Measuring the Deployment of Local Root in DNS Recursive Resolvers

RAZVAN STEFAN, University of Twente, The Netherlands

The Domain Name System (DNS) is a critical component of the Internet, which translates human-readable domain names into IP addresses. At the top of this hierarchical system, there are the root servers, which handle billions of queries per day and serve as the starting point for DNS resolution processes. To reduce dependence on these root servers and improve performance and resilience, *RFC 8806* proposes the use of local copies of the root zone, known as *local root*, on recursive resolvers.

Despite the potential benefits, it remains unclear to what extent this feature has been adopted in practice. In this paper, we investigate the possible deployment of locally served root zones. To evaluate the impact of this setup, we first conducted local testing using DNS resolver software, namely BIND and Unbound. We observed a significant decrease in root query response times; when using Unbound, the average dropped from 130 milliseconds per query to as low as 1.5 ms, when a local root zone was in use. These results show the benefits of local root configurations under controlled conditions.

We then analyzed data from 1,000 RIPE Atlas probes and identified around 40 probes that rely on resolvers with DNS resolution patterns that suggest they may be configured with local root. These probes showed response times that were likely too fast to involve querying the root servers. However, due to factors like aggressive caching or infrastructure configurations, we cannot confirm with certainty that these probes were using a local root zone.

These findings highlight the performance increase potential of local root deployments in controlled environments and the difficulty of identifying their presence in the wild based solely on observational data.

Additional Key Words and Phrases: DNS, recursive resolvers, root zone, local root, RFC 8806, RIPE Atlas, resolver deployment, performance measurement

1 Introduction

The Domain Name System (DNS) is a critical component of Internet infrastructure, in charge with translating human-readable domain names into IP addresses. This system has a hierarchical structure that begins with the DNS root. When a recursive resolver can't answer a query from its cache, it starts the resolution process by contacting a root server, which points the resolver to the correct top-level domain (TLD) name server (e.g., .com, .org, .net).

The root server system consists of 13 named authorities (labeled A through M), operated by independent organizations and distributed globally via Anycast [18]. This architecture ensures availability and resilience, even under high traffic or denial-of-service attacks. However, the need for recursive resolvers to constantly contact these root servers can still cause high latency and dependency on external infrastructure.

Recursive resolvers are typically operated by ISPs (Internet Service Providers), cloud providers, enterprises, or public DNS services like Google [6] and Cloudflare [3]. They play a key role in DNS performance and reliability. To improve efficiency and decrease dependence on root servers, resolvers can be configured with a publicly available local copy of the root zone maintained by IANA (Internet Assigned Numbers Authority) [9] standardized in *RFC 8806* [8]. This setup, also known as "local root", allows resolvers to respond to root queries directly, removing the need to contact external root servers. The expected benefits include lower RTTs (round-trip-time), reduced DNS traffic, and greater resilience during root server outages.

Despite these advantages, the adoption of local root zone configurations remains largely unmeasured. It is unclear how many resolvers currently implement this standard and how exactly this configuration impacts overall query latency.

This study aims to answer this question by measuring the deployment of local root configurations with a three-phase methodology. First, controlled experiments are conducted using a virtual testbed to compare the behavior of BIND and Unbound resolvers with and without a local root zone. These experiments measure latency, caching effects, and resilience during simulated network failures. Second, the study uses the RIPE Atlas platform [17] to perform global DNS measurements from thousands of geographically diverse probes. These measurements are designed to determine whether recursive resolvers are using local root zones based on patterns observed in the first phase. Finally, traffic logs from the B-root server are analyzed to validate whether resolvers observed in the RIPE Atlas measurements actually contacted the root system, or potentially relied on local root data.

2 Problem Statement

While extensive research has been conducted on DNS performance, the deployment and effects of the local root configuration, as defined in *RFC 8806*, remain unclear. Therefore, this paper will first attempt to set up and evaluate a testbed environment that makes use of the local root standard with the goal of understanding the effects on performance and resilience on the DNS. Secondly, this study will investigate whether resolvers in the wild actually use local root by analysing metrics such as RTT and traffic logs of root servers.

2.1 Research Questions

The following research question results from the problem statement:

> What is the current adoption rate of the local root configuration among DNS recursive resolvers?

This question can be answered with the help of the following sub-questions:

- How does the implementation of a local root configuration affect DNS resolution performance metrics (e.g., RTT, query success rate) in a controlled testbed environment?
- To what extent can measurement platforms, such as RIPE Atlas, effectively detect the use of local root configurations in recursive resolvers?

3 Related Work

Direct studies on the deployment of local root in recursive resolvers have not been conducted yet, but there is a substantial amount of work done in the topic of DNS, specifically about caching behavior and robustness of the DNS architecture. The references cited in

Author's address: Razvan Stefan, r.stefan@student.utwente.nl, University of Twente, Enschede, The Netherlands.

this paper are selected examples that support and illustrate these broader trends in DNS research.

