A Probabilistic Model of Namespace Influence in the Domain Name System☆

Casey Deccio[a,*], Jeff Sedayao[b], Krishna Kant[b], Prasant Mohapatra[c]

[a] *Sandia National Laboratories, PO Box 969, Livermore, CA, USA*
[b] *Intel Corporation, 2200 Mission College Blvd., Santa Clara, CA, USA*
[c] *University of California Davis, 1 Shields Avenue, Davis, CA, USA*

Abstract

Name resolution using the Domain Name System (DNS) is integral to today's Internet. The resolution of a domain name is often dependent on namespace outside the control of the domain's owner. In this article we review the DNS protocol and several DNS server implementations. Based on our examination, we propose a formal model for analyzing the name dependencies inherent in DNS, and experimentally validate the model with actual domain names. Using our name dependency model we derive metrics to quantify the extent to which domain names affect other domain names. It is found that under certain conditions, more than half of the queries for a domain name are influenced by namespaces not expressly configured by administrators. This result serves to quantify the degree of vulnerability of DNS due to dependencies that administrators are unaware of. When we apply metrics from our model to production DNS data, we show that the set of domains whose resolution affects a given domain name is much smaller than previously thought. However, behaviors such as using cached addresses for querying authoritative servers and chaining domain name aliases increase the number and diversity of influential domains, thereby making the DNS infrastructure more vulnerable.

Keywords: DNS, Networks

1. Introduction

The Domain Name System (DNS) is one of the systems most critical to the Internet. Nearly all Internet applications rely on the DNS for proper function. Although Internet hosts communicate using the Internet Protocol (IP) and are identified by IP addresses, the DNS abstracts IP addressing from users and clients, so they can identify Internet hosts with domain names representative of human-friendly words. In addition to IP address lookup, the DNS is also necessary for email delivery, service discovery, and host identification.

Because applications rely on the DNS for name resolution, its integrity and security are critical. While temporary failures due to misconfiguration may cause inconvenience, targeted attack by malicious parties could be much less discernible, and the repercussions more severe. Tainted DNS responses may transparently redirect user applications to servers within adversarial control, where sensitive information can be stolen.

While the concept of name resolution is relatively simple, the overall system is complex. The DNS allows configurations that create dependencies for a domain name on servers well outside the control of the domain's owner and managed by third parties. Domain names with such configurations are in turn affected by the security and accuracy of the namespaces they depend on. An understanding of a domain's context in the entire system will allow architects and administrators to better design and configure their DNS services to maximize the reliability of DNS name resolution.

In this work we present a quantitative analysis of name dependencies in DNS. We review pertinent aspects of the DNS protocol and derive a model to probabilistically determine the namespace influencing resolution of a domain name. Based on our model, we define several metrics to assess the quality of name resolution for a domain name, based on the other names that affect its resolution. The behaviors of name server implementations, some of which are configurable, can affect these metrics. We analyze a large sample of recent DNS data in light of the presented model. The results show the impact of name dependencies in terms of the size of the namespace that influences a domain name, and in the percentage of queries that will reach namespace not explicitly configured by DNS administrators.

We list the following as primary contributions of this article:

- A detailed study of DNS protocol specification and name server implementation, with respect to DNS name dependencies
- A formal model for analysis of DNS name dependencies, based on specification and implementations
- Metrics for quantifying the influence domain names have on other domain names

☆This research was supported in part by the National Science Foundation under the grant CNS-0716741.

*Corresponding author

Email addresses: ctdecci@sandia.gov (Casey Deccio), jeff.sedayao@sandia.gov (Jeff Sedayao), krishna.kant@intel.com (Krishna Kant), pmohapatra@ucdavis.edu (Prasant Mohapatra)

[1] Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

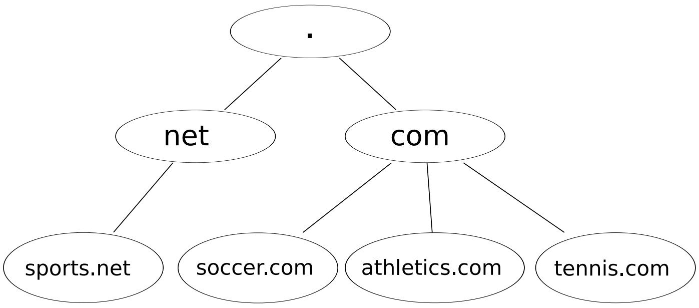

Figure 1: The zone hierarchy for the the zone data shown in TABLE 1.

Our analysis shows that 92% of influential namespace of domain names is explicitly configured by domain administrators. However, certain caching behaviors reduce that figure, and increase the probability that resolution of a domain name is influenced by names or organizations not explicitly configured by administrators. A diverse set of dependencies has the potential to affect the reliability of name resolution for a domain name. Based on our findings, we discuss best practices for design and administration of DNS services to contain influence of domain names.

We begin our analysis with a review of the DNS protocol in Section 2. We describe previous work in this area in Section 3. Pertinent behaviors of DNS server implementations are described in Section 4. In Section 5 we introduce the concept of DNS name dependencies, present a graph model for their analysis, and derive methodology for quantifying influence. We describe methodologies employed for data collection and an analysis of the observed quality of name resolution in Section 6. We explore considerations that increase outside influence in Section 7 and discuss extensions to our model in Section 8. We conclude our analysis in Section 9.

2. DNS Background

The Domain Name System (DNS)[1, 2, 3] provides a name-based lookup service for the Internet. The DNS namespace is organized hierarchically, and each *domain name* reflects its ancestry. For example, the ancestry of *www.soccer.com* is: *www.soccer.com*, *soccer.com*, *com*, and the *root* domain (represented by a single dot with no labels: “.”). All domain names are descendants of the root.

DNS data is managed within the DNS *zone* to which it pertains. A zone includes all the subdomain namespace below its *origin*, except that whose management has been explicitly *delegated* to child zones. Figure 1 illustrates a zone hierarchy in which the *com* zone delegates management of the *soccer.com* namespace to the *soccer.com* zone.

2.1. Zone data

DNS zone data consists of *resource records* (RRs), each identified by an owner name (e.g., *www.soccer.com*), a class (e.g., `IN`)[2], and a type (e.g., `A`). Among the RR types defined, the following are pertinent to this article: `A` (IP address)[3], `CNAME` (canonical name), and `NS` (name server). Each RR contains record data specific to its RR type. For example, the record data for an `A` RR is an IPv4 address (e.g., 192.0.2.1). The record data for RRs both of type `CNAME` and `NS` is another domain name—a *target* for further lookup (e.g., *www.tennis.com*). An *RRset* is comprised of all RRs matching a given owner name and type.

QUESTION			
www.soccer.com.		`A`	
ANSWER			
www.soccer.com.	3600	`CNAME`	www.tennis.com.
www.tennis.com.	3600	`A`	192.0.2.3
AUTHORITY			
soccer.com.	3600	`NS`	ball.soccer.com.
soccer.com.	3600	`NS`	racket.tennis.com.
soccer.com.	3600	`NS`	ns1.sports.net.
ADDITIONAL			
ball.soccer.com.	3600	`A`	192.0.2.1
racket.tennis.com.	3600	`A`	192.0.2.4
ns1.sports.net.	3600	`A`	192.0.2.6

Figure 2: A response for *www.soccer.com* issued by *ns1.sports.net*, which is authoritative for *soccer.com*.

TABLE 1 contains zone data for some fictitious zones.

The `NS` RR type is used to designate the names of servers *authoritative* for a zone. The `NS` RRset for a zone must not only exist in the child zone, but also in its parent—as a set of *delegation records*. In TABLE 1 the delegation and authoritative `NS` RRsets for *soccer.com* are found on lines lines 4–6 of the *com* zone and lines 1–3 of the *soccer.com* zone, respectively.

2.2. Name resolution

DNS servers have two primary roles: *resolver* and *authoritative server*. The resolver queries authoritative servers to find the answers to domain name lookups. The answers, comprised of RRsets, may be stored in the resolver’s cache for later retrieval. Each RRset includes time-to-live value (TTL), which designates the lifetime for which it may persist in a resolver’s cache.

DNS responses are comprised of several sections. The *question* section contains the name and type of the current query. Answers are returned in the *answer* section. Information about the servers authoritative for the answer are contained in the *authority* section. The *additional* section contains supplemental information that may be helpful or necessary for the resolver. The anatomy of a response that might be issued by a server authoritative for *soccer.com* in response to a query for *www.soccer.com* is shown in Figure 2.

A resolver begins the name resolution process by issuing a query to one of the servers authoritative for the root zone. The root server responds with a referral, populating the authority section of its reply with the `NS` RRset for the top-level domain (TLD) of the name in question. The resolver then re-issues the

[2] We abstract RR class from the remainder of our analysis, as our research focuses on specific RR types in the `IN` class.

[3] This work focuses on DNS name dependencies, which affect name resolution to both IPv4 and IPv6 addressing using RRs of types `A` and `AAAA`, respectively. For simplicity, however, our analysis references only RRs of type `A`.

$ORIGIN soccer.com. (SoccerMania, Inc.)

	Name	Type	Value
1	soccer.com.	NS	ball.soccer.com.
2	soccer.com.	NS	racket.tennis.com.
3	soccer.com.	NS	ns1.sports.net.
4	ball.soccer.com.	A	192.0.2.1
5	www.soccer.com.	CNAME	www.tennis.com.

