the same measurement using just a single vantage point by using the client subnet field in DNS queries. This is defined in RFC 7871 [18] and allows us to pretend our queries are coming from a client in an arbitrary IP address prefix. The name server will then provide a localized response for to this address. The results of this measurement are shown in Figure 21.

E Offset Bounds and the Effect of Path Asymmetry

Figure 1 illustrates the standard four-timestamp exchange used in NTP measurements, where T_1 and T_4 are recorded by the client, and T_2 and T_3 are recorded by the server. Let θ denote the true offset of the server clock relative to the client, and let δ be the round-trip time measured as $\delta = (T_4 - T_1) - (T_3 - T_2)$.

Symmetric paths. Assuming known and constant client-server and server-client delays (d_1, d_2), the offset can be calculated from either the measured sending or receiving delay:

$$\theta = T_2 - T_1 - d_1 = -(T_4 - T_3 - d_2)$$

If d_1 and d_2 are unknown, but assumed equal, then we can calculate them using the round-trip time $d_1 = \frac{\delta}{2} = \frac{d_1+d_2}{2} = \frac{(T_4-T_1)-(T_3-T_2)}{2} = \frac{(T_4-T_3)+(T_2-T_1)}{2}$, yielding

$$\begin{aligned}\theta_{\text{NTP}} &= T_2 - T_1 - d_1 = -(T_4 - T_3 - d_2) \\ &= T_2 - T_1 - \frac{(T_4 - T_3) + (T_2 - T_1)}{2} \\ &= \frac{(T_2 - T_1) - (T_4 - T_3)}{2}\end{aligned}$$

which is the classical NTP offset calculation, which will give exact offsets for equal and constant delays.

Asymmetric paths. In practice, delays are neither equal nor constant, but depending on Internet routing. Network asymmetry is common, i.e. $d_1 \neq d_2$, introducing a bias in the measured offset. This creates a difference between θ and θ_{NTP}:

$$\begin{aligned}\theta &= \frac{(\theta + \theta)}{2} \\ &= \frac{T_2 - T_1 - d_1 - (T_4 - T_3 - d_2)}{2} \\ &= \frac{(T_2 - T_1) - (T_4 - T_3) + d_2 - d_1)}{2} \\ &= \frac{(T_2 - T_1) - (T_4 - T_3) + d_2 - d_1}{2} \\ &= \frac{(T_2 - T_1) - (T_4 - T_3)}{2} + \frac{d_2 - d_1}{2} \\ &= \theta_{\text{NTP}} + \frac{d_2 - d_1}{2}\end{aligned}$$

Because $d_1, d_2 \geq 0$, it holds that $-(d_2 - d_1) \leq d_2 - d_1 \leq d_2 + d_1$, with those bounds occurring when one delay is 0. Therefore and the corresponding bounds expand to

$$-\frac{d_1 + d_2}{2} \leq \theta - \theta_{\text{NTP}} \leq \frac{d_1 + d_2}{2},$$

i.e. $|\theta - \theta_{\text{NTP}}| \leq \frac{\delta}{2}$.

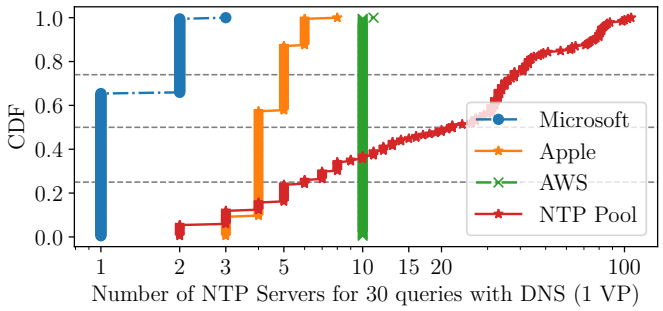


Figure 21: Number of NTP servers for each time provider, using EDNS0 measurements.

These expressions show that even modest asymmetry shifts the apparent offset, namely by half the forward-reverse delay difference, and this cannot be corrected without an independent time reference (e.g., GPS-synchronized clocks). The bounds derived here provide a rigorous interval in which the true server offset (ignoring transmission) must lie, given the measured timestamps.