3.1 Using RFC 7706 to Enhance Privacy

A notable study that highlights the benefits of local root configurations is Hardaker's analysis of privacy in the DNS Root Service [7]. He proposes local root deployment, as described in *RFC 7706* [19], the precursor of *RFC 8806*, as a privacy enhancing technique. His findings indicate that enabling a local root setup significantly reduces traffic to the B-Root server and mitigates all identified privacy attacks by eliminating the need to contact root servers during routine query processing.

3.2 Techniques to Enhance Resolver Robustness

Two techniques with similar goals as local root deployments are *serve-stale* and *aggressive NSEC caching*. Serve-stale, described in *RFC 8767* [11], allows resolvers to return stale cached responses when authoritative servers are unavailable, making it possible for DNS resolution to work when root servers are not available. Aggressive NSEC caching, as defined in *RFC 8198* [5], enables resolvers to generate negative responses from cached NSEC/NSEC3 records, decreasing the number of queries to the root zone.

The 2025 study by Deccio and Tessem [4] explored how DNS resolvers can leverage DNSSEC's NSEC and NSEC3 records to aggressively cache negative responses. This significantly reduces unnecessary queries to authoritative servers, cuts latency, decreases resolver-authoritative traffic, and enhances overall DNS efficiency and security. Their findings indicate that approximately half of the observed resolvers have this feature enabled.

Resolvers that implement aggressive NSEC caching have a similar behavior to those using a local root when queried for non-existent TLDs, as both can respond with local data, instead of querying the root server. This similarity in behavior underscores the importance of the methodology used in this study: in order to distinguish between these two techniques and accurately assess resolver behavior, it was crucial to query valid TLDs also, not only non-existent TLDs.

3.3 DNS Caching

In their 2019 study, Moura et al. [14] investigated how DNS caching behavior, particularly the time-to-live (TTL), impacts query performance and load on authoritative servers. Using real-world DNS data from multiple TLD operators, they experimented with different TTL configurations and observed how cache hit rates and latency were affected. Their analysis showed that increasing TTLs can significantly reduce DNS query latency. For example, in one country-code TLD scenario, median latency dropped from 183 ms to 28.7 ms with longer caching durations. They also demonstrated that longer TTLs substantially decrease query volume to authoritative servers, improving both scalability and robustness of the DNS infrastructure.

3.4 Root Server Traffic

Liang et al. [12] measured query latency of top level DNS Servers in 2013. They used DNS probes to issue queries to root and TLD servers, measuring RTTs across different locations and network conditions. The study revealed that the median latency to root servers was relatively low, typically under 20 ms for most clients, due to Anycast routing. However, they also found significant geographic and ISP-related disparities, with some regions experiencing latencies several times higher than others.

3.5 Security Threats to the DNS

Alieyan et al. [2] reviewed a wide range of DNS-targeted DDoS attacks, including reflection and amplification attacks. Such attacks can have a massive impact on DNS services, potentially making Internet resources inaccessible. This threat can be lowered by local root configurations because they allow recursive resolvers to operate independently of root server availability.

3.6 Root Infrastructure Resilience

Moura et al. [13] examined the resilience of the root DNS infrastructure during the large-scale DDoS attack of November 2015. Their analysis focused on the behavior of Anycast routing, a technique widely used by root server operators, under high-stress conditions. While Anycast effectively distributed query loads, it also led to regional disparities in success rates and, in some cases, collateral damage to other services.

3.7 Implications for Local Root Deployment

Together, these studies highlight the difficulty of maintaining the DNS infrastructure and the possible advantages offered by local root configurations. If root zone data is served locally, recursive resolvers can minimize queries and reduce reliance on the global root system, increasing performance and robustness. In particular, local roots serve as a mechanism that can continue operating during large scale attacks or outages, making them a good strategy for improving DNS infrastructure.

4 Methodology

This study followed a three-phase methodology that combined local testbed experiments, global Internet measurements using RIPE Atlas, and analysis of real DNS root server traffic logs. This approach was used to evaluate both the technical behavior and practical adoption of local root setups in recursive resolvers.

4.1 Local Testbed Experimentation

The first phase involved setting up a controlled environment to observe the behavior of recursive resolvers with and without a local root configuration. A virtual testbed was created using *VirtualBox* [16], hosting resolver instances configured in two modes: a standard setup that queries real root servers directly, and the configuration with the local root zone.

The resolver software used was BIND 9.18 [10] and Unbound 1.13 [15]. These two commonly used open-source resolvers simulated a realistic recursive resolver environment. Using the `dig` command, a list of 140 real and 40 non-existent TLDs were queried for various available records (e.g A, NS, SOA). Various record types were queried to verify if there are any substantial differences in how they are resolved. Performance metrics, such as query latency and response codes (RCODEs), were collected to analyse the impact of the local root zone on performance and availability. At first, the cache was cleared to make sure it doesn't impact the results [1]. Afterwards, the effect of caching was tested by forcing data in the resolver cache and querying for it again. This test made it possible to compare how local root compares to traditional caching for performance and reliance on root servers.