$ORIGIN tennis.com. (Tennis Pro, Inc.)

	Name	Type	Value
1	tennis.com.	NS	ns1.tennis.com.
2	tennis.com.	NS	ball.soccer.com.
3	tennis.com.	NS	ns1.sports.net.
4	ns1.tennis.com.	A	192.0.2.2
5	www.tennis.com.	A	192.0.2.3
6	racket.tennis.com.	A	192.0.2.4

$ORIGIN athletics.com. (Sports Central, Inc.)

	Name	Type	Value
1	athletics.com.	NS	ns1.athletics.com.
2	ns1.athletics.com.	A	192.0.2.8

$ORIGIN com. (VeriSign, Inc.)

	Name	Type	Value
1	com.	NS	ns1.com.
2	ns1.com.	A	192.0.2.5
3	athletics.com.	NS	ns1.athletics.com.
4	soccer.com.	NS	ball.soccer.com.
5	soccer.com.	NS	racket.tennis.com.
6	soccer.com.	NS	ns1.sports.net.
7	tennis.com.	NS	ball.soccer.com.
8	tennis.com.	NS	ns1.tennis.com.
9	tennis.com.	NS	ns1.sports.net.
10	ball.soccer.com.	A	192.0.2.1
11	ns1.tennis.com.	A	192.0.2.2
12	ns1.athletics.com.	A	192.0.2.8

$ORIGIN sports.net. (Sports Central, Inc.)

	Name	Type	Value
1	sports.net.	NS	ns1.sports.net.
2	sports.net.	NS	ns1.athletics.com.
3	ns1.sports.net.	A	192.0.2.6

$ORIGIN net. (VeriSign, Inc.)

	Name	Type	Value
1	net.	NS	ns1.net.
2	ns1.net.	A	192.0.2.7
3	sports.net.	NS	ns1.sports.net.
4	sports.net.	NS	ns1.athletics.com.
5	ns1.sports.net.	A	192.0.2.6

Table 1: The zone data from several fictitious zones, whose hierarchy is shown in Figure 1. Any coincidence with actual zones of the same name is unintentional.

query to one of the TLD servers. The resolver iteratively continues this process until it receives a response from an authoritative source containing either an answer or an indication that the requested RRset does not exist. If a CNAME (*canonical name*) RR is returned as an answer in an authoritative response, the resolver must subsequently resolve the target of the alias to obtain an address. For example, *www.soccer.com* is an alias for *www.tennis.com* in TABLE 1.

The NS RRset in a referral specifies the authoritative servers for a delegated zone by name, but a resolver must have IP addresses to actually query the servers. Any addresses already known by the resolver initially populate the authoritative server list for the zone, and it initiates requests in parallel to acquire the addresses for any others. For any NS targets that are subdomains of the delegated zone the referring server must provide the addresses in the additional section of its response to avoid a chicken-and-egg problem with resolving the server names. Such *glue records* are maintained in the referring parent zone, so name resolution can be properly “bootstrapped” [1]. The A glue record for *ns1.tennis.com* is on line 11 of the *com* zone in TABLE 1.

If a server issuing a referral has pertinent non-glue A RRsets available locally, they may be included in the additional section of the response. Such RRsets might be available if the referring server is also authoritative for the zones to which the names belong or if it has in its cache the A RRset for one of the names in question [1]. However, RRsets with names that are not subdomains of the referring zone are considered *out-of-bailiwick* (i.e., outside its authority). Resolver implementations should independently obtain an authoritative answer for the out-of-bailiwick target names before querying such servers.

3. Previous Work

In this section we review related research. First we summarize previous studies analyzing DNS dependencies, misconfiguration, and availability. We then describe some of the solutions which have been proposed to protect name resolution from unwanted influence.

3.1. Analysis

Ramasubramanian, et al. [4] demonstrate the far-reaching effects of DNS dependencies by surveying the DNS namespace and tracing the dependencies of a large number of domain names. They identify the set of all authoritative name servers potentially involved in resolution of each name, which comprise the potential attack target for that name. Their results show that a domain name relies on 44 authoritative servers on average and 6% of names depend on more than 200 servers. We perform further examination of name resolution behaviors to create a formal model of name dependencies in DNS and quantify the significance of such dependencies.

A survey of common DNS misconfigurations is presented by Pappas, et al. [5]. The authors identify lame delegation, diminished server redundancy, and cyclic dependencies as problem areas, and an empirical study is performed to show their pervasiveness in DNS. The formal model presented in this article creates a foundation to systematically identify and quantify the impact of such misconfigurations on security and availability.

3.2. Solutions

The DNS Security Extensions (DNSSEC) [6] are the leading standard for cryptographically validating DNS queries. DNSSEC extends the DNS protocol [1, 2] to allow resolvers to authenticate DNS responses using cryptographic public keys and signatures embedded in new RR types [6, 7].

The IKS Jena (Information, Communication, and Systems) [8] and SecSpider [9, 10] projects have monitored the deployment of DNSSEC, maintaining an online listing and status of signed zones. The SecSpider and IKS project sites report the number of production signed zones at roughly 24,000 and 44,000, respectively, at the time of this writing [8, 10]. Only 0.02% of the the zones we used in our analysis (see Section 6) were signed with DNSSEC, as observed by the presence of `DNSKEY` and `RRSIG` RRs.

Proper application of DNSSEC foils attempts at cache poisoning and other attacks on DNS integrity. However, just like unsigned names, DNSSEC-signed names also rely on the proper configuration and availability of their dependencies. We address DNSSEC considerations in more detail in Section 8.

Several non-cryptographic approaches have been proposed for defending against response spoofing attacks. Dagon, et al. [11] present a technique in which a resolver mixes the case of the name being queried and requires the authoritative server to preserve the case of the name in the question section of the response. This increases the probability space that attackers have to work with. Another technique for increasing entropy is WSEC DNS [12], which introduce wildcard DNS records into a zone, which aliases the actual DNS records. Clients generate a random string which is prepended to the wildcard name, and attacks require a larger effort to spoof responses for the randomized queries.

Additional entropy makes guesswork by a third-party more difficult. However, if an authoritative server on which a domain name is dependent for its resolution is run by a nefarious organization or otherwise compromised by offenders, then no guesswork is needed. An understanding of DNS dependencies will aid administrators in employing practices that will help them maintain control of the name resolution for their domains.

4. Characteristics of DNS Server Implementations

An analysis of certain behaviors exhibited by name server implementations is relevant to our analysis of DNS name dependencies. We reference several products in our research. The Berkeley Internet Name Daemon (BIND, version 9.6) [13] implements both a resolver and an authoritative server. `unbound` (version 1.0.2) [14] and `dnscache` (version 1.05) [15] implement resolvers. The Name Server Daemon (NSD) [16] and `tinydns` [15] implement authoritative name servers.

4.1. Trust ranking

In certain situations, a resolver or authoritative server may receive in a DNS response RRsets whose owner name and type match RRsets previously obtained from other data sources, such as from earlier queries or local zone data. It is therefore essential that a mechanism be employed for establishing precedence of RRsets, based on their sources. RFC 2181[17] outlines a relative ranking of trustworthiness of data for name servers to consider as part of operation. Among the total ranking are the following (in decreasing order of trustworthiness):

- Data from a zone for which the server is authoritative, other than glue data
- The authoritative data included in the answer section of an authoritative reply
- The data in the authority section of an authoritative reply
- Glue from a zone for which the server is authoritative
- Data from additional section of a response

This ranking affects behavior of both resolvers and authoritative servers. A resolver must initially use the delegation records provided in a referral. However, when it receives an answer from a server authoritative for the zone itself, the authoritative server often includes the authoritative `NS` RRset for the zone in its authority section. If these sets differ, the resolver will use the `NS` RRset returned in the authoritative response in preference to the delegation records received in the referral.

If a resolver receiving a referral has the `A` RRset for an `NS` target in its cache, previously received in the answer section of an authoritative response, then the resolver deems the cached data more trustworthy than any data received in the additional section of the referral. Thus, it will use the locally cached data in preference to any glue records.

Both the BIND and `unbound` resolvers respect the notion of trustworthiness described in RFC 2181, using the `NS` RRset from an authoritative response in preference to that received in the authority or additional section of a referral. However, `dnscache` updates its cache with the most recently received RRset, regardless of the section in which it was received.

The combined service of cached and authoritative data by a single name server is allowed by specification, although best practices suggest that these functionalities not co-exist [18]. BIND's implementation allows an authoritative server to additionally maintain a cache, unlike NSD and `tinydns`, which only serve locally available zone data. If an authoritative server so configured has cached the `A` RRset for an `NS` target from the answer section of an authoritative response (i.e., from another server), it will use the data from its cache to populate the additional section of a referral to a resolver, rather than the glue record, which might be different.

Section 7 describes practical examples of these caching behaviors in more detail.

4.2. Name server selection

RFC 1035[2] dictates that a resolver associate historical statistics, such as response time and success rate, with each server address. After trying all authoritative servers at least once, it prefers the server with the best performance record, thus fine-tuning the performance for lookups for a zone [2].

The BIND and `unbound` resolvers both follow the performance-based selection guideline, giving preference to authoritative servers with historically better responses. The `dnscache` resolver, however, selects an authoritative server uniformly at random for each query.

