Analysis and Measurement of Zone Dependency in the Domain Name System

Jian Jiang*†, Jia Zhang*, Haixin Duan*, Kang Li‡ and Wu Liu*
*Institute for Network Science and Cyberspace, Tsinghua University
Email: zhangjia2017@mail.tsinghua.edu.cn
†University of California, Berkeley
‡Department of Computer Science, University of Georgia

***Abstract*—The Domain Name System (DNS) is a hierarchical distributed system organized through top-down zone delegation. Consequently resolution of a zone depends on its ancestors. However, since the delegation in DNS is designed by name rather than address, the dependency could further extend to other zones. If not configured well, the dependency of a zone could be large and complicated, potentially harmful to its availability and integrity. In this paper, we propose a graph-based model to comprehensively analyze zone dependency in DNS. Our approach classifies zone dependency into four different relations: general dependency, explicit dependency, critical dependency and essential dependency. We also propose an empirical method to quantitatively measure the zone dependencies of given zones. Our survey with over 1 million DNS zones shows that more than 99% of the zones depend on some 3-rd party zone; about 41% of the zones critically rely on more than 2 zones except their ancestors; some TLDs such as `.org`, `.info` and `.cn` tend to have more dependencies than others.**

I. Introduction

The Domain Name System (DNS) is a fundamental infrastructure of the Internet, providing a global mapping service between domain names and IP addresses. Almost every Internet conversation today involves various DNS interactions. A scalable, efficient and secure DNS is critical to the whole Internet.

DNS is a highly scalable and efficient distributed system, with caching and redundancy used in DNS servers. In DNS, all the domain names are organized by a hierarchy of zones. A parent zone server contains information about the delegated servers of child zones, and the resolution of a zone is done by searching through the DNS tree structure.

The hierarchical delegation in DNS naturally introduces *zone dependency*: resolution of a child zone depends on its parent and transitively upper level ancestors; further, since DNS delegation is designed by name rather than by address, the dependency could extend to other branches of the DNS tree. If not configured well, the dependency of a zone could become large and complicated. Large and complicated zone dependency could be harmful to a zone's availability and integrity for several reasons including long initial resolution delay, low reliability, and high security risk for domain hijacking and poisoning. In general, large zone dependency is undesirable from both the reliability and security perspectives, as it introduces large Trusted Computing Base (TCB) to a DNS zone [1].

Zone dependency has been recognized by implementers[2], operators [3] as well as academic researchers [4] [1] [5] [6] [7]. However, previous research limited in certain effects of this problem, yet to have a comprehensive analysis. This paper makes the following contributions on the study of DNS zone dependency. First, we introduced a graph-based model for DNS zone dependencies and classified them into four different dependency relations between DNS zones: general dependency, explicit dependency, critical dependency and essential dependency. Second, we proposed an empirical method to quantitatively measure the zone dependencies of given zones. Third, we applied the method and measured 1 million popular DNS zones.

Our study results in multiple interesting findings: 1) more than 99% zones depend on some 3-rd party zones, 2) about 41% of the zones critically rely on more than 2 zones in attention to ancestors, and 3) some TLDs such as `.org`, `.info` and `.cn` tend to have more dependencies than others.

The rest of the paper is organized as follows. The next section we review some previous work and Section III introduces some background of DNS and the zone dependency problem. Section IV explains our graph-based model and metrics. In Section V we present the results of our survey. We discuss the implications in Section VI, and conclude in Section VII.

II. Related Work

Some previous work have been concerned about the zone dependency problem and some of its negative effects on availability and integrity of DNS resolution.

Ramasubramanian *et al.* were first aware of the potential integrity harm of transitive dependencies in DNS [1]. They defined the TCB size to measure a zone's dependencies, which is essentially the number of name servers of all zones in its general dependency graph in our model. They surveyed the TCB sizes of DNS zones of 593,160 website names and further assessed known vulnerabilities in their DNS software versions. From this study they found that more than 30% names were easy to be compromised. Matthew Dempsky further developed an online website [8] to show the dependencies of TLDs with the same concepts in [1]. Both of their work raised some

978-1-5386-3180-5/18/$31.00 ©2018 IEEE

discussions in DNS operation community [9] [3]. Phokeer *et al.* introduced their case study of DNS lame delegations [10]. They found that a bit chunk(about 55%) of reversed domains registered at AFRINIC is lame.