This local environment also made it possible to test under edge conditions. A root server outage was simulated by blocking outgoing traffic on port 53 using firewall rules [2], the default port used by DNS resolvers. This made it possible to see whether the resolvers could still respond to queries only using local root data, not being able to rely on the root servers.

4.2 Internet-Wide Measurements Using RIPE Atlas

In the second phase, the study moved from the controlled environment to the open Internet using the RIPE Atlas measurement platform. RIPE Atlas provides a global network of thousands of vantage points, called "probes", distributed across a wide range of autonomous systems, ISPs, and geographic locations.

Measurements were created, using these probes' local resolvers to send DNS queries. Based on the patterns learned in the first part of the study, the goal was to select the probes that relied on recursive resolvers which showed patterns found while using local root configurations. Specifically, queries targeted obscure country-code top-level domains (ccTLDs) to reduce the likelihood of cached responses and better reflect resolver behavior.

Initial measurements used RIPE Atlas's random probe selection to ensure broad global distribution. From this measurements, probes with very low latency, consistent with the values seen in the local testing, were flagged for further testing. Our assumption is that probes that respond with low RTT for the requested domain name would be candidates for using local root.

However, not all probes returned correct results for the issued queries. Some probes returned incorrect `NXDOMAIN` responses for valid ccTLDs. `NXDOMAIN` is a DNS response indicating that the domain name queried does not exist. These incorrect responses may have been due to stale data. Such probes were excluded from further analysis to avoid skewed results.

To validate the remaining probes, additional queries were issued for obscure and fake TLDs to see how resolvers handled negative responses. This additional queries were performed to make sure previous answers were not just cached. By consistently resolving TLDs quickly, confidence grew in these resolvers to be configured with local root. Additionally, non-existent TLDs are very unlikely to be cached, therefore, answering such TLDs fast could be possible with the use of local root.

Another test was performed to observe how these probes handle caching. The domain *google.com* was queried on each probe to force the response into the cache. At a two minute interval, the domain was queried again to measure the RTT. Typically, an entry like *google.com* would have a TTL of five minutes in the resolver's cache, therefore the second query should not have required the resolver to contact any other server. This RTT was then considered as a baseline to compare against RTTs from earlier uncached TLD queries. In the case that the RTT for the uncached TLD query would be very close to the baseline value, it would be highly unlikely for the resolver to also contact the root server, meaning it would respond with locally available data.

4.3 Access to Root Server Traffic Logs

In the final phase, data collected via RIPE Atlas was checked further with traffic logs from the B-root server. By knowing the resolver IP addresses (identified via RIPE Atlas), query arguments and the time at which these queries were sent, it was possible, with support of the operators of B-root, to look at the traffic logs and try to find whether specific resolvers actually contacted the root server to solve the probe request. Our assumption is that a resolver that uses local root would only contact root servers, such as the B-root server, only for AXFR requests, which is a protocol for zone transfers. Therefore, if any other queries would be observed in the logs, it would be a clear indication that the specific resolver does not implement local root.

Unfortunately, log data was granted only for the B-root server, thus limiting the coverage of this test. It is possible that resolvers contacted any of the other 12 root servers, which could not be confirmed from the available data. Additionally, other factors, such as packet loss or network failure, could have occurred, limiting the completeness of this phase.

5 Results

This section presents the results obtained from local testbed experiments and RIPE Atlas measurements. The primary goal was to measure the deployment of the local root standard in resolvers in the wild. To understand how to distinguish local root resolvers, another objective was to evaluate the behavior of different DNS resolvers when configured with local root.

5.1 Local Testbed

To learn how local root changes resolver behavior, a controlled environment was created, so experiments could be conducted on a local testbed. The goal was to measure how Unbound and BIND handle queries when configured with and without a local root zone and to compare their performance, resilience to network failures, and handling of invalid TLDs. All of the tests were performed in the Netherlands, on a home network. The exact configurations to enable local root on both Unbound and BIND can be found in the Appendix A .

5.1.1 Unbound

The first resolver tested was Unbound. The aim here was to determine how Unbound behaves under different configurations, namely default and local root, and whether it could successfully resolve DNS queries without external root server contact.

With its default configuration, Unbound was tested by querying for various available records for 140 real TLDs and 40 fake ones. The cache was flushed just before running this test to make sure it doesn't skew the results. The average response time was approximately 161 ms per query for real TLDs and only 26 ms for fake TLDs, as shown in Figure 1. There was no clear performance difference

[1] cache clearing was achieved using the command `unbound-control flush_zone .` for Unbound and `rndc flush` for BIND
[2] the command to add this rule to the firewall is `sudo ufw deny out to any port 53`

between different record types. This confirmed that Unbound, by default, performs recursive resolution via the global root server system.

When outbound traffic on port 53 (used for DNS queries) was blocked, Unbound failed to resolve any new queries, although previously cached entries remained accessible with response times dropping to 1–2 ms.

