On Borrowed Time: Measurement-Informed Understanding of the NTP Pool's Robustness to Monopoly Attacks

Robert Beverly
San Diego State University
rbeverly@sdsu.edu

Erik Rye
Johns Hopkins University
rye@jhu.edu

***Abstract*—Internet services and applications depend critically on the availability and accuracy of network time. The Network Time Protocol (NTP) is one of the oldest core network protocols and remains the de facto mechanism for clock synchronization across the Internet today. While multiple NTP infrastructures exist, one, the "NTP Pool," presents an attractive attack target for two basic reasons, it is: 1) administratively distributed and based on volunteer servers; and 2) heavily utilized, including by IoT and infrastructure devices worldwide. We gather the first direct, non-inferential, and comprehensive data on the NTP Pool, including: longitudinal server and account membership, server configurations, time quality, aliases, and global query traffic load.**

We gather complete and granular data over a nine month period to discover over 15k servers (both active and inactive) and shed new light into the NTP Pool's use, dynamics, and robustness. By analyzing address aliases, accounts, and network connectivity, we find that only 19.7% of the pool's active servers are fully independent. Finally, we show that an adversary informed with our data can better and more precisely mount "monopoly attacks" to capture the preponderance of NTP pool traffic in 90% of all countries with only 10 or fewer malicious NTP servers. Our results suggest multiple avenues by which the robustness of the pool can be improved.

I. Introduction

Accurate time is critical to the function and security of distributed systems. The Network Time Protocol (NTP) is the long-standardized and well-adopted protocol for synchronizing time between systems on the Internet [1]. The security of the protocol itself has been well-studied, with prior work demonstrating e.g., time-shifting and denial of service attacks [2], [3], [4], while recent efforts standardize NTP security mechanisms for authentication and integrity [5]. However, these NTP security mechanisms do not fully protect against malicious time servers or availability attacks.

In part due to outstanding security concerns, and to mitigate potential risk, several commercial operating system vendors and network providers operate their own NTP infrastructure, including Microsoft, Apple, Cloudflare, and Google [6], [7]. However, open source operating system distributions based on Linux and BSD utilize the NTP Pool project [8] for time synchronization by default. In addition to the prevalence of these operating systems in server and network infrastructure, embedded Linux is widely deployed on Internet of Things (IoT) devices such as printers, webcams, WiFi routers, and home automation. Thus, the NTP Pool (herein referred to simply as the "pool") is well-used and critical to the operation of a large number of in-the-wild Internet devices and services.

This work develops new measurements to shine new light on the NTP Pool and bring new insights into its overall robustness. Whereas prior efforts to characterize the NTP Pool rely on *indirect* inferences, e.g., via large-scale querying of the DNS [9], [10], [11], we develop a means for *direct* (non-inferential) measurement. Our method permits exhaustive and longitudinal measurement of the pool, and affords insight into previously unavailable information including: 1) all servers that are members of the pool, including poor-quality, offline, and "monitor-only" servers not returned in query responses; 2) query traffic volume and distribution; 3) server speed and configuration; 4) aliased servers that distort the perceived diversity; 5) accounts controlling a large number of servers; and 6) server lifetime. In sum, we make the following primary contributions:

1) Development and validation of a custom scraper to exhaustively and longitudinally gather granular data on the pool, including servers, accounts, zones, addresses, scores, traffic, and popularity (§III).
2) Fingerprinting to identify and characterize NTP server "aliases" present in the pool, including across IP protocol versions (§III-F).
3) Measurement-based characterization of the pool, including showing an inferred global pool rate of about 100k DNS queries per second (§IV).
4) Analysis and evaluation of pool server independence, including account owners, network diversity, aliases, and lifetimes, showing that only approximately 20% of the active servers are independent (§V).
5) Demonstration and validation of how the netspeed data we gather can be utilized to better and more precisely mount targeted "monopoly attacks" [4] whereby the at-

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.240541
www.ndss-symposium.org

tacker captures the preponderance of NTP pool traffic in 90% of all countries with 10 or fewer attack servers (§VI).

The remainder of the paper is organized as follows. We briefly review NTP and the NTP Pool in §II, while §III details our methodology. §IV describes our datasets and important macro characteristics of the pool. We then investigate the degree to which participating pool servers are independent from one another in §V. Next, we explore the feasibility of the monopoly attack using our data in §VI. Finally, we conclude with a discussion on ethical considerations and recommendations for improving the robustness of the pool.

II. Background

NTP, standardized in 1985, is a protocol to synchronize system clocks among a distributed set of servers across a variable latency, best effort packet switched network [12], [1]. NTP organizes distributed devices into a hierarchy, rooted in a reference clock (stratum 0, with high-precision time sources). Stratum 1 servers synchronize with stratum 0 time sources, while stratum 2 servers synchronize with stratum 1 servers, and so on. The history and evolution of NTP over its four-decade lifespan has been described in detail by prior research [13], [14], [15], [16].

Significant prior work has demonstrated attacks against the NTP, e.g., time shifting [2], [17], [11], [18]. A wide-ranging host of applications and services rely accurate time, from TLS certificate validation [19], [2] to authentication [2], [20]. It is well-accepted that inaccurate or incorrect time synchronization can enable multiple attacks including e.g., denial of service or incorrect trust calculations.

NTP can also be used as a means for network reflection and amplification attacks [21], [3], [22], [23], [24], [25] due to its use of connectionless UDP as its transport-layer protocol. Certain NTP message types (such as the `monlist` request, which induces servers to provide NTP client statistics) have amplification factors ranging from 100s to 1000s of times the size of the request. While `monlist` in particular has been deprecated since 2014 [26], other NTP queries (such as `version` and `showpeers`) are also well-suited for amplification attacks [3].

Rather than developing new attacks, this effort instead focuses on characterizing the NTP Pool through new measurements and understanding the robustness, resilience, and independence of its volunteer-operated, distributed network of NTP servers.

A. The NTP Pool

The NTP Pool is a system to coordinate queries to a distributed set of volunteer-run NTP servers; in particular, the NTP Pool itself does not operate NTP servers. Instead, the NTP Pool provides a public website with information, statistics, and the ability for volunteers to register their own servers. Servers are assigned to one or more "zones" based on their geographic location; these zones consist of country codes and continents, as well as a global "@" zone. The availability and quality of time provided by participating servers is monitored by the pool, via a distributed set of dedicated "monitors," to form a "score." The monitoring and scoring algorithm has been described in detail in prior work [10], [27]. In short, however, scores are initially set to 0 and can range from -100 to 20. A server's score increases when it responds with accurate time to a monitor's queries, and decreases when it is unresponsive or provides inaccurate time.

The NTP Pool then apportions servers to clients using the Domain Name System (DNS) based on multiple factors, including the server's score, registered "netspeed," and geographic zone. The NTP Pool further uses the client's geolocation to prefer servers from the same or nearby zones. Clients from countries with no NTP Pool servers receive DNS responses with servers from their continent zone. To ensure high-quality, reliable servers are provided to clients, only "active" servers – those with a score greater than 10 – are included in responses provided by the NTP Pool. Server operators can influence the frequency with which their server (and, hence, the traffic load) is provided by adjusting their netspeed value. By using hierarchical geographic server zones, a continuous monitoring and scoring system, and server administrator query load tuning, the NTP Pool is designed for dynamic allocation, load-balancing, robustness, and resilience to failures.

B. Prior Work on The NTP Pool

Prior work has sought to characterize the NTP Pool. Rytilahti et al. [11] scanned the entire IPv4 address space for responsive NTP hosts, and further used a crawler to repeatedly query the NTP Pool's DNS and query the returned NTP servers. Similarly, the work of Moura et al. subsequently used the RIPE Atlas infrastructure to issue DNS queries against the NTP Pool from a wider geographic region, again to discover NTP Pool servers [9]. These studies used large volumes of active DNS measurements to indirectly infer properties of the NTP Pool. Thus, if a participating server is in the pool, but not served in the DNS (e.g., because it is offline, not providing incorrect time, or configured to receive no NTP query load), these DNS-based methods will not discover the server. In contrast, our approach does not use the DNS, but rather directly queries the NTP Pool web site to produce more accurate and complete information, as well as a rich set of additional data including accounts, scores, and DNS answer rates crucial to our analysis.

Most closely related to our exploration of the feasibility of monopoly attacks against the NTP Pool, Perry et al. demonstrated the potential for malicious time servers to join the NTP Pool and carry out time shifting attacks by using large netspeed values [4]. Whereas their approach empirically determined the number of NTP servers an attacker would need to impact five large pool zones, our data affords new insights into the broader feasibility of these attacks and the NTP Pool's robustness. In particular, we mathematically deduce the number of servers required to monopolize the traffic of *any* zone – without needing to add servers to the zone *a priori*.

