ActiveDNS: Is There Room for DNS Optimization Beyond CDNs?

Yue Wang
Computer Network Information Center, CAS
Beijing, China

Changhua Pei*†
Hangzhou Institute for Advanced Study, UCAS
Beijing, China

Zexin Wang
Yingqiang Wang
Computer Network Information Center, CAS
Beijing, China

Guo Chen
Hunan University
Hunan, China

Yuchao Zhang
Beijing University of Posts and Telecommunications
Beijing, China

Yi Li
Jingjing Li
Computer Network Information Center, CAS
Beijing, China

Jianhui Li
Computer Network Information Center, CAS
Beijing, China

Gaogang Xie‡
Computer Network Information Center, CAS
Beijing, China

***Abstract*—Domain Name System (DNS) converts domain names into IP addresses. The specific IP it returns to a client has significant implications for optimizing user experiences on web services. The establishment of Content Distribution Networks (CDNs) facilitates the spread of content across diverse cache servers, thereby enabling quick responses to user requests. But not all distributed Internet services have sufficient budget or resources to deploy CDN servers. For these services, is there a cost-efficient way to enhance users' access experience to them? In this paper, we build a data-driven DNS resolver system named ActiveDNS. We conduct a historical analysis of billions of DNS request-response logs from a public DNS resolver. Inspired by these findings, we probed dozens of public DNS servers to obtain available IPs behind carefully selected target domain names and measure their performance metrics. We built a domain ip performance database of 2.9 million records that can be easily incorporated into the open-source DNS framework BIND via DLZ technology. ActiveDNS has been deployed in real-world scenarios and the evaluations reveal promising results. Compared to currently used DNS resolver, the 75th service round-trip time (RTT) for the IP returned by ActiveDNS decreased from 67 ms to 43 ms.**

***Index Terms*—Domain Name System, DNS Measurement**

I. Introduction

Domain Name System (DNS), a pivotal component of network infrastructure, is tasked with the conversion of domain names into IP addresses thereby enabling convenient user access. Many Internet service providers, including social networks, e-commerce platforms, and streaming video services, often deploy the same service across servers in different geographical locations. These servers have different IP addresses. The advent of DNS allows users to avoid memorizing these complex IP addresses and instead only need to remember a domain name. This also ensures a certain level of flexibility, as service providers can add or remove machines without notifying users, simply by updating DNS records.

Content Distribution Networks (CDNs) [21] are engineered to store specific content copies across multiple cache servers. This configuration allows for the selection of optimal servers close to the clients, thereby enhancing the speed at which users can access the content. Given the overlapping functionalities of CDNs and DNS, modern CDNs possess capabilities akin to DNS resolvers. However, not all services have sufficient budget to deploy CDN services. According to the report from Builtwith [5], only 60% of the top 100k domains have enabled CDN services. Furthermore, among the services that have enabled CDN support, we find that they often choose multiple CDN providers, with multiple CDN entry points available for selection.

Therefore, is there a more **cost-efficient** method to automatically select the optimal IP address behind a certain service domain for users without the need for CDN deployment, in order to enhance the user experience? To achieve "cost-efficient" in this context, three conditions need to be met:

Transparency to Users. Beyond CDN-like optimization methods, another promising approach is SmartDNS [35]. Unlike CDN, which operates server-side, SmartDNS functions on the client-side by actively measuring the RTT to anticipate user requests for optimal IPs. Nevertheless, this approach has two notable limitations. First, SmartDNS requires software installation on the client side to take active measurements, which consume client resources. Second, SmartDNS needs to conduct real-time measurements for any domain query requests initiated by the user, which can significantly increase the query time for each request, providing users with a poor experience (see results in Section V).

* Corresponding author. Email: chpei@cnic.cn.
† Also with Computer Network Information Center, CAS.
‡ Also with University of Chinese Academy of Sciences.

Transparency to Service Providers. For CDN providers, obtaining a list of available IP addresses for a domain and their performance metrics is relatively straightforward, as the service providers will inform the CDN providers of this information prior to the CDN deployment. However, acquiring the list of available IP addresses without the service provider's proactive disclosure poses a challenge. In principle, authoritative servers may hold multiple IP addresses for different domains, but typically an authoritative server stores only a subset of these domains. In this paper, we employ active probing techniques to request information from various public DNS providers, thereby acquiring multiple available IPs behind a service.

Low Deployment Cost. Even though it is possible to obtain multiple IP addresses for a domain without requiring server-side support by querying several public DNS servers (such as 8.8.8.8 and 1.1.1.1), the task is immensely challenging due to the sheer volume of domain names worldwide, which amounts to 700 million [36]. Enumerating through public recursive servers to acquire the list of available IPs behind each domain name incurs a significant overhead. Fortunately, our analysis of over a billion records from a public DNS reveals that 0.01% of domain names account for more than half of the total traffic. Therefore, in this paper, we have designed a domain selector that requires measurement of only a small, carefully selected subset of domain names to achieve substantial coverage and effectiveness.

Starting with the principle of cost-efficient and grounded in real-world investigation, we introduce a data-driven DNS resolver system, ***ActiveDNS***, designed to operate DNS-side. When ActiveDNS receives user requests, it uses an IP selection algorithm to choose the appropriate IP address from a continuously updated and maintained domain name-IP quality of service database, then returns this IP to the user. ActiveDNS has been deployed in real-world scenarios for 5 months. Compared to online DNS resolver, the 75^{th} service RTT for the IP returned by the DNS resolvers decreased from 67 ms to 43 ms.

