Impact of Configuration Errors on DNS Robustness

Vasileios Pappas, Duane Wessels, Daniel Massey, Songwu Lu, Andreas Terzis, and Lixia Zhang

***Abstract*—During the past twenty years the Domain Name System (DNS) has sustained phenomenal growth while maintaining satisfactory user-level performance. However, the original design focused mainly on system robustness against physical failures, and neglected the impact of operational errors such as misconfigurations. Our measurement efforts have revealed a number of misconfigurations in DNS today: delegation inconsistency, lame delegation, diminished server redundancy, and cyclic zone dependency. Zones with configuration errors suffer from reduced availability and increased query delays up to an order of magnitude. The original DNS design assumed that redundant DNS servers fail independently, but our measurements show that operational choices create dependencies between servers. We found that, left unchecked, DNS configuration errors are widespread. Specifically, lame delegation affects 15% of the measured DNS zones, delegation inconsistency appears in 21% of the zones, diminished server redundancy is even more prevalent, and cyclic dependency appears in 2% of the zones. We also noted that the degrees of misconfiguration vary from zone to zone, with the most popular zones having the lowest percentage of errors. Our results indicate that DNS, as well as any other truly robust large-scale system, must include systematic checking mechanisms to cope with operational errors.**

***Index Terms*—Domain Name System, configurations errors, resiliency, measurements, performance.**

I. Introduction

THE DOMAIN Name System (DNS) is one of the most successfully designed and operated Internet services today. It provides a fundamental service for end users, i.e., name resolution, and it is used by various applications ranging from load balancing to service discovery. In essence, DNS is a global scale hierarchical database, managed in a fully distributed manner. It seeks to provide acceptable performance without setting any limit on the size of the database [26]. According to ISC [5], the number of host records, one of several record types carried by DNS, has grown from 20,000 in 1987 to 541,677,360 in January 2008. Despite such phenomenal growth in size, DNS has been able to deliver satisfactory performance at the user level. Jung *et al.* [18] showed that, on average, a DNS query is answered by sending 1.3 messages and the mean resolution latency is less than 100ms. Moreover, DNS has been able to meet unforeseen challenges. For example, DNS survived widely publicized DDoS attacks targeted at the root name servers in October 2002 [12] and in February 2007 [4], demonstrating the robustness of its distributed design against brute-force attacks.

Manuscript received 1 March 2008; revised 15 November 2008.
V. Pappas is with IBM Research (e-mail: vpappas@us.ibm.com).
D. Wessels is with the Measurement Factory.
D. Massey is with the CS Department at Colorado State University.
A. Terzis is with the CS Department at Johns Hopkins University.
S. Lu and L. Zhang are with the CS Department at UCLA.
Digital Object Identifier 10.1109/JSAC.2009.090404.

Despite its tremendous success, DNS is not without weakness. The critical importance of the DNS places a high standard on its resilience and its design warrants further examination, as evidenced by the following example. During January 2001 all the authoritative servers for the Microsoft DNS domain became inaccessible [11]. This failure was due to a simple configuration mistake where Microsoft placed all its DNS servers behind the same network router, despite the well documented guidelines on geographically dispersing DNS servers [9], and the switch failed. During this event, the number of DNS queries for the Microsoft domain seen at the F root server surged from the normal 0.003% of all the queries to over 25%. We speculate that other root servers were likely to have observed similar increase of queries for Microsoft domain.

The previous example illustrates that a mistake in configuring a specific DNS zone can have globally adverse impact. While several studies have evaluated various aspects of the DNS performance [13], [18], [20], [21], there has been no systematic study to quantify the pervasiveness and impact of DNS configuration errors. In this work, we study the effect of misconfigurations on DNS robustness and performance. We used a combination of passive and active measurements to study the type and the extent of misconfigurations observed in the global DNS infrastructure, and the impact these misconfigurations have on DNS query response times and service availability. The passive measurement traces were collected from a typical university campus environment. The active measurements were done by querying a sample set of DNS zones randomly selected from the ISC reverse zone files [5]. The active measurements span a period of multiple years, from 2004 to 2007. Interestingly enough, the appearance of these errors persisted throughout this long period of time. In this paper we report our measurements results on four major classes of DNS configuration errors.

The first type of error that we consider is *delegation inconsistencies*. This error occurs when a parent zone points to a set of name servers for the child zone that is different from the set stored at the child zone. Although it is not mandatory for both zones to provide identical delegation records, this inconsistency can reduce the total number of nameservers that a DNS resolver may use to reach the child zone. Our measurement results show that this type of configuration appears in 21% of the DNS zones on average, and around 40% of zones with delegation inconsistencies list less than half of the servers at the parent zone.

Second, we present results on *lame delegation*, an error that occurs when either the parent or the child zone point to one or more non-existing name servers for the child zone. Our measurements show that on average 15% of the DNS zones

suffer from lame delegations, and that 70% of these lame delegation cases reduced the number of available nameservers for a zone by half. Because DNS queries sent to non-existing servers do not get a reply and have to be resent to a different server, lame delegation translates to increased query delay. While DNS queries are normally resolved within 100ms, queries sent to lame delegated servers resolve in 3 seconds on average, a thirty-fold increase in response time.

Third, although DNS design documents [9], [17] call for diverse placement of a zone's authoritative servers, our measurements show that this guideline is frequently violated which leads to *diminished server redundancy*. Among the zones we measured, 45% of the zones have all their servers under the same /24 address prefix, 82% of the zones place their servers in the same geographical region, and 77% of the zones have all their servers in a single Autonomous System (AS). Servers placed behind the same /24 prefix are vulnerable to routing anomalies affecting that prefix, servers located in the same geographic region are vulnerable to regional failures, such as fiber cuts or power outages. Similarly, servers placed within the same AS can be simultaneously affected by failures within the AS.

Fourth, we report results on a subtle type of configuration error, that we call *cyclic zone dependency*. These errors happen when information required to resolve a name in zone X depends on information in zone Y which in turn depends back on zone X. While the percentage of the zones involved in such misconfigurations seems relatively small, 2% based on our measurements, these cases are more difficult to detect. Cyclic zone dependencies reduce a zone's availability and may substantially increase query response times. Our results show that, for zones involved in cyclic zone dependencies, the error may cause more than 25% of the zone's servers to become unreachable.

Our key contribution in this work goes beyond the above specific results. Our data shows that, even for a seemingly well-functioning system such as DNS, human errors exist to a surprising extent. These errors lurk under the surface, making the degree of system resiliency for those affected zones significantly lower than that indicated by the number of redundant servers. Such unanticipated reduction of resiliency may incur grave consequences when unexpected failures occur. Enhancing DNS resilience requires systematic detection and reaction mechanisms to handle operational errors.

The rest of the paper is structured as follows. In Section II we give a brief overview of the DNS design. We present the methodology used to measure the different types of DNS misconfigurations in the global DNS system in Section III. In Section IV we describe results from our measurements. We discuss the implications of our findings in Section V, and compare with the related work in Section VI. We conclude the paper in Section VII.

II. Background

In DNS parlance, the namespace is divided into a large number of *zones*. Roughly speaking, each zone is authoritative for the names that share the same suffix with the zone's domain name. A domain may delegate part of its namespace to another domain known as a subdomain. For example, the *ucla.edu* zone delegated the *cs.ucla.edu* namespace to create a subdomain. This delegation procedure results in an inverted tree structure with each node being a domain and each edge representing a delegation point. The *root* zone resides at the top of this tree structure. Generic top-level domains (gTLDs), such as *edu*, and country code top-level domains (ccTLDs) appear directly below the root domain.

Each zone stores the resource records (RRs) associated with the domains under its authority. There are a variety of RR types, with the most common being the A records used to map names to IPv4 addresses. Each record has a time to live value (TTL) which specifies the maximum lifetime it may be cached. For example, the A record for *www.ucla.edu* contains a value of 169.232.33.135 and may be cached for 300 seconds.

All the resource records belonging to a zone are available from a set of *authoritative name-servers* (ANS) operated by the zone. Each authoritative server is identified by an NS record stored at both the zone itself and its parent. NS records point to the *name* of the authoritative name-server rather than its IP address. As such, one needs both the NS and A records of a domain's servers to make contact. We call the set of NS and A records that are associated with the authoritative name servers *infrastructure resource records* (IRRs). Essentially, IRRs construct the DNS hierarchical relations between domains.

Client applications typically retrieve a desired record, of domain D, by querying a stub resolver: a DNS element which is implemented in every operating system. A stub resolver will then forward the query to a *local caching server* (LCS) which attempts to obtain the requested record.

Before querying other servers for the target record, the local caching server will search it own cache. If the record is present and its TTL has not expired, the LCS will immediately respond to the stub resolver. Otherwise, the local caching server will attempt to find the desired record by querying one of D's authoritative name servers. Despite the absence of the original request from the its cache, the LCS may still have a record of one of D's name-servers on hand. Without the ANS's records, the local caching server is forced to query D's parent zone. If the LCS does not know any infrastructure resource records for D or D's parent, it will try to find an IRR of the nearest ancestor. Every local caching server is hard-coded with the IRRs of the root zone and may start there if no better records are known.

Although zone administration is autonomous, some inter-zone coordination is required to maintain the DNS hierarchy. In particular, the parent zone must provide information on how to reach its children's ANS. Each child provides its parent with a list of authoritative servers, more precisely an NS RRset for the child zone. The parent stores this NS RRset, and refers the resolvers to the child zone by including this NS RRset in the responses. Ideally, the NS RRset stored at the parent should match the child's set of authoritative servers exactly, although the DNS design provides no mechanism to ensure this consistency. DNS continues to work as long as the parent NS RRset correctly identifies at least one authoritative server.

III. Measurement Methodology

The DNS design, as described in Section II, assumes that operators correctly configure and manage the zones. In particular, reliable DNS operations depend on the following correct actions: appropriate placement of redundant servers for high availability, manual input of each zone's database for correct setting, and coordination between parent and child zones for consistency. We broadly define human errors in any of these actions as *configuration errors*.