Pappas *et al.* empirically measured various configuration errors in DNS [4]. One such error is cyclic zone dependency. In our model, the state of cyclic zone dependency can be assessed with the global dependency graphs using standard graph algorithms. Deccio *et al.* noted the potential impact of zone dependency on a zone's availability, They developed a probabilistic model to describe dependencies between DNS names [5]. They also defined two practical metrics for measuring the impact: the minimum number of servers queried and the actual redundancy [6]. This two metrics can also be derived with the dependency graphs in our approach.

A article from Osterweil *et al.* [7] briefly introduced the zone dependency problem to raise awareness in DNS operation and research community. Our work is inspired by this article and the aforementioned efforts.

III. The Zone Dependency Problem

We first briefly review how DNS works, then introduce some background of the zone dependency problem.

A. DNS Overview

DNS is essentially a distributed database organized as a hierarchical tree. On top of the hierarchy is the root zone which delegates the global name-space to various Top Level Domain (TLD) zones. The delegation then continues from TLDs to Second Level Domain (SLD) zones and further to lower levels. Each zone has one or more machines called *authoritative servers* holding DNS data for all names under it. A DNS data entry is called a *Resource Record* (RR), which has a type, a time-to-live (TTL) value and some type-specific data. DNS has defined many RR types. Specifically, `A` and `AAAA` records associate a name with IPv4 and IPv6 addresses; `NS` record delegates a sub-zone to one or more name servers.

Resolving a name (typically its `A` record) involves a client (stub resolver) sending a DNS query to a recursive resolver (resolver for short), which then iteratively goes through the DNS hierarchy. Assuming an empty cache, the resolver starts from the root, then follows the name's delegation chain to find its zone and the corresponding authoritative servers, from which the resolver finally gets answer.

An important detail is that DNS delegation is designed by name, which means the delegation record (*i.e.* , `NS` record) in a delegating zone (*i.e.* , the parent zone) only gives names rather than addresses of authoritative servers of the delegated zone (*i.e.* , the child zone). While in some cases, the resolver could start another round of iteration to resolve an `NS` name; if the `NS` name is under the child zone, this is infeasible because of self-loop. For example, if zone `example.com` has `NS` name `ns.example.com`, since the name is under the zone, then resolving `example.com` requests the address of `ns.example.com` which is circularly dependent on resolving `example.com`. In this case the parent zone must provide additional address information for `ns.example.com`; this is called *glue* record.

Specifically, an `NS` name in DNS could be in four categories [11]: *in-zone delegation* that is directly under the child zone; *grandchild delegation* which belongs to a grandchild zone; *sibling delegation* which is under the parent zone but outside the child zone; *unrelated delegation* which is out of authority of the parent zone. The four categories of `NS` names have different needs for glue records. In the first two cases, a corresponding glue record is expected. Glue record for a sibling delegation is legitimate yet not necessary. An unrelated delegation cannot have glue records in the parent zone because it is out of the authority of the parent [12]. This restriction, called *the bailiwick rule* in DNS community, is dedicated to counter a specific DNS cache poisoning attack [13] [14].

B. The Zone Dependency Problem

The reverse side of the top-down delegation is bottom-up dependency: resolution of a DNS zone essentially depends on its ancestors. However, the dependencies could further extend to other zones if the zone has some out-of-zone `NS` names that fall into the categories of grandchild delegation, sibling delegation and unrelated delegation. For example, in Figure 1, resolution of `cernet.net` depends on `edu.cn` because it has a `NS` name `dns.edu.cn`.

Clearly the zone dependency is transitive, therefore a zone's dependency set might grow unawarely and become very large. In Figure 1, the dependency set of `cernet.net` eventually grows to 21 zones. From security and reliability perspective, this is undesirable. It could have some negative effects such as long initial lookup latency, reduced server redundancy, increased risk of temporary failure and outage, and increased risk of zone hijacking.

This problem has been recognized by research community. However previous work either limited on evaluating certain effects [4] [1] [5] [6] or sketched the surface to raise awareness [7] (see Section II for detailed review). Our goal in this work is to develop a comprehensive analysis of this problem and have a big-picture view of its current state in DNS.

IV. Analysis

We analyze the zone dependency problem with a graph-based model: we view dependency as a binary relation between two zones with transitive property; then we construct a directed *dependency graph* based on the relation to fully describe how many zones are included in the dependency set of a given zone and how the dependencies are introduced. While the concept of dependency graph has been adopted by previous work, our approach clarifies four increasingly fine-grained dependency relations: general dependency, explicit dependency, critical dependency and essential dependency.