This paper's contributions can be summarized as follows:

- For the first time, we build a data-driven DNS optimization method at DNS side, named ***ActiveDNS***, which requires no assistance from the server side and no modifications on the user side. Through carefully designed URL Selector, Measurement Worker, and IP Selector, ActiveDNS achieve significant results with low deployment and measurement costs.
- We analyze the optimization potential of the existing DNS system through real-world measurements. Simultaneously, by actually deploying ActiveDNS and conducting a comprehensive evaluation, we verify that even without the assistance of CDNs, the existing DNS system can achieve substantial performance improvements.
- Our optimization of ActiveDNS is implemented within the open-source DNS framework BIND, adopting DLZ techniques. This allows for easy integration into current recursive DNS resolvers, a crucial factor for widespread deployment [1].

The remainder of this paper is structured as follows: Section II discusses previous related work. Section III gives a real-world investigation. Section IV discusses ActiveDNS system design. An evaluation of our approach is conducted in Section V, and the paper concludes with a summary and future directions in Section VI.

II. Related Work

Based on user experience with DNS, DNS resolution performance can be optimized in two ways: shortening the DNS query time and speeding up user access to the target server [23]. Reducing DNS query time can be achieved through several optimization techniques, including deploying local caching [26], [27], optimizing cache TTL [3], [16], [26]–[28], and implementing anycast multi-location deployments [14], [18], [22], [31]. However, according to Table II, DNS query time accounts for a relatively small proportion of the time users spend accessing target servers. The focus of this paper is on optimizing the time it takes for users to access the target server, especially when a domain name maps to multiple IP addresses, which raises the issue of server selection. This problem can be solved on the DNS side, server side, or client side.

A. DNS-side Optimization

For DNS-side server selection, the EDNS Client Subnet (ECS) [12] extension provides an optimization mechanism for authoritative DNS servers to return the optimal server IP addresses based on IP prefix of the user. By providing information about the client's subnet, ECS allows for more precise traffic distribution and server selection strategies, thus optimizing access speed and enhancing overall network performance. However, the use of ECS introduces significant drawbacks that cannot be overlooked because it need to cache resolved IPs separately for each distinct user IP prefix.

B. Server-side Optimization

On the server side, in addition to the optimization of traditional data center strategy [37] and routing algorithm [10], finding the fastest server among multiple redundant deployments for different users is the simplest and most effective approach. Round Robin strategy, as delineated in [4], seeks to distribute the load among multiple target servers. Nonetheless, [15] posits that this Round Robin strategy is most suitable for scenarios where a few servers share the same subnet and exhibit comparable request processing capabilities.

Furthermore, [33] introduces a DNS query preprocessor that selects the optimal target server based on dynamic RTT measurements. However, this method not only increases the average query time by 9.1 ms, but it also lacks a comparison with traditional DNS methods to validate its effectiveness.

Additionally, in the context of CDNs, alternative server selection methodologies have been explored [19], [32]. For instance,

[1]We release our code at **https://github.com/CSTCloudOps/ActiveDNS**

[6] utilizes partial round-trip time measurements and historical server load information to make server selections. However, due to cost considerations, not all Internet services have deployed CDN services.

C. Client-side Optimization

[24] proposes a server selection method based on Quality of Service (QoS). In this method, clients make their current selection based on QoS data collected over a period of time from various server IP addresses. This requires clients to maintain real-time QoS information for each IP address, which can consume significant resources. [25] evaluates five client-based server selection methods and summarizes a set of principles for client selection. SmartDNS [35], which is a local DNS server running on the client's side. It accepts DNS query requests from local clients, fetches DNS query results from multiple upstream DNS servers, and returns the fastest result to the client, thereby improving network access speed. Although SmartDNS can achieve good optimization effects, it requires modifications to the client or home router, and in addition, its measurement method increases overhead on the client. In this paper, we aim to design a completely client transparent DNS optimization system.

III. Real-world Investigation

In this section, measurements were conducted based on real-world recursive DNS data to investigate the resolution conditions of recursive DNS in the real world. This serves a dual purpose: firstly, to explore the necessity of optimizing the response IPs of recursive DNS, and secondly, to provide feasible data support for optimizing recursive DNS. The measurements in this section are based on a conventional recursive DNS that does not enable ECS. Hereinafter, we refer to it as *DefaultDNS*.

A. Is there still room for optimization in the server IPs provided to users with co-existance of CDN?

The latency between clients and servers on the internet directly affects user experience and business revenue. A second delay in web page response time leads to a 16% decrease in customer satisfaction [29]. Amazon has also reported that every 100 ms increase in page load time costs them 1% in sales [2]. Therefore, reducing the client-server latency by even a millisecond is crucial on the internet. Fortunately, based on the two measurements below, we observe that existing DNS returns multiple IPs for the same domain and there is still room for optimization in RTT between these IPs.

1) **Geographical distribution of different IPs for the same domain.** We selected 10 out of the top 20 high-traffic domains within *DefaultDNS* and conducted multiple accesses to these domains from the same location. The result indicates that, even with the existing of CDN, the geographical locations of some domain resolution IPs are quite decentralized.
2) **RTT distribution of different IPs for the same domain.** Fig. 1 shows significant variations in RTT for the same IP at different times and across different IPs. For the same user, the RTT variation range across different IPs under the same domain name is significant, ranging from 20ms to 80ms. This underscores the necessity of selecting the optimal IP.