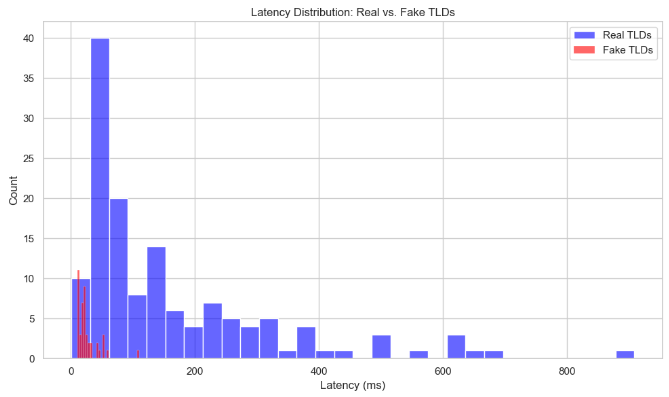

Fig. 1. Histogram of query times using standard configuration in Unbound

Next, Unbound was configured to use a local root zone, using the configuration suggested in the RFC 8806 document. The resolver had servers *b, c, d, f, g, and k* as master and their data was stored locally in a `root.zone` file. Fallback was enabled, meaning that the resolver could contact any of the mentioned root servers in case of an unexpected failure.

With this configuration, the same test had average response times of 1.5 ms for both real and fake TLDs, similar to cached results. These results can also be seen in Figure 2. Traffic was monitored using the `tcpdump` command on port 53. This monitoring showed no outgoing DNS traffic, confirming that all queries were resolved locally. After this, outgoing traffic on port 53 was blocked using firewall rules. The resolver was still able to resolve non-cached TLDs, highlighting the resilience of local root.

An additional test targeted second-level domains (SLDs) under various TLDs. The local root setup was again significantly faster, with an average response time of 132 ms, while the standard configuration took on average 190 ms. This performance gain can be seen in Figure 3. Although these insights were not used in the later phases of this paper, they again show that using local root has a performance impact in the complete resolution process.

In summary, Unbound configured with local root performs significantly faster and continues to function, even when network access to root servers is blocked, demonstrating the benefits of local root deployment.

5.1.2 BIND

The second resolver tested was BIND (Berkeley Internet Name Domain), using the same methodology. With the standard configuration, BIND's average query time was around 154 ms for real TLDs. It is interesting to note that BIND resolved all fake TLDs immediately,

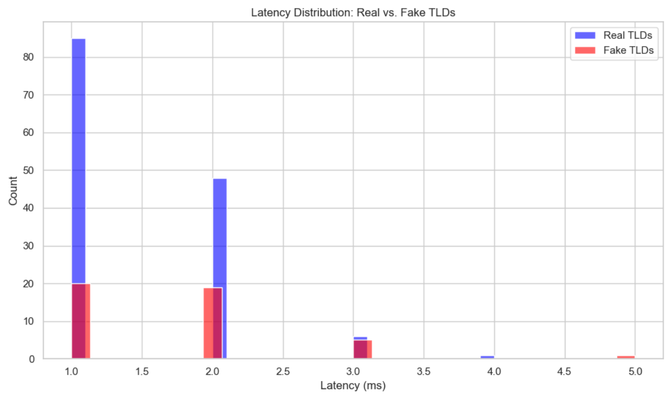

Fig. 2. Histogram of query times using standard configuration in Unbound

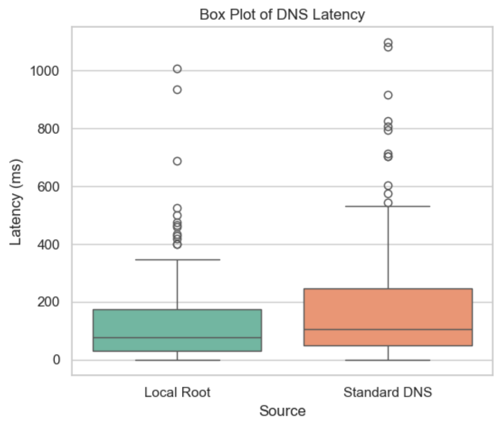

Fig. 3. Boxplot comparing query times on SLDs using standard and local root configuration

unlike Unbound. The results of these measurements can be observed in Figures 4 and 5. As expected, blocking port 53 prevented BIND from resolving real TLDs.

When configuring BIND with local root, two approaches were used:

1. Mirror Configuration

BIND was configured to mirror the root zone using the `mirror` zone type, following the configuration suggested in the *RFC 8806* document. However, this did not result in the expected performance improvements. Query times staid high, and `tcpdump` showed that the resolver still contacted root servers. Blocking port 53 made the resolver unable to resolve real TLDs, but fake TLDs continued to resolve instantly. Consultation with BIND maintainers revealed that BIND may still query upstream for SOA (Start of Authority) validation or other metadata, even when root zone data is available locally.