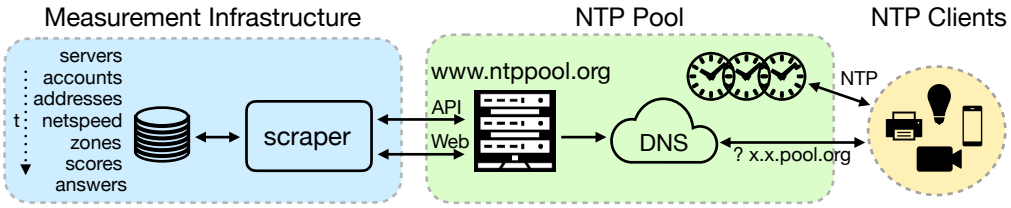


Fig. 1: Methodology: The NTP Pool website maintains statistics and APIs (green box) that we periodically query (blue box) to exhaustively enumerate participating servers and their properties. We gather multiple longitudinal datasets described in Table II.

Finally, studies have used the NTP Pool as a vehicle to measure other network properties. For instance, Durairajan et al. used data collected by NTP Pool server operators to measure one-way delays at scale on the Internet [28]. Rye and Levin subsequently used the NTP Pool to collect active IPv6 addresses, particularly from clients, which are difficult to discover using active measurements [29]. And Syamkumar et al. used data collected by NTP Pool server operators to detect network events, such as route changes or outages [30].

C. Motivation

A key motivation of our work is the widespread use of, and dependence on, the NTP Pool, effectively rendering it *critical infrastructure*. While many desktop and mobile operating systems utilize different closed NTP servers, e.g., NTP servers operated by Apple, Google, and Microsoft, a large number of Internet of Things (IoT) and infrastructure devices utilize the NTP Pool. We base this assertion on three observations:

- Data from the IPv6 Observatory [29], which collects IPv6 addresses from NTP Pool clients, supports the NTP Pool's use by many embedded Linux and IoT devices. For example, during the week beginning on April 20, 2025, the IPv6 Observatory was visited by over 5.5M unique Fritz!Box routers, 1.6M Amazon devices, and 289k Sonos speakers, all identified through their use of Extended Unique Identifier-64 (EUI-64) addresses in IPv6, which embed an interface's Media Access Control (MAC) address. Of note, end users typically do not (or cannot) reconfigure the chosen NTP server for such devices.
- Prior work from Moura et al. [9] examined DNS queries at root name servers to estimate the popularity of the NTP Pool, and find that, in their dataset, the NTP Pool receives 90M of the 126M total NTP DNS queries. Thus, the NTP Pool is the most popular time provider by a large margin.
- As we will show in §V, the DNS "answer" statistics maintained by the NTP Pool show a global DNS query rate to the pool of over 100k queries per second. Given DNS caching, the number of unique clients utilizing the NTP Pool is orders of magnitude higher.

This large-scale use, combined with the unique volunteer nature of the NTP Pool implies that traffic capture and time manipulation attacks would be highly impactful. Our work represents a comprehensive characterization of the pool's robustness to such attacks.

III. Methodology

Figure 1 provides an overview of our measurement infrastructure in relation to the NTP Pool, while Table II summarizes the data we collect with our infrastructure. Our methodology is primarily based on: 1) probing APIs of the public NTP Pool website; 2) a custom NTP Pool website scraper; 3) an NTP server fingerprinter; and 4) continuous measurements. We discuss each of these in turn, but first present the threat model.

A. Threat Model

Within the confines of the existing NTP Pool, we consider an attacker that seeks to capture a preponderance of NTP traffic in a country or region. The ultimate goal of the attack could be either passive surveillance and monitoring (e.g., collecting live IP addresses [29]), active back-scanning (e.g., port scanning live hosts) [31], [32], or time skew (e.g., to incorrectly influence clients' notion of time [10]).

We assume the adversary is capable of: 1) setting up hosts running NTP servers physically in a particular country or region, e.g., using cloud providers, tunnels, or virtual private servers; 2) configuring both IPv4 and IPv6 server reachability; and 3) joining the NTP Pool and accessing its public services, e.g., by creating accounts, adding servers, and viewing statistics. However, we assume that the adversary cannot circumvent the NTP Pool's access controls, DNS and load balancing mechanisms, or scoring algorithms. Further, the adversary cannot control the servers or behavior of other NTP Pool participants.

B. Historic Score Data

The only dataset we analyze that was not collected using our measurement infrastructure is the historic per-server score data. The NTP Pool maintainers archive complete historic score data within Google's cloud-based BigQuery, with per-year tables from 2008 to the present day [33], [34]. These tables contain approximately 12B rows (∼710GB) and include per-server timestamp and score rows with a distinct server ID column. Notably, however, this data does *not* contain the IP addresses of the servers or any other meta-data. Because of its longitudinal coverage, we use this historic data to infer both participating server lifetimes as well as server availability (fraction of time the server is a member of the pool and has a score that allows it to participate in serving queries to the pool).

TABLE I: HTTP and API endpoints provided by `https://www.ntppool.org`. The server ID is an internal monotonically increasing integer. By querying the space of server IDs, we obtain all server IPs.

Endpoint	Parameters	Response Type	Returns
`/scores/{%d}`	Server ID	HTTP 301	Redirect to `/scores/{ip}`
`/scores/{%s}`	Server IP	HTML	Server Info
`/scores/{%s}/json`	Server IP	JSON	Server Scores
`/api/data/server/dns/answers/{%s}`	Server IP	JSON	Per-zone DNS answers that include the server IP
`/api/data/zone/counts/{%s}`	Zone	JSON	Per-zone server counts and aggregate "netspeed"

C. Scraper

Prior efforts to characterize and understand the NTP Pool have relied on issuing large numbers of DNS queries from multiple geographic locations to discover participating servers [9], [11]. In addition to being inefficient, potentially inaccurate, and inducing undue operational load on the production system, this DNS-based probing approach cannot discover servers in the pool that are inactive, have a low score, or are in monitor mode, as these will never be returned in a DNS response.

In contrast, we discover the ability to exhaustively enumerate all servers, past and present, active and inactive, by directly HTTP querying the NTP Pool website. For each server, the NTP Pool website provides a "score" history page that plots the accuracy of the timing information from that server as observed by a network of sentinel monitors. The public URL to issue an HTTP GET for these server statistic pages requires the NTP server's IP address – which we do not know a priori. However, in examining the NTP Pool backend infrastructure, we observe that each server is assigned a monotonically increasing integer identifier. We then find a URL endpoint that maps (via an HTTP-level redirection) NTP Pool internal server integer identifiers to their corresponding IP address. Table I provides the specific endpoints our scraper and measurement infrastructure query. For example, to map the server with identifier `59105`, we HTTP query: `ntppool.org/scores/59105` which returns an HTTP 301 response with the URL: `/scores/2001:470:1f07:c21:1::123`. Once we have this ID to IP mapping, we can query the other endpoints to obtain score and answer data.

We leverage the relatively small and monotonically increasing integer NTP server identifier space to create an NTP Pool website scraper (Figure 1) to enumerate all NTP Pool servers and then query for the next unused identifier every 90 minutes on average. Thus, we discover new servers shortly after they are added to the system. In addition, we use a separate scraper to query the website for statistics, scores, and metadata of all servers every day. The retrieved metadata includes the user account associated with the server, the country and region zones served, the server's score, and the server's netspeed. We are also able to detect when servers are deleted. Note that a server may have a low score, and thus not be included in NTP Pool responses, but servers are only deleted if a user requests deletion, or the server is offline or otherwise unresponsive to NTP requests for an extended period of time.

We began collecting the `pool-scrape` data in October 2024 and continued collecting through July 2025, representing approximately 9 months.

D. Netspeed

The NTP Pool management interface allows volunteers to specify a "netspeed" for each server with discrete values in the set: 0, 512kbps, 1.5Mbps, 3Mbps, 6Mbps, 12Mbps, 25Mbps, 50Mbps, 100Mbps, 250Mbps, 500Mbps, 1Gbps, 1.5Gbps, 2Gbps, and 3Gbps. The intent of the netspeed setting is to allow server operators to participate in the pool while providing a coarse-grained method to control the received query rate. A common point of confusion on the discussion boards surrounds how these netspeed settings affect the actual data rate of received NTP queries.

Despite the data rate (e.g., Mbps) labels for netspeeds, the setting is instead a *relative* weighting[1]. The NTP Pool determines the *aggregate* netspeed of all servers actively participating in the zone and then apportions a relative fraction based on each server's netspeed. As a result, while the netspeed will change the received query rate, it may bear no relationship to the true rate.

Consider, for example, a zone with five total servers: four servers set to a netspeed of 25Mbps and one server with a netspeed of 100Mbps. The aggregate netspeed in this hypothetical zone is 200Mbps. Thus, the first four servers will each receive approximately one eighth (12.5%) of the total query traffic for the zone while the fifth will receive half (50%). Of course, this is an approximation as the NTP Pool can only control how frequently it includes a particular server in a DNS query for a given zone, but the final traffic rate is proportional.