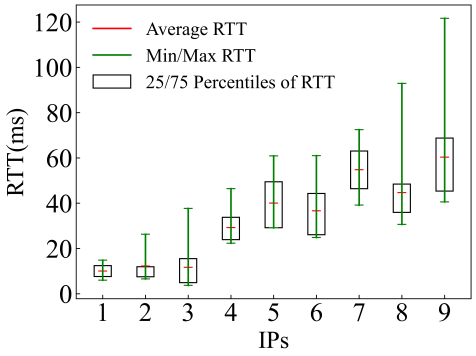


Fig. 1: RTT of multiple IP addresses resolved for the same domain.

B. Is there still room to achieve better resolution IP with less storage space?

It is natural to proactively measure and record the RTTs for IPs associated with each domain and select the best IP for optimization. However, it is unfeasible because the number of domain names and IP addresses in the real-world is enormous and dynamically changing. Whether it is possible to solve this problem with limited storage and measurement overhead is important.

1) **Distribution of query volume.** Approximately 1 billion query logs from *DefaultDNS*, spanning a week, were analyzed to estimate query volumes per domain. Domains were ranked in descending order of query volume, and the CDF of the domain query volumes was calculated. Fig. 2 showed that the top 1% of domains accounted for approximately 95% of the query traffic, and the top 10% of domains accounted for 99% of the query traffic.
2) **Distribution of IP numbers belonging to the same domain.** Using the same one-week query logs from *DefaultDNS*, we count the number of IPs resolved for each domain. Fig. 3 indicated that about 50% of domains were mapped to a single IP, while the remaining 50% were mapped to two or more IPs. Fig. 4 indicated that that when the time frame extended to 20 days, the average number of IPs stabilized around 20. This figure suggests that there is considerable room for optimization in the resolved IPs of recursive DNS domains.

These measurements indicate that DNS queries are concentrated on a small subset of domain names. Targeted optimization of this subset could enhance the user experience for the majority. This is why we later established a database using a portion of these domain names. Furthermore, the measurements also reveal

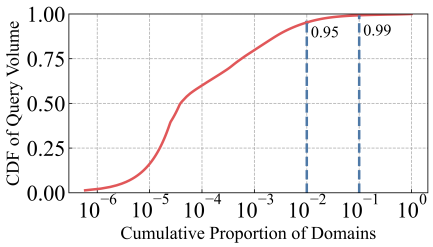


Fig. 2: CDF of query volume across domain proportions.

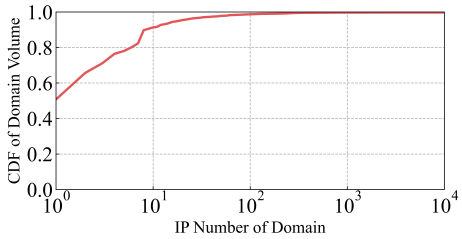


Fig. 3: CDF of domain volume across different IP number of domain.

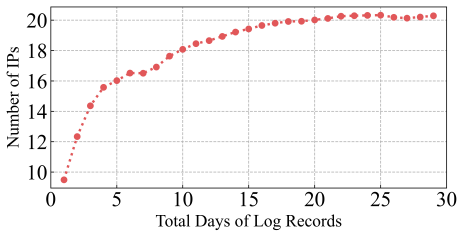


Fig. 4: Average #IP per domain accumulated over time.

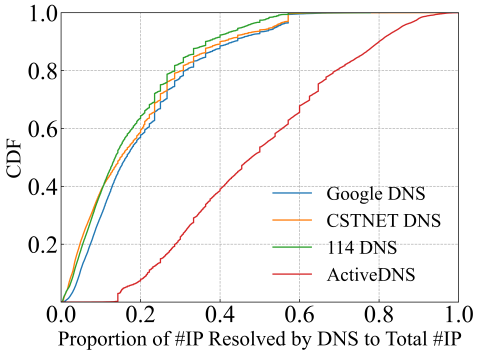


Fig. 5: CDF of resolved IP coverage across different DNS providers.

that a single domain may correspond to multiple IP addresses that increase over time. Therefore, dynamically selecting the optimal IP during DNS queries can significantly reduce service latency.

C. Is there still room to provide a more diverse range of resolution IPs for users?

This measurement aimed to ascertain the distribution of IP numbers resolved by public recursive DNS services. The top 10% of domain names, each having more than five IP addresses, are extracted from *DefaultDNS*, and their resolved IP numbers were measured using probes in four cities across 17 public DNS services. These results were then consolidated to form a comprehensive dataset. The analysis focused on comparing the number of IPs resolved for each domain by three DNS services—*Google DNS*, *114DNS*, and *DefaultDNS*—with those resolved by *ActiveDNS*, which is the DNS contribution of this study, in terms of their proportion of the total dataset.

Fig. 5 indicated that for the majority of the domains, the number of IPs resolved by *Google DNS*, *114DNS*, and *DefaultDNS* constituted only a small portion of the entire dataset. In contrast, *ActiveDNS* was found to resolve a significantly larger proportion of IPs.

This suggests that *ActiveDNS* potentially offers a more extensive range of IP resolutions for the top domains, possibly leading to improved DNS resolution capabilities. By providing a broader set of resolved IPs, *ActiveDNS* may enhance user connectivity and performance, underlining its potential advantages in more effectively meeting the diverse needs of internet users.

IV. System Design

The main goal of this paper is to design a new DNS system under limited resources overhead, without the need for modifications on the client and server sides, to better address the user's DNS requests and provide the most efficient IPs for the user, thereby enhancing the user's network experience. The network experience here is measured by RTT (Round-Trip Time). For this purpose, we've developed a system named *ActiveDNS*, depicted in Fig. 6, consisting of 4 main components: **URL Selector**, **Measurement Worker**, **IP Monitor** and **IP Selector**. The URL Selector's primary function is to identify domain names for optimization. These domain names undergo a comprehensive extension and evaluation process in the Measurement Worker, which leads to the creation of a domain-IP quality of service database. The IP Monitor continuously reviews this database to promptly remove any faulty IP addresses, thereby maintaining the quality of IP address resolution. When an end user queries a domain name, the IP Selector retrieves and returns the most suitable IP from the database.

