ConfiDNS: Leveraging Scale and History to Improve DNS Security

Lindsey Poole and Vivek S. Pai
Princeton University

Abstract

While cooperative DNS resolver systems, such as CoDNS, have demonstrated improved reliability and performance over standard approaches, their security has been weaker, since any corruption or misbehavior of a single resolver can easily propagate throughout the system. We address this weakness in a new system called ConfiDNS, which augments the cooperative lookup process with configurable policies that utilize multi-site agreement and per-site lookup histories. Not only does ConfiDNS provide better security than cooperative approaches, but for up to 99.8% of unique lookups, ConfiDNS exceeds the security of standard DNS resolvers. ConfiDNS provides these benefits while retaining the other benefits of CoDNS, such as incremental deployability, improved performance, and higher reliability.

1 Introduction

The use of distributed computing to address performance and reliability problems in the Domain Name System (DNS) [4] has recently received much research attention, and has spawned two widely-deployed distributed systems, CoDNS [5] and CoDoNS [7]. Both of these systems provide clients with improved reliability when performing DNS lookups by distributing the queries across nodes in the system. These systems fetch name-to-IP translations from the existing legacy DNS infrastructure as needed to provide an upgrade path for users.

These systems can be less secure than traditional local DNS resolvers when interacting with vulnerable legacy DNS infrastructure. If any node performs a DNS resolution and receives an incorrect answer, that answer can be propagated to other nodes, especially if the incorrect answer is returned quickly. The incorrect answer can occur because of a failure or compromise at a local DNS resolver, UDP packet spoofing when the node tries to communicate with an external DNS server, local site filtering policies, etc. CoDNS was designed to mitigate the impact of poisoned DNS responses by disallowing multi-hop request brokering, and by intentionally performing no caching. However, it was assumed that the security of such systems could never beat that of a local resolver, since content distribution networks (CDNs) and load balancers will return different DNS lookup results at different locations, making it difficult to compare lookup results.

Rather than being a fundamental tradeoff in supporting legacy DNS while achieving better reliability, worse security is not integral to cooperative DNS systems. We show that the scale of cooperative DNS systems can provide better security than legacy DNS resolvers for the vast majority of lookups. Where scale cannot be used, observing the history of DNS lookups can provide some assurance that DNS replies have not been modified. Between these two options, fewer than 1% of unique DNS lookups need to trade security for reliability or performance.

Using CoDNS traffic, we gather information on DNS usage, and perform a month-long study of name lookup behavior using multiple vantage points. We observe how DNS is used by various content providers, how name-to-IP mappings change over time, and how some sites' DNS resolvers can poison global DNS caches. Using the stability over time of name-to-IP mappings and/or the agreement of name lookup results at multiple DNS resolvers, we devise a range of security policies for our new DNS lookup system, ConfiDNS. For each policy, we show its performance and what fraction of names it can satisfy. In its weakest configuration, ConfiDNS provides better security than local DNS resolvers alone for 99.8% of unique lookups, and 92% can meet significantly stronger requirements. Lookup times are comparable to CoDNS, and are much better than local DNS resolvers.

ConfiDNS is incrementally deployable, using the same approaches as CoDNS, and requires no change to the existing global DNS infrastructure to reap its benefits. It can be installed as a simple proxy on the client or on the local resolver, requiring no changes to applications.

2 Overview

We first briefly describe DNS, including our terminology for its various components. DNS maps human-readable machine names to IP addresses using a hierarchy of servers, each of which is responsible for a portion of the global namespace. This system, called the server-side (global) DNS infrastructure, is run by the owners of domain names (e.g., example.com) and by organizations that run the top-level servers (e.g., com) that point to the per-domain servers. Clients send DNS lookups to machines within their own organization, called local (client-side) DNS resolvers, or LDNS. These resolvers perform the queries and cache the results, sharing lookup overhead across clients. The CoDNS system stemmed from the observation that many DNS lookup delays were due to failures of the local resolvers. CoDNS brokered queries

to peer DNS resolvers at remote sites when the local resolvers failed, improving performance and reliability.