Although individual resolvers following specification may gravitate towards a particular authoritative server, we assume that requests for a zone arrive from resolvers in diverse network and geographic locations with varying proximity to authoritative servers. Based on this assumption we use a uniform distribution to describe queries issued to authoritative servers for a zone, such that any server has an equal chance of being queried for names within the zone.

5. DNS Dependency Model

In this section we introduce the notion of domain name dependencies and establish a formal model for identifying and quantifying domain name influence.

5.1. Name dependencies

Various DNS dependencies stem from the DNS protocol specification. Three specific components introducing dependencies are the following:

- *Parent zones*: A resolver must learn the authoritative servers for a zone from referrals from the zone's hierarchical parent.
- *NS targets*: The `NS` RR type uses names, rather than addresses, to specify authoritative servers, so a resolver must resolve the names before it can query the authoritative servers.
- *Aliases*: If a name is an alias, then to obtain an address, a resolver must subsequently resolve the alias target.

We say that name u *depends on* domain name v if resolution of v may *influence* resolution of u. Influence is categorized into two classes: *active* and *passive*. If domain name u is actively influenced by domain name v, then with some non-zero probability resolution of v will be *required for* resolution of name u. If domain name u is passively influenced by domain name v, then although v may not be required for resolution of u, resolution of v may *affect* resolution of u with some probability.

Parent zones and aliases are both examples of active influence because both are required for resolution of the dependent name. However, `NS` target dependencies may either be the cause of passive or active influence, depending on the state of server cache and server implementation behavior.

We establish some notation for our discussion of `NS` target dependencies. To distinguish `NS` RRsets from different sources, we use NS_z and NS'_z to respectively denote the sets of `NS` target names for zone z, as configured in zones z and $Parent(z)$, respectively. Let NSA_z and NSA'_z denote the sets of IP addresses corresponding to the server names in NS_z and NS'_z, respectively.

Term	Definition
r	The root name "."
$I_u(v)$	The measure of name v's influence on name u
$I_u(D)$	The aggregate influence of names in set D on name u
$Parent(d)$	The nearest ancestor zone of name d
$Cname(d)$	The alias target of name d
NS_z, NS'_z	The set of `NS` target names authoritative for zone z as configured in z and $Parent(z)$, respectively
NSA_z, NSA'_z	The set of addresses corresponding to the names in NS_z and NS'_z, respectively
$P_{NS}(z)$	The probability that a resolver has cached the `NS` RRset for z from an authoritative source
$P_{\{s,c\}}(v)$	The probability that either s or c has in cache and uses `NS` target name v from an authoritative source
$G_d = (V_d, A_d)$	Name dependency graph for name d
$G'_d = (V'_d, A'_d)$	Active influence dependency graph for name d
$P_q(z, v)$	The probability that `NS` target v is used to resolve z
$w(u, v)$	The weight of edge (u, v) in A_d
S_u	The set of addresses corresponding to name u
$U'_d \subseteq U_d \subseteq Z_d$	The sets of first-order, non-trivial, and all zones in V_d, respectively

Table 2: Notation used in this research.

Suppose $v \in NS_z$ is a subdomain of $Parent(z)$, $Parent(v) \neq z$, and $Parent(z)$ is properly configured with a glue record for v. Such is the case when z = *tennis.com* and v = *ball.soccer.com*. If an authoritative answer for v has previously been resolved and cached by either authoritative server $s \in NSA_{Parent(z)}$ or resolver c, then z is affected by v and its name dependencies, based on the trust ranking discussion from Section 4. This behavior describes passive influence of v on z. The probability of passive influence, $P_{\{s,c\}}(v)$, is the combined probability of $P_s(v)$ and $P_c(v)$, the likelihood that either s or c has and uses a cached authoritative answer for v. Since the probabilities are independent of one another, $P_{\{s,c\}}(v)$ is calculated:

$$P_{\{s,c\}}(v) = P_s(v) \vee P_c(v) = 1 - \Big(1 - P_s(v)\Big)\Big(1 - P_c(v)\Big)$$

A resolver is responsible for resolving any names from NS_z which are out-of-bailiwick or not included in the additional section of a response from an authoritative server. An example is *ns1.sports.net*, which is authoritative for *soccer.com*. Such induced queries indicate active influence of the resolved names on z, since it is directly dependent on their resolution.

5.2. Name dependency graph

To model the dependencies of domain name d we use a directed, connected graph, $G_d = (V_d, A_d)$. The graph G_d contains a single sink, r, which is the root zone. Each node in the graph $v \in V_d$ represents a domain name, and each edge, $(u, v) \in A_d$, signifies that u is directly dependent on v for proper resolution of itself and any descendant names. Each edge, $(u, v) \in A_d$, carries a weight, $w(u, v)$, indicative of the probability that it will be

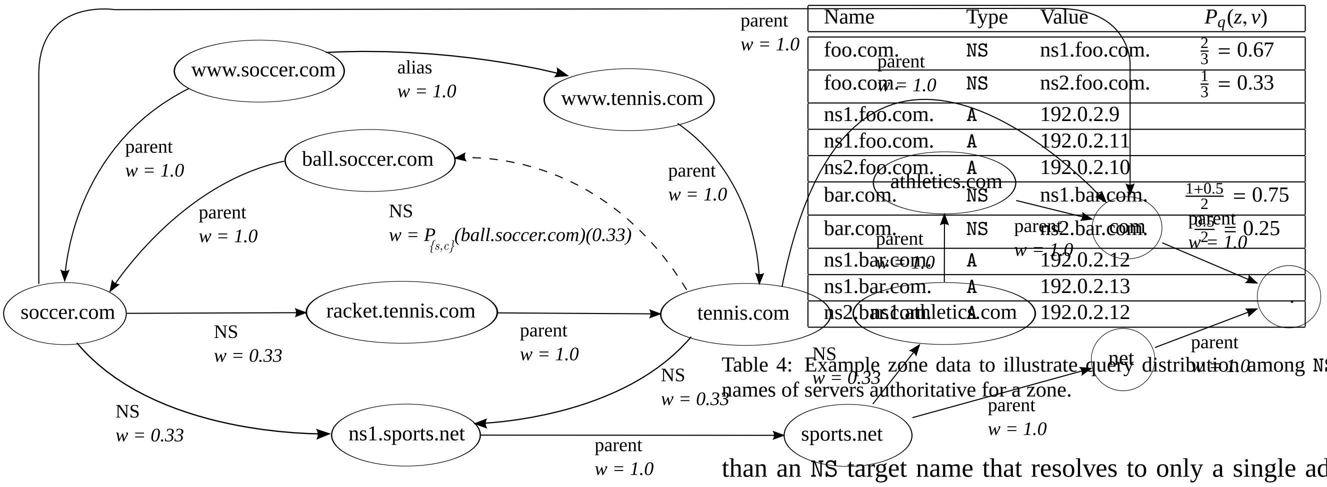

Figure 3: The dependency graph for the domain name *www.soccer.com*, derived from the zone data in TABLE 1. The solid lines represent active influence, and the dashed lines represent passive influence.

Name	Type	Value	$P_q(z,v)$
foo.com.	`NS`	ns1.foo.com.	$\frac{2}{3} = 0.67$
foo.com.	`NS`	ns2.foo.com.	$\frac{1}{3} = 0.33$
ns1.foo.com.	`A`	192.0.2.9	
ns1.foo.com.	`A`	192.0.2.11	
ns2.foo.com.	`A`	192.0.2.10	
bar.com.	`NS`	ns1.bar.com.	$\frac{1+0.5}{2} = 0.75$
bar.com.	`NS`	ns2.bar.com.	$\frac{0.5}{2} = 0.25$
ns1.bar.com.	`A`	192.0.2.12	
ns1.bar.com.	`A`	192.0.2.13	
ns2.bar.com.	`A`	192.0.2.12	

Table 4: Example zone data to illustrate query distribution among `NS` target names of servers authoritative for a zone.

followed for resolving u. A name dependency graph for domain name *www.soccer.com* is shown in Figure 3, built from the data in TABLE 1.

Edges are placed on the graph from each domain name $u, u \neq r$ to its parent $Parent(u)$ with $w\big(u, Parent(u)\big) = 1$; a domain name is always dependent on its parent. If resolution of domain name u yields a `CNAME` RR, then an edge is placed between u and its target name, $Cname(u)$, with $w(u, Cname(u)) = 1$; the resolution of an alias is always dependent on the resolution of its target. Such edges in Figure 3 are those between *www.soccer.com* and its parent, *soccer.com*, and between *www.soccer.com* and its canonical name, *www.tennis.com*.

Placement of edges and weights corresponding to `NS` target dependencies draws from the discussion in Section 5.1 and is summarized in TABLE 3.

We first identify the proportion of queries distributed among each of the `NS` target names in NS_z, which we use as a base for calculating the weights of edges in A_d stemming from `NS` target dependencies. Resolvers prepare their lists of servers for selection from *addresses* rather than *names* of authoritative servers, as explained previously. Although a particular resolver instance will gravitate towards querying a single authoritative server for a zone, we reiterate our assumption that queries from arbitrary resolvers are distributed uniformly among the servers, due to network diversity. Thus, the probability, $P_q(z, v)$, of querying any `NS` target $v \in NS_z$ for resolution of z will be some fraction of $|NSA_z|$ that reflects the proportion of server addresses attributed to v. Let S_v represent the set of addresses to which $v \in NS_z$ resolves. A naïve formula for determining query probability $P_q(z, v)$ is to simply calculate the fraction of total server addresses authoritative for z that correspond to v:

$$P_q(z, v) = \frac{|S_v|}{|NSA_z|}$$

The zone data for *foo.com* in TABLE 4 shows that an `NS` target name that resolves to multiple addresses, such as *ns1.foo.com*, has a higher probability of being queried for names in the zone than an `NS` target name that resolves to only a single address, such as *ns2.foo.com*.