It is well known in the operational community that DNS configuration errors exist [15], [22]. However, there has been no systematic study to quantify their pervasiveness and impact. In this work, we conducted large-scale measurements to assess the pervasiveness of configuration errors, their effect on the user-perceived performance, and their impact on DNS robustness. We conducted two types of measurements, passive and active, as described next.

A. Passive Measurements

We collected two sets of DNS traces from the UCLA Computer Science department network. The first set was collected during the week of 8/29/03–9/5/03, and the second set during 9/27/03–10/4/03. Table I summarizes the total number of queries and responses in each set, as well as the total number of second level domains queried. We use the last two labels of a domain name to estimate the number of second level domains[1].

Our data collection observes DNS packets sent over the department's external links and captures all the DNS packets exchanged between the department and external sites. We count only the DNS traffic exchanges with external sites; we exclude the local DNS traffic between end hosts and the local caching servers. We also exclude all DNS queries sent to the department's authoritative servers. We measure the delay between the first query packet and the final response. In other words, we measure the delay associated with obtaining an answer that is not present in the local server's cache. We also analyze the content of each intermediate response packet to detect whether it reflects any configuration errors.

Given that the data-set is taken from a university environment, it is possible that there is a bias in the interests of the observed user population. Therefore, the visited zones may not be a representative sample of the entire DNS space, and the number of configuration errors among all the visited zones may not give a good quantitative estimate on the degree of configuration errors in general. However, for all the cases of DNS configuration errors that we identified, our measured delay should serve as a good estimate of the DNS query delay in the presence of configuration errors.

B. Active Measurements

To overcome the potential bias in the passive measurement and to gauge the pervasiveness of DNS misconfiguration errors, we also conducted active measurements. We implemented a specialized DNS resolver and used it to query a randomly selected subset of the DNS namespace. Our resolver added the following additional features on top of the standard resolver function. First, when it receives a referral for zone Z with a list of DNS servers for Z, it sends a query to *each* of the servers to verify whether all of them can provide correct replies. Second, it attempts to make use of the DNS zone transfer functionality to retrieve the entire zone data which allows us to determine the number of delegations and compare the results for the various delegations. We utilized external information, such as BGP tables [2] and geographic location information [1], to estimate the topological and geographic locations of the authoritative servers.

Our active measurements use three sample sets of DNS zones. To create Sample 1, we used the ISC reverse zone data [5] to obtain the pool of zones used for the active measurements. The ISC data included PTR records that map IP address to DNS names, and we examined the names in the PTR data field. From these fully qualified domain names, we stripped off the first label to obtain a *potential* zone name. We then used a set of relatively simple sanity checks to eliminate bogus entries that included non-existent top level domains. This left us with a pool of about two million potential zone names, from which we picked 3% of the names through a uniformly random selection. Finally, we queried each selected name to verify that it indeed belonged to an existing zone. This created our measurement set Sample 1, as shown in Table II.

Sample 2 was used in order to measure the pervasiveness of configuration errors across time. Sample 2 was created in a similar way as Sample 1 by sampling 10% of the domains that are listed under the *com* and *net* zones [3]. Sample 2 contains only second level domains. Note also that Sample 2 consists of three different samples taken at the years 2005, 2006 and 2007. The use of different sample for each year was necessary given the fact that the *com* and *net* databases change considerably within a period of one year.

Finally we created Sample 3 in order to measure the pervasiveness of misconfiguration errors in "important" zones, i.e., zones that host at least one popular Web server. We collected the 500 most popular Web servers by combining the Web server rating information given by two independent sources [7], [29]. While Samples 1 and 2 gauge the extent of misconfiguration errors in the current DNS infrastructure, we use Sample 3 to estimate how frequently a typical user may be affected by these errors.

IV. Measurement Results and Analysis

In this section, we describe in more detail the four types of DNS configuration errors and provide measurement results for each of them.

A. Delegation Inconsistency

When a parent zone P delegates part of its namespace to a child zone C, P stores a list of NS resource records for the authoritative servers of zone C. This list of NS resource records are kept both at the parent and the child zone. Whenever the operator of zone C makes changes to one or

[1]Labels are separated by ".". However, the presence of "." in a name does not necessarily signify a zone delegation. For example, "a.b.c.example" may belong to zone "b.c.example" or zone "c.example" if there was no delegation to "b.c.example". This ambiguity can occur at any level, but the presence of a "." near the last labels does tend to indicate a delegation and this allows us to reasonably infer the second level zone name from the query name.

TABLE I
DNS PACKET TRACES USED FOR THE PASSIVE MEASUREMENTS

	Trace Period	Number of Queries	Number of Responses	Level 2 Domains
Trace 1	08/29/2003 - 09/05/2003	2,470,370	2,284,744	54,564
Trace 2	09/27/2003 - 10/04/2003	3,097,028	2,693,907	56,058

TABLE II
DNS ZONES USED FOR THE ACTIVE MEASUREMENTS

	Number of Zones	Type of Sampling
Sample 1	51,515 (Year 2004)	Random Sampling
Sample 2	1,311,909 (Year 2005) 1,857,393 (Year 2006) 2,703,764 (Year 2007)	Random Sampling
Sample 3	500 (Year 2004)	500 most popular

TABLE III
DELEGATION INCONSISTENCY EXAMPLES

Example 1		(Date: 12/07/03)	Example 2		(Date: 12/07/03)
$ORIGIN com.			$ORIGIN com.		
charapro.com.	NS	ns3.firstserver.ne.jp.	cornellmedclaim.com.	NS	ns1.covad.net.
charapro.com.	NS	ns4.firstserver.ne.jp.	cornellmedclaim.com.	NS	ns2.covad.net.
$ORIGIN charapro.com.			$ORIGIN cornellmedclaim.com.		
charapro.com.	NS	ns1.firstserver.ne.jp.	cornellmedclaim.com.	NS	ns1.meganameservers.com.
charapro.com.	NS	ns2.firstserver.ne.jp.	cornellmedclaim.com.	NS	ns2.meganameservers.com.
charapro.com.	NS	ns3.firstserver.ne.jp.	cornellmedclaim.com.	NS	ns3.meganameservers.com.
charapro.com.	NS	ns4.firstserver.ne.jp.			

more of C's authoritative servers, he must coordinate with the operator for zone P to update P accordingly. In reality, there are cases where changes made at the child zone are not reflected at the parent zone, usually due to "bad" coordination between them. As a consequence, the NS RR set of the child zone can be completely different from the NS RR set of the parent zone. However, in most cases the child's NS RR set is either a superset or a subset of the NS RR set listed at the parent zone.

Table III shows the configuration of two zones, captured during our measurements, with delegation inconsistency errors. The first example shows that the *charapro.com* zone had four authoritative servers, the *ns{1,2,3,4}.firstserver.ne.jp*, as it was indicated by the NS RRs stored at this zone. On the other hand, its parent zone, the *com*, stored only two of these four servers. The second example shows a different case of delegation inconsistency, where the parent of the *cornellmedclaim.com* zone provided a completely different set of name servers, compared with the one given by the child zone. The child zone stored three servers, the *ns{1,2,3}.meganameservers.com*, and the parent, the *com* zone, stored two different ones, the *ns{1,2}.covad.net*. By directly asking all these servers for the authoritative servers of the *cornellmedclaim.com* zone, we found that all servers were authoritative and all of them listed as authoritative only the first set of three servers.

In our measurements, we a say a delegation inconsistency occurs anytime the list of servers at the parent zone does not exactly match the list of servers at the child zone. Inconsistencies can occur if a server is listed at the parent, but not listed at the child. For example, server ns1.covad.net is listed in the com (parent) zone, but not listed in the cornellmedclaim.com (child zone). This is a configuration error because an NS RR set is always a property of the child zone [16] and thus the NS records at the parent zone should always appear at the child zone as well. Inconsistencies can also occur if a server is listed at the child, but not at the parent zone. For example, server ns1.firstserver.ne.jp from case 1 is listed at charapro.com, but not listed at com. The DNS specification is ambiguous in this case; it could be an unintentional error or could also be a legitimate configuration. Even though the parent and the child zone are not required to list the same NS RR set, delegation inconsistency errors can affect the availability of a zone. That is because DNS resolvers are allowed to cache either the parent's or the child's NS RR set, but they can never merge both records [16]. Thus, in the case of delegation inconsistency errors, they can use only a subset of the servers that are authoritative for the zone. In the two examples shown above, when a resolver queries the parent zone, it receives replies that list only 50% and 40% of the authoritative servers respectively.

1) Measurement Details: We used our custom-built DNS resolver to assess the pervasiveness of delegation inconsistency errors. The resolver identifies those errors in the following way: First, for each zone C the resolver iteratively queries the DNS system, starting at the root servers, until it reaches the parent zone P of C. At each step, the resolver queries for the SOA (Start of Authority) resource record of zone C; it uses previously cached entries whenever they are available to avoid unnecessary queries. The iterations stop when the resolver queries a server of the parent zone P, which replies with a referral to the servers of zone C, i.e. with the NS RR set of the child that is stored at the parent. Up to this point, our resolver behaves as a standard DNS resolver. Next, the resolver queries each of these servers, that are provided as a referral from the parent zone P, and asks for the NS RR set of zone C. It then checks if these servers can provide authoritative answers, as explained in section IV-B. For the servers that provide authoritative answers, it compares the NS RR set returned by these servers and the NS RR set returned by the parent zone. If these two sets of records are not identical, a delegation inconsistency error is detected.

2) Measurement Results: Our main goal is to obtain a quantitative estimate of the pervasiveness of delegation inconsistency among DNS zones. At the same time, we attempt to identify whether there is any relation between the occurrence of delegation inconsistency errors and the zone's geographic location, the depth of the zone in the DNS hierarchy, and the

TABLE IV
DELEGATION INCONSISTENCY ACROSS TIME (SAMPLE 2)

Year	2005	2006	2007
Delegation Inconsistency	39.5%	47.3%	52.1%

number of delegations associated with the zone. We define the number of delegations associated with a zone as the zone's *family size*.