ConfiDNS tries to provide more confidence in DNS lookups by using peer resolvers at *all* times in order to provide protection against certain attacks and failures, and by using lookup history to detect changes in name-to-IP mappings. Users run a ConfiDNS agent, ideally on their own machines, but it can be run on a (possibly shared) machine near the user (with some increased risk). This agent receives DNS lookup requests from the user, and sends the request to both the local DNS resolver as well as some number of peer ConfiDNS agents located at remote sites that use different resolvers than the client. Examples of policies that could be implemented in ConfiDNS are (a) the local resolver and at least one peer must agree on the result, (b) at least three peer sites must agree, (c) if no peers agree with the local resolver, the IP address must not have changed in the past week, (d) if no peers agree within 5 seconds, use any result.

2.1 Threat Model & Attacks Handled

ConfiDNS is designed to protect against attacks or failures at the client-side DNS infrastructure, including cache poisoning as well as the compromise, spoofing, or non-failstop failure of the local DNS infrastructure [3]. Since client-side DNS infrastructure is also used in cooperative DNS lookup services, protecting the client-side infrastructure also reduces avenues for polluting the global lookups in cooperative DNS systems. We have seen several client-side resolver behaviors that could pollute a cooperative DNS service. In one scenario, we saw a site administrator pollute CoDNS by configuring a resolver to reply instantly to all requests with the IP address of a local webserver that served a page saying that the resolver was being replaced. Unfortunately, if the browser expected an image and received this error message, the web page displayed broken image icons, causing problems. We also measured three other instances of pollution, which are further described in Section 3.1. In all of these cases, the results were returned quickly, so any peer using the resolvers at these sites could find its own lookups poisoned in the process.

We ignore server-side attacks for several reasons, including self-interest – as the developers of a CDN and a name lookup service that is used by the CDN, our most pressing concern is ensuring the CDN does not weaken security. Another practical issue is that protecting against takeover of the server-side infrastructure requires modifying the global DNS infrastructure [2], and is beyond our control. We also believe that server-side takeovers are easier to detect than client-side problems – if an attacker compromises a bank's DNS servers and redirects all traffic to a spoofed Web site, the bank's Web site will see a sharp and easily-detectable drop in activity. However, an attacker who wants to draw less attention could compromise an ISP's resolver, and redirect only lookups for one bank – the resulting drop in traffic may go unnoticed. By using a client-side solution that requires agreement from multiple client sites, this kind of attack would be ineffective unless conducted at scale. However, by attacking at scale, it is also more likely to be caught.

2.2 Applicability

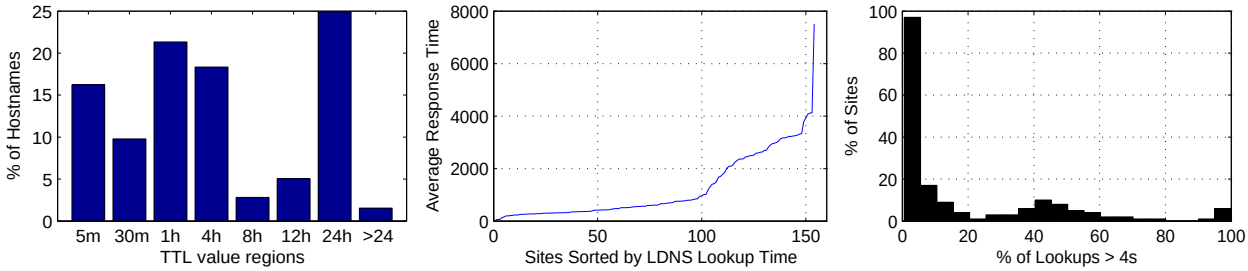
If DNS was used exactly as originally envisioned, with only one result per name no matter where the client resolves it, ConfiDNS would be a trivial and obvious solution to the problems mentioned. We suspect that the reason something like it does not already exist is not that the problem is small, but that researchers (including us) had assumed that the use of content distribution networks (CDNs) and DNS-based load balancers would make such a system unworkable. These systems redirect clients to nearby data centers, and often use very short DNS response TTLs to more aggressively balance load and locality. Particularly for CDNs such as Akamai [1], the number of possible IPs returned per domain name can number in the hundreds in these systems, since they will try to place servers at most large ISPs.