2. Master Configuration

In this configuration, the root zone file was manually downloaded and loaded using the `master` zone type. While this resulted in fast

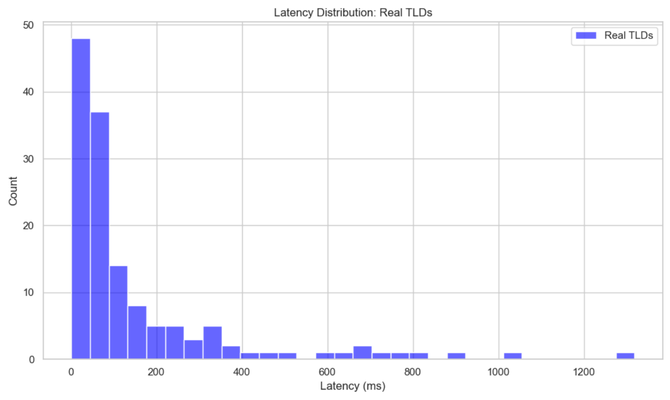

Fig. 4. Histogram of query times for real TLDs using standard configuration in BIND

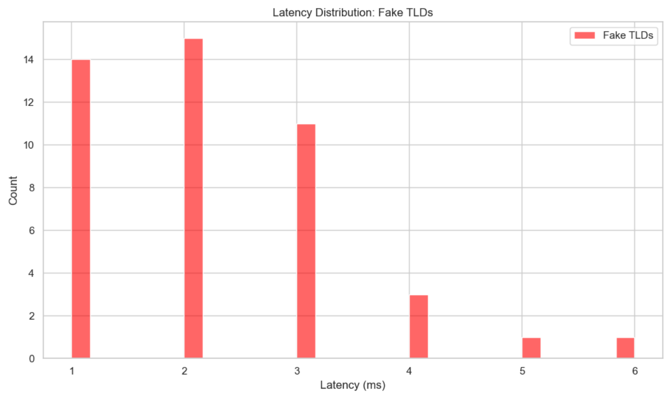

Fig. 5. Histogram of query times for fake TLDs using standard configuration in BIND

query times of approximately 1.3 ms, comparable to the results observed while using Unbound, the behavior of the resolver fundamentally changed. By loading the root zone as a master zone, BIND acted as an authoritative server for the root zone (.). As a result, it would not attempt to resolve further delegations and returned only information present in the zone file. To resolve queries, the RD (Recursion Desired) bit had to be unset (e.g., using the `+norecurse` flag in the `dig` command), which is not normal for standard client requests. This configuration is therefore not practical for real-world use.

Although the results observed in BIND were not as clear as the ones of Unbound, it still brought valuable insights for the local root behavior, especially for resolution of fake TLDs. The various tasks resolved with different configurations are summarized in the table 1.

5.2 RIPE Atlas Results

Based on the insights gained during local resting, RIPE Atlas measurements focused on detecting resolvers that returned results with very low latency, under 5 milliseconds. Such a quick response time would likely be too fast to involve querying root servers.

It is important to note that RIPE Atlas only provides observational data, such as response times and the target of each DNS query, without access to resolver configurations. Because of this, it is not possible to distinguish between standard and local root configurations in the case of high-latency responses. A resolver with high latency could still be using a local root but be experiencing delays due to network conditions or other factors, making it look like a standard resolver. As a result, the analysis concentrated on identifying probes with consistently low-latency responses across a range of obscure TLDs, where the likelihood of cache hits is minimal.

Approximately 1000 globally distributed probes were selected at random. Each was tasked with resolving obscure country-code TLDs (e.g., San Marino, Solomon Islands, Guinea-Bissau), which were unlikely to be cached. Initial filtering yielded a list of promising probes, which were further refined through repeated tests using different TLDs. Each promising probe passed through 6-10 different measurements, responding to each query in under 5 ms. Each measurement used a different TLD to avoid caching. Additionally, some TLDs were repeated at an interval large enough for the TTL to expire in order to see if the RTT remains consistent.

By the end of the measurements, 40 probes consistently responded under the 5 ms threshold across all tests. The list of probes, including the resolver organization on which they rely, and the list of measurement IDs can be found in Appendix B.

These probes were then tested to observe their caching behavior. Each resolver was primed with *google.com*, and then queried again shortly after. Most probes returned *google.com* with response times similar to those of the obscure TLDs, as seen in Figure 6. A few outliers had slightly slower responses, possibly due to TTL expiration or switching between resolvers. By querying for a cached entry, we could observe the baseline RTT for a probe to receive a response from its resolver. The resolvers of the probes that potentially use local root had very similar times as this baseline RTT, for obscure TLDs, that were probably not cached, therefore, it is unlikely that these resolvers contacted a root server.

Although the random probe selection feature provided by RIPE Atlas was useful in selecting diverse probes, the selection focused on country spread rather than true geographic dispersion. Moreover, randomly selected probes could belong to the same ASN and therefore are likely to use the same resolver, thus sharing cached responses, introducing bias and reducing the independence of observations. To address this, the entire procedure was repeated using a script to select geographically dispersed probes, choosing 3–5 probes per region defined by a 10-degree latitude by 10-degree longitude grid, each associated with a unique ASN. This sampling strategy aimed to minimize both geographic and network bias. The results were very similar to those obtained using the original method, reinforcing the earlier findings: a small subset of resolvers consistently returned low-latency responses (under 5 ms) for obscure TLDs, regardless of location or network. These consistent results indicate a higher probability that a subset of resolvers in the wild actually use local root.