The location of ActiveDNS in the network is shown in Fig. 7. It can be seen that ActiveDNS plays the role of a recursive server, by receiving requests from the client side and optimizing the return of specific domain names quickly. For domain names that ActiveDNS cannot cover, it will continue to request authoritative servers. This architecture ensures that ActiveDNS can be compatible with the existing network structure without the need for changes on the client side, service side, and authoritative side. In the following sections, we delve into ActiveDNS and introduce the four major modules of ActiveDNS one by one.

A. URL Selector

The URL Selector is designed to extract a list of domain names for optimization. It integrates three input sources: the query logs from DefaultDNS, the query logs from ActiveDNS and the output from the IP Monitor. The URL Selector merges the query logs from DefaultDNS and ActiveDNS, and selects the top 10% of domain names by query volume to add to the domain name list. Fig. 2 explains that by optimizing the top

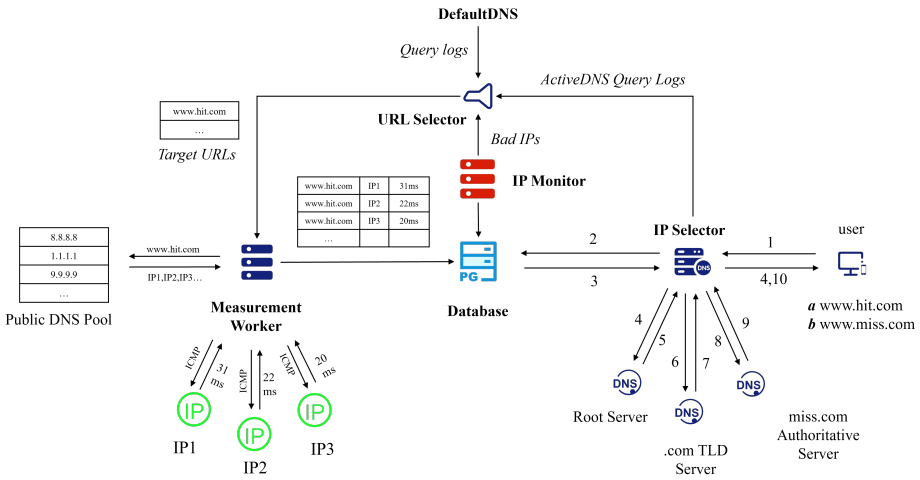


Fig. 6: Architecture of ActiveDNS.

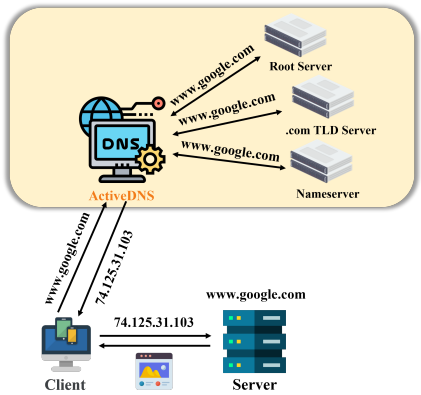


Fig. 7: ActiveDNS in the network.

10% of domain names, we can enhance more than 99% of query traffic. Domain names output by the IP Monitor in the previous cycle were also added to the domain list. This is important because these domains were previously selected by the URL Selector. It is necessary because they have faulty IPs(explained in Subsection IV-C), requiring their IPs to be reevaluated and expanded. The update cycle for the **URL Selector** is set to 24 hours. This duration is chosen because the query requests from end-users vary daily, resulting in differences in the DefaultDNS query logs between dates. Additionally, a 24-hour period is sufficient for establishing and updating the domain-IP service quality database.

B. Measurement Worker

The **Measurement Worker** plays a crucial role in establishing a domain-IP quality database and operates in two main phases: IP expansion of the domain names and measurement and evaluation of the IPs in IP pool. The initial list of domain names is sourced from the URL Selector.

Firstly, the Measurement Worker utilizes the Public DNS Pool to expand the IP for the list of domain names. As illustrated in Fig. 5, the number of aggregated IP sets significantly surpasses those provided by any single public DNS. This extension allows for a broader evaluation of IP quality, aiming for a nearly global optimal solution. The validity of this approach is supported by the experiments described in Subsection V-D.

In the second stage, the Measurement Worker consolidates the IP addresses from all the domain names, refreshes the IP pool, and measures the RTT of all IPs in the pool multiple times. After collecting these RTTs, it assesses the IP quality under the same domain name. In this paper, we mainly assess the IP quality by RTTs. We also evaluate different IP selection strategies in section V-F. Finally, the Measurement Worker records the results in a `<domain, ip, score>` format to the database for persistent storage. The content of the database is persistently accumulated, with updates and additions of measured results and the domain name list. The update cycle is synchronized with the URL Selector, occurring every 24 hours. In Subsection V-E, we discussed the impact of the cumulative time of the database on the performance of ActiveDNS. The results indicate that an accumulation period of 15 days achieves satisfactory results, with the database occupying approximately 400MB of disk space. This implies that the storage capacity only needs to exceed 400MB.