E. Pool DNS answers

We find an additional NTP Pool web server API endpoint, "answers," that takes a server's IP address as input and returns a JSON object containing a count of per-zone DNS responses, i.e., how many times the server was included in response to a client's DNS query to the pool for a given zone. While the JSON does not contain a timestamp, by querying the API endpoint for 100 different servers every minute, we experimentally determine that the NTP Pool web server updates the returned JSON data every 30 minutes.

Therefore, in addition to periodically probing the NTP Pool web server for servers and accounts, we query this answers API endpoint for all of the active servers (those with score

[1]The NTP Pool server management page states that "this speed does not mean the wire speed of your server, it's just a relative value to other servers." [35]

TABLE II: Overview of datasets, sources, and tools: we gather and use the first five datasets in this work; the last two datasets (shaded) represent prior work and are included for comparison.

Dataset	Period	Source	Addresses (v4/v6)	Description
bq-scores (§III-B)	05-Sep-2008 – 01-Jun-2025	BigQuery	39,756	Server scores
pool-scrape (§III-C)	22-Oct-2024 – 10-Jul-2025	scraper	9,955 / 5,725	Servers, accounts, and metadata scraped from public website
pool-answers (§III-E)	28-Jul-2025	scraper	3,867 / 2,228	Per-server, per-zone DNS response rates
server-fp (§III-F)	23-Jul-2025	fingerprinter	3,967 / 2,275	IPv4 and IPv6 aliased servers
ntp-residual (§III-G)	08-Dec-2024 – 24-Jul-2025	NTP Pool Server	198,787,734 / 109,930,726	IPv4 and IPv6 client IPs
Prior Work:				
pool-web [8]	10-Jul-2025	Web page	3,434 / 1,926	Aggregate counts published on public web page
deep-dive [9]	26-Aug-2021 – 31-Aug-2021	Moura et al.	3,056 / 1,479	Discovered via RIPE Atlas active DNS probing

$\geq$ 10) every 30 minutes over the course of one day on July 28, 2025 to obtain the `pool-answers` dataset.

F. Server Fingerprinting

A host running an NTP server application may have multiple physical or virtual network interfaces. These interfaces can be numbered with one or more IPv4 and IPv6 addresses. Whereas a single NTP application may listen and respond to NTP queries sent to different addresses, the pool has a strict one-to-one mapping between an address and a server, i.e., a "server" is an instance of an NTP server daemon bound to a single IP address. We term two IP addresses with the same NTP server application as "NTP aliases."

The NTP protocol defines a mode for control messages [1]. These control messages permit management and diagnosis, for instance "read variables" [36]. However, for privacy and security reasons, this functionality may be disabled or blocked by the network, especially for remote connections. To understand the ability to leverage NTP control messages for fingerprinting, we queried IPv6 addresses active in the pool in May, 2025. Of the 1,658 IPv6 NTP servers in the pool that respond to an NTP time client query, only 28 (1.7%) also respond to a read variables request. Manual investigation of the responses indicates that these few responding servers are running quite old versions of NTP server implementations.

We therefore do not use control messages to find aliases, but rather implement active server fingerprinting starting with the open-source "ntpdedup" code [37]. (We forked this codebase which we keep anonymous for submission, but will make public and will contribute our modifications back via merge requests.)

Instead, we leverage unique features and fields of standard NTP time (mode 4) response packets to provide a fingerprint and identify aliases. In particular, our modified version of the fingerprinting tool ntpdedup collects several NTP time response fields that identify the server's version, time source, the server's last synchronization time, and its precision and maximum allowable polling interval. The time source data includes the reference identifier ("refid"), stratum, and dispersion. The refid is a 32-bit field that identifies the reference clock for stratum-1 servers (analyzed in §IV-B) or the synchronization peer IP address for stratum-2 and higher servers. The stratum field is a single byte; we identify 7 unique stratum values among NTP servers in our data. Similarly, polling interval and precision fields are each one byte. While there are only 11 unique poll and 22 unique precision values within our data, these fields are static and provide course-grained differentiation to identify clear non-aliases.

The "reference timestamp" contains the time since the system's clock was last set or corrected and is represented in "NTP timestamp format:" seconds since January 1, 1900 with 32 bits encoding the integer component of seconds and 32 additional bits encoding the fractional seconds. While NTP server applications synchronize time, their individual internal update intervals are not – hence the precision of this field affords strong discrimination power and, naturally, is the field that exhibits the largest number of unique values in our data.

When these fields are collected from a server over a short time interval, they will be consistent, even across different aliases of the same physical server. Thus, we can identify potential aliases among NTP Pool servers by comparing whether these values are equal across multiple server responses.

Our alias resolution technique can suffer from both false positives and false negatives. Two non-alias server IP addresses are more likely to be incorrectly identified as aliases if they synchronize with the same upstream NTP server; this problem is particularly acute for stratum 1 NTP servers, which synchronize time with a limited number of stratum 0 time sources (e.g., GPS) and therefore have a small set of refids. We examine the distribution of stratum 1 server refids in §IV-B. For this reason, and to focus on alias precision rather than recall, we do not attempt to de-alias 830 active stratum 1 NTP servers. Conversely, two server IP addresses may be erroneously misidentified as non-aliases. For example, in the event that an NTP server resynchronizes its time between queries to two IP addresses assigned to it, its reference timestamp would differ in the two NTP responses. We use ground-truth servers to validate our fingerprinting method in §III-H and further use account owner and ASN as proxies to evaluate the discovered aliases across the entire pool in §V-A.

Other potential NTP protocol features are available for de-aliasing, such as the root delay and root dispersion; however, these can vary significantly between probes and cause false negatives. Similarly, IP-layer features such as IPv4 TTL and DSCP, and IPv6 hop limit and traffic class, can provide intra-protocol alias hints, but are too unreliable as discriminators, especially for identifying cross-protocol aliases. We added collection and correlation of these weak identifiers, as well as the server's polling interval, to ntpdedup, but do not use them by default. To obtain a weaker alias inference for stratum

1 servers, we could omit the refid field as part of the alias determination. However, we opted to exclude stratum 1 servers from alias consideration rather than include false positives.

G. Residual NTP Client Traffic

The NTP Pool warns prospective operators that running an NTP server as part of the pool is a long-term commitment. The pool further indicates that a server operator may continue to receive NTP traffic to their servers "weeks, months, or even YEARS before the traffic goes completely away" due to NTP server IP address caching [38]. This means that an adversary that joins the NTP Pool to receive potential victims in the form of NTP clients may be able to continue to attack those victims (e.g., by shifting their clocks) even if detected and evicted from the pool.

In order to understand the quantity and duration of client requests to a pool server after it has been evicted, we conduct some small-scale experiments with an NTP server under our control. We add and remove this server from the NTP Pool, but continue to monitor received traffic levels for months after it was evicted. This helps us understand how long and how many clients may be at risk even in the event that a malicious NTP server is removed from the NTP Pool.

H. Validation

For the purposes of validation, we configured eight virtual private servers (VPS) in various geographic regions. We added both the IPv4 and IPv6 address of one of the VPS to the NTP Pool. For the remaining seven VPSes, we configured two IPv6 addresses each and added them to the NTP Pool. Thus, we added a total of 16 servers to the NTP Pool. We then examined the operation of our scraper and measurement infrastructure by validating: 1) presence of the server in our database; 2) correct zones and meta-data; and 3) the inferred server age, i.e., the duration of time it was participating in the NTP Pool. For all 16 servers, across these three metrics, we achieve perfect accuracy, providing an additional degree of confidence in the correctness and completeness of the data we gather.

We next evaluated the accuracy of our NTP server fingerprinting code in identifying NTP aliases. In addition to the eight different pairs of aliases in our ground truth set of servers (one mixed IPv4/IPv6 alias, and seven IPv6 pair aliases), we selected 100 active servers with a score $\geq$ 10 at random. We then used our fingerprinter to probe all 116 addresses. Within this sample experiment, our alias detection correctly identified all eight aliases and did not produce any additional false positive aliases.

On one of our ground truth VPSes, we then added 10 different IPv6 addresses to a single interface and again ran our fingerprinting code. The fingerprinter correctly identified this cluster of 10 addresses as belonging to a single server. Finally, we experimented with running stock configurations of two popular NTP server applications, `ntpd` and `chronyd`, both running on the same physical VPS, but with `ntpd` bound and listening to two different IPv6 addresses and `chronyd` bound to a third address. In this instance, our fingerprinter identified one cluster of two addresses and one cluster of a single address, indicating that NTP server implementation plays a role in cluster determination.