A. Dependency Relations in DNS

General Dependency. Following the description in section III, we define *general dependency*: let a and b be two different DNS zones, we say a depends on b if i) b is the

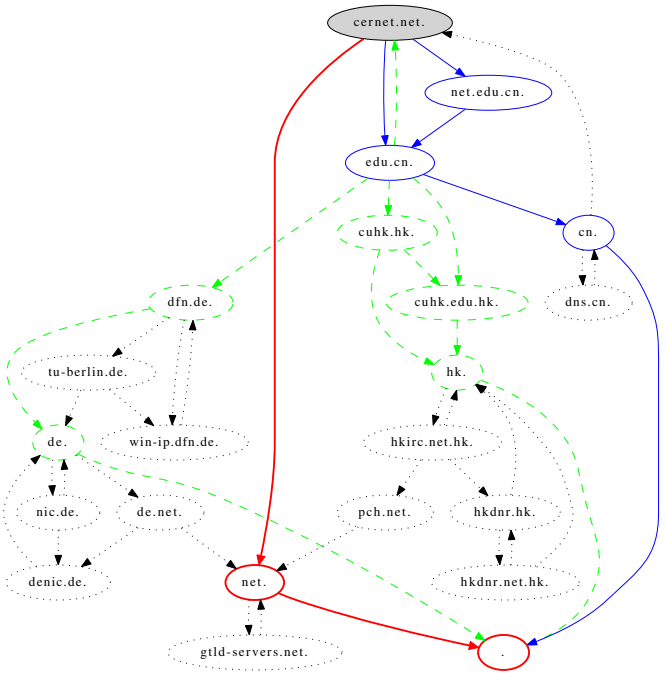


Fig. 1. Dependency graphs of `cernet.net`. $G_{essential}$ includes the source node, bold nodes and edges; $G_{critical}$ includes $G_{essential}$ plus solid nodes and edges; $G_{explicit}$ includes $G_{critical}$ plus dashed nodes and edges; $G_{general}$ includes all nodes and edges.

Algorithm 1 $general\_dependency\_graph(a, g)$

Input: a DNS zone a
Output: a's general dependency graph g

```
g ← DirectedGraph()
_iter_construct_gdgraph(a, g)
return g

procedure _iter_construct_gdgraph(a, g)
if a in g then
    return
end if
add_node(g, a)
/* dependencies introduced by parent */
p ← parent_zone(a)
_iter_construct_gdgraph(p, g)
add_edge(g, a, p)
/* dependencies introduced by NS names */
names ← get_ns_names(a)
for all name in names do
    z ← find_zone(name)
    _iter_construct_gdgraph(z, g)
    add_edge(g, a, z)
end for
```

parent zone of a, or ii) one or more `NS` names of a belong to b, denoted as $D_{general}(a, b)$.

Since the dependency is transitive, we further define the transitive rule: if a, b and c are three different DNS zones and have $D_{general}(a, b)$ and $D_{general}(b, c)$, we then say a is transitively dependent on c through b, denoted as $D_{general}(a, b, c)$.

Following the definition, given a zone a, we can construct its *general dependency graph* by applying a simple recursive algorithm (algorithm 1). We denote the graph as $G_{general}(a)$. The node set of $G_{general}(a)$ includes all zones that resolution of a potentially depends on (both directly and transitively), and the edges indicate how the dependencies are introduced. Figure 1 illustrates the general dependency graph of `cernet.net`.

The relation $D_{general}$ is loosely defined to include all possible dependent zones. However it does not clarify which zones are more important. For example, $G_{general}(a)$ does not label a's parent which is apparently more important than others; while some zones must be reachable for resolving a, some are just occasionally involved. These information are useful to derive some properties of a such as its minimum iterative steps. We therefore need more fine-grained dependency relations to capture the different importance of dependent zones.

Explicit Dependency. We note that an important difference between delegations is whether an `NS` name has corresponding glue records. If has, the dependency it introduces is essentially *implicit* as a resolver could reach its address by glue records, without resolving the dependent zone. Nevertheless, the dependency still exists since a resolver still needs to resolver the `NS` name if its glue expires from cache. That is why we include this type of dependency in the general dependency relation. In comparison, an `NS` name without glue records introduces *explicit* dependency: the resolver must get its address with another round of iteration before using it to resolve the targeted names.