C. IP Monitor

The IP addresses associated with domain names are constantly changing, which leads to numerous **faulty IPs**. An IP that functions correctly one moment may become invalid the next due to system maintenance, server outages, service expirations, etc. To mitigate the frequency of such issues, traditional recursive DNS adjusts the TTL. These DNS systems cache domain name resolution results and set a TTL for each. Once the duration of a cached result exceeds the TTL, the recursive DNS system will query again. When the TTL is set to 15 minutes, the cache hit rate can reach approximately 80%, which also helps to reduce the likelihood of returning invalid IPs [20].

The domain-IP quality of service database is updated every 24 hours, effectively setting a 24-hour 'cache' for the domain names within it. This increases the risk of returning faulty IPs to users. The IP monitor checks all valid IPs in the database every 10 minutes. If a valid IP fails to respond to ICMP requests consecutively, the module will remove it from the database. If no other valid IPs are available for that domain name, IP monitor will re-query to obtain new available IPs and write these into the database. However, the quality of these newly acquired IPs has not been rigorously evaluated by the measurement module. Therefore, they need to be recorded as input sources for the next cycle of the URL Selector, so that the quality of these IPs can be assessed in the next cycle.

TABLE I: 17 public DNS providers in public DNS pool of ActiveDNS.

Category	Name of Public DNS
Academic Institution	CSTNET DNS, CNNIC DNS
Service Provider	China Telecom DNS, China Unicom DNS
	China Mobile DNS, China Telecom (Shanghai) DNS
Internet Company	114 DNS (2 instances), AliDNS (2 instances), Tencent DNS (2 instances)
	Baidu Public DNS, OpenDNS, Google DNS, AliDNS (Shanghai), YAMU DNS

D. IP Selector

Unlike traditional recursive DNS, *ActiveDNS* operates differently when handling domain name queries from users. If the queried domain name matches an entry in the database, *ActiveDNS* selects an appropriate IP for the domain using the **IP Selector** to provide as the resolution result. If there's no match in the database, *ActiveDNS* proceeds with an iterative query.

Fig. 6 illustrates two distinct processes: steps 1-4 show the query process for a domain (`www.hit.com`) that finds a match in the database, demonstrating how *ActiveDNS* quickly provides a resolution by selecting an appropriate IP from its database. Steps 1-10 depict the query process for a domain (`www.miss.com`) that does not find a match, outlining the more extensive iterative query process that *ActiveDNS* undertakes to resolve the domain name. This comparison highlights the efficiency gains possible when *ActiveDNS* can utilize its database for domain name resolutions.

The IP Selector is implemented with BIND [11] and DLZ [1]. BIND provides the iterative query functionality, while DLZ enables BIND to retrieve outcomes from the database.

This structured approach not only systematically maintains the system's efficiency but also enhances the user's experience by ensuring faster and more reliable domain name resolutions.

V. Evaluation

To validate the efficacy of our proposed ActiveDNS, we conducted comprehensive experiments based on the following four research questions.

RQ1: How does ActiveDNS perform in comparison to state-of-the-art methods?

RQ2: What is the impact of the number of candidate IPs corresponding to each domain in the database on the effectiveness of ActiveDNS?

RQ3: What is the influence of the database size on the effectiveness of ActiveDNS?

RQ4: What is the impact of different IP selection strategies on the effectiveness of ActiveDNS?

In ActiveDNS, we selected 17 public DNS (listed in TABLE I) and accumulated data over 21 days. The reasons for such a selection will be elaborated in subsequent sections. From the top 10% of domain names in real-world DNS logs, we randomly picked 10,000 domain names that had at least 2 IP addresses for measurement.

A. Evaluation Metric

The evaluation metric of DNS optimization can be categorized into static metrics such as geographical distance and dynamic metrics such as RTT [8], [13]. Empirical study in [9] demonstrate that dynamic metrics notably outperform static metrics. This is because geographical distance does not represent end-to-end network experience well [15]. Therefore, in this paper, we employ service RTT and query time to assess the performance of different DNS optimization methods.

- **Service RTT.** According to [7], more than 60% of users exhibit a behavior of selecting only the first IP address when querying the DNS server. As a result, the RTT of the first IP address becomes paramount and is referred to as the "Service RTT" in this paper.
- **Query time.** Query time refers to the time it takes for the DNS server to process and respond to a query. For fairness, we have enabled the caching feature for all sets of DNS during the measurement of query time.

By evaluating service RTT, query time and total time (which is the sum of service RTT and query time), we gain a holistic view of the user experience when accessing websites or online services via DNS systems. To comprehensively evaluate the overall performance of ActiveDNS across a large user base, we utilize the 25^{th} percentile, 50^{th} percentile, 75^{th} percentile, and average values of these metrics as evaluation metrics.

B. Baseline

- **GoogleDNS.** Google Public DNS ranks among the most widely used public DNS services worldwide [17]. Since its launch, it has garnered favor from numerous users and organizations for its stability, speed, and security.
- **SmartDNS.** SmartDNS was selected as the benchmark due to its ability to efficiently direct queries to upstream DNS servers and prioritize the quickest IP response for users. More specifically, SmartDNS boasts three response methodologies [34]: fastest ping response, fastest IP address and immediate DNS response. The first mode, also the default setting for SmartDNS, was chosen for its ability to deliver the optimal user experience.
- **114DNS.** 114DNS, a renowned public DNS provider in China, handles approximately 45 billion DNS queries daily for its users [30]. It aims to offer fast and stable DNS resolution services. Known for its excellent access speed and stability within mainland China, it also provides some level of malicious website blocking functionality.
- **DefaultDNS.** The DefaultDNS offers approximately 2 billion DNS queries daily for public users. It aims to provide trustworthy and reliable DNS resolution results, ensuring safe internet access for users.