Table 1. Summary of Resolver Behavior Across Different Tasks and Configurations

Resolver	Task	Type	Result	Response Time
Unbound	Dig common TLD	Local root	Success	1.5 ms
	Dig fake TLD	Local root	Success	1.5 ms
	Dig cached TLD	Local root	Success	1-1.5 ms
	Dig with blocked port	Local root	Success	1.5 ms
	Dig common TLD	Standard	Success	130 ms
	Dig fake TLD	Standard	Success	30 ms
	Dig cached TLD	Standard	Success	1-2 ms
	Dig with blocked port	Standard	Fail	—
BIND	Dig common TLD	Local root - Mirror	Success	120 ms
	Dig fake TLD	Local root - Mirror	Success	1-2 ms
	Dig cached TLD	Local root - Mirror	Success	1-1.5 ms
	Dig with blocked port	Local root - Mirror	Fail	-
	Dig common TLD	Local root - Master	Success	1.3 ms
	Dig fake TLD	Local root - Master	Success	1-2 ms
	Dig cached TLD	Local root - Master	Success	1-1.5 ms
	Dig with blocked port	Local root - Master	Success	1-2 ms
	Dig common TLD	Standard	Success	120 ms
	Dig fake TLD	Standard	Success	1–2 ms
	Dig cached TLD	Standard	Success	1–1.5 ms
	Dig with blocked port	Standard	Fail	—

Altogether, the RIPE Atlas measurements suggest that a small number of recursive resolvers may be using a local root configuration. The consistent sub-5 ms response times for obscure, likely uncached TLDs point toward resolution mechanisms that do not involve querying the global root system. By also having non-existent TLDs, that are very unlikely to be cached, be answered so quickly, the likelihood of just optimized caching techniques decreased significantly. Therefore, the carried out measurements provide a strong indication that local root usage, though limited in deployment, is already present in the DNS ecosystem.

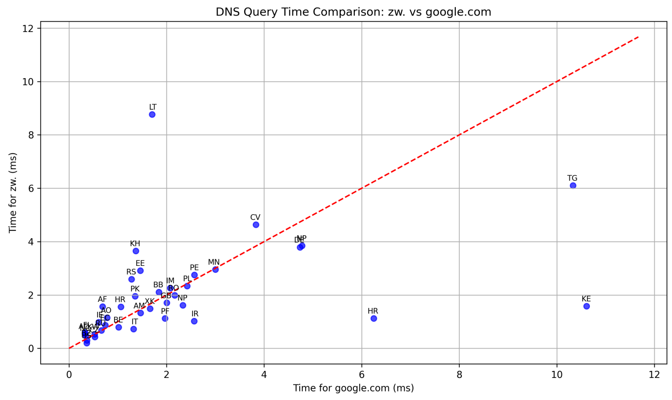

Fig. 6. Scatter plot comparing response times between the cached google.com and the zw. TLD

5.3 Root Server Traffic Logs

To correlate probe activity with real-world root server traffic, resolver IP addresses were extracted using the `whoami.akamai.net` query. `whoami.akamai.net` is a special DNS hostname provided by Akamai [1], and it's commonly used to help determine the public IP address of the machine that made the request. This allowed identification of the public IPs of each probe's recursive resolver. Most observed resolvers were operated by Google or Cloudflare, though some probes relied on multiple resolvers.

Unfortunately, neither Google nor Cloudflare explicitly state that they use local root. However, in a blog post about a major DNS outage on October 4th, 2023, Cloudflare covered a technique they implement similar to the one described in RFC 8806 [20]. Specifically, they mentioned that their DNS infrastructure automatically retrieves and serves a local copy of the root zone twice, daily. This strategy enables their resolvers to continue answering queries for root zone data even if the actual root servers become unreachable.

With the resolver IP, timestamp, and query domain in hand, it became possible to search for corresponding events in root server traffic logs, specifically those of the B-root server, for which access was granted.

None of the expected resolver queries were observed in the B-root logs. However, this absence does not conclusively indicate that the resolvers avoided the root system with a local root zone. Queries could have been directed to any of the other twelve root servers, or blocked, filtered, or dropped in transit. Even full visibility across all root logs would not fully eliminate these uncertainties.

Nonetheless, this methodology could be used as a foundation for broader studies. During measurement events like *DITL*, when data is available from more root servers, such a methodology could be used for a better understanding of resolver behavior and potential local root adoption.

6 Conclusion

This study set out to evaluate the adoption and impact of locally served DNS root zones among recursive resolvers, a technique proposed to enhance DNS performance, increase resilience, and reduce reliance on external root infrastructure. By combining controlled experiments with real-world measurements, we examined how resolvers behave when configured to operate with a local copy of the root zone and tried to determine the deployment rate of this standard in recursive resolvers in the wild.