We therefore define *explicit dependency*: we say a DNS zone a is explicitly dependent on another zone b if i) b is the parent zone of a, or ii) one or more `NS` names belong to b and do not have glue records, denoted as $D_{explicit}(a, b)$.

$D_{explicit}$ has similar transitive rule as $D_{general}$. We can also construct a zone a's *explicit dependency graph* with a similar algorithm as algorithm 1, denoted as $G_{explicit}(a)$. Apparently $D_{explicit}$ is a sub-relation of $D_{general}$, therefore $G_{explicit}(a)$ is a subgraph of $G_{general}(a)$ that recursively marks dependencies caused by parent and glueless delegations. Figure 1 also presents the explicit dependency graph of `cernet.net`.

Critical Dependency. Among a zone's dependent zones, some are critical that outage of these zones can cause it being unreachable. More precisely, this is a relation between a zone a and a set of zones that if all zones in the set are unavailable then a will be unresolvable. We then call such sets as a's *critical zone sets*.

Firstly, zone a has a few critical zone sets from its ancestors. For example, in Figure 1, `cernet.net` has two critical zone sets {".net"} and {"."} from its ancestors. Further, if all delegations of zone a are glueless, then a will have an extra critical zone set because resolution of a must rely on at least one of the dependent zones introduced by its delegations. In Figure 1, all delegations of `cernet.net` are glueless, so it

has an extra critical zone set {"edu.cn", "net.edu.cn"}. The extra critical zone set could then transitively introduce more such sets. In summary, cernet.net has 5 critical zone sets: {".net"}, {"edu.cn", "net.edu.cn"}, {"edu.cn"}, {".cn"}, and {"."}.

For zone a and all zones in its critical zone sets, we define *critical dependency*. Following the above explanation, for two different zones a and b, we say a critically depends on b if i) b is the parent of a, ii) $D_{explicit}(a, b)$ holds, and all of a's delegations are glueless. We denote as $D_{critical}(a, b)$. $D_{critical}$ is also transitive and similarly we have $G_{critical}(a)$ for a.

$D_{critical}$ is a special case of $D_{explicit}$ indicating a zone does not have any glue in its parent. Consequently a fresh resolver needs more steps to resolve it. A large $G_{critical}$ also potentially implies increased risk of outage because of 3-rd party zones.

Essential Dependency. Finally we define *essential dependency* for the dependencies between a zone and its parent and transitively its ancestors. We denote the relation as $D_{essential}$, and the corresponding dependency graph as $G_{essential}$. $G_{essential}(a)$ includes the minimum dependencies of a in DNS. Comparing its other dependency graphs with $G_{essential}(a)$ can have an intuitive view of its extra dependencies.

B. Metrics

By interpreting the four dependency graphs of a zone, we can derive its many useful properties. For example, we can get the number of servers a zone potentially depends on by counting all name servers of zones in its general dependency graph (essentially, this is the definition of a zone's TCB size in Ramasubramanian *et al.* 's work [1]). We can also infer the minimum iterative steps for resolving a zone with the explicit and critical dependency graphs of itself and its dependent zones.

Recall that our goal in this work is to have a big-picture view of zone dependency in DNS, we thus limit our scope to a few macroscopic metrics reflecting the sizes and shapes of the dependency graphs.

Our first concern is the number of extra dependent zones of a zone. Formally, given a zone a and its dependency graphs, while $G_{essential}(a)$ includes the minimum dependencies, for $G_n(a) \in \{G_{general}(a), G_{explicit}(a), G_{critical}(a)\}$, the set of extra dependent zones is:

$$Z_n = V(G_n(a)) - V(G_{essential}(a))$$

we denote the *extra dependency size* as:

$$ExtraSize(G_n(a)) = |Z_n|$$

We are also interested in how the extra dependencies are introduced, for which we adopt the maximum and average distance from a zone to its extra dependent zones. Formally, we denote the *maximum extra dependency depth* as:

$$MaxExtraDepth(G_n(a)) = max_{z_i \in Z_n} d(a, z_i)$$

TABLE I
TLD DISTRIBUTION OF THE SAMPLE DNS ZONES.

Rank	TLD	# of zones	%	Rank	TLD	# of zones	%
1	.com	550,959	52.10%	6	.uk	22,894	2.16%
2	.net	74,739	7.07%	7	.jp	20,956	1.98%
3	.org	42,766	4.04%	8	.br	18,300	1.73%
4	.ru	40,819	3.86%	9	.info	15,980	1.51%
5	.de	40,671	3.85%	10	.pl	15,208	1.44%
Total 1,057,359 DNS zones in 256 TLDs							

and the *average extra dependency depth* as:

$$AvgExtraDepth(G_n(a)) = \frac{\sum_i d(a, z_i)}{|Z_n|}$$

where $d(a, z_i)$ is the shortest path from a to z_i in $G_n(a)$.