C. *RQ1:* Overall Performance

In the same experimental setting, we conducted detailed experiments using various DNS methods, and the experimental results are presented in Table II.

TABLE II: Comparison of Service RTT, Query Time, and Total Time across different DNS services.

	Service RTT(ms)				Query Time(ms)				Total Time(ms)			
	25%	50%	75%	avg	25%	50%	75%	avg	25%	50%	75%	avg
DefaultDNS	4.39	27.14	67.06	31.79	**0.50**	**0.55**	**0.60**	9.24	**5.12**	30.66	74.12	36.38
GoogleDNS	4.57	28.61	82.26	34.83	36.66	39.30	82.14	76.15	65.75	95.39	223.33	111.55
114DNS	9.73	25.89	75.45	36.31	19.09	20.56	21.07	25.21	29.90	46.40	100.97	57.80
SmartDNS	4.18	15.56	64.48	29.82	8.56	20.98	72.00	129.57	13.38	34.84	134.04	64.84
ActiveDNS	**2.72**	**7.67**	**43.61**	**25.59**	3.29	3.51	3.95	**8.71**	6.37	**15.15**	**53.87**	**33.74**

TABLE III: Comparison of Service RTT, Query Time, and Total Time between two different measurement indicators.

	Service RTT(ms)				Query Time(ms)				Total Time(ms)			
	25%	50%	75%	avg	25%	50%	75%	avg	25%	50%	75%	avg
AVG_RTT	3.07	11.41	52.88	25.10	3.40	3.82	4.55	**19.19**	7.29	16.87	83.98	41.29
MIN_RTT	**2.85**	**10.60**	**49.29**	**24.71**	**3.33**	**3.70**	**4.54**	22.42	**6.43**	**16.39**	**82.86**	**39.22**

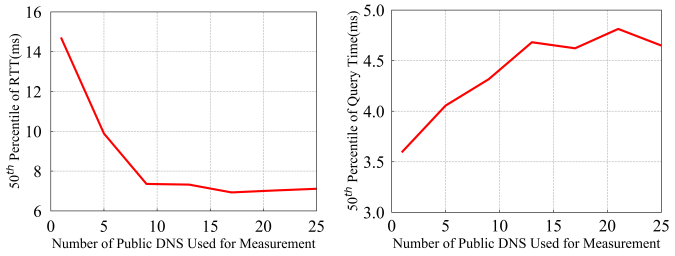
- **Service RTT.** The results demonstrate that ActiveDNS outperforms the current state-of-the-art method in terms of service RTT across various percentiles and average values. Particularly noteworthy is the improvement seen at the 25^{th} and 50^{th} percentiles, surpassing state-of-the-art by 35% and 51% respectively. This indicates that through ActiveDNS, the majority of users have significantly reduced the time taken to access domain-related services, thereby enhancing user experience substantially. Despite a significant enhancement at the 75^{th} percentile with ActiveDNS, some domain services accessed were located at a considerable distance, a challenge that even the current famous DNS server cannot entirely overcome. Common public DNS servers like GoogleDNS, 114DNS, and DefaultDNS typically perform iterative queries from root to top-level and down to authoritative servers, caching the results locally for a period without filtration. This approach often yields suboptimal results in practice, particularly evident with 114DNS where the 25^{th} percentile reaches 10ms. SmartDNS, by querying multiple upstream DNS servers to obtain an IP address list and locally testing and filtering for the optimal IP, achieves superior performance compared to public DNS, especially at the 50^{th} percentile. In contrast, our novel approach, ActiveDNS, takes a proactive stance by storing potential resolution results from public DNS based on historical data and implementing a more refined IP selection mechanism, resulting in superior resolution performance and an enhanced user experience.
- **Query Time.** The experiment was conducted under the same conditions for DefaultDNS, SmartDNS, and ActiveDNS, ensuring the comparability of their query times. However, the query times for GoogleDNS and 114DNS were not directly comparable due to unknown geographical locations and Autonomous System (AS) affiliations. From Table II, it is evident that DefaultDNS holds a significant advantage across various percentiles in query time, attributed to the caching of the majority of domain query results. Nonetheless, its average query time remains relatively long, indicating that numerous uncached queries still consume a substantial amount of time. On the other hand, SmartDNS exhibits poor performance in query time. As previously mentioned, it necessitates waiting for results from multiple upstream DNS servers, coupled with local testing, resulting in an average query time reaching up to 130ms, which is deemed unacceptable for many users. In contrast, ActiveDNS achieves overall optimal performance by sacrificing a certain amount of query time. The results demonstrate that ActiveDNS maintains stability around 3ms across various percentiles and surpasses the state-of-the-art in terms of average query time.
- **Total Time.** The results indicate that ActiveDNS remains the best option, followed by DefaultDNS, with SmartDNS ranking last. Even though ActiveDNS is a few milliseconds slower in query time compared to traditional DNS, its lead in service RTT at both the 50^{th} and 75^{th} percentiles, by approximately 20ms, ensures its superiority in total time. While SmartDNS has some advantages in service RTT, its mechanism leads to excessive query time consumption, resulting in the longest total time.

D. ***RQ2:*** *Impact of Candidate IP Pool Size*

The essence of DNS server resolution quality lies in the fact that a single domain name can be mapped to multiple IP addresses, and these different IPs may exhibit significant variations in RTT due to complex factors such as network topology and traffic conditions. Therefore, we hypothesize that expanding the IP candidate pool for a domain name will increase the number of alternative IPs, consequently enhancing the likelihood of identifying superior IPs.