The questions that determine the applicability and effective of ConfiDNS is what fraction of content providers are using these techniques, and how do these systems behave in practice. While the answers to these questions may change over time, we expect that the Web will continue to have a mix of sites hosted at single locations, sites at a small number of data centers, and some sites hosted by commercial large-scale CDNs.

3 Continuous Monitoring

To create a DNS trace for analysis, we collected daily data from all 400+ PlanetLab [6] nodes running CoDNS, which produced lookups for over 100K unique names per day. Most of this traffic is generated by the CoDeeN content distribution network [8], which serves over 25 million requests per day to a global client population of over 50K daily users. To create a representative trace, we gathered one month of data, and selected names based on how frequently they appeared in the daily logs of CoDNS. The names that occurred daily contained virtually all of the Alexa top 500 Web site list, excluding some financial sites. Note that we select *unique* names, and weight their chance of inclusion in the list based on how many days they appear in our trace. The final list contains 40,000 names – the size was chosen to speed the lookup process. For one month, we resolved these names at every PlanetLab site (about 150 on most days), at the rate of one name per second. The list was randomized on a per-site basis to avoid overloading any server-side DNS infrastructure.

Figure 1(a) shows the TTL values for these names –



(a) TTL Breakdown (b) Average Lookup Times (c) Failure Rate Breakdown

Figure 1: TTL and lookup time statistics for names in the DNS trace.

most have cacheable lifetimes of 1 hour to 1 day, but a sizable fraction includes sub-hour TTLs, suggesting the use of CDNs, load balancers, or dynamic DNS systems for machines without static IP addresses. The average lookup time (per site) for all 40K unique names is shown in Figure 1(b), with a wide range of performance. Lookup times for individual names are bimodal – generally requiring less than 200ms or more than 4 seconds. The high lookup times reflect retries issued after a failure to receive a response, and their fraction is shown in Figure 1(c). Many sites require retries on half their lookups, resulting in high average lookup times. These results are worse than the LDNS measurements in CoDNS because they reflect the increased failure rate for uncached lookups in the wide area. Typical resolvers would have lower lookup times as seen by their clients, due to popular names being cached and requested multiple times.

3.1 CDNs and Data Centers

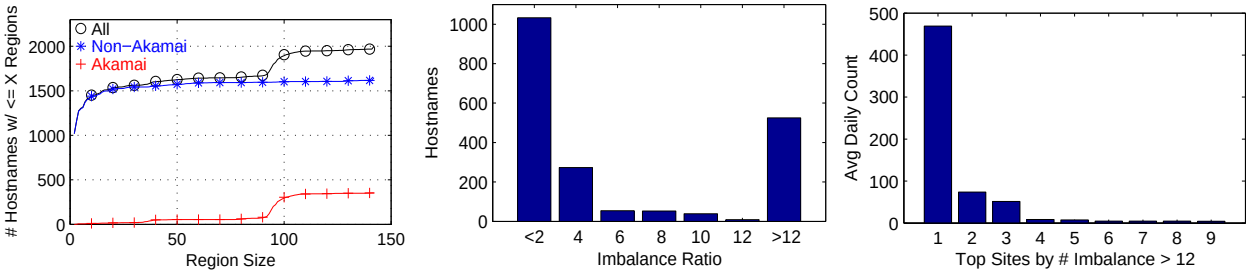
ConfiDNS's protection-by-agreement depends on how many sites receive the same IP address for each name. The number of IP addresses returned per hostname is less important than their pattern. If a domain has two data centers with addresses IP_1 and IP_2, and returns one IP based on which data center it believes is closer to the DNS resolver, we say that the hostname has two *regions*. From our 40K name list, we find that 91.5% return the same single IP address to all queries. Another 4.2% return multiple IP addresses, but return the same set of IP addresses to all queries. The specific order of IPs may vary to aid in load balancing, but the contents of the set are the same.