The above metrics are for measuring zone dependencies of individual zones. It would be nice to have a direct measure of the global state. In fact, assuming we can construct dependency graphs for all zones in DNS, we then can construct the global dependency graphs by merging all the dependency graphs, which we denote as $\mathbb{G}_{general}, \mathbb{G}_{explicit}, \mathbb{G}_{critical}, \mathbb{G}_{essential}$ respectively. These global graphs have same node set yet with different amount of edges. While $\mathbb{G}_{essential}$ includes minimum number of edges (*i.e.* , essential zone dependencies in DNS), for $\mathbb{G}_n \in \{\mathbb{G}_{general}, \mathbb{G}_{explicit}, \mathbb{G}_{critical}\}$, we define its *relative density* as:

$$RelativeDensity(\mathbb{G}_n) = \frac{|E(\mathbb{G}_n)|}{|E(\mathbb{G}_{essential})|}$$

Intuitively, this metric is a comparison between reality and ideal of zone dependencies in DNS from different angles.

V. SURVEY RESULTS

A. Global Graphs

We collected DNS zone samples by crawling delegation chains of the Alexa's top 1 million sites. Finally we got 1,057,359 DNS zones distributed in 256 TLDs. Table I presents the TLD distribution of the sample zones.

We first construct the global dependency graphs with all sample zones. Like other networks in Internet such as Internet topology and WWW, the in-degree distributions of these dependency graphs follow power-laws, as presented in Figure 2. The relative density of $\mathbb{G}_{general}$ is 2.14, indicating all potential dependencies in DNS (represented by surveyed zones) are nearly 2 times of the minimum. The values of $\mathbb{G}_{explicit}$ and $\mathbb{G}_{critical}$ are 1.65 and 1.50 respectively. Figure 3 visualizes these dependency graphs, from which we can have an intuitive observation of how many extra dependencies exist in DNS from different angles.

B. Most Critically Depended-upon Zones

We are also interested in which zones are the most important except root and TLDs. For this purpose, we rank the in-degree of the transitive closure of the global critical dependency

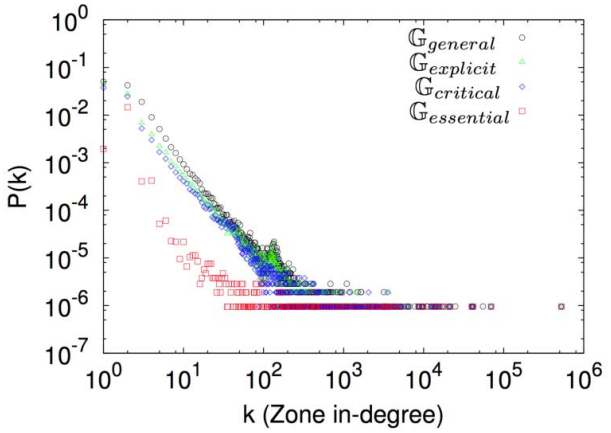


Fig. 2. In-degree distribution of the global dependency graphs. The power-law exponents of $\mathbb{G}_{general}$, $\mathbb{G}_{explicit}$, $\mathbb{G}_{critical}$ and $\mathbb{G}_{essential}$ are 1.86, 1.83, 1.85, 1.42 respectively.

TABLE II
MOST CRITICALLY DEPENDED-UPON DNS ZONES EXCEPT ROOT AND TLDS.

Rank	Zone	# of depended-upon zones	%
1	ovh.net	14,917	1.41%
2	dnsv2.com	13,935	1.32%
3	dynect.net	13,273	1.26%
4	domaincontrol.com	12,871	1.22%
5	dnspod.net	12,737	1.20%
6	ui-dns.[de, biz, com, org]	12,589	1.19%
7	hostgator.com	6,453	0.61%
8	mediatemple.net	5,684	0.54%
9	cloudflare.com	5,451	0.51%
10	nic.ru	4,463	0.42%

graph. Because critical dependency is the strongest dependency relation except dependencies between descendants and ancestors, and the in-degree counts how many zones directly and transitively depend on a zone. Table II presents the top 10 most important zones excluding root and TLDs. The results are not surprising as all listed zones belong to large DNS registrars and DNS hosting providers. These zones are critical to DNS, and even to the whole Internet. As an illustration, users of China Telecom experienced a large-scale Internet outage in May 19 2009. The root cause was a relatively small DDoS attack targeting `dnspod.net` (listed in Table III) which caused many popular domain names were unresolvable, then some misbehaved softwares further congested local resolvers of China Telecom by massive DNS retry-queries.