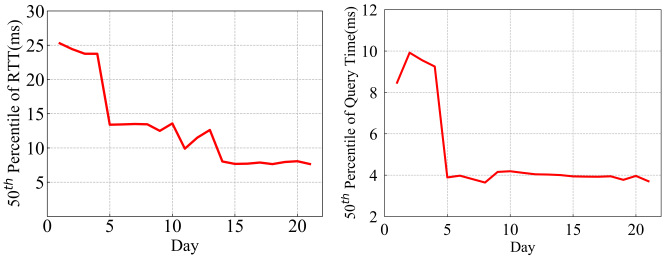
To investigate the impact of IP candidate pool size on DNS resolution quality, we conducted experiments while keeping other configurations constant, only varying the number of source DNS servers (1, 5, 9, 13, 17, 21, 25) from which IP addresses for the measured domains were obtained. The experimental results are depicted in Fig. 8.

As the number of source DNSs retrieving <domain, IP> pairs in our measurement system grows, there's an observed improvement in ActiveDNS's service RTT. However, we believe there's a diminishing return on this enhancement. Specifically, when the number of source DNSs increased from 17 to 25, the service RTT improvement was minimal with a marginal increase in query time. Hence, judiciously augmenting the quantity of



(a) Service RTT (b) Query Time

Fig. 8: The effects of number of IP candidates on the resolution quality.



(a) Service RTT (b) Query Time

Fig. 9: The median service RTT and query time with the expansion of the database.

candidate IP addresses can enhance the efficacy of ActiveDNS. Nevertheless, an excessive augmentation in candidate IPs may disproportionately inflate the database size, subsequently imposing additional overhead in terms of database query time.

*E. **RQ3:** Impact of Database Size*

Domain mapping database is an essential part of ActiveDNS. The content within this database directly dictates the quality of IPs that ActiveDNS resolves. Thus, we endeavored to analyze the changes in ActiveDNS's optimization efficacy as this database continuously expands and updates over time. We selected databases from different cumulative days and measured using random domain names from the subsequent day, calculating the 50^{th} percentile for service RTT and query time. From Fig. 9, we discerned that as the database expands in scale, the quality of IP optimization also proportionally improves. This improvement is attributed to the database's growing pool of superior IPs, which, in turn, enhances the IP quality for each domain. Concurrently, query duration diminishes with database expansion. The underlying reason being, as the database enlarges, there's a heightened likelihood for domain queries to directly hit the database and retrieve the corresponding IP. This reduces the proportion of recursive queries, resulting in significant time savings. From Fig. 9, it is observed that query time converges when the database's cumulative days reach five, while service RTT converges at fifteen days of database accumulation. At this point, the database holds a sufficient number of domain names, and the pool of candidate IPs for these domains is close to its limit.Therefore, further accumulation of additional IP addresses provides limited improvement to the optimization of the best IP addresses.

*F. **RQ4:** Impact of IP Selection Strategies*

Due to the dynamic nature of the network environment, to obtain stable measurement results, for each IP of a domain, we continuously measure the RTT ten times. When using this data for IP selection, we have two strategies: whether to use the average RTT (AVG_RTT) or the minimum RTT (MIN_RTT) as the indicator for choosing this IP. To evaluate the effectiveness of these two selection strategies, we conducted experiments under environments with consistent other settings. The results of the experiments are shown in Table III. From the table, it can be seen that the MIN_RTT selection strategy is superior to AVG_RTT in multiple evaluation metrics and achieves better overall performance. Thus we ultimately chose MIN_RTT as the basis for IP selection.

VI. CONCLUSION

In this paper, we presented *ActiveDNS*, a data and measurement-driven method that offers a promising solution to improving the speed of services represented by domains while harmoniously coexisting with the existing CDN mechanism. Our method leverages the power of DNS request-response logs and active probing, demonstrating the feasibility of simultaneously reducing service RTT and DNS query time, even in the context of services that are already optimized by CDN-like mechanisms. The implementation of ActiveDNS is done within the open-source DNS framework BIND, adopting DLZ techniques. This allows for easy integration into current DNS resolvers, a crucial factor for its widespread deployment. The real-world deployment of ActiveDNS has shown promising results. When compared to our currently used DNS, the 75^{th} service RTT for the IP selected by DNS decreased from 67 ms to 43 ms. The results of this study demonstrate the potential of ActiveDNS in optimizing DNS performance and improving user experiences on web services. We believe that this work lays a solid foundation for future studies in this field. We release our code in the hope that it can contribute to the broader community's efforts in improving DNS performance.

VII. ACKNOWLEDGMENTS

This work is supported by National Key R&D Program of China (Grant No. 2022YFB3103000), in part by the National Natural Science Foundation of China (No.62202445), in part by the State Key Program of National Natural Science Foundation of China under Grant 62321166652.

REFERENCES

[1] Dynamically loadable zones. https://bind-dlz.sourceforge.net/. [Online; accessed 2023-09-16].

[2] Amazon. Common elasticache use cases and how elasticache can help. [Online; accessed 2024-02-06].

[3] Soumya Basu, Aditya Sundarrajan, Javad Ghaderi, Sanjay Shakkottai, and Ramesh K. Sitaraman. Adaptive ttl-based caching for content delivery. *CoRR*, abs/1704.04448, 2017.

[4] Thomas P. Brisco. DNS Support for Load Balancing. RFC 1794, April 1995.

[5] Builtwith. Content delivery network usage statistics. [Online; accessed 2024-02-06].