Figure 1 shows the percentage of zones that have delegation inconsistency errors, grouped by top level domains. These results are based on measurements done with Sample 1 and the selected TLDs are a representative sample of gTLDs and ccTLDs. By representative we mean two things: i) there is a large number of zones in our samples that belong to the selected TLD; and ii) for the case of ccTLDs, the selected TLDs cover different continents. The figure shows that many ccTLDs of the RIPE region[2] have zones that are rarely affected by delegation inconsistency errors. On the other hand most gTLDs and many ccTLDs of the APNIC region[3] have a considerably higher percentage of delegation inconsistency errors.

We repeated the same measurements using Sample 3 in order to gauge the pervasiveness of these errors on the "most popular" zones. The results show that that 25% of the most popular zones have delegation inconsistency errors, and 18% of the top 100 zones suffer from these errors, compared to 21.4% for randomly chosen zones. In addition, in 15.6% of the top 500 zones the delegation inconsistency is due to the fact that the parent zone does not list a server appearing at the child zone, and in 18.8% of the top 500 zones the error is because the child zone does not list a server that appears at the parent zone. For a randomly chosen zone the above results are 13.14% and 16.82% respectively.

Table IV gives the delegation inconsistency errors as measured in 2005, 2006 and 2007, by using Sample 2. These results are indicative that year by year delegation inconsistency errors become more widespread.

3) Impact of Delegation Inconsistency: Figure 2 depicts the impact that delegation inconsistency errors have on the availability of the DNS zones. The upper graph of that figure gives the CDF for the percentage of servers that are authoritative for a zone and that they do not appear in the NS RR set of its parent zone. It is evident that in 40% of the cases the parent zone lists less than half of the total servers. Similarly the lower graph at the same figure gives the CDF for the percentage of servers that do not appear at the child zone. Again we see that in 40% of the cases the child zone lists less than 50% of its authoritative servers.

We must note that in cases that the child zone does not list a server appearing at the parent zone, it is quite possible that this server may not be authoritative (lame server) for the zone and thus by including it may introduce additional problems. We found that the delegation inconsistency errors are correlated with lame delegation errors, another type of DNS misconfiguration presented in the next section. Specifically, 31% of zones with delegation inconsistency errors appear to suffer from lame delegation errors, as well.

[2]The uk, de, fr, nl, gr, ch, se, be, dk and pl zones.
[3]The jp, kr, cn, tw, au zones.

B. Lame Delegation

A lame delegation occurs when a DNS server that is listed as an authoritative server for a zone cannot provide authoritative answers for names in that zone. Zones with lame delegation often have delegation inconsistencies, but lame delegation and delegation inconsistency are distinct problems. Recall that in a delegation inconsistency, the set of servers listed at the parent is not equal to the set of servers listed at the parent. In case 2 from the previous section, ns1.covad.net was listed at the parent and not listed at the child. This is an inconsistency. However, ns1.covad.net does provide authoritative answers for the child and thus is not a lame delegation. At the same, it possible that all servers are consistently listed in both the parent and child zones and lame delegation can still occur, if any of the listed servers does not provide authoritative answers. Thus one can have delegation inconsistencies that are not lame delegations and can have lame delegations that are not inconsistencies.

But in practice, the lame delegation and delegation inconsistencies are often correlated. For example, suppose the authoritative servers for a zone change. These changes are reflected in the NS RRs of the child zone, but are not reflected at the parent zone. The delegation is now inconsistent since the child has the new server set and the parent still has the old server set. Some of the old NS RRs at the parent zone now point to either non-existing or non-authoritative servers. These servers don't respond and thus the delegation is also lame.

Table V shows the configuration of two DNS zones that had lame delegation errors. The *com* zone had three NS records for the *araconstruction.com* zone, pointing to servers *ns.cliftontechnology.com*, *ns-east.cerf.net*, and *ns-west.cerf.net*, respectively. When a query for a name belonging to the *araconstruction.com* zone was sent to each of the three servers, only the first one replied with an authoritative answer; the other two servers replied with a referral to the servers for the *com* zone, indicating that they were not authoritative servers for the *araconstruction.com* zone. In the second example, *com* zone indicated that the zone *virginiagardens.com* was served by two name servers. When queried, however, only the first one provided authoritative answers; the second server, *ns2.whro.net*, did not respond at all.

The existence of lame delegations can affect DNS performance in at least two ways. First, it decreases the zone's availability: in the previous examples, out of a seemingly redundant set of servers for both zones, only a *single* server served each zone. Second, it increases the query response time. Queries sent to lame servers either do not elicit any answer, in which case a resolver waits until the query timer expires (usually set to 3-4 seconds), or receive a useless referral. In either case the resolver has to contact the next authoritative server until it receives an authoritative answer for the zone, or gives up after retrying all the known authoritative servers for the zone [24]. In the best case a lame server may reply with a non-authoritative answer to a query if it happens to have cached the queried name.

1) Measurement Details: We use our custom-built DNS resolver to gauge the pervasiveness of lame delegation errors.

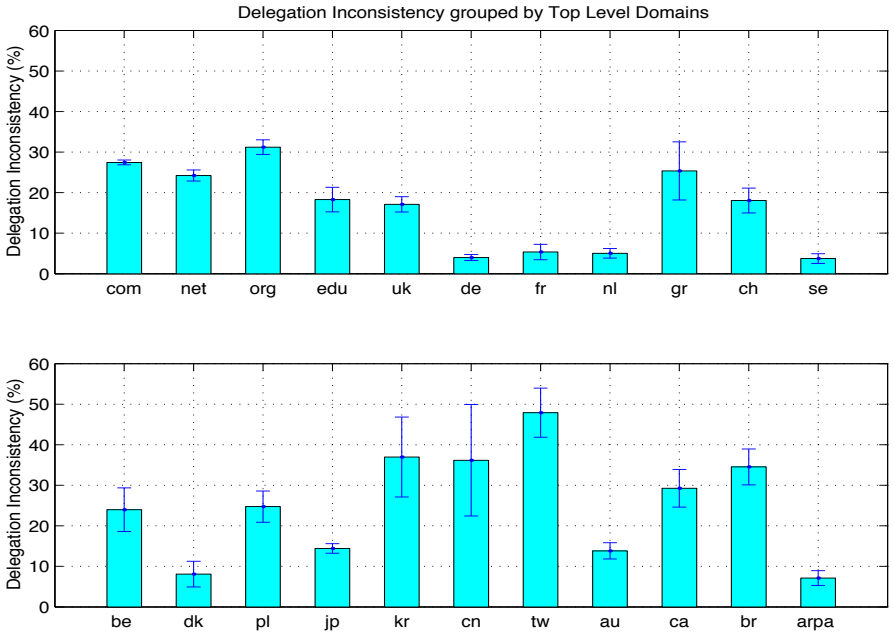


Fig. 1. Percentage of zones with delegation inconsistency errors, grouped by TLDs (based on Sample 1). Delegation inconsistency errors appear on average in 21% of the zones.

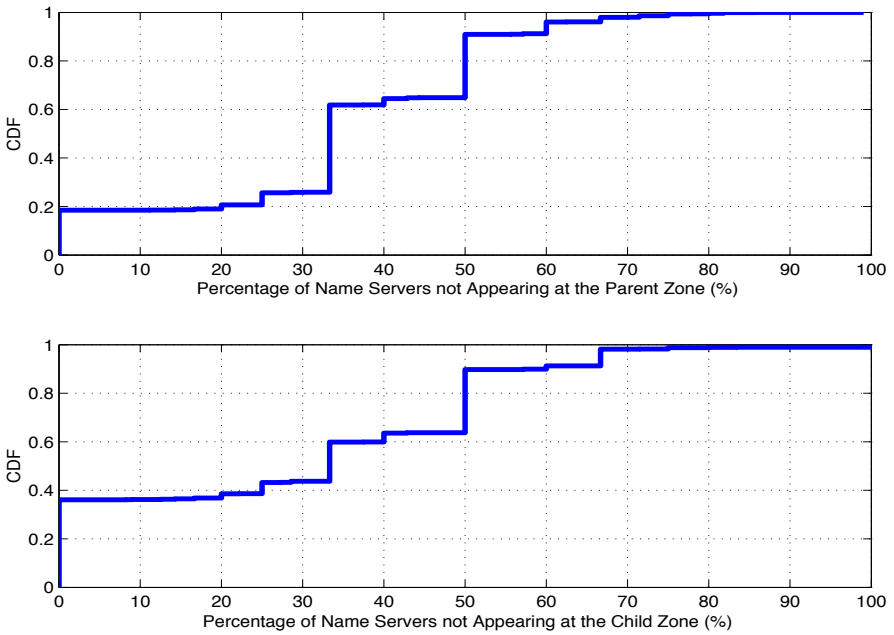


Fig. 2. A) Percentage of servers that do not appear at the parent's NS RR set. B) Percentage of servers that do not appear at the child's NS RR set.

TABLE V
LAME DELEGATION EXAMPLES

Example 1		(Date: 12/07/03)	Example 2		(Date: 12/07/03)
$ORIGIN com.			$ORIGIN com.		
araconstruction.com.	NS	ns.cliftontechnology.com.	virginiagardens.com.	NS	ns1.whro.net.
araconstruction.com.	NS	ns-east.cerf.net.	virginiagardens.com.	NS	ns2.whro.net.
araconstruction.com.	NS	ns-west.cerf.net.			

TABLE VI
TYPES OF LAME DELEGATION

gTLD	Type I (%)	Type II (%)	Type III (%)
com	47.51±2.75	4.11±1.90	48.22±2.75
net	52.61±6.45	3.48±2.37	43.41±6.41
org	42.78±6.96	3.61±2.62	53.61±7.02
edu	45.83±14.10	6.25±5.85	47.92±14.13

First, as in the case of delegation inconsistency errors, for each zone C the resolver iteratively queries the DNS system, starting at the root servers, until it reaches the parent zone P of C, which replies with a referral to the servers of zone C. Note that in a small number of cases, one server may be authoritative for both the child and its grandparent zones and thus our resolver never encounters the parent servers. We account for this case by using the child NS RR set to test for lame delegations in such rare cases [4].