TABLE III: NTP Pool Summary Statistics (pool-scrape dataset) as of July 10, 2025

	IPv4	**IPv6**
Servers (IP addresses)	9,955	5,725
Autonomous Systems	2,107	841
Zones	29	29
Active Servers	3,967	2,275
Servers w/ Accounts	4,277	2,948
Stratum 1 Servers	548	282
Monitor-only Servers	1,672	1,007
Anycast Servers	7	5

I. Limitations

While we believe our dataset represents the most accurate characterization of the NTP Pool to-date, we note several limitations of our methodology. First, the public webpages for each server do not contain account names or organization information in cases where the user has configured their account to be private. While we discover 1,332 unique accounts and 15,680 servers, we are only able to infer the accounts responsible for 7,225 (46%) of the servers. Thus, our account visibility is limited.

Second, while the BigQuery data include historic scores dating back to 2008, this data can only be used for inferring server lifetimes. The remainder of our analyses are based on the scraping and measurement infrastructure which includes only 9 months of longitudinal data.

Finally, while we perform multiple experiments to validate the accuracy of our fingerprinting, we note that corner cases may exist that lead to false positive or false negative aliases. For instance, in the case that two different NTP server application implementations are running on different interfaces of the same physical machine, our fingerprinter cannot ascertain that these are aliases. However, we believe such instances to be uncommon.

IV. Pool Characterization

Because NTP Pool membership requires only that a volunteer have a publicly accessible NTP server, many types of individuals, institutions, and organizations might consider adding their server. To characterize the types of servers and server operators that make up the NTP Pool, we analyze both current and historical IP addresses that comprise the NTP Pool as well as the accounts linked to those IPs over several axes.

A. Server Lifetimes

We first consider the "lifetime" of the servers in the pool and analyze the `bg-scores` data covering 17 years and nearly 40,000 servers. Figure 2 shows the cumulative fraction of servers as a function of their lifetime, as inferred by the presence of scores in the dataset. We see that the median lifetime is approximately one year, while more than 10% of servers

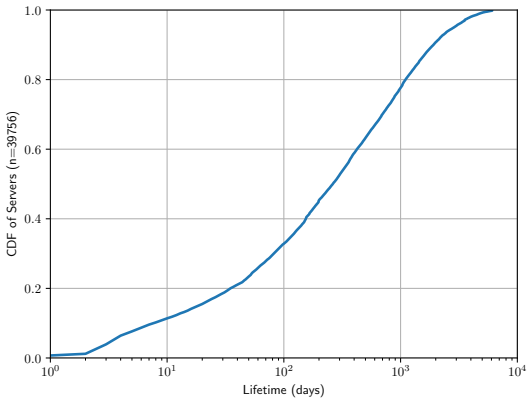


Fig. 2: Lifetime of NTP Pool servers. More than 10% of servers participate for less than 10 days.

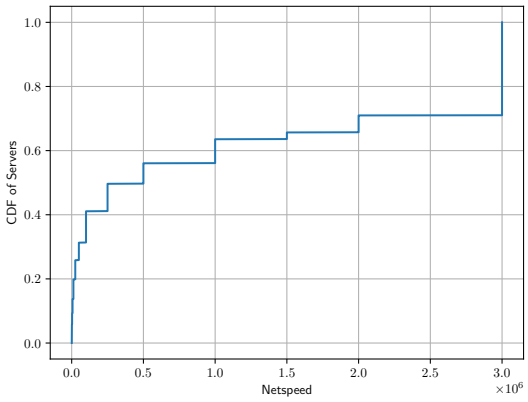


Fig. 3: Distribution of the netspeeds of the 5,333 servers with nonzero netspeed in the NTP Pool. Another 57,049 servers have zero netspeeds, either because they are set to "monitor-only" or have been deleted.

participate for less than 10 days. Conversely, approximately 20% of the servers have been participating in the pool for 3 or more years. Based on this analysis, the pool might consider prioritizing returning servers with higher uptimes to improve the pool stability, as well as mitigate short-lived or ephemeral attacks.

B. Clock sources

As described in §II, each NTP server synchronizes with servers at a lower stratum and serves servers with higher stratum. Within our data, we find 830 stratum 1 servers, 548 of which are IPv4 and 282 IPv6. Stratum 1 servers depend on stratum 0 high-precision time sources for their reference clock. Per the NTP protocol, the reference identifier ("refid") in NTP packets from stratum 1 servers encode their clock source in ASCII [39]. Table IV provides the distribution of the top 10 most common reference clock identifiers. Global Navigation Satellite System (GNSS) sources are by far the most common, including the PPS, GPS, and GNSS identifiers. However, there

TABLE IV: Top 10 Most Common NTP Pool Stratum 1 Server Reference Clocks

Clock	IPv4 Count	IPv6 Count
PPS	200 (36.5%)	105 (37.2%)
GPS	151 (27.6%)	63 (22.3%)
GNSS	25 (4.6%)	16 (5.7%)
MRS	19 (3.5%)	10 (3.5%)
PPS0	18 (3.3%)	14 (5.0%)
MBGh	15 (2.7%)	0 (0.0%)
PTP0	9 (1.6%)	14 (5.0%)
PHC0	6 (1.1%)	2 (0.7%)
kPPS	6 (1.1%)	3 (1.1%)
DCF	5 (0.9%)	2 (0.7%)

is a large range of identifiers with 59 unique refids across all servers, many of which are not standardized.

The small number of false positives we find in our NTP server dealiasing (§III-F) are due to collisions among stratum 1 servers caused by the limited number of refids. We therefore exclude stratum 1 servers from the dealiasing component of our analysis.

C. Netspeeds

The "counts" API endpoint (see Table I) provides a direct means to query the pool on a per-zone basis for the count of active IPv4 and IPv6 servers, as well as the *aggregate* netspeed. Using our ground-truth servers in sparsely populated zones, we modified the server's netspeed and experimentally verified the effect on the reported aggregate netspeed, as well as the received query volume.

Figure 3 displays a CDF of the 5,333 NTP Pool servers with nonzero netspeed. More than half of all NTP servers have a netspeed of 500 Mbps or less, which may permit an attacker with a significantly higher rate to dominate a zone's NTP queries (§VI).

D. Traffic Load

Our `pool-answers` dataset allows us to compute the aggregate rate of DNS responses the NTP Pool returns for different zones. Note that this rate is distinct from the number of NTP queries arriving at a pool server or the servers within a zone due to DNS caching effects. Thus, the rates we compute are a strict lower bound of the total number of actual NTP queries.

Figure 4a displays the number of servers participating in each pool zone for both IPv4 and IPv6. Note that a server may belong to more than one zone, hence the total sum of counts in the plot is larger than the number of active servers. Consistent with prior work, we also find that the distribution of servers across zones is highly skewed and that many zones remain underserved. As a result, the global "@" zone experiences a very high query rate and the operators have sought to include anycast servers in the pool. We analyze anycast servers in the next subsection.

Figure 4b provides the distribution of inferred DNS response rates per zone, while Table V shows the number of servers for the top 10 highest DNS answer rate zones, where "@" is

TABLE V: NTP Pool DNS answer aggregate rates across the top 10 zones. While the rates vary significantly by zone, we infer a large aggregate system rate of over 100k DNS queries / second (each DNS response contains up to 4 NTP servers).

	IPv4		IPv6	
Zone	**Servers**	**Rate (servers/sec)**	**Servers**	**Rate (servers/sec)**
@	3,211	194,653	1,941	17,201
us	677	54,924	428	4,804
br	29	15,608	20	1,454
de	562	8,599	496	914
cn	34	8,238	43	768
uk	216	7,235	122	611
ru	404	6,917	77	509
in	45	6,244	29	658
fr	219	5,122	138	487
ca	119	4,473	71	402
Total	3,867	389,257	2,228	34,399

the global zone. We estimate that the entire pool system is returning approximately 390k and 34k IPv4 and IPv6 servers in DNS responses per second. Since the pool returns up to four addresses per DNS query, this equates to a global rate of approximately 106k queries per second.

We see an order of magnitude higher rate for IPv4 as compared to IPv6, likely due to the fact that the pool will only return a DNS response containing an IPv6 server if the "2.xxx.pool.ntp.org" name is queried. We see that the global zone and the United States and Brazil zones account for approximately 68% of all DNS answers.

E. Anycast

Among the servers we discover in `pool-scrape`, we find 12 registered in two or more different continent zones. Seven of these servers have IPv4 addresses, while five have IPv6 addresses. Two of these servers are located in Türkiye and are in both the Asia and Europe continent zones, which is consistent with Türkiye's physical location. Four servers, consisting of two IPv4/IPv6 aliases belong to the same account – a regional ISP. The remaining six servers are all IP anycast, as determined by looking up their addresses in the MAnycast anycast census [40]. Four of these anycast servers belong to Cloudflare, while two are within a Dutch academic network.

We again use the `pool-answers` dataset to examine the rate of NTP DNS answers for these multiple continent anycast servers. We find that the IPv4 anycast servers are returned at a rate of 41.8k DNS responses per second (approximately 11% of the global traffic), while the five IPv6 anycast servers are returned at a rate of 4.0k DNS responses per second (approximately 12% of the global traffic). In investigating these anycast servers, we find an online discussion including the pool operators who emphasize that these servers help provide service for zones with few or no servers [41]. Thus, this small set of servers are disproportionately important in the pool.