The remaining 4.3% of hostnames are not automatically out of reach for ConfiDNS – if they have relatively few regions, we may have enough ConfiDNS peers in a region to reach agreement. CDNs with many regions may use the same region continuously for a particular ISP, so the rate-of-change of name-to-IP mappings may be low.

The breakdown of the 1738 hostnames with two or more regions is shown in Figure 2(a), divided into those served by Akamai (347 hostnames) and the rest (1391 hostnames). The difference between the Akamai and non-Akamai region counts are quite sharp. Most non-Akamai hostnames have only two regions, and few have more than 10 regions, while most Akamai-served hostnames have 80-90 regions, as seen from our 150 vantage points.

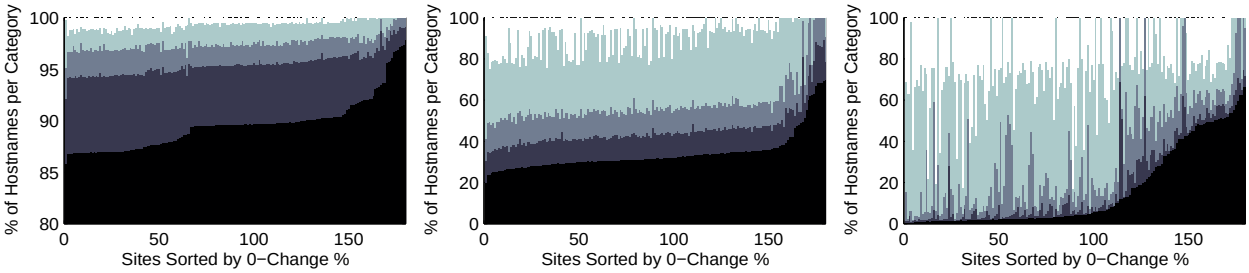
By comparing the number of sites in each region versus how many would appear in perfectly-balanced regions, we calculate a region imbalance factor for each multi-regioned name in our trace. Given the set of regions with the number of nodes per region, we calculate a geometric mean of a series of terms, where each term is either the ratio of actual region size to average region size, or its inverse, whichever is larger. For example, if a hostname has three regions with 15, 55, and 80 sites, its imbalance ratio is $\sqrt[3]{\frac{50}{15} \times \frac{55}{50} \times \frac{80}{50}}$. This calculation is designed to identify names that have gross imbalances in the sizes of their regions. While most names have region sizes that are within a factor of 2-4 of being fully balanced, we see a spike where the imbalance ratio exceeds 12 – in these cases, only one site disagrees over the IP address, and all of the other sites form a second region.

To get a sense of the origins of these heavily-imbalanced regions, we counted how often a site disagreed with all others, and show the daily average for the top ten sites in Figure 2(c). The worst site has an average of 469 hostnames per day whose lookups differ from all others. This set of names is fairly stable, and an examination of their contents suggests that it is policy-driven censorship, since they are resolved to IP addresses that provide no responses. Users will be able to seemingly resolve the name, but will be unable to contact any machine at the address, and may conclude the server does not exist. The second-worst site appears to have a traffic-sniffing virus checker working in conjunction with the local DNS resolver. When it activates, all lookups from the client are directed to a local Webserver with a message warning that your client is infected. Unfortunately, the virus snif-