Next, the resolver tests whether all the servers, returned by the referral from the parent zone P, are indeed authoritative servers for the child zone C. An authoritative server should reply with the actual answer and the DNS header AA (authoritative answer) bit set. Note that if the zone C exists then the SOA resource record is always stored in C's zone file, and thus we can always expect a valid answer. On the other hand if we do not get an answer to the SOA query, the server is considered to be lame for the zone. We sort lame delegations into the following three types based on what happens during the querying process:

- *Type I: non-responding server*. The server does not respond to DNS queries. This could be due to multiple reasons. For example, no machine is assigned to that IP address, a machine does exist but no DNS server listens on port 53, or even a (misconfigured) firewall is blocking DNS queries[5].
- *Type II: DNS error indication*. The server replies with an error indication (ServFail or Refused error code). These errors indicate that the server is not properly configured. This can possibly happen in cases of wrongly configured access lists and/or incorrectly defined views. [6].
- *Type III: non-authoritative answer*. The server does not return authoritative answers (the AA bit is not set). The server either replies with a referral to another zone, likely higher up in the DNS tree and usually the root zone, or replies with the actual answer, if the requested name happens to be locally cached [6].

2) Measurement Results: Figure 3 shows the percentage of zones that are lame delegated, grouped by top level domains. These results are based on measurements done with Sample 1. The figure shows that there is a relation between the pervasiveness of lame delegation and geographical distribution: most of the zones that belong to the RIPE region have a low percentage of lame delegations, lower than 10% in most cases, whereas many zones in the APNIC region have a lame delegation percentage higher than 20%. Zones belonging to gTLDs lie somewhere in-between, with the exception of the zones under $arpa$, which have a considerably higher percentage of lame delegations. Notably, the lame delegation and delegation inconsistency errors follow a similar pattern for most TLDs, which may serve as another indication that delegation inconsistencies and lame delegation errors are correlated.

Table VI shows the frequency for each of the three different types of lame delegations, grouped by the same four gTLDs. We observe that the first and third types of lame delegations be responsible for the majority of the cases. The first type, non-responding servers, accounted for nearly 50% of all the lame delegation cases and, as we will show later, this type of lame delegation can increase query response time by an order of magnitude.

We repeated the same measurements using Sample 3 in order to gauge the pervasiveness of these errors on the "most popular" zones. The results show that 7.6% of the most popular zones are lame delegated, and none of the top 100 zones has any lame delegation errors (compared with 15.1% of lame delegation for randomly chosen zones). These results indicate that, as one might expect, popular zones are much better administered compared with a randomly chosen zone. On the other hand, the results on the different types of lame delegation for the popular zones are consistent with the ones we obtain from Samples 1 and 2, with type I appearing in nearly 50% of the cases and type III in the other 50%.

TABLE VII
LAME DELEGATION ACROSS TIME (SAMPLE 2)

Year	2005	2006	2007
Lame Delegation	16.3%	27.4%	22.2%

Table VII gives the lame delegation errors as measured in 2005, 2006 and 2007, by using Sample 2. While these results do not provide any strong indication that lame delegation errors are on the rise (as in the case of delegation inconsistencies), they show that roughly 1 out of 5 zones has a lame delegation error.

3) Impact of Lame Delegation: The upper graph in Figure 4 shows the impact of lame delegation on zone availability. We plot the CDF for the percentage of unavailable servers in a lame delegated zone. We note that for about 70% of the zones which suffer from lame delegations, the number of available servers is reduced by 50% or more, that is, those zones are served by half or less of their servers.

The lower graph in Figure 4 shows the impact of lame delegation on query response time by using Trace 2 (results from Trace 1 were similar). They plot the CDF of the total response time for queries that encountered at least one lame server of type I, II or III, and the CDF of total response times for all the other queries. The results show that lame delegations increase the DNS response time considerably. For normal queries (those that do not encounter lame servers), the mean response time is about 60 msec; for queries that encountered at least one lame server of type I, the response time is longer than 3 seconds in most cases, and can even

[4]These cases appeared in 0.95% of the sampled zones

[5]In this paper we consider non-responding servers as lame if they haven't responded within one week period, while other definitions of lame delegation exclude this type of error

[6]Note that certain implementation of DNS resolvers cache servers with this type of lame delegation, so as not to query them in the future. Given that we use our own implementation of resolver, our active measurements are not affected.

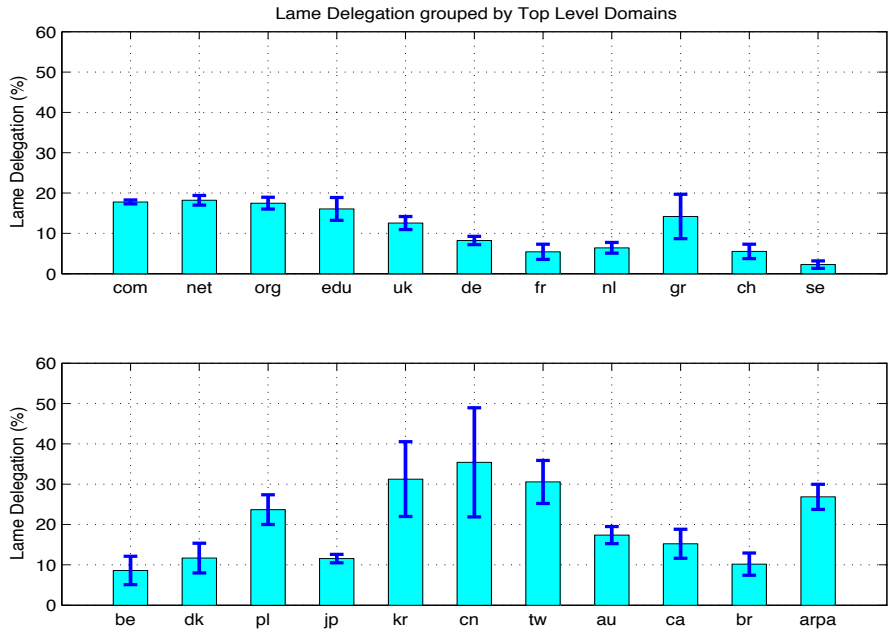


Fig. 3. Percentage of zones with lame delegation errors, grouped by TLDs (based on Sample 1). Lame delegation errors appear on average in 15% of the zones.

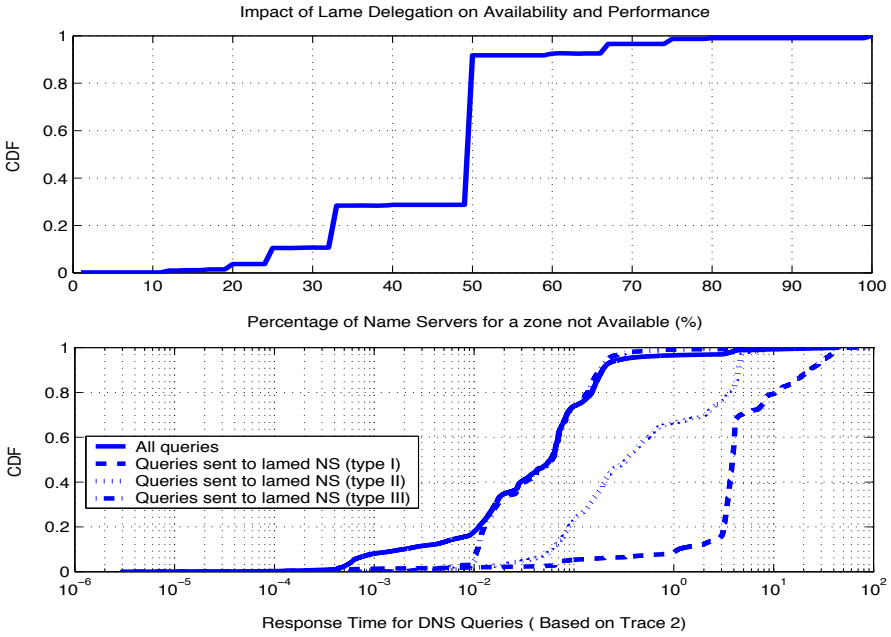


Fig. 4. Impact of lame delegation errors on DNS availability and performance

exceed 30 seconds in rare cases. Moreover, the response time for queries that encountered at least one lame server of type II is increased by several hundreds of milliseconds, compared to the response time for normal queries. Finally, queries sent to type III lame servers experience response times similar to normal queries. A possible explanation is that, during our measurements, the non-authoritative servers replied with the correct answer, which had happened to be locally cached due to a previous query sent from a caching server.

Finally, Table VIII gives the number of queries sent to lame servers. It shows that lame delegation related queries contributed around 8% of the total number of queries in Trace 1, and 13% of the queries in Trace 2. The total number of queries sent to lame servers depends highly on the users' profiles, thus we cannot conclude that these numbers represent typical cases for sites other than the ones we observed. Note that the percentage of queries that went to non-responding servers is much larger than other types of lame related queries; this is because the resolvers in our traces repeatedly sent queries to non-responding servers. One may also note that the number of type II queries is much higher than type III, while the number of zones containing a type II lame delegation is

TABLE VIII
NUMBER OF QUERIES SENT TO LAME SERVERS

Number of queries sent to lame servers	Type of Lame Delegation			
	Type I	Type II	Type III	All Queries
Trace 1	117,753	59,440	25,180	2,470,370
Trace 2	310,975	66,881	24,904	3,097,028

relatively small (see Table VI). Further examination shows that 92.6% of type II queries went to the *arpa* domain for Trace 1 and 88.4% for Trace 2, and queries that went to the *arpa* domain were about 20-30% of the total queries. Overall, these numbers show that traffic due to lame delegations can make a considerable portion of the total DNS traffic.