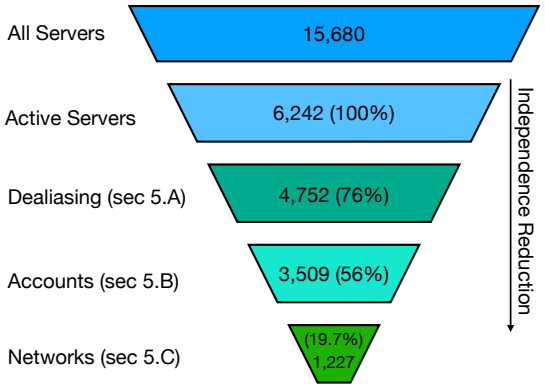


Fig. 5: Overall Independence analysis

V. Server Independence

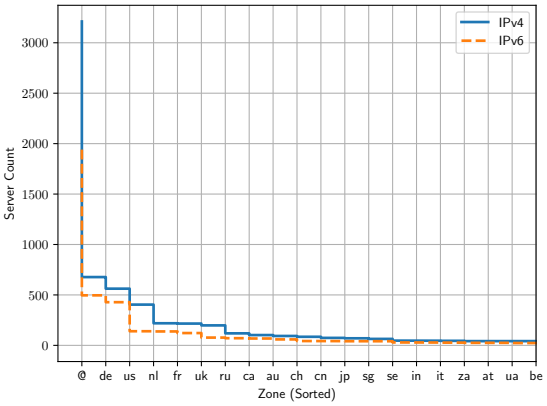
In this section, we analyze the structure and composition of the NTP Pool as inferred from our measurements. The primary contribution of this analysis is to demonstrate the ways that the seemingly large number of servers within the pool *are not independent.* In other words, we find that many of the servers have correlated behaviors and share fate in the event of a failure. To better understand correlations between servers and how servers may share fate, we employ a series of independence reduction steps using tools and data described in the previous sections.

As illustrated in Figure 5, we begin with the full set of servers from the `pool-scrape` dataset. We then consider the snapshot of 6,242 servers that had a score $\geq$ 10 on July 10, 2025 and count this set as the full set (100%) of servers available on that day. We then use our fingerprinting method to dealias these to unique hosts (§V-A). Next, we examine account uniqueness in detail to further winnow the set (§V-B). Finally, we examine the server network connectivity and Autonomous System (AS) Numbers (ASNs) (§V-C). As we will show, while our dataset includes 6,242 active pool servers, only 1,227 of these are fully independent servers – a reduction of more than 80%.

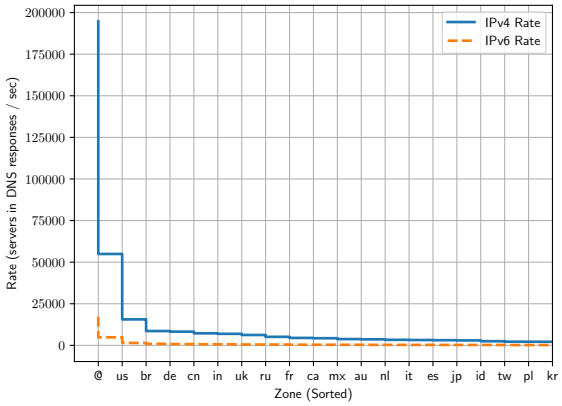
A. Server Aliases

Recall that a "server" registered within the pool is identified by its IP address. Thus, the IPv4 address and the IPv6 address of a server may correspond to the same physical machine, or the same physical machine may have multiple IP addresses within the same protocol family assigned to its interfaces. In this subsection, we turn to identifying these "NTP aliases" using our fingerprinting method of §III-F.

We define "NTP aliases" as two IP addresses that respond to NTP queries from the same NTP daemon running on a single host. Let an "NTP alias cluster" be the set of IP addresses that are NTP aliases, and the "cluster size" be the number of addresses in the alias set. Let a "singleton cluster" be a single IP address with no identified aliases, i.e., a cluster with size one. Last, we refer to the cluster "covering prefix length" as the longest IPv4 or IPv6 network prefix mask such that the prefix encompasses all IPv4 or IPv6 addresses within the set.



(a) Number of servers within each pool zone



(b) Inferred DNS response rates across NTP pool zones

Fig. 4: Pool DNS response statistics: both the distribution of servers across zones as well as the per-zone DNS answer rates are highly skewed.

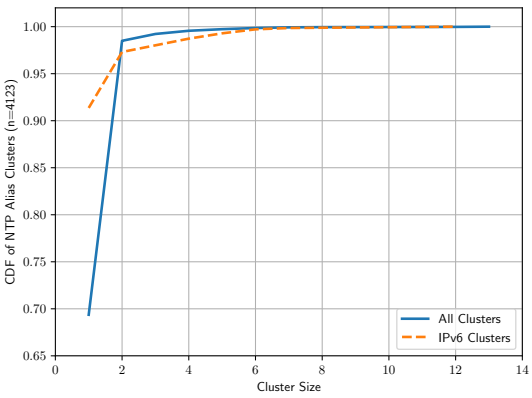


Fig. 6: NTP Alias Clusters: while 69% of addresses belong to a singleton cluster, there are a significant number of clustered IPv4 and IPv6 pairs, and there exist large IPv6 clusters.

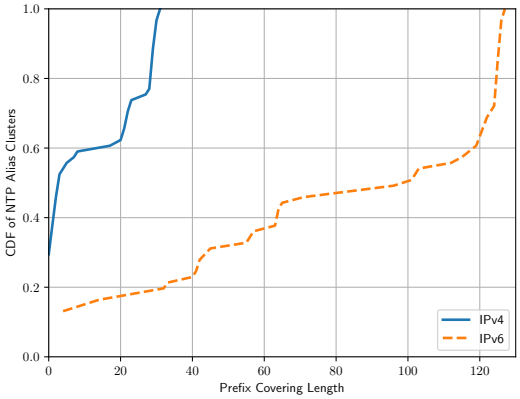


Fig. 7: NTP Cluster Covering Length (Clusters of size ≥ 2): 50% of the addresses in IPv6 alias clusters are within the same /64.

For example, if an alias cluster contains three IPv4 addresses: `1.2.1.10`, `1.2.3.200`, `1.2.14.30`, the covering prefix is `1.2.0.0/20` and the covering prefix length is 20.

We probed 6,242 servers (3,967 IPv4 and 2,275 IPv6) within the `pool-scrape` data with score ≥ 10 on July 23, 2025. Of these, a total of 5,687 responded (91.1%) to our fingerprinter. We identify 4,123 NTP alias clusters, 2,860 (69%) of which are singletons (i.e., a cluster with a single address and no aliases). We exclude from analysis 74 clusters containing one or more stratum 1 servers (a total of 160 addresses).

Figure 6 displays the cumulative fraction of clusters as a function of their size, both for all clusters and IPv6-only clusters. 29% of the clusters are of size two, and 90% of these consist of an IPv4 and IPv6 pair. However, the distribution has a long tail; for instance we find larger IPv6 clusters containing as many as 13 addresses.

We then examine cluster covering sizes; Figure 7 displays the CDF of alias clusters as a function of their cover size for IPv4 and IPv6 (note, it is not possible to obtain a prefix that covers the mixed protocol clusters). Approximately 40% of IPv4 clusters have a covering prefix length of /20 or longer (more specific), while the 50% of IPv6 clusters have a covering prefix length of /64 or longer.

While this largely maps to our intuition that aliases in the same address family should be numerically close, as they represent addresses on the same host, the smaller prefix covering lengths were unexpected. To better understand these results and measure inter-cluster consistency, we examined both the account owner and the ASN to which the servers in the clusters belong.

Among the 1,264 non-singleton clusters, we find that, for all servers within the cluster, 1,160 have consistent, i.e., matching,

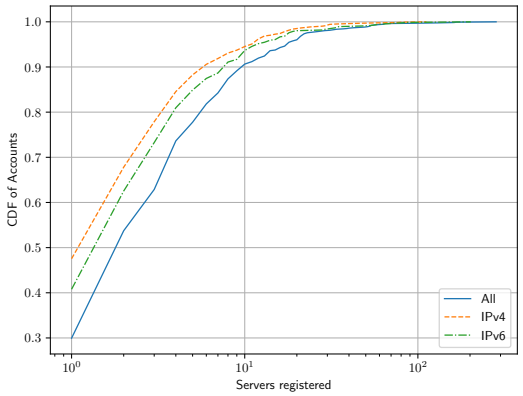


Fig. 8: Distribution of the number of NTP Pool servers registered per account. A small number of accounts control a disproportionate number of servers.

account owners and ASNs. A further 90 have consistent account owners, but inconsistent ASNs, and 7 have matching ASNs but inconsistent account owners. Only 6 clusters have both inconsistent ASNs and account owners. Thus, we believe our cluster inferences to be largely correct. Among the inconsistent ASNs, we find that some providers have two different ASNs for their IPv4 and IPv6 networks. Even within the same ASNs, that network may own and advertise multiple different prefixes that are not numerically close in the address space. Finally, we find examples of IPv6 tunnels from third-party providers that account for some of the inconsistent ASNs as well as small covering prefix lengths. Among the inconsistent account owners, we discover instances of accounts with different names that are, however, clearly related. For instance, one account uses an individual's full name, while a second account uses the individual's abbreviated name.