(a) Region Sizes (b) Region Imbalances (c) Site Poisoning

Figure 2: Region information for names mapping to multiple disjoint IP addresses



(a) All names (b) Names with 2-10 regions (c) Names with 10+ regions

Figure 3: Rate of change for name-to-IP mappings. The bars, from bottom to top, are for zero changes, 1 change, 2-3 changes, 4-14 changes, and 15+ changes (in white). **Note: Y axis truncated to improve detail in first graph.**

fer returns false positives, and indicated that our Linux-based boxes were infected with Windows viruses. The third-worst site appeared to be having sporadic failures, and was randomly returning the IP address of the school's main Web server for queries, with no discernible pattern to its behavior. The remaining sites show no strong patterns of poisoning, with most of the imbalances stemming from slowly-deployed changes in name-to-IP mappings. In all of these cases, any multi-site agreement policy in ConfiDNS would automatically prevent these sites from poisoning the lookup results.

3.2 IP Address Changes

By monitoring over a period of 30 days, we can see how often name-to-IP mappings really change. While past history is no guarantee, if a mapping has been the same for an extended period of time, users may have more confidence in it. Conversely, a previous stable mapping that suddenly changes may be cause for concern – it may be as simple as a server being replaced or migrated, or it may be that an attacker trying to divert traffic.

We calculate the rate of change of name-to-IP mappings during our test period by counting the number of times the returned IP differs from the previous day's value on each site. The change counts are shown in Figure 3(a). For each site, we group names by the number of changes observed, and represent these counts as a stacked bar. For example, node 0 sees no changes for 85% of names, one change for 7% of names, 2-3 changes for 3% of names, 4-14 changes for 3% of names, and 15 or more changes for the final 2% of names. At every site, more than 85% of names did not change at all in 30 days. The remaining bars group the number of changes and show that while some names change on virtually every lookup, others change much more slowly. We see that most names are stable for a month at a time, and more than half of the names that change are stable for two weeks. On average, only 2% of these names change IP addresses more than once per week.

Figure 3(b) examines names with a small number of regions, and indicates that even here, a large number of names have long periods of stability – decisions to send clients to nearby data centers are likely to be stable over time. Figure 3(c) shows the same statistics for those names that map to more than 10 regions, including most of the Akamai-served domains, some domains served by LimeLight Networks (another CDN), and others. The in-

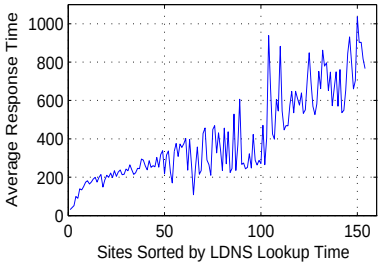


Figure 4: CoDNS

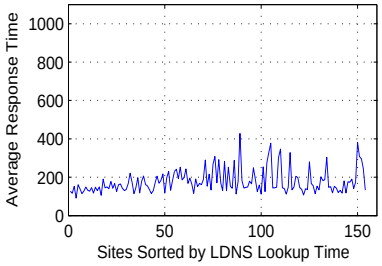


Figure 5: 3 sites from 10 peers

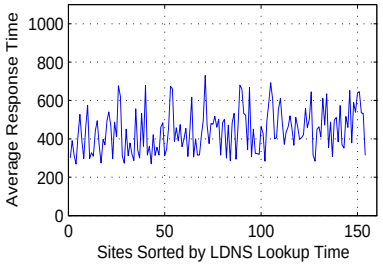


Figure 6: 7 sites from 30 peers

crease in the count of zero changes beginning near node 100 is largely a function of the size and deployment of Akamai clusters – these do not appear to use hardware load balancers, so the larger the cluster, the more IP addresses get exposed and rotated, causing high rates of IP address changes. In contrast, Google clusters, despite having thousands of nodes, advertise only a small number of IP addresses as entry points.

4 Evaluation

We use this data to evaluate a number of ConfiDNS policies, beginning with policies that relate only to agreement, and then combining agreement and history. Our focus in this evaluation will be coverage (applicability) and latency, so we focus on how many hostnames and how many sites benefit from the various types of security the different policies provide.