C. Diminished Server Redundancy

DNS uses redundancy as one of the two mechanisms for high availability - the other one is caching. The level of availability provided by redundant servers is a function not only of their number but also of their location. An operational server may not be able to answer DNS requests if the network path between the server and the clients is unavailable due to physical failures or routing problems. If all the replicated servers are connected to the same local network, then the redundancy is lost when that network fails. If all the servers are assigned addresses from the same address prefix, they will all be unavailable when that prefix becomes unreachable due to routing problems. If all the servers are in the same geographic location, natural disasters (e.g. earthquakes or floods) or large scale power failures may again cause all the replicated servers to fail. Therefore, to build redundancy against *unexpected* failures, the replicated servers must be placed in diverse locations that are unlikely to fail at the same time. Diverse server location not only increases service reliability but also reduces query response time since diversely placed servers can better cover widely distributed client populations.

Table IX shows different configurations of redundant servers for two sample zones. In the first case all the authoritative servers for the *bieszczady.pik-net.pl* zone were assigned addresses from the same /24 prefix, which was advertised by a single autonomous system. Furthermore, all of the servers were located behind the same last hop router, and they were also placed in the same geographic location. In contrast, the three authoritative servers for the *saxcompany.nl* zone were assigned addresses from different address prefixes, advertised from three different autonomous systems, and were located in three different cities (based on the NetGeo database [1]).

The need for diverse placement of redundant servers is a well known requirement in the DNS community. Several RFCs [17], [9] state the requirement for geographic distribution of the authoritative servers. However despite these RFC guidelines, operators often decide where to place the zone's DNS servers based on their *perception* of expected failures. In the case of Microsoft DNS incident in January 2001, operators (correctly) believed that redundant servers would have protected the domain against individual *server* failures but overlooked the possibility of *network* failures. The result was that the entire *microsoft.com* zone became unreachable when an unexpected router failure occurred. Unfortunately, in the absence of any systematic mechanism to enforce adequate diversity in server placement, the actual decision is often made under the constraints of cost and management by operators who may have partial understanding of the consequence.

We believe that diverse placement of redundant servers should be a requirement for both large and small organizations. One could imagine a scenario where all the hosts of a small organization depend on a single upstream link to the Internet. If this upstream link fails, none of the hosts in the organization will be reachable. Even in this case, diverse server placement has clear advantages: If all the organization's DNS servers are placed behind the failed upstream link, resolvers will slowly try each server before finally abandoning the query. This potentially adds a long delay before the application learns the host is unresolvable. Furthermore, applications (and users) may respond differently based on a "hostname unresolvable" or "host unreachable". One should not assume these two errors are equivalent for all applications. By keeping DNS service always available, applications (and users) can learn the desired host IP address exists. Then they can determine whether the host is reachable and respond appropriately.

The need for diverse placement of redundant servers goes even beyond the availability issue of affected zones. According to DNS specification (Section 5 of [23]), a resolver may go back to query the root servers if it fails to receive a response from any of the servers for a requested zone. There is at least one popular DNS resolver implementation which, after failing to receive response from a requested zone, continuously queries the DNS servers of higher level zones [19]. Such DNS implementations introduce a coupling between the availability of any specific zone's DNS servers and the load put on the top level DNS servers, resulting in undesired global impact due to local failures.

Unlike the lame delegation problem, the lack of diversity in server placements is not easily observed, making the problem difficult to detect. Because failures are rare events, and large scale failures are more so, even zones with vulnerable server placements appear to work correctly under normal conditions. Thus, administrators often discover the underlying vulnerability only *after* a failure disables the service.

1) Measurement Details: Our main goal is to estimate the number of authoritative servers that have independent failure modes. In order to measure the server redundancy we need to estimate whether two servers share a common point of failure. Specifically, we try to identify if two servers share the same network subnet, which in most cases implies that they are behind the same router, if they are served by the same autonomous system, and if they are placed in the same geographic location.

Since we do not know how remote networks allocate addresses, the results presented in this section assume a subnet prefix length of /24 since it is the most commonly used one. We also tried a variable size prefix length, ranging from /24 to /30, to decide whether two servers were in the same subnet, and did not observe any substantial differences compared with using /24 prefixes. We use the BGP routing tables provided by the RouteViews project [2] to locate the AS that each server is located in. We perform a longest prefix match in the BGP table for the server's IP address; the server is assumed to reside in the last AS on the AS Path associated with that prefix.

TABLE IX
DIMINISHED SERVER REDUNDANCY EXAMPLES

Example 1		(Date: 12/07/03)	Example 2		(Date: 12/07/03)
$ORIGIN pik-net.pl.			$ORIGIN nl.		
bieszczady.pik-net.pl.	NS	ns3.pik-net.pl.	saxcompany.nl.	NS	ns.vuurwerk.nl.
bieszczady.pik-net.pl.	NS	ns1.pik-net.pl.	saxcompany.nl.	NS	ns2.vuurwerk.net.
bieszczady.pik-net.pl.	NS	ns2.pik-net.pl.	saxcompany.nl.	NS	ns3.vuurwerk.net.
ns3.pik-net.pl.	A	213.241.68.129	ns.vuurwerk.nl.	A	62.250.2.2
ns1.pik-net.pl.	A	213.241.68.198	ns2.vuurwerk.net.	A	212.204.221.71
ns2.pik-net.pl.	A	213.241.68.146	ns3.vuurwerk.net.	A	213.136.0.173

Finally, we estimate the geographic location of different servers using the NetGeo database [1], which provides a mapping between AS numbers and geographic locations. NetGeo provides location information in three levels: city level, region (state) level, and country level. We use the city level to estimate the servers' geographic location, since servers in the same city are likely to be affected by the same external failures. In cases where we cannot extract city level information, we use the most specific common level to compare the two locations. We note that geographic mapping techniques may not be accurate [30], nevertheless we believe they provide adequate results for our aggregate measurements.

2) Measurement Results: Table X shows the degree of redundancy for different definitions of redundancy. Degree of redundancy refers to the number of independent failures that the system can tolerate. For example degree 2 of redundancy at the host level means that there are two authoritative servers, while at the prefix level (/24 prefix) that there are two /24 prefixes where the servers are placed at. At the host level we can see that most zones (65%) have two authoritative servers, and a small percentage (20%) of them have three or more servers. At the prefix level, 55% of the zones have two or more authoritative servers located at different /24 prefixes. An even smaller percentage of zones, i.e. less than 25%, have two or more servers located at different ASs or different geographic locations (cities).

We also computed the server redundancy at the /24 prefix level for zones grouped by their TLD. From Figure 5, we can observe that there is a relation between server redundancy at the prefix level and the TLD where the zone belongs to. We did not observe any substantial difference among the TLDs for the other levels of redundancy. The main reason is that in most cases all servers for the same zone are placed in the same location or advertised by the same AS.

We repeated the same measurements for the popular zones of Sample 3. The results show again that popular zones are better administered compared to a randomly chosen zone. For example only 1% of the zones have one DNS server and 24.4% of the zones have all their authoritative servers under the same prefix. 42% of the zones have at least two of their servers placed at different ASs, and 32% of the zones have servers located in multiple cities.

3) Impact of Diminished Server Redundancy: While it is intuitive that a smaller degree of redundancy translates to a reduced level of availability, we conducted quantitative measurement of server placement's impact on availability. To assess the availability of a zone, we sent probes every 30 minutes for a duration of four weeks to all the authoritative servers of every zone of Sample 1 that had three servers. We define each 30 minute probing as a round, and a zone as *unreachable* in one round if none of the authoritative servers replied to the probing. The *availability* of the zone is defined as the ratio of rounds that received at least one answer, coming from an authoritative server, over the total number of rounds.

Table XI and Figure 6 show the results of our measurements. We group zones based on their redundancy (1, 2 or 3) at the prefix (/24), AS, and geographic level. For each of these groups we calculate the percentage of the zones that are unreachable for at least one round. We also calculate the average zone availability for each of the three groups. The results show that zones with all their servers under the same prefix tend to have the worst availability: around 16% of such zones are at least once unavailable during the first two weeks (a number that becomes 31% in four weeks) and their availability on the average is under 99.9%. In contrast, zones whose servers are placed at three different geographic locations, or in three different ASs, have notably higher availability (99.99%). Moreover, the number of zones that are unavailable at least once is considerably lower when servers are correctly placed, 5% and 4% respectively in the first two weeks and around 6% and 5% in the four weeks.

D. Cyclic Zone Dependency

To achieve the desired geographic and network diversity for a zone's authoritative servers, operators often establish mutual agreement to host each other's DNS services. For example, *ns.bar.com* may serve as a secondary authoritative server for zone *foo.com*. Authoritative servers located in other zones are normally identified by their names instead of IP addresses. As a result, to resolve a DNS name in zone *foo.com* requires one to first resolve the IP address of *ns.bar.com*. A cyclic zone dependency happens when two or more zones' DNS services depend on each other in a circular way: to resolve a name in zone $Z1$, one needs to first resolve a name in zone $Z2$, which in turn requires some information from zone $Z1$. This type of inter-dependency creates a "chicken and egg" problem; one cannot resolve a name in zone $Z1$ without first resolving a name in $Z2$ and vice versa. This scenario can occur due to configuration errors in either or both of the zones, however it is more often the case where none of the involved zones has any noticeable configuration error, yet the combination of two or more correctly configured zones results in cyclic zone dependency.