B. Server Control

While the `bg-scores` dataset provides an extended historical view, the data only includes server IDs and scores. We next turn to our gathered `pool-scrape` dataset to examine accounts registering and controlling the participating servers. Figure 8 displays the cumulative distribution of accounts as a function of the number of servers they control, broken into IPv4 and IPv6 servers.

While the median number of servers per account is approximately two, the distribution has a long tail. In particular, we find one account that controls over 340 servers, while the top 10 accounts control nearly 1,300 servers in total – a significant overall fraction of the entire pool's active servers. Note that this concentration is a lower bound; as noted in the limitations (§III-I), we are unable to map anonymous servers to their account owners.

C. Server Networks

We next consider the networks of the participating pool servers by mapping the IP addresses of all of the servers we discover in `pool-scraper` to the Autonomous System (AS) to which they belong. For this, we utilize a complete Routeviews BGP table snapshot from July 23, 2025 [42] and perform longest prefix matching. Figure 9 shows the cumulative distribution of servers as a function of the total number of ASes to which they belong. We further separate the analysis between IPv4 and IPv6 servers, as well as those servers that are active (score $\geq$ 10) or inactive. We find that the servers are concentrated in a small number of networks, with 50% of the servers belonging to fewer than 100 ASes. Restricting the scope to just those servers that are active reduces the overall ASN diversity. Further, the addition of the IPv6 servers does not add additional AS diversity, as evidenced by the intersection of the IPv4 and IPv6 lines.

We used IPinfo.io [43] to categorize the *types* of ASes that the NTP Pool's servers are located within. Of the 15k total servers our enumeration discovered (§III-C), slightly more than half (8,500, 54%) were located in cloud hosting providers. Over 800 hosting companies are represented in this count, but the NTP server IPs in cloud providers are disproportionately concentrated in only a few providers. Hetzner has hosted 1,049 unique NTP IP addresses, or nearly 7% of all NTP Pool IP addresses. Similarly, OVH hosts or has hosted nearly 5% (732) of all NTP Pool IPs; Vultr, DigitalOcean, Akamai, Oracle and Amazon all contribute more than an additional 1% each as well.

TABLE VI: NTP Pool Server Classification: the majority of servers in the pool are within cloud hosting or service provider infrastructures.

AS Type	Count	%
Hosting	8,500	54.2
ISP	6,036	38.5
Education	476	3.0
Business	395	2.5
Unknown	189	1.2
Government	84	0.5

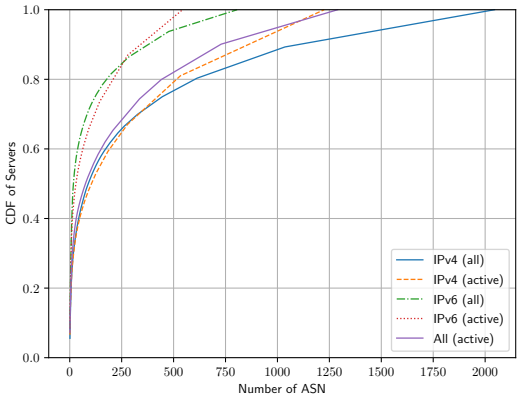


Fig. 9: Server AS distribution: the majority of servers are located within a small number of networks, while IPv6 does not substantially contribute to the network diversity.

An additional 6,036 (38%) NTP server IPs were labeled with IPInfo's "ISP" category, which encompasses both large transit providers like Hurricane Electric, but also customer ASes like Comcast, KPN, and Vodafone. Of these, Hurricane Electric is most common (397 NTP Pool IPs), with Comcast (331) and Deutsche Telekom (237) rounding out the top three.

The remaining 8% of NTP Pool server IPs belong to educational networks (476), businesses (395) such as Alibaba, Apple, and Facebook, and governmental networks (84) such as Hungary's KIFU Governmental Information Technology Development Agency and the US National Institute of Standards and Technology (NIST). One-hundred eighty-nine server IPs could not be categorized by IPInfo.

These results indicate that NTP Pool servers are being run predominantly in cloud hosting networks. While the high availability of most cloud providers might be viewed as a boon to the NTP Pool's resilience, the fact that large swathes of server IPs are found in only a small number of ASes indicates that the NTP Pool relies heavily on a few underlying providers.

Finally, note that because most of the 15k NTP Pool addresses are no longer active NTP servers, it is possible that the IP addresses formerly in the NTP Pool have since been reassigned to a different AS. This might confound our AS type analysis if the new AS is of a different type. However, we believe that this type of error is relatively uncommon and does not meaningfully affect the overall distribution of NTP Pool server ASes.

D. IPv6 Servers

Next, we examine the IPv6 servers within the pool, their connectivity, and inferences we can make from the addresses. First, we note that one method for obtaining IPv6 connectivity in the absence of native IPv6 is to utilize an IPv6-in-IPv4 tunnel, and a popular service provider of this is Hurricane Electric's "Tunnel Broker" [44]. We identify Hurricane Electric tunnels via their registered IPv6 prefix of `2001:470::/32`. A total of 337 servers are within this prefix and likely connected via this tunnel broker.

Next, we use the `addr6` tool to characterize the interface identifiers (IIDs) of the IPv6 server addresses. Table VII shows the distribution of IIDs, across both active (score $\geq$ 10) and inactive IPv6 servers. Approximately 5% of the servers use EUI-64 and embed their interface's MAC address, while over 20% have random, Privacy Extension (PE) addresses. Approximately 50% of the addresses are "low-byte," where the most significant bytes of the IID are zero indicating that the address was likely manually configured.

PE and EUI-64 addresses are more typical of client hosts, whereas low-byte and embedded are more typical of long-lived infrastructure hosts. Interestingly, the active servers are more likely to use a low-byte address, while the inactive servers are more likely to use PE and EUI-64. However, server AS-specific nuances are at play here, as well: for instance, 169 of 205 (82%) of the hosting provider Vultr's (AS20473) IPv6 IP addresses are EUI-64. On the other hand, 0 of the 89 Ionos (AS8560) IPv6 IP addresses are EUI-64.

TABLE VII: Pool IPv6 Server Address IID Categorization: Low score servers are more likely to use EUI64 and Privacy Extension (PE) addresses.

Type	All	score$\geq$10	score$<$10
Low-Byte	48.1%	50.7%	46.5%
Embed-IPv4	8.2%	8.5%	8.1%
Embed-port	8.8%	12.0%	6.8%
EUI64	5.4%	3.1%	6.8%
PE	21.0%	16.7%	23.7%
Other	8.5%	9.0%	8.1%

VI. Pool Monopolization

The canonical NTP Pool definition of a "server" is an IP address. A single physical machine may have multiple network interfaces, both physical and virtual, and may have multiple IP addresses assigned to an interface. Thus, there is a many-to-one mapping of NTP Pool server IP addresses to hosts. For example, an NTP host may have a single interface with one IPv4 address and two IPv6 addresses assigned to it – if the operator of this host registers all three addresses in the NTP Pool, the NTP Pool sees these as three servers.

A. Monitor Only Mode

Of note, one of the netspeed rates is zero, which corresponds to a "monitor only" mode. In this mode, the server is a member of the pool, but will not be included in any DNS responses (and, hence, should not receive any client NTP queries as a result of the pool). However, monitor only mode servers are queried by the pool monitors to determine the quality of time they are providing.

The IP addresses of the pool's monitoring infrastructure are not published. However, as described in prior works [10], monitor only mode permits a server to readily determine the pool's current monitors. With knowledge of the monitors, a malicious server can selectively respond, providing good time to the pool monitors, while sending a different time to other clients, e.g., as a part of a time skew attack.

Our scraper finds all instances of servers with a netspeed of zero. We find that 2,679 of the 15,680 servers (17.1%) in our dataset are operating in monitor only mode. Among these, 1,672 are IPv4 servers (62.4%) while 1,007 are IPv6 servers (37.6%). While these monitor only mode servers may be innocuous, adopting prior recommendations to use ephemeral addresses for the monitors is necessary to defend against attacks that rely on discovering the monitors.