We evaluate four agreement policies for ConfiDNS, varying the numbers of sites agreeing and peer set sizes. In one policy, we require that the agreement include the local DNS, but in the rest, the local DNS is just one of the sites in the agreement process. In each case, we restrict the number of peers that can be queried in order to reach agreement. Peers are selected solely by lowest RTT value from the node.

All policies are evaluated on every node, and per-node average latencies for a subset of policies are reported in Figures 4– 6. The baseline policy, using only the local DNS resolver, suffers from the problem of retries that we described in Section 3. Likewise, the policy of requiring that at least one other site (out of the five closest peers) agree with the local resolver has similar performance because the local DNS lookup time is the bottleneck. A simplified form of the CoDNS policy is shown in Figure 4, and takes the first response of the local resolver and the three nearest peers. The queries to the peers are staggered using the same delays CoDNS uses in deployment. It shows a significant response time improvement over the local DNS resolver, precisely because it does not have to wait on the local resolver when it is slow.

The more aggressive agreement policies for ConfiDNS require 3 of the 10 closes sites agreeing (Figure 5), 5 sites out of 20 agreeing (not shown), or 7 sites out of 30 agreeing (Figure 6). For practicality, we stagger lookups in these policies at the rate of 1000ms every 10 lookups for fairness. A policy of taking either the local DNS resolver and one peer, or 3 other peers, is not shown because its latency characteristics are identical to Figure 5. The most important latency observation for these more secure policies is that they perform much better than local DNS resolvers, and are in fact generally better than CoDNS. The 3-agreement policy performs surprisingly well, with an average latency almost half of CoDNS's.

	# Days IP is Stable				
	none	2-3	7	15	30
LDNS only	1	2	3	3	4
3 peers	2	3	3	4	4
LDNS + 1	3	3	4	4	5
5 peers	3	4	4	5	5
7 peers	4	4	5	5	6

Table 1: For simpler analysis, we linearize and collapse the range of protection policies down to single values, as shown above. Higher numbers indicate better protection than lower numbers.

We combine agreement and rate-of-change to determine the spectrum of protection policies that are possible, and how many hostnames per site can be satisfied with each. To simplify the analysis, we linearize the range of possibilities, as shown in Table 1. The process of assigning values to policies is subjective, but our goal was to give an idea of the strength of combinations, with higher numbers indicating better protection. The per-node breakdown for each label is shown in Figure 7.

The average breakdown for lookups that satisfy only label 1 (local DNS only) averages only 0.18%, which indicates that ConfiDNS improves the security of 99.82% (on average) of unique lookups from our collection of vantage points. Even if we pick a stronger security requirement, such as label 4, which indicates seven peers agreeing, 30 days of stability, or some intermediate combina-

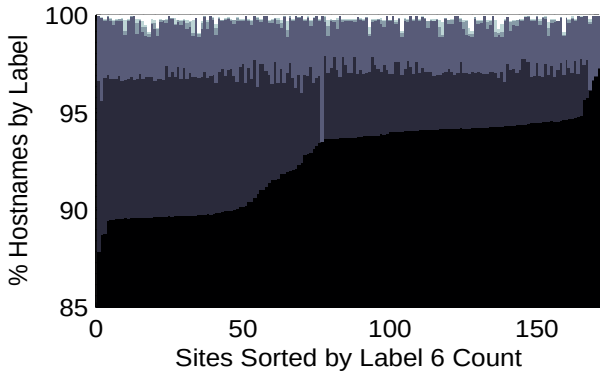


Figure 7: Breakdown of linearized policy labels by node. The bottom bar is linearized label 6, and the top bar (white) is label 1. **Note: Y axis truncated to show detail**