It is possible that multiple `NS` target names in NS_z resolve to the same address, so a single address in S_v may also be attributed to other names in NS_z. A more complete approach to determining query probability therefore is to evenly divide the probabilistic weight attributed to a server address among all the names that resolve to that address:

$$P_q(z, v) = \frac{\sum_{s \in S_v} |\{u \in NS_z | s \in S_u\}|^{-1}}{|NSA_z|}$$

For example, in TABLE 4 both *ns1.bar.com* and *ns2.bar.com* resolve to 192.0.2.12, so the weight of that server is split evenly among both names. The result is that *ns1.bar.com* is queried with 0.75 probability for *bar.com* because it also resolves to 192.0.2.13, and *ns2.bar.com* is queried with only 0.25 probability.

When $NS_z \neq NS'_z$, the query probability of an edge to `NS` target v depends on v's existence in NS_z and NS'_z and the likelihood that the resolver has cached the `NS` RRset for z from the answer or authority section of an authoritative response received previously. We use $P_{NS}(z) \in [0, 1]$ to denote that likelihood:

$$P_q(z, v) = P_{NS}(z) P\big(v \in NS_z\big) \frac{\sum_{s \in S_v} |\{u \in NS_z | s \in S_u\}|^{-1}}{|NSA_z|} + \big(1 - P_{NS}(z)\big) P\big(v \in NS'_z\big) \frac{\sum_{s \in S_v} \big|\{u \in NS'_z | s \in S_u\}\big|^{-1}}{\big|NSA'_z\big|}$$

For simplicity we assume that $NS_z = NS'_z$ unless specified otherwise.

If `NS` target $v \in NS_z$ is not a subdomain of $Parent(z)$, edge (z, v) is added to G_d with $w(z, v) = P_q(z, v)$. Resolution of v is required for (i.e., actively influences) resolution of z. An example is *soccer.com*'s dependency on *ns1.sports.net*.

If target name $v \in NS_z$ is a subdomain of z, the $Parent(z)$ zone should include a glue record for v. If no glue record exists for v in the $Parent(z)$ zone, then resolution of v is required for (i.e., actively influences) resolution of z, and an edge (z, v) is added to G_d with $w(z, v) = P_q(z, v)$. Such is the case with *soccer.com*'s dependency on *racket.tennis.com*.

If v is in-bailiwick, and a glue record for v exists, then resolution of v is not required for resolving z because the resolver will use the address provided from glue in the additional section of the referral from $Parent(z)$. When $Parent(v) = z$, there

v subdomain	Glue				
of *Parent*(z)	exists	*Parent*(v) = z	$w(z, v)$	Influence type	Example (TABLE 1 and Figure 3)
no			$P_q(z, v)$	Active	*soccer.com* → *ns1.sports.net*
yes	no		$P_q(z, v)$	Active	*soccer.com* → *racket.tennis.com*
yes	yes	no	$P_{\{s,c\}}(v)P_q(z, v)$	Passive	*tennis.com* → *ball.soccer.com*
yes	yes	yes	0	None	*soccer.com* → *ball.soccer.com*

Table 3: Rules for determining whether or not and with what weight $w(z, v)$ a edge is placed between a zone z and an NS target $v \in NSA_z$.

is no edge (z, v) in G_d; all servers authoritative for z have the authoritative data for v, such as with *ball.soccer.com*'s relationship to *soccer.com*. However, when $Parent(v) \neq z$ an edge (z, v) is added with $w(z, v) = P_{\{s,c\}}(v)P_q(z, v)$; the name v passively influences z, dependent on the probability that either the resolver or the authoritative server has the address for v cached from an authoritative source. An example is *tennis.com*'s dependency on *ball.soccer.com*. Passive influence is analyzed more closely in Section 7.

5.3. Level of influence

The *trusted computing base* (TCB) for domain name d is the set of all domain names in its dependency graph, G_d. An analysis of the raw size of domain name's TCB or even an enumeration of its constituents may be interesting, but insufficient for understanding the makeup of its influence. Using the well-formed dependency graph and its edges we can calculate the *level of influence*, which is the probability that one domain name will be utilized for resolving another. Quantifying influence of name dependencies shows that some names may be much more influential than others. We let $I_u(v) \in [0, 1]$ denote v's level of influence on u.

An analysis of the *dependency paths* in G_d is necessary to determine the level of influence of the domain names $v \in V_d$ on d. The dependency paths in G_d are modeled by performing a depth-first traversal of G_d, beginning with d. This depth-first traversal produces the exhaustive set of acyclic intermediate paths of name dependencies for resolving d. The level of influence is calculated by determining the probability that paths leading from d will reach v during resolution:

$$I_d(v) = P(d, \ldots, v)$$

To calculate $P(d, \ldots, v)$, the probabilities of encountering v in the dependency paths beginning with each of u's direct dependencies must first be recursively calculated and aggregated. The probability of encountering v in a path beginning with edge $(u, j) \in A_d$ is calculated by multiplying the probability, $w(u, j)$, of following edge (u, j) by the probability of encountering v in a path beginning with j:

$$P(u, j, \ldots, v) = \begin{cases} w(u, j) & \text{if } j = v \text{ (direct dep)} \\ 0 & \text{if } j = r \text{ (root)} \\ w(u, j)P(j, \ldots, v) & \text{otherwise} \end{cases}$$

A particular domain name $u \in V_d$ may be directly dependent on multiple names, resulting in multiple out-edges from u. Some of these dependencies are mutually exclusive, and others are independent of one another. At most one NS target dependency of u is necessary for resolution (assuming the server queried is responsive), but alias and parent dependencies exist independently of the NS target dependencies. For example, when resolving names in *tennis.com* using the zone data from TABLE 1, either *ns1.tennis.com*, *ball.soccer.com*, or *ns1.sports.net* will be selected for query, each with equal probability, and only the latter two result in a name dependency. However, resolution of *tennis.com* remains entirely dependent on its parent, *com*, regardless of which server is selected for query.

Aggregating the probability of encountering v in paths beginning with each of u's direct dependencies is as follows. First the probability of encountering v through any NS-type dependencies is determined by calculating the sum of encountering it in each of the NS-type dependency edges because the probabilities are dependent on one another:

$$P(u, [NS\ dep], \ldots, v) = \sum_{j \in NS_u} P(u, j, \ldots, v)$$

This probability is then combined independently with the probability of encountering v in paths beginning with any alias- or parent-type dependencies:

$$\begin{aligned} P(u, \ldots, v) = \quad & 1 - \Big(1 - P\big(u, Parent(u), \ldots, v\big)\Big) \\ & \Big(1 - P\big(u, Cname(u), \ldots, v\big)\Big) \\ & \Big(1 - P\big(u, [NS\ dep], \ldots, v\big)\Big) \end{aligned}$$

Using these expressions, we calculate the level to which *sports.net* influences *soccer.com*:

$$\begin{aligned} & I_{www.soccer.com}(sports.net) = \\ & \quad 1 - \big(1 - P(www.soccer.com, soccer.com, \ldots, sports.net)\big) \\ & \quad \big(1 - P(www.soccer.com, www.tennis.com, \ldots, sports.net)\big) \\ & \ldots \\ & = 0.62 + 0.06P_{\{s,c\}}(ball.soccer.com) \end{aligned}$$

5.4. Dependency metrics

Calculating the individual influence of all dependencies of a domain name is computationally complex and ultimately not useful for a comparable analysis of two domain names. A more useful metric would aggregate influence of a domain name in a representative way such that can be meaningfully compared with that of other names.

In this section we present several qualitative graph properties describing the makeup of the dependency graph for a domain name and use TABLE 5 to exemplify these properties for

Zone	I	NT	FO
soccer.com	yes	yes	yes
tennis.com	yes	yes	yes
sports.net	yes	yes	yes
athletics.com	yes	yes	no
com	yes	no	no
net	yes	no	no
.	yes	no	no

Table 5: Influential (I), non-trivial (NT), and first-order (FO) zones for *www.soccer.com*.

```
procedure NonTrivialZones($d$)
    $D \leftarrow \{Parent(d)\}$
    for all $(u, v) \in A_d$ do
        if $(u, v)$ is an NS target or alias dependency then
            $D \leftarrow D \bigcup \{Parent(v)\}$
    return $D$
```

Figure 4: The NonTrivialZones algorithm identifies all non-trivial zones in the dependency graph for a domain name, *d*.

www.soccer.com (Figure 3). We then use these properties to create normalized metrics for a quantitative analysis of domain names.

Influential zones. The set of zones influencing domain name d is a subset of all the influential domain names in d's dependency graph, $Z_d \subseteq V_d$. It represents the diversity of the namespace influencing resolution of d.