B. Residual Traffic

Our work demonstrates that an attacker need not even keep their server active in the NTP Pool to mount some attacks against NTP clients. The NTP Pool indicates that running an NTP server is a long-term commitment and that traffic may persist long after a server's removal from the pool. To understand the contours of this "residual" NTP traffic after a server has been removed from the pool, we provisioned a new, dual-stacked NTP server in a US cloud hosting provider. Figure 10 depicts the course of our experiment, displaying the

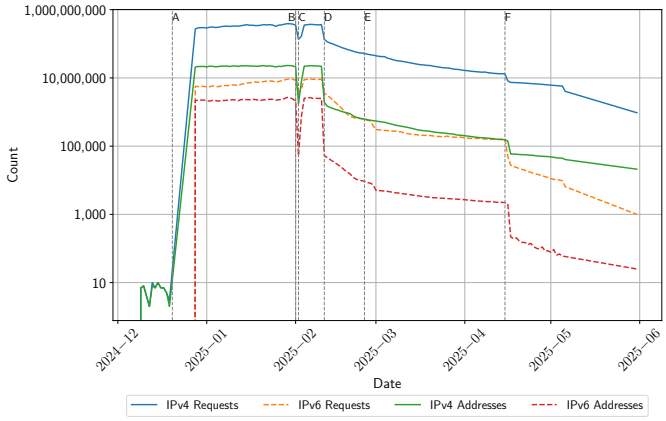


Fig. 10: Residual Traffic Experiment: Queries and unique addresses observed at a server per day after joining the pool (**A**), scheduling for removal (**B**), canceling the removal (**C**), scheduling for removal again (**D**), removal (**E**), and termination of the NTP daemon (**F**).

total number of IPv4/IPv6 NTP requests and number of unique IPv4/IPv6 clients in per-day bins.

Before adding our NTP server to the NTP Pool, it received some intermittent NTP scanning on IPv4. Several days after configuring our NTP server, we added it to the NTP Pool (time **A**). Our server quickly reached a steady state rate of IPv4 and IPv6 NTP requests (∼350M IPv4/7M IPv6 requests per day). At time **B**, we scheduled our server for removal from the NTP Pool via the NTP Pool's web interface. Although the deletion date was set four days in the future, it immediately began receiving fewer NTP requests. At time **C**, we canceled the impending deletion for our server. It soon returned to the steady state rate of NTP requests. We again scheduled our NTP server for deletion at time **D**, this time with a deletion date of two weeks in the future (time **E**). At time **E**, our server was no longer associated with the NTP Pool system, but continued to receive a reduced steady state number of NTP requests over the course of the next month (∼25M IPv4/200k IPv6). This is likely due to NTP clients that had obtained our server's address continuing to keep it cached for unexpectedly long periods of time. Because our server still served accurate time, these clients were unaware of the server's removal from the NTP Pool. Finally, at time **F**, we stopped our NTP daemon. This caused another significant decrease in the number of requests and clients.

Our results show that many clients will cache an NTP server's IP address and continue to use the server for time synchronization even months after removal from the NTP Pool. Continuing to provide time suffices for hundreds of thousands of clients to keep using our NTP server. This fact enables an attacker interested in e.g., skewing the time of their victims, to do so days or weeks after they are no longer being monitored by the NTP Pool's monitors.

C. Monopolization Attack

Finally, we consider the power of an informed adversary to execute a monopolization attack. Prior work also observed that some zones are well populated, while others contain few servers – as a result, an adversarial node could join and potentially "take over" these zones [4], [9]. In particular, Perry et al. empirically determined the number of NTP servers an attacker would need to contribute to five large pool zones (US, CA, UK, DE, and FR) to reach 50% of the total traffic for those zones [4]. In these zones, they discovered that an adversary need contribute 50-250 servers to reach the 50% threshold. In contrast, we use the computed netspeed values derived from the "counts" API endpoint (§III-D) to mathematically deduce the number of servers required to monopolize the traffic of *any* zone – without needing to add servers to the zone *a priori*.

More precisely, for a given country or zone, let n represent the current aggregate netspeed as gathered via the NTP Pool API and m be the maximum possible netspeed of any individual server; currently $m = 3000000$. Assuming an attack to capture a fraction f of total NTP queries, then the number of attack servers S required is:

$$S = \left\lceil \frac{nf}{m(1-f)} \right\rceil \tag{1}$$

We posit an adversary that wishes to receive at least half of the traffic for a particular country, i.e., contribute at least half of the aggregate netspeed. Further, we assume that the adversary sets their netspeed to the maximum possible speed (3Gbps). Figure 11 displays, across all country zones, the number of servers the adversary would require to capture half of the traffic. More than half of the countries would be compromised in this fashion by a single attacking server, while the next 40% of countries would require only 10 attack servers.

We conclude that the per-zone robustness to such attacks is relatively low for 90% of all countries. Further, an attacker with the ability to create IPv6 aliases can today effectively create an arbitrary number of servers to perform the traffic monopolization attack on any country.

D. Monopolization Attack in Practice

Last, we execute a limited version of the monopolization attack in practice to demonstrate its feasibility in the wild, as well as to validate our findings. Note that while we "attack" the zone, all of the servers we use in the experiment return good, valid time – hence, we did not disrupt the NTP Pool, its clients, or any host's notion of time. Instead, we demonstrate the ability to gain the preponderance of traffic within a zone via a capacity informed adversary.

We performed our experiment on August 5, 2025. We elected to target the `.hu` zone (Hungary) as it contained six IPv6 active (score ≥ 10) servers. To understand our effect on clients within the zone, we examine both the "counts" and the "answers" pool API endpoints (Table I). Recall that the "answers" endpoint reports the total count of times that each server was included in DNS answers – thereby allowing us to observe how the pool shifts traffic.

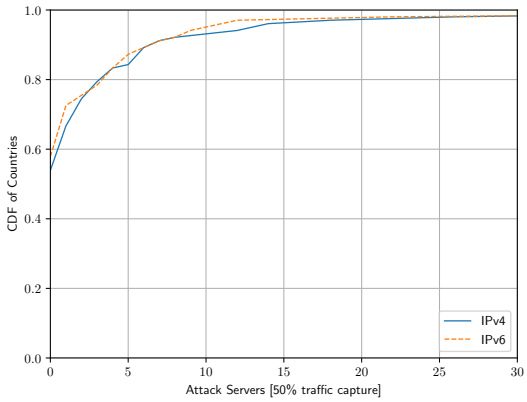


Fig. 11: Potential for traffic monopolization attack: per-country number of required servers to capture at least half of the zone's traffic. 90% of countries require 10 or fewer attacking servers to successfully execute the attack.

As a baseline, prior to the attack, we observe that the zone has a combined reported aggregate netspeed of 4.101Gbps. Among the six participating servers, four of the servers were apportioned for 24.4% of the DNS responses each (∼71,000 answers per hour), one server apportioned 2.4% (∼13,600 answers per hour) and one server apportioned less than 0.1% (134 responses per hour).

We then configured and added two IPv6 servers in the `.hu` zone to mimic a monopoly attack. Each attacking server was set to the maximum netspeed of 3Gbps. After approximately one day, we re-examined the distribution of netspeed and DNS answer counts. The two attack servers were each apportioned 29.7% of the netspeed and were included in ∼61,000 DNS answers per hour. The previous six servers went down to 9.9% for four (∼36,000 answers per hour), 1.0% for one (∼4,500 answers per hour), and almost 0% (45 answers per hour) for the last. By DNS answer count, the attack servers were each included in 23.4% of the total DNS answers within the period, for a total of 46.8%. Through manual investigation, we find that this answer fraction was slightly lower than expected as the two servers are also included in the global and continent zones, thereby lowering their overall contribution to the country zone. However, this small experiment validates the ability for a weak adversary to obtain a large fraction of the total country's pool query traffic with only minimal resources (two servers in-country). This problem is particularly acute in IPv6, as obtaining large quantities of IPv6 addresses to volunteer as NTP servers is trivial (some VPS providers assign as much as /64 prefixes to individual servers).

VII. Conclusions

g In this work, we take a fresh look at the NTP Pool – a volunteer driven and widely used NTP infrastructure, by gathering more complete and rich data than previously possible. By analyzing aliases, accounts, and network connectivity, we find that only approximately 20% of the participating servers are truly independent, and that the NTP Pool is less robust than previously believed. We then examine monopoly attacks, wherein the adversary captures the preponderance of NTP traffic in a particular country or region, and show that most zones in the pool are readily vulnerable to such attacks by a capacity informed adversary.

Our results suggest that the pool should consider longevity and reputation, as well as server independence, in the scoring algorithm. As a first step, the pool's backend DNS server selection algorithm could be modified to consider the account owner, protocol family, lifetime, and ASN in its decision process. We have shared our findings and recommendations with the NTP Pool operators.