Table XII shows two real examples of cyclic zone dependencies we captured during our measurements. The first example involves a single zone only and the problem was due to a configuration error in the parent zone, which should have included the glue A record for the *ns3.nlc.net.au*. Without

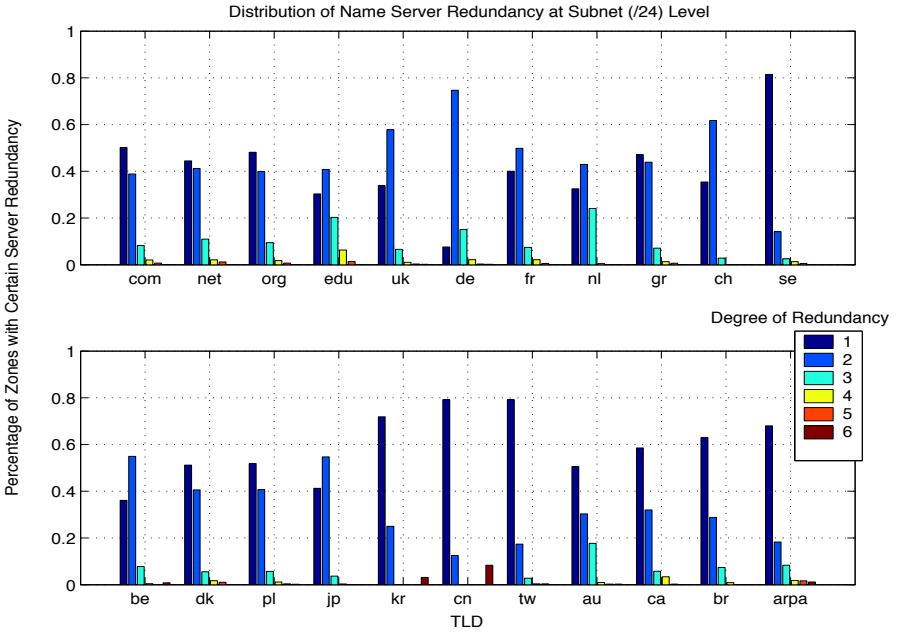


Fig. 5. Distribution of redundancy at /24 prefix level grouped by TLDs (based on Sample 1).

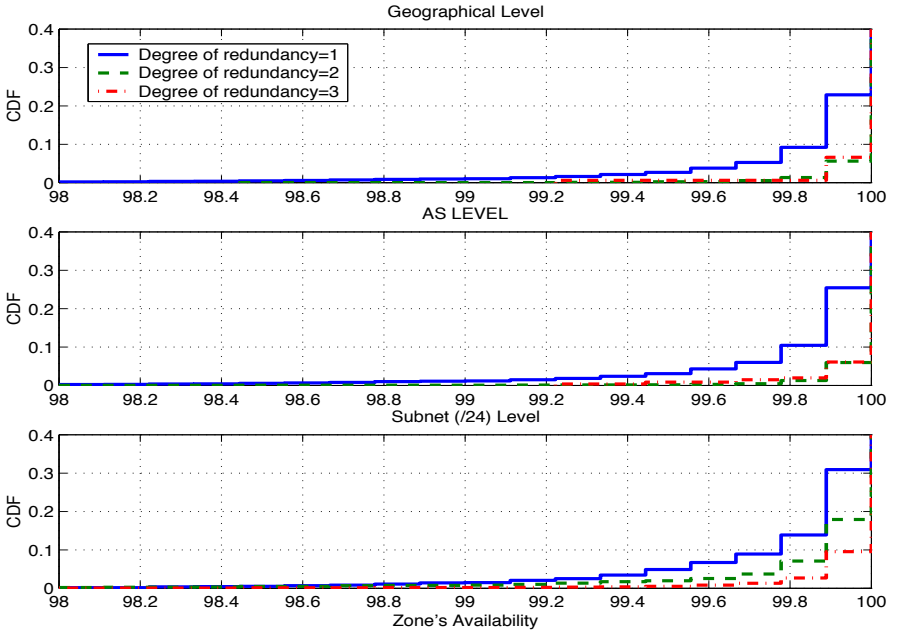


Fig. 6. The CFD for the zones's availability at different degrees of redundancy at the subnet, AS and geographical level

TABLE X
NAME SERVER REDUNDANCY FOR DIFFERENT REDUNDANCY DEFINITIONS.

Degree of redundancy	Geographic Level	AS Level	Prefix (/24)	Host
1	82.30%	77.19%	45.80%	14.69%
2	16.51%	19.56%	42.73%	65.20%
3	0.78%	2.26%	8.94%	14.45%
4	0.36%	0.56%	1.81%	4.50%
5	0.03%	0.36%	0.57%	0.86%
6	0.02%	0.04%	0.10%	0.23%

TABLE XI
IMPACT OF DIMINISHED SERVER REDUNDANCY ON AVAILABILITY.

		Prefix Level		AS Level		Geographic Level	
Degree of Redundancy		14 Days	28 Days	14 Days	28 Days	14 Days	28 Days
1	unreachable	15.90	30.89	14.73	25.43	13.32	22.87
	availability	99.89	99.91	99.93	99.93	99.97	99.94
2	unreachable	11.39	17.91	4.23	6.08	3.95	6.58
	availability	99.94	99.95	99.98	99.98	99.99	99.99
3	unreachable	6.11	9.53	3.84	5.94	4.49	5.62
	availability	99.98	99.98	99.99	99.99	99.99	99.99

TABLE XII
CYCLIC ZONE DEPENDENCY EXAMPLES

Example 1		(Date: 12/07/03)	Example 2		(Date: 12/07/03)
$ORIGIN .net.au.			$ORIGIN com.		
nlc.net.au.	NS	ns1.nlc.net.au.	abacoweb.com.	NS	ns1.abacoweb.com.ar.
nlc.net.au.	NS	ns2.nlc.net.au.	abacoweb.com.	NS	ns3.abacoweb.com.ar.
nlc.net.au.	NS	ns3.nlc.net.au.	abacoweb.com.	NS	dns1.abacoweb.com.
			abacoweb.com.	NS	dns2.abacoweb.com.
ns1.nlc.net.au.	A	203.24.133.1	dns1.abacoweb.com.	A	200.49.93.26
ns2.nlc.net.au.	A	203.24.133.2	dns2.abacoweb.com.	A	200.49.93.27
			$ORIGIN com.ar.		
			abacoweb.com.ar.	NS	dns1.abacoweb.com.
			abacoweb.com.ar.	NS	dns2.abacoweb.com.

that glue record, the third server was reachable only after one was able to resolve its IP address by querying one of the other two servers. In case these two servers became unavailable, it was impossible to obtain the IP address of *ns3.nlc.net.au*. The second example is slightly more complicated and involves two correctly configured zones. The parent zone, *com*, listed four servers for the *abacoweb.com* zone. However, in order to reach both *ns{1,3}.abacoweb.com.ar* servers, one had to first contact the *abacoweb.com.ar* zone. The *abacoweb.com.ar* zone was served by two servers, *dns{1,2}.abacoweb.com*. In case both of them became unavailable, the zone *abacoweb.com* was unavailable. Even though *ns1.abacoweb.com.ar* and *ns3.abacoweb.com.ar* might have functioned correctly, they were not reachable because their IP addresses couldn't be resolved.

The above examples illustrate the failure dependency between zones, the failure of some servers in one zone leads to unreachability of all authoritative servers in another zone. We have found cases where, due to cyclic zone dependency, a zone that appears to have several redundant servers actually relies solely on the availability of one server, which effectively becomes a single point of failure. Similar to the diminished server redundancy problem, these errors can significantly reduce the system's redundancy and they are not immediately visible to the operators. Moreover, it is possible that incorrectly implemented DNS resolvers may be trapped in a query loop when they encounter a cyclic dependency case and certain servers are unavailable (note that current versions of BIND do not suffer from such implementation problems).

Among the three configuration problems discussed in this paper, cyclic zone dependency is the most difficult to detect. The system operates correctly in the absence of server failures and, when inspected individually, each of the zones involved in a cyclic zone dependency appears to be configured correctly. The inter-dependency loop can be discovered only when one brings together the combined configuration of all the involved zones. Although our measurements show that this configuration error is not as widespread as the previous types of errors, most of the existing cases reduce system redundancy substantially. There is also a concern that, without a systematic mechanism to detect this problem, it may spread more widely in the future as the DNS system continues to grow and the operators responsible for configuring new DNS servers may not be aware of this subtle problem.

1) Measurement Details: In order to detect cyclic zone dependency errors, our DNS resolver follows all the possible paths in resolving a query. When the resolver gets a referral answer, it tries to resolve the IP address for all the NS records that do not have a glue A record in the additional section of the answer. This starts a new sequence of iterative queries, and puts the previous question in a pending state. If during that process it happens to ask information from a zone that is already in a pending state, then a cyclic dependency error is detected. With a careful examination of the servers that are part of the cycle, we can identify the servers that depend on the availability of other servers in the loop.

Cyclic zone dependencies could be eliminated by the inclusion of specific glue A resource records. For example in the second case shown in Table XII, had the *com* zone server included glue A RRs for the two servers *ns{1,3}.abacoweb.com.ar*, one would have been able to resolve names in *abacoweb.com* zone even when the two *dns{1,2}.abacoweb.com* servers were unavailable. However the current DNS implementations discourage the use of *unnecessary* glue A RRs. Glue A RRs are considered "unnecessary" if they point to servers that belong to a domain different from the delegated domain. In the second example in Table XII, all the glue A RRs defined at the *com* zone are necessary; glue A RRs for *ns{1,3}.abacoweb.com.ar* are unnecessary and thus are not included in the *com* zone. Since operational practice may differ regarding the inclusion of unnecessary glue A RRs, we repeated each type of measurement with and without accepting unnecessary glue A records.

2) Measurement Results: Figure 7 shows the percentage of zones, grouped by top level domains, whose authoritative servers have names that are defined outside the zones' domain. For instance, at Table XII all the servers of the *abacoweb.com.ar* zone are defined outside the zone's domain. We term this servers as out-of-domain servers. Zones with out-of-domain servers are more prone to cyclic zone dependency errors, given that a resolver needs to follow some levels of indirection in order to resolve names belonging to that zones. Moreover, it has been reported that some resolver implementations are unable to follow more than one level of indirection [19]. The figure gives the percentage of zones which have at least one out-of-domain server, and at least half and all their server being out-of-domain. It shows that for some TLDs the majority of the zones have only out-of-domain servers.