Non-trivial zones. Non-trivial zones are the result of explicitly configured inter-zone dependencies. Included in this set are the parent zones of any NS or alias targets in A_d: $U_d \subseteq Z_d$. A non-trivial zone *foo.bar.com* that influences d may contribute up to four zones to Z_d, itself and each of its ancestors. However, if no in-edges resulting from alias- or NS-type dependencies exist for any of its ancestor zones, then they exist in Z_d only because *foo.bar.com* is explicitly configured as a dependent zone and are thus trivial. The algorithm in Figure 4 identifies non-trivial zones by iterating the set of edges A_d and adding the parent zones of NS and alias targets.

First-order zones. A subset of non-trivial zones $U'_d \subseteq U_d$ are explicitly configured by d (or *Parent*(d), if d is not a zone) and comprise *first-order zones*. This subset also includes the non-trivial zones in the ancestry of each explicitly configured zone. Figure 5 finds all the alias (lines 5–7) and NS target (line 11) dependencies for a name d and then includes the parent zone for each target (line 15) and each non-trivial zone in its ancestry (lines 16–21).

The sizes of the sets of all, non-trivial, and first-order zones influencing a domain name can be used for a quantitative analysis. However, these metrics are unbounded and may not be entirely representative. It might be the case, for example, that a zone has a large number of influential zones, but 90% of its influence is contained within 5% of those zones. The following metrics use the preceding graph properties to quantify influence with values normalized between 0 and 1, for representative comparison.

First-order ratio. The *first-order ratio* is used to determine the percentage of non-trivial zones that are expressly configured by

```
 1: procedure FirstOrderDeps($d$)
 2:     $N \leftarrow$ NonTrivialZones($d$)
 3:     /* $M$ is the set of explicitly configured names for $d$ */
 4:     $M \leftarrow \{d\}$
 5:     if $d$ is not a zone then
 6:         if $d$ is an alias then
 7:             $M \leftarrow M \bigcup \{Cname(d)\}$
 8:         $d \leftarrow Parent(d)$
 9:     /* Add NS target edges for zone $d$ to $M$ */
10:     $M \leftarrow M \bigcup \{u \in V_d | \exists(d, u) \in A_d$, NS target dep.$\}$
11:     $D \leftarrow \{d\}$
12:     /* Add non-trivial zones in $M$'s ancestry to $D$ */
13:     for all $u \in M$ do
14:         $v \leftarrow Parent(u)$
15:         while $v \neq r$ do
16:             if $v \in N$ then
17:                 $D \leftarrow D \bigcup \{v\}$
18:             $v \leftarrow Parent(v)$
19:     return $D$
```

Figure 5: The FirstOrderDeps identifies all the non-trivial zone dependencies for domain name *d* which are explicitly configured by DNS administrators.

the administrators of d: $\frac{U'_d}{U_d}$. Values closer to 1 indicate that the administrators are largely in control of the zones comprising the TCB.

Third-party influence. Third-party influence (TPI) is used to quantify how much domain name d is influenced by names not explicitly controlled by its administrators, i.e., $I_d(U_d - U'_d)$. Figure 6 details the ThirdPartyInf algorithm, which is used to calculate TPI. The TPI of d's alias, if any (lines 5–7), is combined (lines 15–16) with the TPI of its parent zone (lines 9–10) and that of its collective NS target dependencies (lines 11–14). Two helper algorithms are utilized: the ControlledAlias algorithm (lines 17–26) analyzes a name to determine whether or not it aliases (directly or indirectly) another name outside of the set of zones in set D. The ThirdPartyInfD algorithm (lines 27–42) determines the probability that resolution of u will utilize a name outside the set of zones in set D.

6. Data Collection and Analysis

In this section we describe the methodology we employed for collecting data from the DNS infrastructure, and provide analysis of the data collected. We analyze several different characteristics of our data using the metrics presented in Section 5 to assess quality of name resolution.

6.1. Data collection

We populated a database of name dependencies by crawling the namespace of known domain names. A set of over 3 million hostnames was extracted from URLs submitted to the Open Directory Project (ODP) at DMOZ [19]. These names were combined with over 100,000 names received as queries by the recursive servers at the International Conference for High-performance Computing, Networking, Storage and Analysis (SC08) [20]. The ODP/SC08 seed names represent both names with some intention of being resolved for access to production services (ODP names) and names actually queried by clients (SC08 names).

Measurement	Values
ODP/SC08 hostnames	3,167,594
Total domain names collected	8,439,927
Total zones	2,996,460
NS target dependencies	6,855,379
NS targets requiring glue	3,723,203 (54%)
NS targets missing required glue	901 (0.024%)
Zones for which $NS_z \neq NS'_z$	587,865 (20%)

Table 6: A summary of results collected from surveying the DNS namespace, seeded with ODP/SC08 hostnames.

We recursively analyzed and identified dependency relationships for each ODP/SC08 name by issuing directed queries for the name itself and each of its dependencies, consisting of names in its ancestry and the targets of NS, MX, and CNAME RRs.

For each zone z, we obtained the delegation and authoritative NS RRsets using the following methodology. We directed a query for z to each server authoritative for $Parent(z)$ to elicit a referral, seeking a response without the authoritative answer (AA) flag set. Since it is possible for a server to be authoritative for both z and $Parent(z)$, only if the AA flag was clear could we be sure that the NS RRset returned reflected the delegation records maintained in the $Parent(z)$ zone. We obtained the authoritative NS RRset by querying a server authoritative for z. A mismatch between the two RRsets indicated an inconsistency.

We searched the additional section for the presence of glue records in referrals. Additional records may not necessarily indicate the presence of glue. For example, a referring server might be authoritative for the NS target for which the address record is being supplied. However, we optimistically assume that an additional record corresponding to an in-bailiwick NS target reflects a glue record in the parent.

If we were unable to elicit a non-authoritative referral for z, then we could not determine inconsistencies between NS'_z and NS_z nor detect the presence of glue records. However, in practice, if the set of servers authoritative for $Parent(z)$ are a subset of those authoritative for z, then consistency is satisfied implicitly since all servers will send the authoritative records from z over corresponding records from $Parent(z)$ [17]. For all zones in our analysis we assumed a 0.5 likelihood that a resolver had in its cache the authoritative NS RRset for z (i.e., $P_{NS}(z) = 0.5$), such that NS target names in both NS_z and NS'_z were considered for server selection with equal probability.

The results from our analysis are summarized in TABLE 6.

One interesting caveat to our analysis is that the authoritative server names for several TLDs are not subdomains of the TLDs for which they are authoritative. For example, the authoritative servers for *com* are in the *gtld-servers.net* zone. These server names don't result in active influence on affected TLDs because they are in-bailiwick (subdomains of the root zone) and glue records exist. However, these TLDs are subject to passive influence if resolvers have A RRsets in their caches from authoritative responses.

As part of our analysis we evaluated TCB size and third-party influence with different values of $P_{\{s,c\}}(v)$, the likelihood that an authoritative A RRset for any NS target v exists in the cache of the resolver or the referring authoritative server. The third-party influence for subdomains of affected TLDs was always at least the value of $P_{\{s,c\}}(v)$ in such cases. When $P_{\{s,c\}}(v) = 1$, for example, any domain names ending in *com* or *edu* had a third-party influence of 1.0.

Metric	$P_{\{s,c\}}(v)$	Avg.	Max.
Influential zones	0	5.26	72
Influential zones	> 0	16.53	180
Non-trivial zones	0	2.26	45
Non-trivial zones	> 0	11.65	146
First-order zone ratio	0	0.92	1.0
First-order zone ratio	> 0	0.63	1.0
Third-party zone influence	0	0.08	1.0
Third-party zone influence	0.5	0.38	1.0
Third-party zone influence	1.0	0.55	1.0
Non-trivial organizations	0	1.67	37
Non-trivial organizations	> 0	8.41	113
Third-party organization influence	0	0.04	1.0
Third-party organization influence	0.5	0.34	1.0
Third-party organization influence	1.0	0.49	1.0

Table 7: TCB and influence statistics for the ODP/SC08 hostnames.

While the passive influence of such TLDs is legitimate, its presence in our analysis obscures the level of passive influence incurred due to configuration of subdomains. Because the naming convention of TLD authoritative servers is deliberate, and the effects of passive influence on TLDs are likely negligible, we did not consider their passive influence, with the exception of country-code TLDs (e.g., *us*, *fr*), which we felt might be more interesting because influence indicates the potential crossing of political boundaries.

6.2. Trusted computing base

The raw size of the TCB for hostnames collected in terms of influential zones and non-trivial zones is shown in Figure 7 as a cumulative density function (CDF), and the statistics are shown in TABLE 7. Nearly all hostnames have a TCB smaller than 20 zones when $P_{\{s,c\}}(v) = 0$, and the average size of the TCB was 2.26 non-trivial zones and 5.26 total zones—both of which are reasonably small. When $P_{\{s,c\}}(v) > 0$, the average size of the TCB increases several times to 11.65 non-trivial and 16.53 total zones. Only about 80% have fewer than 20 zones; most of the remaining 20% have between 30 and 90 non-trivial and total zones in their TCB. Caching and using NS target names from authoritative sources, rather than glue, can increase the size of the TCB of a domain by several times.

6.3. Controlled influence

The first-order ratio of the ODP/SC08 hostnames is shown in Figure 8. The average first-order ratio was 0.92 for $P_{\{s,c\}}(v) = 0$ and 0.63 for $P_{\{s,c\}}(v) > 0$, indicating that control of the TCB is lost as caching of NS target names is introduced. When $P_{\{s,c\}}(v) > 0$, third-party zones comprise more than half of the the non-trivial zones in the TCB of roughly 40% of the hostnames surveyed.