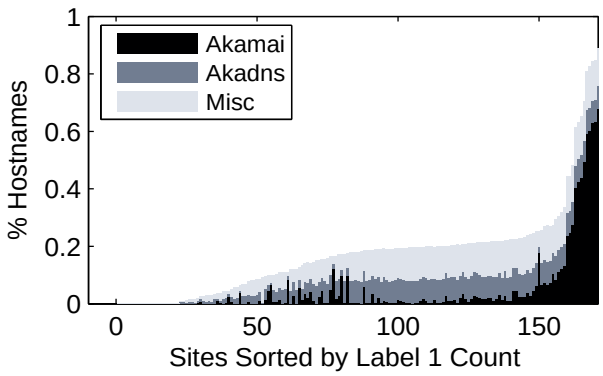


Figure 8: Breakdown of names in policy label 1

tions, ConfiDNS is able to satisfy 99.63% of the unique lookups. Even the strongest policy, with 7 peers agreeing and the lookup being stable for 30 days, still works for over 92% of unique lookups on average. As we showed in the earlier results, this extra security does not come at a high latency cost – latency is better than LDNS alone, and is comparable to CoDNS.

By using a policy that uses the best available security and reverts to LDNS otherwise, ConfiDNS can provide security that is never worse than LDNS. Users that are interested in lookups with security that is strictly better than LDNS will not be able to resolve 0.18% of unique names, assuming policies that meet at least label 2. To understand the types of names affected, we break down this fraction into three categories, as shown in Figure 8: those served by Akamai, those served using Akamai's DNS service, and other assorted names. Depending on how much trust the user has in one or more of these CDNs or DNS services, they could whitelist these names to further improve resolvability. The whitelist could take several forms, ranging from automatically whitelisting these names, whitelisting them if they fail to meet other security policies, or even whitelisting them as long as they map to a given set of vetted IP addresses. A full discussion of the options and their implications is beyond the scope of this paper.

5 Conclusion

We show that by using peer agreement and storing some past history, our new cooperative resolver, ConfiDNS, can provide *better* security than both traditional DNS resolvers as well as the previous cooperative approaches for the vast majority of domain names. This study also provides us with information on the real usage of DNS mappings, at a variety of domains ranging from small, singly-hosted sites to sophisticated replicated data centers with DNS redirection, and finally to commercial third-party content distribution networks. In all cases, we find that it is possible to leverage scale, history, or both, and provide a much more secure result than local DNS alone. All of these benefits are obtained without changing any server-side DNS infrastructure, making ConfiDNS incrementally deployable, requiring only a minimal agent running on either client machines or on client-side resolvers.

Acknowledgments

We would like to thank our shepherd, Mike Afergan, and the anonymous reviewers for their useful feedback on the paper. We would like to thank Onat Yazir for his data analysis. This work was supported in part by NSF Grants ANI-0335214, CNS-0439842, and CNS-0520053.

References

[1] Akamai. Content Delivery Network. http://www.akamai.com/.

[2] R. Arends, R. Austein, M.Larson, D. Massey, and S. Rose. Protocol modifications for the DNS security extensions. RFC 4033, March 2005.

[3] A. Lioy, F. Maino, M. Marian, and D. Mazzocchi. DNS security. In *TERENA Conference*, Lisbon, May 2000.

[4] P. Mockapetris and K. Dunlap. Development of the Domain Name System. In *SIGCOMM*, Stanford, CA, Aug 1988.

[5] K. Park, V. S. Pai, L. Peterson, and Z. Wang. CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups. In *OSDI*, San Franscisco, CA, December 2004.

[6] PlanetLab. An open testbed for developing, deploying and accessing planetary-scale services, September 2002. http://www.planet-lab.org/.

[7] V. Ramasubramanian and E. G. Sirer. The Design and Implementation of a Next Generation Name Service for the Internet. In *SIGCOMM*, Portland, OR, Aug 2004.

[8] L. Wang, K. Park, R. Pang, V. Pai, and L. Peterson. Reliability and Security in the CoDeeN Content Distribution Network. In *In Proceedings of the USENIX Annual Technical Conference*, Boston, MA, June 2004.