Table XIII shows the percentage of zones that have cyclic zone dependency errors, based on measurements of the zones of Sample 1. About 2% of the zones were involved in a cyclic dependency error, however the number of zones that were affected by these problems is around 5%. A zone is

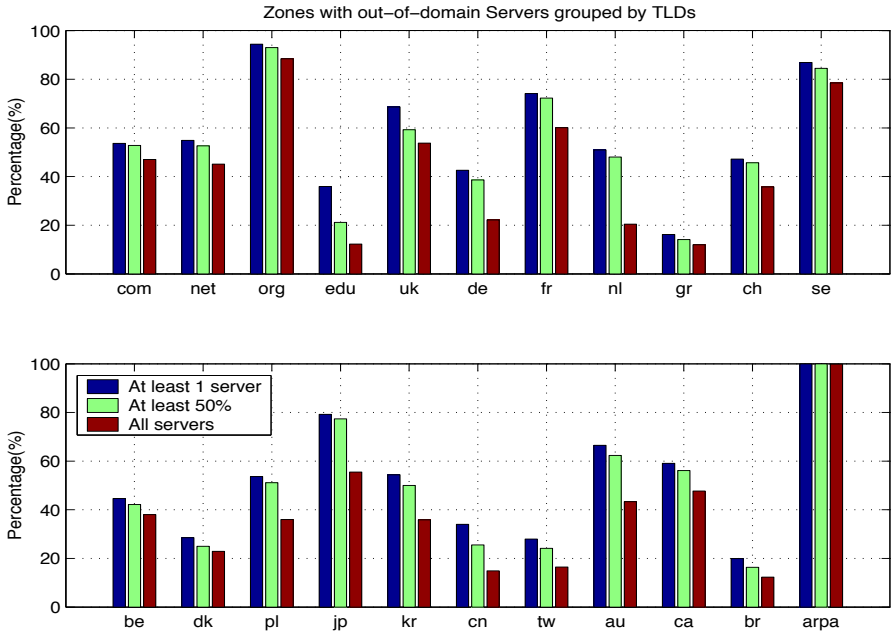


Fig. 7. Percentage of zones with out-of-domain servers grouped by TLDs (based on Sample 1)

TABLE XIII
PERCENTAGE OF ZONES WITH CYCLIC ZONE DEPENDENCY ERRORS

	Percentage of Zones Involved	Percentage of Zones Affected
All glue A records	2.39%	5.95%
Necessary glue A records only	2.99%	33.34%

affected if any of the following three conditions hold: *i)* the zone is directly involved in a cyclic dependency, *ii)* one or more authoritative servers lay in a zone with a cyclic dependency, or *iii)* if any ancestor zone is involved in a cyclic zone dependency. Note that measurements based on Sample 3 show that none of the popular zones is involved in cyclic dependencies. Figure XIII also shows the percentage of zones that are involved and the percentage of zones that are affected by the cyclic zones dependency errors, when only necessary glue A RRs are accepted. While the percentage of involved zones is not increased much by the elimination of unnecessary glue A RRs, the number of affected zones is increased to 33%, primarily because some top level domains are involved in cyclic zone dependency which are currently concealed with the use of unnecessary glue A records.

3) Impact of Cyclic Zone Dependency: Cyclic zone dependency errors can reduce the DNS service availability. If a zone is involved in a cyclic dependency, the failure of DNS servers in some other zones can affect its own DNS service availability. The upper graph of Figure 8 shows the CDF for the percentage of a zone's authoritative servers that are a part of a cyclic zone dependency problem and may become unavailable due to dependency on other servers. As a result of cyclic dependency, a zone loses more than 25% of its servers in most cases. The graph also shows the CDF when no unnecessary glue A records are accepted, which has an even greater impact on the server availability.

As the number of zones involved in a specific cyclic zone dependency increases, the manual detection of the problem becomes harder because one must bring together the configuration data from each involved zone. Fixing the problem becomes even harder because it requires coordinated changes in multiple zones. The lower graph of Figure 8 gives the number of zones that are involved in each cyclic zone dependency, with and without considering unnecessary glue A RRs. If all glue A RRs are accepted, cyclic dependency errors involve four zones in most cases. If only necessary glue A records are accepted, the number of two-zones cyclic dependency errors increases greatly. This indicates that cyclic dependency errors exist in the current system, but are concealed due to the unnecessary glue A RRs. Including these unnecessary glue A RRs in a zone increases chances of lame delegation, yet excluding these unnecessary RRs leads to increased cyclic zone dependency.

V. DISCUSSION

The DNS design is an example of great engineering. It tries to balance the essential functionality requirement – scalable name resolution – against the complexity of different design choices. The result is a disarmingly simple system which has been proven extremely successful in both achieving the original objectives and adapting to changes. However the original design focused mainly on robustness against physical failures and gave no consideration to operational errors such as misconfigurations. As our measurements show, left unchecked, configuration errors have effectively reduced the level of redundancy in the deployed system. Thus the actual degree of redundancy is likely to be lower than the number of servers suggests for a significant portion of DNS zones. Although the system seems to be operating satisfactorily today, unexpected partial failures may occur at any time and may easily disable DNS services to those zones having one or more types of errors lurking in their configurations.

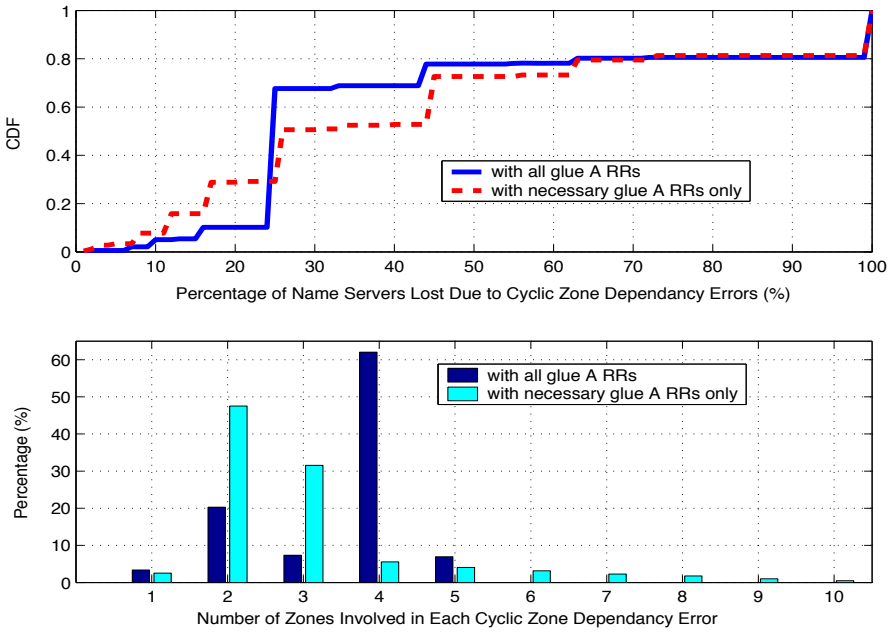


Fig. 8. A) Servers lost due to cyclic zone dependency errors; B) Number of zones involved in cyclic dependency errors.

A. Detecting Misconfigurations

One can develop a set of simple mechanisms to detect all of the lurking errors identified in this study. Delegation inconsistency and lame delegation errors can be detected by a simple protocol between parent and child zones to periodically check the consistency of the NS records stored at each place. Cyclic zone dependency can be detected via automatic checking by trying to resolve a name through *each* of the authoritative servers in the zone. Although there may not be a single check to detect the diminished server redundancy problem, automatic periodic measurement between servers of the same zone on their IP address distance, hop count distance, and AS distance can effectively reflect the diversity degree in their placement. These simple checks are absent from the original DNS design, not because they are difficult to do but a lack of appreciation of the severity of human-introduced errors. As part of our ongoing effort we are developing a software tool [31] that can proactively detect DNS configuration errors.

B. Comparison with Other Efforts

The IETF community has initiated several efforts to address DNS resilience issues. One is deploying anycast routing for root servers. The anycast routing solution removes the limit on the number of replicated servers for a zone. Moreover, it also eliminates the need to change configurations when the number or locations of the servers change. While anycast is well suited to the root and certain heavily used top-level servers, it is less practical for lower-level zones due to the scalability concern in global routing. It is also important to note that, unless redundant servers are placed in diverse locations and the anycast address is announced from those locations, anycast itself does not automatically solve the diminished server problem. It simply shifts the burden from DNS configurations to routing configurations. Furthermore, anycast raises its own issues in problem debugging. For example, if a lame server exists among a set of anycast enabled servers, it can be difficult to pin down which one is the culprit.

The extent of configuration errors can also have an important impact on new DNS additions. The DNS Security Extensions (DNSSEC) [8] are another effort to improve DNS resilience. The original DNS design provided no authentication and current DNS service is vulnerable to a wide range of attacks [10]. The DNS Security Extensions add cryptographic authentication into the system, allowing a resolver to authenticate that the data received in the response matches the data entered by the zone operator. However authentication does not address any of the configuration errors presented in this study. In fact, DNSSEC deployment may exacerbate these problems as cryptographic misconfigurations may further reduce the availability of DNS services. Moreover, DNSSEC defines an authentication chain where a new DS RR at the parent zone identifies a public key (DNSKEY) at the child zone. Just as a traditional zone must ensure the parent and child's NS RR sets are consistent, a secure zone must also ensure the DS RR at the parent matches the DNSKEY RR at the child. The addition of DNSSEC will only increase the importance of consistency checks and correct configurations.

To a large extent, DNSSEC (and cryptographic techniques in general) do not easily tolerate inconsistencies. For example, a cryptographic signature does not typically work well if one hopes to prove the signed data is "mostly correct". Recall that in the delegation inconsistency, the set of servers listed at the parent is not equal to the set of servers listed at the child. Ideally, the servers would be identical and the DNS design requires that the set of servers listed at the parent must be a subset of the servers listed at the child. But our results show that the system continues to work even though this requirement is not met for a large percentage of zones. This is an example of where a system design works despite errors.

DNSSEC does not attempt to sign the NS records stored at the parent. This creates some security limitations, but our results on delegation inconsistencies would suggest that fixing these limitations would be difficult at best since operational practice does not match the system design.

VI. Related Work

Danzig *et al.* [25] provided an extensive analysis of the DNS traffic observed on a root name server. They identified a number of DNS implementation bugs and found that such bugs incurred unnecessary wide-area DNS traffic by a factor of twenty. In a followup work, Brownlee *et al.* [11] measured the impact of some new implementation errors on the DNS traffic directed toward the F root server. In this work, our focus is to identify configuration errors in the DNS system and to measure their impact on the zone's availability and performance.