VIII. Ethics

Our measurements require continued periodic polling of the NTP Pool website, including both an API end-point and per-server status web pages that we scrape. We reviewed our web scraping methodology with our institute's digital librarian who provided several guiding principles: 1. scrape only public data; 2. ensure data is not covered by copyright; 3. obey any terms of service; 4. obey any rules provided to scrapers via robots.txt; 5. use the minimal query load possible; 6. use APIs when available; and 7. do not redistribute the data. We followed these principles and note that the NTP Pool website is publicly available, does not contain any personally identifiable information or information on individuals and is not covered by copyright. Further, we follow the terms of service and the robots.txt rules, and the API for data available via an API.

The website primarily provides statistics and is decoupled from the operation of the pool's time service – i.e., if the website were to fail, the pool's DNS and NTP servers would continue to operate and provide time. We followed established best practices for ethical network measurements and minimized instantaneous load or any potential service impact on the website by querying for new servers on average once every 90 minutes and gathering updated statistics on existing servers only once a day, where we wait an average of 5 seconds between any two queries.

Because our findings have potential security implications for the pool, we do not make our complete dataset publicly available and have shared our findings with the NTP Pool project operators such that they can make informed decisions on improving the project's resilience.

Acknowledgment

The authors would like to thank Peter Bartoli, Giovane Moura, Georgios Smaragdakis, and the anonymous reviewers for invaluable constructive feedback. Thanks to Adam Shotland and Liam Carter for gathering and providing the BigQuery data.

REFERENCES

[1] J. Martin, J. Burbank, W. Kasch, and P. D. L. Mills, "Network Time Protocol Version 4: Protocol and Algorithms Specification," RFC 5905, Jun. 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc5905

[2] A. Malhotra, I. E. Cohen, E. Brakke, and S. Goldberg, "Attacking the Network Time Protocol," *Cryptology ePrint Archive*, 2015.

[3] J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, and M. Karir, "Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks," in *ACM Internet Measurement Conference (IMC)*, 2014.

[4] Y. Perry, N. R. Schiff, and M. Schapira, "A Devil of a Time: How Vulnerable is NTP to Malicious Timeservers?" in *Network and Distributed System Security Symposium (NDSS)*, 2021.

[5] D. F. Franke, D. Sibold, K. Teichel, M. Dansarie, and R. Sundblad, "Network Time Security for the Network Time Protocol," RFC 8915, Sep. 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8915

[6] "How the Windows Time Service Works," 2025, https://learn.microsoft.com/en-us/windows-server/networking/windows-time-service/how-the-windows-time-service-works.

[7] "Google Public NTP," 2025, https://developers.google.com/time.

[8] A. B. Hansen, "NTP Pool," 2025, https://www.ntppool.org.

[9] G. C. Moura, M. Davids, C. Schutijser, C. Hesselman, J. Heidemann, and G. Smaragdakis, "Deep Dive into NTP Pool's Popularity and Mapping," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 8, no. 1, pp. 1–30, 2024.

[10] J. Kwon, J. Song, J. Hur, and A. Perrig, "Did the Shark Eat the Watchdog in the NTP Pool? Deceiving the NTP Pool's Monitoring System," in *USENIX Security Symposium*, 2023.

[11] T. Rytilahti, D. Tatang, J. Köpper, and T. Holz, "Masters of Time: An Overview of the NTP Ecosystem," in *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2018.

[12] D. Mills, "Network Time Protocol (NTP)," RFC 958, Internet Engineering Task Force, Sep. 1985. [Online]. Available: http://www.ietf.org/rfc/rfc958.txt

[13] D. L. Mills, "A Brief History of NTP Time: Memoirs of an Internet Timekeeper," *ACM SIGCOMM Computer Communication Review (CCR)*, 2003.

[14] A. N. Novick and M. A. Lombardi, "Practical Limitations of NTP Time Transfer," in *IEEE International Frequency Control Symposium & The European Frequency and Time Forum*, 2015.

[15] R. Durairajan, S. K. Mani, J. Sommers, and P. Barford, "Time's Forgotten: Using NTP to Understand Internet Latency," in *Workshop on Hot Topics in Networks (HotNets)*, 2015.

[16] D. L. Mills, *Computer Network Time Synchronization: The Network Time Protocol on Earth and in Space*. CRC Press, 2017.

[17] R. Annessi, J. Fabini, and T. Zseby, "It's About Time: Securing Broadcast Time Synchronization with Data Origin Authentication," in *International Conference on Computer Communication and Networks (ICCCN)*, 2017.

[18] O. Deutsch, N. R. Schiff, D. Dolev, and M. Schapira, "Preventing (Network) Time Travel with Chronos," in *Network and Distributed System Security Symposium (NDSS)*, 2018.

[19] L. Zhang, D. Choffnes, D. Levin, T. Dumitraş, A. Mislove, A. Schulman, and C. Wilson, "Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed," in *ACM Internet Measurement Conference (IMC)*, 2014.

[20] J. Kohl, C. Neuman *et al.*, "The Kerberos Network Authentication Service (V5)," RFC 1510, September, Tech. Rep., 1993.

[21] C. Rossow, "Amplification Hell: Revisiting Network Protocols for DDoS Abuse," in *Network and Distributed System Security Symposium (NDSS)*, 2014.

[22] L. Rudman and B. Irwin, "Characterization and Analysis of NTP Amplification Based DDoS Attacks," in *Information Security for South Africa (ISSA)*. IEEE, 2015.

[23] M. Jonker, A. King, J. Krupp, C. Rossow, A. Sperotto, and A. Dainotti, "Millions of Targets Under Attack: A Macroscopic Characterization of the DoS Ecosystem," in *ACM Internet Measurement Conference (IMC)*, 2017.

[24] D. Kopp, C. Dietzel, and O. Hohlfeld, "DDoS Never Dies? An IXP Perspective on DDoS Amplification Attacks," in *Passive and Active Network Measurement Conference (PAM)*, 2021.

[25] M. Luckie, R. Beverly, R. Koga, K. Keys, J. A. Kroll, and k. claffy, "Network Hygiene, Incentives, and Regulation: Deployment of Source Address Validation in the Internet," in *ACM Internet Measurement Conference (IMC)*, Nov. 2019.

[26] U. C. . I. S. A. (CISA), "NTP Amplification Attacks Using CVE-2013-5211," https://www.cisa.gov/news-events/alerts/2014/01/13/ntp-amplification-attacks-using-cve-2013-5211.

[27] J. Song, J. Kwon, and J. Hur, "Analysis of NTP Pool Monitoring System Based on Multiple Monitoring Stations," in *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2022.

[28] R. Durairajan, S. K. Mani, P. Barford, R. Nowak, and J. Sommers, "Timeweaver: Opportunistic One Way Delay Measurement via NTP," in *International Teletraffic Congress (ITC)*, 2018.

[29] E. Rye and D. Levin, "IPv6 Hitlists at Scale: Be Careful What You Wish For," in *ACM SIGCOMM*, 2023.

[30] M. Syamkumar, S. K. Mani, R. Durairajan, P. Barford, and J. Sommers, "Wrinkles in Time: Detecting Internet-Wide Events via NTP," in *IFIP Networking Conference (IFIP Networking) and Workshops*, 2018.

[31] "shodan.io Actively Infiltrating ntp.org IPv6 Pools for Scanning Purposes ," 2025, https://seclists.org/oss-sec/2016/q1/219.

[32] B. Hein, "The Rising Sophistication of Network Scanning," 2025, https://netpatterns.blogspot.com/2016/01/the-rising-sophistication-of-network.html.

[33] "Google BigQuery," 2025, https://console.cloud.google.com/bigquery?project=ntppool.

[34] "NTP Pool Monitoring Logs," 2023, https://news.ntppool.org/docs/logscores/.

[35] "NTP Pool Server Management Page," 2025, https://manage.ntppool.org/manage/servers.

[36] B. Haberman, "Control Messages Protocol for Use with Network Time Protocol Version 4," RFC 9327, Nov. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9327

[37] M. Lichvar, "ntpdedup," 2017, https://github.com/mlichvar/ntpdedup.

[38] ntp.org, "How do I join," 2025, https://www.ntppool.org/en/join.html.

[39] IANA, "Network Time Protocol (NTP) Parameters," 2025, https://www.iana.org/assignments/ntp-parameters/ntp-parameters.xhtml.

[40] R. Hendriks, M. Luckie, M. Jonker, R. Sommese, and R. van Rijswijk-Deij, "Manycast reloaded: a tool for an open, fast, responsible and efficient daily anycast census," 2025. [Online]. Available: https://arxiv.org/abs/2503.20554

[41] "Why is Cloudflare in the pool?" 2022, https://community.ntppool.org/t/why-is-cloudflare-in-the-pool/2455/20.

[42] Routeviews, "University of Oregon Route Views Project," 2025, http://www.routeviews.org/routeviews/.

[43] "IPinfo – The Trusted Source For IP Address Data," 2025, https://ipinfo.io.

[44] Hurricane Electric, "IPv6 Tunnel Broker," 2025, https://tunnelbroker.net/.