Figure 9 shows the third-party influence of the ODP/SC08 hostnames. When $P_{\{s,c\}}(v) = 0$, 85% of the hostnames are not influenced at all by third parties. At $P_{\{s,c\}}(v) = 0.5$ only 60% of the hostnames are influenced less than 50% by third parties.

When $P_{\{s,c\}}(v) = 1$ nearly half of the hostnames are influenced almost certainly by third parties. Again the behavior of caching preference of NS target names from authoritative sources at the resolver and authoritative servers greatly affects third-party influence of domain names.

6.4. Alias chains

One behavior affecting the third-party influence of domain names is the practice of chaining aliases. As an example, *www.example.com* is configured as an alias for *www.example.net* which, in turn, is an alias for *www.example.org*. The practice is common among content distribution networks for offloading the burden of addressing from the administrators of the original name (e.g., *www.example.com*). In our survey, we found that 33,873 (1%) of the ODP/SC08 names were affected by alias chains. Using that subset of affected names, we graphed the third-party influence for analysis in Figure 10 using $P_{\{s,c\}}(v) = 0.5$. Only 10% of the names dependent on alias chains exhibit no third-party influence, compared to 42% of the entire set of ODP/SC08 names. However, when we removed the effects of the alias chains in our analysis, nearly half of the names avoided third-party influence completely, even with $P_{\{s,c\}}(v) > 0$.

6.5. Organizational dependencies

Throughout Sections 5 and 6 we have used zones as the unit of measurement for describing properties of the DNS name dependency graph. In reality the grouping of names or servers for analysis is arbitrary, and the methodology employed depends on certain assumptions and the desired results. Using zones as a unit of measurement may seem reasonable because of the assumption that security and reliability across the servers whose names are in a common zone are consistent. A similar assumption may hold for use of administering organizations as a unit of measurement. Analysis of domain name influence may yield different results when zones are grouped by organization. For example, *sports.com* and *athletics.net* (TABLE 1) are both run by Sports Central, Inc. Although the *athletics.net* zone is a third party to *www.soccer.com*, when analyzed by organization it is grouped with *sports.com* under Sports Central, Inc., a first-order dependency.

To further our analysis, we used the domain suffix from the contact email in theSOA (start of authority) RR of a zone as a method for grouping different zones into a single organization, under the assumption that such reflect the organization responsible for maintenance of the zone. The results are shown in Figure 11. Using this organizational grouping, we found that the number of domain names with no third-party influence increased from 85% to over 90% when $P_{\{s,c\}}(v) = 0$ and from 42% to 48% when $P_{\{s,c\}}(v) > 0$.

7. Reducing Passive Influence

We have observed an increased third-party influence on dependent domain names when there is a higher likelihood of passive influence caused by the existence of an A RRset from an authoritative response in cache for an NS target name for which a glue record exists. This likelihood, $P_{\{s,c\}}(v)$, is comprised of the probability that the A RRset from authoritative source is in the cache of the authoritative server, s, offering the referral or in the cache of the resolver, c, $P_s(v)$ and $P_c(v)$, respectively. We discuss in this section considerations that affect passive influence and make recommendations to contain third-party influence with proper configuration.

7.1. Authoritative servers

The obvious way to minimize $P_s(v)$ is to remove caching functionality from authoritative servers, a DNS best practice [18]. Both NSD and `tinydns` act as authoritative servers without caching. BIND allows dual functionality but provides a configurable option to disallow the inclusion of cached RRsets in the additional section of responses. However, this option is not enabled by default, so if a BIND authoritative server with default settings has records in cache, it will use them to populate the additional section in preference to any glue records in its local configuration.

During our survey of the production DNS space we queried 7,781 distinct servers authoritative for a parent zone in an attempt to detect glue records and discrepancies in the delegation records for a child zone. We found that 1,486 (19%) of these servers returned additional records whose TTL decreased when issued a subsequent request seconds later, indicating that they had come from the cache of the authoritative server. This demonstrates the pervasiveness of caching functionality on production authoritative servers.

7.2. Resolvers

Many factors may contribute to an increased probability of RRsets from authoritative sources in a resolver's cache, $P_c(v)$. Some are environment-specific. For example, the rate of queries to the resolver requesting an authoritative answer for the A RRset of an NS target name would certainly affect this probability, but this behavior is local to the resolver's environment. One consideration that has effects in any environment is the dynamics of TTL values for related RRsets.

Passive influence is minimized when the TTL of a glue record is equal to that of the NS RRset for which it is an NS target. When the TTL of the glue record is less than that of the NS RRset, there is an increased chance that queries for names within the zone will result in induced query for an authoritative response for the A RRset.

We consider a practical example using the zone data from TABLE 1 and illustrate it in Figure 12. Suppose the TTL of the NS RRset for *tennis.com* is 160 and the TTL for the glue record for *ball.soccer.com* is 80. If a resolver with empty cache receives a query for *tennis.com* at time $t = 0$, it caches the *tennis.com* NS RRset and *ball.soccer.com* A RRset from the authority and additional sections of the referral, respectively. For lookups in *tennis.com* during $0 < t \leq 80$ the resolver can query *ball.soccer.com* using the address from its glue record. When $80 < t \leq 160$ the *tennis.com* NS RRset is still cached, but now a lookup is required to obtain the address for *ball.soccer.com*, resulting in an A RRset cached from an authoritative source. Such

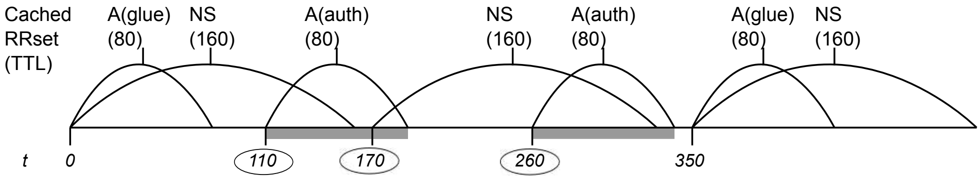

Figure 12: A timeline of queries received by a resolver for names within a particular zone, not found in the resolver's cache. The cache lifetimes for the zone's `NS` RRset and corresponding glue and authoritative `A` RRsets are shown above the timeline, labeled by TTL. Ticks below the timeline mark query arrival times. Queries inducing lookups of `A` RRsets are circled, and the shading highlights the lifetime of authoritative `A` RRsets in cache.

is the case with the query at $t = 110$. Not until $t > 160$, when the *tennis.com* `NS` RRset expires and a referral from a *com* server is again required, does the resolver again receive a glue record for *ball.soccer.com* in the additional section of a response. The query at $t = 170$ causes this lookup. However, because of the relative ranking of trustworthiness explained in Section 4, the RRsets from the additional section do not override data from authoritative sources already in cache. Thus at $t = 260$, when the `A` RRset from authoritative source has expired, the query induces another authoritative lookup, and passive influence persists until both the `NS` and the `A` RRsets expire from cache. After both RRsets have expired, the query at $t = 350$ induces a query by the resolver resulting in a referral response with the `NS` RRset and the `A` RRset from glue, which will both be used for subsequent queries.

We observed this pattern of passive influence in both the BIND and `unbound` resolvers. However, the behavior exhibited by `dnscache` varies because of its cache renewal policy. BIND and `unbound` do not renew the lifetime of an `NS` RRset cached from the authority section of an authoritative response, even when subsequent responses are received with the same contents. However, every time `dnscache` receives a response, it updates its cache with RRsets from all sections. This means that additional RRsets from referrals will replace any answer RRsets from authoritative responses. It also means that `dnscache` does not require a referral from a *com* server as long as the cached *tennis.com* `NS` RRset is renewed from the authority section of responses. With such behavior passive influence may enter less frequently on a `dnscache` resolver, but it may persist longer, depending on query rates.

We note that the presence of authoritative data for an `NS` target in cache does not necessarily mean that a resolver selects the respective server for query—only that the source of the address is now a product of passive influence, and that if selected, it exhibits dependencies of that name, following the model described in Section 5. Also, our example assumes for simplicity that only queries for names in *tennis.com* are received by the resolver during the time span shown.

We use the difference in TTL value between the `NS` RRset for zone z and the `A` RRset for $v \in \mathcal{NS}_z$ as a measure of the impact on cache probability, $P_c(v)$. This difference represents the window in which a resolver has the `NS` RRset for z cached, but the `A` RRset has expired, following an initial query for z. We normalize this value by dividing it by the TTL of the `NS` RRset. For simplicity, we assume that the TTLs for delegation and glue records in the parent zones are the same as the corresponding RRsets in the child zone.

Figure 13 shows a CDF of the normalized difference between authoritative `NS` and glue `A` RRsets for the 40% of total `NS` target dependencies meeting the criteria for potential passive influence. Over 92% of the dependencies are configured such that there is no difference between the TTLs of the `NS` and `A` RRsets. But for nearly 6% of the dependencies, the TTL of the `NS` RRset is two or more times that of the `A` RRset for the corresponding target.

8. Further Model Considerations

The dependency model described in this article considers only the basic process for resolving domain names to IP addresses. In this section we describe how dependency relationships stemming from other DNS processes and record types might be integrated into the model. We also describe the how our model is affected by the deployment of the DNS Security Extensions (DNSSEC).