The controlled testbed experiments were crucial for understanding how different resolver software, namely BIND and Unbound, implement local root configurations. These tests highlighted clear differences in behavior, particularly in query response times and handling of non-existent TLDs. These findings made it possible to interpret the behavior of public resolvers observed via RIPE Atlas.

However, the Internet-wide measurements had more ambiguous results. Several resolvers had consistent low-latency responses to unlikely to be cached TLDs, possibly due to the use of local root, but this results are not conclusive. Similar results could be due to aggressive caching or other configurations than local root. Additionally, resolvers that showed high latencies could not be properly examined. Although the local tests with BIND showed that even local root resolvers can have higher latencies, it would not have been possible to distinguish such resolvers based solely on the available observational data. Furthermore, because recursive resolvers typically do not disclose configuration details, external verification remains difficult.

To conclude, while our observations suggest that some resolvers may be using a local root zone, definitive proof remains elusive. Our findings highlight both the potential of local root configurations to increase DNS performance and resilience and the challenges in detecting and validating their adoption in the wild.

7 Future Work

To more accurately assess the adoption of locally served DNS root zones, future research could focus on two key directions:

7.1 Day-in-the-Life of the Internet

Day-in-the-Life (DITL) of the Internet is an annual event organised by DNS-OARC (DNS Operations, Analysis, and Research Center) in which DNS traffic is captured at various points in the infrastructure. This traffic is then anonymised and made available to OARC members. This event would be an opportunity to test the presence of local root through a series of queries, similar to the ones conducted in this paper, especially since most root servers participate in this event, thus increasing the coverage. Unfortunately, this event took place this year before beginning this paper, so it was not possible to take advantage of it.

Alternatively, cooperating with authoritative root server could also be a way to increase the visibility. Either of the mentioned approaches could provide a more complete picture of resolver behavior and local root adoption.

7.2 Collaboration with Resolver Operators

Collaborating with public DNS providers and ISPs would make it possible to have ground truth about real-world resolver configurations. These operators could confirm if their infrastructure uses local root, and under what conditions. Such collaboration would help validate and interpret external measurements, making it more clear if local root is actually used in the real-world.

8 Acknowledgements

We would like to express our sincere gratitude to Wes Hardaker for granting us access to the B-root server logs and for assisting in their examination. Their support was crucial for the analysis presented in this work.

9 AI Statement

This paper makes limited use of ChatGPT for some tasks. ChatGPT was used to generate lists of fake TLDs and SLDs, used in the first phase experiments, and to help improve the language and clarity of the text. All code, experiments, data collection, analysis, and original content were developed and written by me.

References

[1] Akamai Technologies. [n. d.]. whoami.akamai.net Service. https://www.akamai.com/blog/developers/introducing-new-whoami-tool-dns-resolver-information.

[2] Kamal Alieyan, Mohammed M. Kadhum, Mohammed Anbar, Shafiq Ul Rehman, and Naser K. A. Alajmi. 2016. An overview of DDoS attacks based on DNS. In *2016 International Conference on Information and Communication Technology Convergence (ICTC)*. 276–280. https://doi.org/10.1109/ICTC.2016.7763485

[3] Cloudflare. [n. d.]. https://www.cloudflare.com/learning/dns/what-is-1.1.1.1/.

[4] Casey Deccio and Benjamin Tessem. 2025. Aggressive Negative Caching in DNS Resolvers. In *Passive and Active Measurement Conference (PAM)*. https://tma.ifip.org/2025/wp-content/uploads/sites/14/2025/06/tma2025_paper26.pdf

[5] Kazunori Fujiwara, Akira Kato, and Warren Kumari. 2017. Aggressive Use of DNSSEC-Validated Cache. RFC 8198. https://www.rfc-editor.org/rfc/rfc8198

[6] Google Public DNS. [n. d.]. https://developers.google.com/speed/public-dns.

[7] Wes Hardaker. 2018. Analyzing and Mitigating Privacy with the DNS Root Service. In *Proceedings of the ISOC NDSS Workshop on DNS Privacy*. The Internet Society. https://ant.isi.edu/~hardaker/papers/2018-02-ndss-analyzing-root-privacy.pdf

[8] Paul Hoffman and Kazunori Fujiwara. 2020. *Running a Local Root Server*. Technical Report RFC8806. RFC Editor. https://www.rfc-editor.org/info/rfc8806

[9] Internet Assigned Numbers Authority. 2025. Root Zone File. https://www.internic.net/domain/root.zone.

[10] Internet Systems Consortium. [n. d.]. BIND - Berkeley Internet Name Domain. https://www.isc.org/bind/.