Jung *et al.* [18] measured the DNS performance in term of query response time perceived by DNS resolvers, and studied the effectiveness of caching. They found that the query response time is highly related to the number of referrals, and that the majority of queries complete in less than 100 milliseconds. They further concluded that DNS caching works effectively even when the TTL value of host names is as low as a few hundred seconds, as long as the domain servers' A RRs are cached. Our interest is in identifying the performance degradation, in terms of query response time, due to the configuration errors.

Cohen *et al.* [13] studied the effects of resource record prefetching on the DNS system performance. They showed that prefetching can considerably improve the DNS performance, measured as query response time, with a moderate penalty on the DNS traffic. They also proposed an alternative method, simultaneous validation, for improved DNS performance: cached records are used even if they are expired, but they are validated with a concurrent query that shows if they are in agreement with the actual records.

Liston *et al.* [21] studied the diversity in DNS performance perceived by a number of geographically distributed clients. They showed that the mean response time for name lookups at different sites varies greatly, and the performance of root servers and TLD servers have the least impact for non-cached entries. In this paper we examine the diversity in server placement and its impact on zones availability.

Parka *et al.* [28] studied the reliability of the DNS caching servers and proposed a cooperative architecture, which improves the DNS performance and reliability as perceived by the DNS clients. Pang *et al.* [27] measured the availability of the individual DNS authoritative and caching servers and studied the different server deployment strategies. Both these works measure the reliability of individual components of the system, whereas in our study we measure the reliability of the DNS infrastructure, and more specifically we show how it is affected by human errors.

Finally there are a number of companies and individuals that look into the problem of lame delegation. Men & Mice [22] periodically measures the lame delegation as it appears under the *com* domain; Team Cymru [15] collects the BIND log files from a number of DNS servers and extracts from them a list of lame servers; and Credentia [14] provides lame delegation statistics on the TLD zones.

VII. Conclusion

DNS is one of the best-known Internet systems providing indispensable name resolution services for end users and applications. Its design relies on redundant servers to achieve reliability. Adverse events, such as DDoS attacks against the DNS root servers, illustrate the critical dependence on distributed replicated servers. However, our measurements show that lame delegations and cyclic zone dependencies reduce the number of reachable servers and thus the actual system redundancy can be much lower than expected. Our results also show that a large portion of the zones have all their DNS servers placed either behind the same routing prefix or in the same geographical location, thus a physical disaster or network disruption can simultaneously incapacitate all of these servers, invalidating the assumption that server redundancy should provide resilience in the face of failures.

Distributed management is crucial in achieving DNS system's scalability, however our measurements show that it also leads to inconsistencies due to mistakes in coordinating zone configurations and changes. While human induced configuration errors are a well-known fact, DNS delegation configurations require consistency *across* administrative boundaries, a condition that is even more prone to errors. Unfortunately the current system does not provide any automated means to communicate for coordination. Today configurations are communicated manually, and as we have seen this process is highly subject to errors.

We draw two broad conclusions from our measurement study on the DNS configuration errors. First, large-scale, distributed systems should *expect* human errors and therefore should *proactively* develop systematic checking mechanisms against such errors. Second, in distributed systems such as DNS, acceptable performance perceived at the user level is not a reliable indicator that the system is error free. Minor errors may be lurking under the surface, and when a number of these errors cascade or get triggered simultaneously, the system can fail in unexpected and catastrophic ways. Other fields have provided numerous examples on how complex systems may fail through a number of cascaded small failures (e.g., failures of airplanes, nuclear plants and the power grid); the same lessons apply to the Internet as well. We are developing a set of active mechanisms to detect and eliminate neglected configuration errors in today's DNS system. Through this process we hope to gain further understanding on how to design such proactive checking into distributed systems in general.

References

[1] NetGeo Database. http://netgeo.caida.org/aslatlong.txt, 2004.
[2] Route Views Project. http://www.routeviews.org/, 2004.
[3] DNS Surveys. http://dns.measurement-factory.com/surveys/, 2008.
[4] ICANN Factsheet. http://www.icann.org/announcements/factsheet-dns-attack-08mar07.pdf, 2008.
[5] Internet Domain Survey. http://www.isc.org/index.pl?/ops/ds/, 2008.
[6] ISC BIND. http://www.isc.org/sw/bind/, 2008.
[7] Alexa.com. http://www.alexa.com/, 2004.

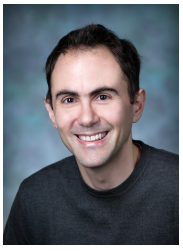
[8] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. Work in progress: draft-ietf-dnsext-dnssec-intro-08, July 2004.
[9] D. Barr. Common DNS Operational and Configuration Errors. RFC 1912, 1996.
[10] S. M. Bellovin. Using the Domain Name System for System Break-Ins. In *Proc. Usenix UNIX Security Symposium*, 1995.
[11] N. Brownlee, k claffy, and E. Nemeth. DNS Measurements at a Root Server. In *Proc. IEEE Globecom' 01*, 2001.
[12] CAIDA. Nameserver DoS Attack October 2002. http://www.caida.org/funding/dns-analysis/oct02dos.xml, 2008.
[13] E. Cohen and H. Kaplan. Proactive caching of DNS records: Addressing a performance bottleneck. In *Proc. SAINT 01*. IEEE, 2001.
[14] Credentia. http://www.credentia.cc/research/cctlds/, 2004.
[15] T. Cymru. http://www.cymru.com/DNS/lame.html, 2004.
[16] R. Elz and R. Bush. Clarifications to the DNS Specification. RFC 2181, 1997.
[17] R. Elz, R. Bush, S. Bradner, and M. Patton. Selection and Operation of Secondary DNS Servers. RFC 2182, 1997.
[18] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS Performance and the Effectiveness of Caching. In *Proc. First ACM SIGCOMM IMW*, pages 153–167. ACM Press, 2001.
[19] M. Larson and P. Barber. Observed dns resolution misbehavior. Work in progress: draft-ietf-dnsop-bad-dns-res-02, February 2004.
[20] T. Lee, B. Huffaker, M. Fomenkov, and kc claffy. On the Problem of Optimization of DNS Root Servers' Placement. In *Passive Measurement and Analysis Workshop '03*, 2003.
[21] R. Liston, S. Srinivasan, and E. Zegura. Diversity in DNS Performance Measures. In *Proc. Second ACM SIGCOMM IMW*, pages 19–31. ACM Press, 2002.
[22] Men & Mice. http://www.menandmice.com/, 2004.
[23] P. Mockapetris. Domain Names–Concepts and Facilities. RFC 1034, 1987.
[24] P. Mockapetris. Domain Names–Implementation and Specification. RFC 1035, 1987.
[25] P. B. Danzig and K. Obraczka and A. Kumar. An Analysis of Wide-Area Name Server Traffic. In *Proc. ACM SIGCOMM'92*, pages 281–292. ACM Press, 1992.
[26] P. Mockapetris and K. J. Dunlap. Development of the Domain Name System. *SIGCOMM Comp. Com. Rev.*, 18(4):123–133, 1988.
[27] J. Pang, J. Hendricks, A. Akella, S. Seshan, B. Maggs, and R. Prisco. Availability, Usage and Deployment Characterisitics of the Domain Name System. In *Proc. ACM IMC 04*, 2004.
[28] K. Parka, V. Pai, L. Peterson, and Z. Wang. CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups. In *Proc. USENIX OSDI'04*, 2004.
[29] Ranking.com. http://www.ranking.com/, 2004.
[30] V. N. Padmanabhan and L. Subramanian. An Investigation of Geographic Mapping Techniques for Internet Hosts. In *Proc. ACM SIGCOMM'01*, pages 173–185. ACM Press, 2001.
[31] V.Pappas, P. Faltstrom, D. Massey, and L. Zhang. Distributed DNS Troubleshooting. In *Proc. ACM SIGCOMM workshop on Network Troubleshooting*. ACM Press, 2004.

Vasileios Pappas received his Ph.D. and M.Sc. degrees both in Computer Science from the University of California, Los Angeles, in 2006 and 2003, respectively, and his B.Eng. degree from National Technical University of Athens, in 2001. He is a member of the research staff at the IBM T.J. Watson Research center. His research interest include Internet systems, distributed systems and wireless networks. More specifically he is interested in the dependability and management issues that arise in large scale distributed systems.

Duane Wessels received a B.S. degree in Physics, followed by a M.S. degree in Telecommunications. He worked for a number of years on the IRCache and Squid Web Caching projects before becoming a partner in The Measurement Factory. He now works for the DNS Operations Analysis and Research Center (DNS-OARC).

Dan Massey is an associate professor at Computer Science Department of Colorado State University. Dr. Massey received his doctorate from UCLA and is a senior member of the IEEE, IEEE Communications Society, and IEEE Computer Society. His research interests include protocol design and security for large scale network infrastructures and he is currently the principal investigator on research projects investigating techniques for improving the Internet's naming and routing infrastructures.

Songwu Lu is an associate professor in the Computer Science Department at UCLA. He received both his M.S. and Ph.D. degrees from the University of Illinois at Urbana-Champaign (UIUC). He received an NSF CAREER award in 2001. His research interests include wireless networking, mobile systems, sensor networks, and wireless network security.



Andreas Terzis is an Assistant Professor in the Department of Computer Science at Johns Hopkins University, where he heads the Hopkins InterNetworking Research (HiNRG) Group. His research interests are in the broad area of wireless sensor networks, including protocol design, system support, and data management. Dr. Terzis is a recipient of the NSF CAREER award.

Lixia Zhang received her Ph.D in computer science from the Massachusetts Institute of Technology. She was a member of the research staff at the Xerox Palo Alto Research Center before joining the faculty of UCLA's Computer Science Department in 1995. In the past she has served as the vice chair of ACM SIGCOMM, and on the editorial board for the IEEE/ACM Transactions on Networking. Zhang is currently serving on the Internet Architecture Board and is co-chair of the Routing Research Group under IRTF. She is a fellow of ACM and a fellow of IEEE.