[6] Lin Cai, Jun Ye, Jianping Pan, Xuemin (Sherman) Shen, and Jon W. Mark. Dynamic server selection using fuzzy inference in content distribution networks. *Computer Communications*, 29(8):1026–1038, May 2006.

[7] Thomas Callahan, Mark Allman, and Michael Rabinovich. On modern dns behavior and properties. *ACM SIGCOMM Computer Communication Review*, 43(3):7–15, 2013.

[8] Robert L Carter and Mark E Crovella. Dynamic server selection using bandwidth probing in wide-area networks. Technical report, Citeseer, 1996.

[9] Robert L. Carter and Mark E. Crovella. On the network impact of dynamic server selection. *Computer Networks*, 31(23):2529–2558, 1999.

[10] Peizhuang Cong, Yuchao Zhang, Lei Wang, Wendong Wang, Xiangyang Gong, Tong Yang, Dan Li, and Ke Xu. Dit and beyond: Inter-domain routing with intra-domain awareness for iiot. *IEEE Internet of Things Journal*, 2023.

[11] Internet Systems Consortium. Berkeley internet name domain version 9. https://www.isc.org/bind/. [Online; accessed 2023-09-16].

[12] Carlo Contavalli, Wilmer van der Gaast, David C Lawrence, and Warren "Ace" Kumari. Client Subnet in DNS Queries. RFC 7871, May 2016.

[13] M.E. Crovella and R.L. Carter. Dynamic server selection in the internet. In *Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems*, pages 158–162, 1995.

[14] Xun Fan, John Heidemann, and Ramesh Govindan. Evaluating anycast in the domain name system. In *2013 Proceedings IEEE INFOCOM*, pages 1681–1689. IEEE, 2013.

[15] Z.-M. Fei, S. Bhattacharjee, E.W. Zegura, and M.H. Ammar. A novel server selection technique for improving the response time of a replicated service. In *Proceedings. IEEE INFOCOM '98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No.98CH36169)*, Nov 2002.

[16] Pawel Foremski, Oliver Gasser, and Giovane CM Moura. Dns observatory: The big picture of the dns. In *Proceedings of the Internet Measurement Conference*, pages 87–100, 2019.

[17] Google. Google public dns and location-sensitive dns responses, 2014. [Online; accessed 2024-02-06].

[18] Ted Hardie. Distributing Authoritative Name Servers via Shared Unicast Addresses. RFC 3258, April 2002.

[19] Hadrien Hours, Ernst Biersack, Patrick Loiseau, Alessandro Finamore, and Marco Mellia. A study of the impact of dns resolvers on cdn performance using a causal approach. *Computer Networks*, 109:200–210, 2016.

[20] Jaeyeon Jung, Arthur W Berger, and Hari Balakrishnan. Modeling ttl-based internet caches. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, volume 1, pages 417–426. IEEE, 2003.

[21] Tom Leighton. Improving performance on the internet. *Communications of the ACM*, 52(2):44–51, Feb 2009.

[22] Kurt Erik Lindqvist and Joe Abley. Operation of Anycast Services. RFC 4786, December 2006.

[23] Richard Liston, Sridhar Srinivasan, and Ellen Zegura. Diversity in dns performance measures. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 19–31, 2002.

[24] Kenichi Mase, Takayuki Kuribayashi, and Akihiko Tsuno. A dynamic server selection method using qos statistics. *Electronics and Communications in Japan Part I-communications,Electronics and Communications in Japan Part I-communications*, Mar 2003.

[25] Nabor C. Mendonça, José Airton F. Silva, and Ricardo O. Anido. Client-side selection of replicated web services: An empirical assessment. *Journal of Systems and Software*, 81(8):1346–1363, Aug 2008.

[26] P. Mockapetris. Domain names: Concepts and facilities. RFC 882, November 1983.

[27] P. Mockapetris. Domain names: Implementation specification. RFC 883, November 1983.

[28] Giovane CM Moura, John Heidemann, Ricardo de O Schmidt, and Wes Hardaker. Cache me if you can: Effects of dns time-to-live. In *Proceedings of the Internet Measurement Conference*, pages 101–115, 2019.

[29] Zhou Munyaradzi, Giyane Maxmillan, and Mutembedza Nyasha Amanda. Effects of web page contents on load time over the internet. *Int Journal of Science and Research*, 2(9):75–79, 2013.

[30] Ltd Nanjing XinFeng network technology Co. 114dns news. [Online; accessed 2024-02-06].

[31] Sandeep Sarat, Vasileios Pappas, and Andreas Terzis. On the use of anycast in dns. *ACM sigmetrics performance evaluation review*, 33(1):394–395, 2005.

[32] Kyle Schomp, Mark Allman, and Michael Rabinovich. Dns resolvers considered harmful. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, pages 1–7, 2014.

[33] Toshihiko Shimokawa, Norihiko Yoshida, and Kazuo Ushijima. Flexible server selection using dns. Jan 2000.

[34] SmartDNS. Response mode configuration. [Online; accessed 2024-02-06].

[35] SmartDNS. Smartdns. https://pymumu.github.io/smartdns/. [Online; accessed 2023-09-16].

[36] Domain Name Stat. Domain name registration's statistics. [Online; accessed 2024-02-06].

[37] Yuchao Zhang, Haoqiang Huang, Ahmed M Abdelmoniem, Gaoxiong Zeng, Chenyue Zheng, Xirong Que, Wendong Wang, and Ke Xu. Flair: A fast and low-redundancy failure recovery framework for inter data center network. *IEEE Transactions on Cloud Computing*, 2024.