[11] David Lawrence, Warren Kumari, and Puneet Sood. 2020. Serving Stale Data to Improve DNS Resiliency. RFC 8767. https://www.rfc-editor.org/rfc/rfc8767

[12] Jinjin Liang, Jian Jiang, Haixin Duan, Kang Li, and Jianping Wu. 2013. Measuring Query Latency of Top Level DNS Servers. In *Passive and Active Measurement (PAM) (Lecture Notes in Computer Science, Vol. 7799)*. Springer, 145–154. https://doi.org/10.1007/978-3-642-36516-4_15

[13] Giovane C. M. Moura, Ricardo de O. Schmidt, John Heidemann, Wouter B. de Vries, Moritz Müller, Lan Wei, and Cristian Hesselman. 2016. Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event. In *Proceedings of the 2016 Internet Measurement Conference (IMC '16)*. ACM. https://doi.org/10.1145/2987443.2987446

[14] 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, 101–115. https://doi.org/10.1145/3355369.3355568

[15] NLnet Labs. [n. d.]. Unbound DNS Resolver. https://www.nlnetlabs.nl/projects/unbound/about/.

[16] Oracle Corporation. 2024. *Oracle VM VirtualBox*. https://www.virtualbox.org/

[17] RIPE Atlas. [n. d.]. RIPE Atlas: A Global Internet Measurement Platform. https://atlas.ripe.net/.

[18] A. S. M. Rizvi, L. Bertholdo, J. Ceron, and J. Heidemann. 2022. Anycast Agility: Network Playbooks to Fight DDoS. In *Proceedings of the 31st USENIX Security Symposium*. https://www.usenix.org/conference/usenixsecurity22/presentation/rizvi

[19] Duane Wessels. 2015. *Decreasing Access Time to Root Servers by Running One on Loopback.* RFC RFC 7706. RFC Editor. https://doi.org/10.17487/RFC7706

[20] Ólafur Guðmundsson. 2023. *1.1.1.1 lookup failures on October 4, 2023.* https://blog.cloudflare.com/1-1-1-1-lookup-failures-on-october-4th-2023/

A Resolver Software Configurations

A.1 Unbound

The following configuration was added to the file /etc/unbound/unbound.conf.d/root-zone.conf:

```
auth-zone:
    name: "."
    master: "b.root-servers.net"
    master: "c.root-servers.net"
    master: "d.root-servers.net"
    master: "f.root-servers.net"
    master: "g.root-servers.net"
    master: "k.root-servers.net"
        fallback-enabled: yes
    for-downstream: no
    for-upstream: yes
    zonefile: "root.zone"
```

A.2 BIND

The following configurations was added to the file /etc/bind/-named.conf:

```
zone "." {
    type mirror;
};
```

B RIPE Atlas results

B.1 Table of probe IDs that rely on recursive resolvers that potentially use local root

B.2 List of measurement IDs

The measurement IDs to find probes that may rely on resolvers that use local root can be found here:

- 103388001
- 103388173
- 103785191
- 103785871
- 103803525
- 103803658
- 103821089
- 104742037
- 105092988
- 105093111
- 105253854
- 105254113
- 105258159
- 105258383
- 105309962
- 105310546
- 105310553
- 105454327
- 105454338
- 105762447
- 105762775
- 105932994
- 106106332
- 106152690
- 106153451
- 106153486
- 107865625
- 108152000
- 108153037
- 108158105
- 108163730
- 109298950
- 109299044
- 109299678

Probe ID	Country	Resolver Organization
6882	KE	Wavex Internet Service Provider LTD
11667	BE	EDPNET Belgium BV
12496	KW	Wataniya-Telecom-Infrastructure
18371	PE	America Movil Peru S.A.C.
19578	TG	African Network Information Center
21472	IT	Cloudflare, Inc. Google LLC
22110	IE	ANX-NET-3725223116631
29415	IM	DOM-INFRA Google LLC
30741	RU	Google LLC Yandex enterprise network
35556	BZ	Latin American and Caribbean Regional Registry
51971	EE	EE-ESTPAK
52908	BB	Cable & Wireless (Barbados) Limited
53098	PF	Cloudflare, Inc. WoodyNet, Inc.
53155	AO	Cloudflare, Inc. Google LLC
54670	LT	Cloudflare, Inc.
55631	MN	Mobinet LLC
60962	CV	ORG-CVTS1-AFRINIC
61388	GB	AA-THN
62224	PK	Cloudflare, Inc. Google LLC
62917	KH	Cloudflare, Inc.
64071	RO	Cloudflare, Inc.
65050	AM	Google LLC
1007966	NP	WLINK STATIC WIRELESS KTM
1002816	EC	AS7195
1002874	FI	WoodyNet, Inc.
1004493	US	Google LLC
65292	NP	Simple Media Network Private Limited
1008771	DE	Contabo GmbH
1009198	CZ	Cloudflare, Inc.
1009709	HR	Cloudflare, Inc.
1010086	AF	Google LLC
1010467	RS	Cloudflare, Inc.
1010679	AD	ORG-SdTd1-RIPE
1008625	NC	Not known
29936	HR	Not known
16825	IR	Not known
50050	PL	Not known
61581	XK	Not known

Table 2. Resolvers observed in measurements, including probe ID, country, and organization.