8.1. Extensions

Our DNS dependency model is based on the three relationships fundamental to resolution of domain names to IP addresses, specifically those between parent and child zones, between `CNAME` RRs and their targets, and between `NS` RRs and their targets. However, the model may be extended to consider other dependency relationships in the DNS. We provide several such examples, though fully integrating them into our model is beyond the scope of this article.

The record data of an `MX` (mail exchange) RR contains a target domain name designating the host to which mail addressed to the owner name of the RR should be delivered, as well as a numerical preference for that target. Multiple `MX` RRs may exist for a single owner name, each with different target and preference values. A mail server will first attempt to send a message to the target with the lowest preference value. If connectivity fails, it will attempt delivery to hosts referenced in other `MX` RRs, in increasing order of preference. The dependency model for mail exchange involving domain name d extends the dependency model for resolving d to an IP address. To the already constructed dependency graph G_d, we simply add edges from d to the targets of each of the `MX` RRs. The weight of each such edge is based on the availability of the targets with lower preference values. The dependencies for each `MX` target are then added to the graph, following the original dependency model (i.e., without further considering `MX` RRs), since the behavior

```
procedure ThirdPartyInf($d$)
    $D \leftarrow$ FirstOrderDeps($d$)
    $P_A \leftarrow 0$
    if $d$ is not a zone then
        /* If $d$ is an alias, calculate the TPI of $Cname(d)$ */
        if $d$ is an alias then
            $P_A \leftarrow$ ThirdPartyInfD($Cname(d), D$)
        $d \leftarrow Parent(d)$
    /* Calculate the TPI of $Parent(d)$ */
    $P_P \leftarrow$ ThirdPartyInfD($Parent(d), D$)
    /* Calculate the TPI of each NS target of zone $d$ */
    $P_{NS} \leftarrow 0$
    for all $u \in V_d | \exists(d, u) \in A_d$, NS target dep. do
        $P_{NS} \leftarrow P_{NS} + w(d, u)$ThirdPartyInfD($u, D$)
    /* Aggregate the TPI of all name dependencies */
    return $1 - (1 - P_P)(1 - P_A)(1 - P_{NS})$

procedure ControlledAlias($u, D$)
    $H \leftarrow \{u\}$
    while $u$ is an alias do
        if $Parent(Cname(u)) \notin D$ then
            return False
        else if $Cname(u) \in H$ then /* Loop detected */
            return True
        $H \leftarrow H \bigcup \{u\}$
        $u \leftarrow Cname(u)$
    return True

procedure ThirdPartyInfD($u, D$)
    if $u$ is not a zone then
        /* $u$ aliases a name outside of $D$ */
        if $\neg$ControlledAlias($u, D$) then
            return 1.0
        $u \leftarrow Parent(u)$
    $P \leftarrow 0$
    /* Aggregate influence outside $D$ for $u$'s ancestors */
    while $u \neq r$ do
        $P_u \leftarrow 0$
        for all $v \in V_d | \exists(u, v) \in A_d$, NS target dep. do
            if $Parent(v) \notin D$ or $\neg$ ControlledAlias($v, D$) then
                $P_u \leftarrow P_u + w(u, v)$
        $P \leftarrow 1 - (1 - P)(1 - P_u)$
        $u \leftarrow Parent(u)$
    return $P$
```

Figure 6: The ThirdPartyInf algorithm returns the third-party influence for domain name d. The ControlledAlias helper algorithm returns False if domain name u directly or indirectly aliases a name outside of a set of zones, D; otherwise it returns True. The ThirdPartyInfD helper algorithm returns the influence on domain name u by names outside of a set of zones, D.

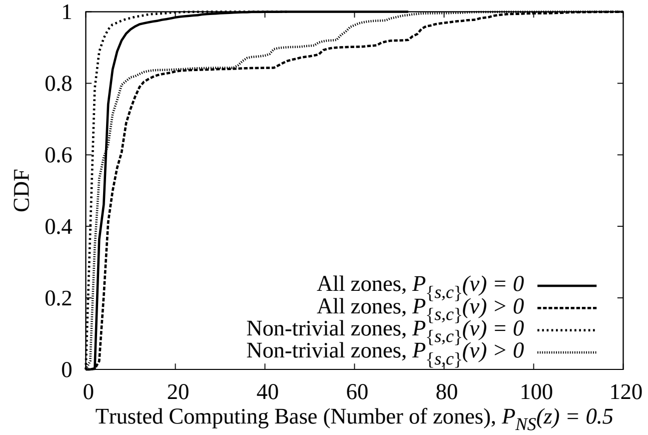

Figure 7: The CDF for the size of the TCB of ODP/SC08 hostnames. Included are the CDF for the number non-trivial and total zones in the TCB, for $P_{\{s,c\}}(v) = 0$ and $P_{\{s,c\}}(v) > 0$.

of a mail server is to resolve each target to an IP address for delivery.

The DNAME RR type [21] was introduced to allow an entire domain to alias another domain, e.g., for seamless organizational transition. The net effect is that any name which is a subdomain of a domain name for which a DNAME RR exists results in a substitution of the DNAME owner for its target in the name's suffix. For example, if *example.com* has a DNAME RR whose target is *example.net*, then *www.example.com* will alias *www.example.net* in the DNS response. Our dependency model can be extended to include the DNAME RR type by simply adding a dependency edge from the node with the owner name to the DNAME target, and recursively following the dependencies of its target.

8.2. DNSSEC

The DNS Security Extensions (DNSSEC) [6, 7, 22] introduce authentication into the DNS. DNS data is signed on a per-zone basis and the signatures and public keys for the data are published in the zone and returned in DNS responses. Public keys for zones are authenticated by their parent zones, which creates a chain of trust from a zone through its ancestry to the root zone. We provides a few insights on DNSSEC from the perspective of our dependency model.

First, DNSSEC enables authentication of RRsets. Thus, barring key compromise, third parties in a domain name's TCB cannot ultimately change the result of RRsets having a complete chain of trust to the root zone without being detected by a validating resolver that trusts the root keys. However, in the case of alias chains involving one or more aliases that span multiple zones, every RRset in the chain must be authenticated for security to be complete. If, from our DNS example, *www.soccer.com* is signed, but *www.tennis.com* is not, then

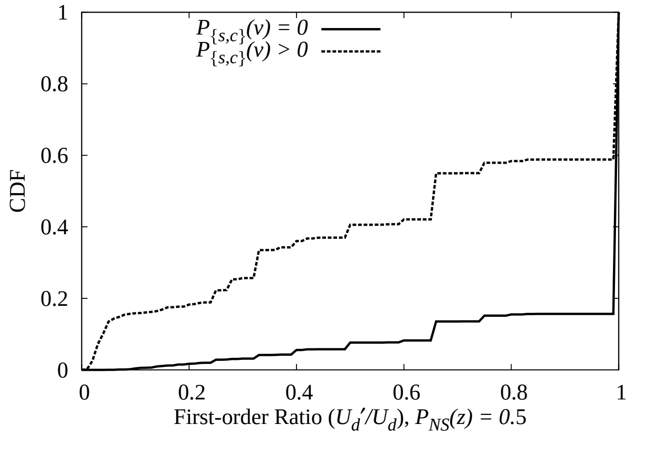

Figure 8: The CDF for the first-order ratio of the ODP/SC08 hostnames.

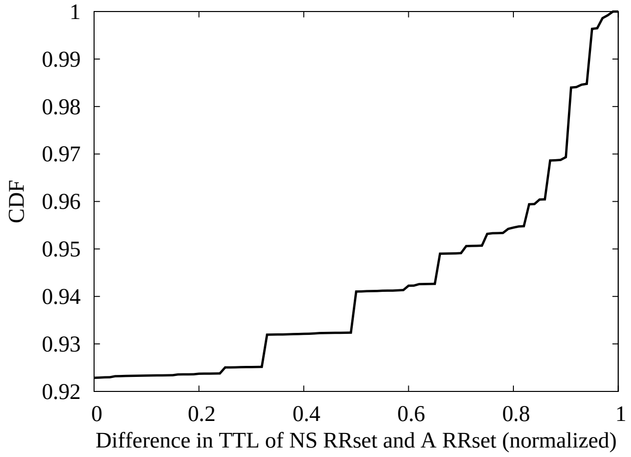

Figure 13: The CDF showing the normalized difference between NS and A RRsets for NS-type dependencies in the data seeded by the ODP/SC08 names.

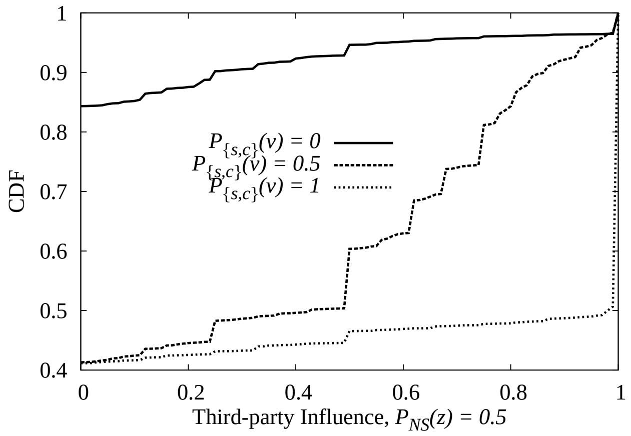

Figure 9: The CDF for the third-party influence of the ODP/SC08 hostnames, grouped by zone.

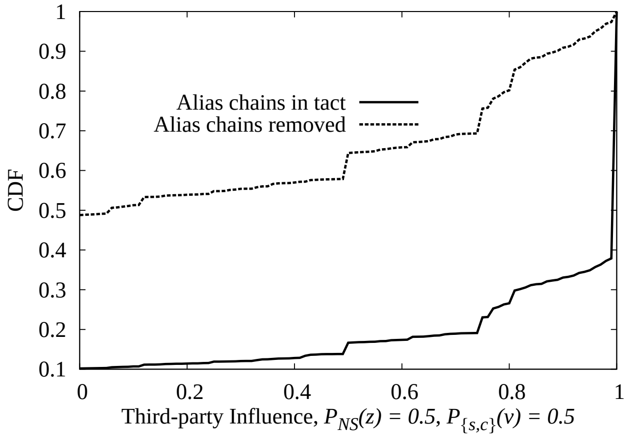

Figure 10: The CDF for the third-party influence of ODP/SC08 hostnames affected by alias chains, grouped by zone.

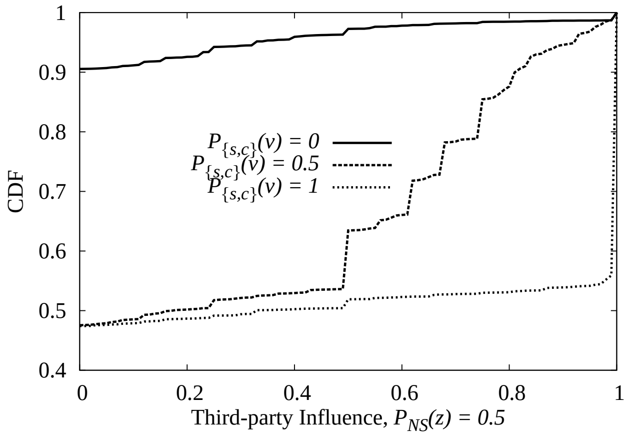

Figure 11: The CDF for the third-party influence of ODP/SC08 hostnames, grouped by organization.

authentication does not prevent tampering of *www.tennis.com*, which the ultimate source of information of data for *www.soccer.com*.

While DNSSEC provides a means to protect against forgery, it also introduces additional complexity into the DNS, both in protocol and administrative overhead. Research has shown that early deployment has been challenging, and misconfiguration has been pervasive [23]. A larger TCB for a domain name leads to a larger set of zones that must be properly configured, and DNSSEC adds potential complications to that. Administrators should be particularly aware of their DNS deployment in a system that employs DNSSEC to avoid name resolution outage.

9. Conclusion

In this article we have presented a graph model for analysis of name dependencies in DNS, which was based on specification and behavior of deployed DNS servers. We defined the trusted computing base (TCB) of a domain name in terms of zones and organizations. We also derived metrics for assessing the dependency model of a domain name. Among these were the level of influence of influential domain names, and third-party influence—the probability that resolution of a domain name will utilize namespace outside the explicit configuration of domain administrators.

We observed that the TCB of domain names, when measured by influential zones and organizations, is much smaller than previously thought. On average 92% of the non-trivial zones in the TCB of a domain name were explicitly configured by the domain administrators. However, the practice of resolver and authoritative servers using address records corresponding to `NS` targets from cache, rather than from additional records in a response or from glue, can increase the size of the TCB and the influence of third-party namespace significantly.

To maximize the reliability of name resolution from the perspective of both resolver and authoritative server, administrators and designers of DNS services should be aware of their server configurations as well as the names and organization comprising the TCBs of their domain names. Administrators should review the role of name servers in their environment to minimize the influence of third parties. The practice of chaining domain name aliases increases the potential for third-party influence and should be avoided, as suggested in RFC 1912 [24]. Additionally, we recommend that the roles of authoritative server and caching server be kept distinct or that authoritative servers be configured to not include information from their caches, as this creates additional dependence. Also, DNS administrators should minimize the difference in TTL value between `NS` RRsets and the glue records for their corresponding `NS` targets.

A better understanding of DNS dependencies and an application of that understanding will put more control into the hands of DNS administrators and mitigate the risks associated with large and diverse TCBs and high third-party influence.

[1] P. Mockapetris, RFC 1034: Domain names - concepts and facilities (1987).

[2] P. Mockapetris, RFC 1035: domain names - implementation and specification (1987).

[3] P. Mockapetris, K. J. Dunlap, Development of the domain name system, SIGCOMM Comput. Commun. Rev. 18 (4) (1988) 123–133. doi:http://doi.acm.org/10.1145/52325.52338.

[4] V. Ramasubramanian, E. G. Sirer, Perils of transitive trust in the domain name system, in: IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement, USENIX Association, USENIX Association, Berkeley, CA, USA, 2005, pp. 379–384.

[5] V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis, L. Zhang, Impact of configuration errors on DNS robustness, in: SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, ACM, ACM, New York, NY, USA, 2004, pp. 319–330.

[6] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, RFC 4033: DNS security introduction and requirements (2005).

[7] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, RFC 4034: Resource records for the DNS security extensions (2005).

[8] IKS, DNSSEC.
URL `https://www.iks-jena.de/leistungen/dnssec.php`

[9] E. Osterweil, D. Massey, L. Zhang, Deploying and monitoring DNS security (DNSSEC), in: 25th Annual Computer Security Applications Conference (ACSAC '09), 2009.

[10] SecSpider.
URL `http://secspider.cs.ucla.edu/`

[11] D. Dagon, M. Antonakakis, P. Vixie, T. Jinmei, W. Lee, Increased DNS forgery resistance through 0x20-bit encoding: security via leet queries, in: Proceedings of the 15th ACM conference on Computer and communications security, ACM, 2008, pp. 211 – 222.

[12] R. Perdisci, M. Antonakakis, X. Luo, W. Lee, WSEC DNS: Protecting recursive DNS resolvers from poisoning attack, in: Dependable Systems & Networks, 2009. DSN '09. IEEE/IFIP International Conference on, IEEE, 2009, pp. 3 – 12. doi:10.1109/DSN.2009.5270363.

[13] ISC BIND. [link].
URL `http://www.isc.org/products/BIND/`

[14] Unbound. [link].
URL `http://www.unbound.net/`

[15] djbdns. [link].
URL `http://cr.yp.to/djbdns.html`

[16] NSD. [link].
URL `http://www.nlnetlabs.nl/projects/nsd/`

[17] R. Elz, R. Bush, RFC 2181 - clarifications to the DNS specification (1997).

[18] R. Chandramouli, S. Rose, Secure domain name system (DNS) deployment guide.
URL `http://csrc.nist.gov/publications/nistpubs/800-81r1/sp-800-8`

[19] Open Directory Project. [link].
URL `http://www.dmoz.org/`

[20] SC08: The International Conference for High-performance Computing, Networking, Storage and Analysis. [link].
URL `http://sc08.supercomputing.org/`

[21] M. Crawford, RFC 2672 - non-terminal DNS name redirection (1999).

[22] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, RFC 4035: Protocol modifications for the DNS security extensions (2005).

[23] C. Deccio, J. Sedayao, K. Kant, P. Mohapatra, Quantifying and improving dnssec availability, in: Proceedings of the 20th International Conference on Computer Communication and Networks (ICCCN 2011), 2011, pp. 1 – 7.

[24] D. Barr, RFC 1912 - common DNS operational and configuration errors (1996).

Dr. Casey Deccio is a Senior Member of Technical Staff at Sandia National Laboratories in Livermore, CA. He joined Sandia in 2004 after receiving his BS and MS degrees in Computer Science from Brigham Young University, and he earned his doctoral degree from the University of California, Davis, in 2010. Dr. Deccio's research interests lie primarily in modeling and availability analysis of DNS and DNSSEC.

Jeff Sedayao is an enterprise architect in Intel's IT Research Group. He focuses on applying distributed systems technologies such as cloud computing to enterprise IT problems. Jeff has participated in IETF working groups, published papers on policy, network measurement, network and system administration, and authored the O'Reilly and Associates book, *Cisco IOS Access Lists*.

Dr. Krishna Kant has been with Intel Corporation since 1997 and is currently on a visiting appointment at George Mason University, Fairfax, VA. He is also serving as a program director at the National Science Foundation where he manages the computer systems research program. His current areas of research include energy efficient and sustainable computing, robustness in the Internet, and cloud computing security. He carries 29 years of combined experience in academia, industry, and government. He has published in a wide variety of areas in computer science and authored a graduate textbook on performance modeling of computer systems.

Dr. Prasant Mohapatra is currently the Tim Bucher Family Endowed Chair Professor and the Chairman of the Department of Computer Science at the University of California, Davis. In the past, he has been on the faculty at Iowa State University and Michigan State University. He has also held Visiting Scientist positions at Intel Corporation, Panasonic Technologies, Institute of Infocomm Research (I2R), Singapore, and National ICT Australia (NICTA). He was/is on the editorial board of the IEEE Transactions on Computers, IEEE Transactions on Mobile Computing, IEEE Transaction on Parallel and Distributed Systems, ACM WINET, and Ad Hoc Networks. He has been on the program/organizational committees of several international conferences.

Dr. Mohapatra received his doctoral degree from Penn State University in 1993, and received an Outstanding Engineering Alumni Award in 2008. He is a Fellow of the IEEE.

Dr. Mohapatra's research interests are in the areas of wireless networks, sensor networks, Internet protocols, and QoS.