We then collect statistics of the dependency graphs of individual zones. Figure 4 plots the Cumulative Distribution Function (CDF) of the extra dependency size. We observe that nearly all zones (99.46%) have some extra dependencies. While this might not be surprising because the loose definition of the general dependency, 52.28% sample zones have some glueless delegations (have extra explicit dependencies) and resolution of 41.27% sample zones must rely on some 3rd-party zones (have extra critical dependencies), which are fairly

TABLE III
STATISTICS OF THE DEPENDENCY GRAPHS OF SURVEYED DNS ZONES.

Metric	Dependency Graph	Non-zero %	Non-zero Avg.	Non-zero Median	Non-zero 95%
$ExtraSize$	$G_{general}$	99.46	8.89	4	28
	$G_{explicit}$	52.28	3.17	2	8
	$G_{critical}$	41.27	2.72	2	7
$MaxExtraDepth$	$G_{general}$	99.46	3.74	3	7
	$G_{explicit}$	52.28	2.18	2	3
	$G_{critical}$	41.27	2.09	2	3
$AvgExtraDepth$	$G_{general}$	99.46	2.46	2.00	4.35
	$G_{explicit}$	52.28	1.59	1.50	2.33
	$G_{critical}$	41.27	1.54	1.50	2.00

high rates indicating glueless delegations are pervasive in DNS. Figure 5 and Figure 6 further show the CDF plots of the maximum and average extra dependency depth. For most of the explicit and critical dependencies, the depths are less than 3. Table III presents more statistical values.

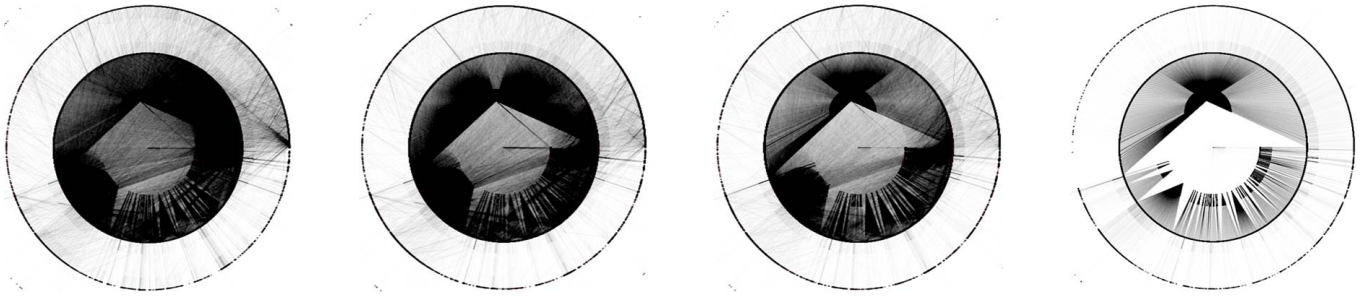
C. Correlation Between Extra Size and Zone Rank

We further analyze the relationship between the extra dependency size of a zone and its popularity. A reasonable hypothesis is that popular zones should have less extra dependencies. However, we do not observe obvious correlation between the two properties from the data of individual zones. On certain aggregation level, the data does show positive correlation, yet the differences are insignificant(Figure 7). Overall the correlation is weak.

Lastly we look at the differences of the extra dependency size between popular TLDs. For the general dependency, the average values of DNS zones under some TLDs (notably `.cn`, `.org` and `.info`) are significantly larger than other TLDs because these TLDs themselves have a large set of extra dependent zones. For the explicit dependency and critical dependency, the values are also quite different between TLDs. Figure 8 shows the percentage of delegation categories of zones in some popular TLDs. Some TLDs (such as `.org`, `.info`, `.cn`, `.fr`, `.eu`) have much higher percentage of unrelated delegations while others are much lower (such as `.com.ru`, `.jp`, `.br`, `.pl`). Consequently much more zones in former TLDs have extra explicit and critical dependencies than those in latter TLDs. Figure 9 shows the differences between these TLDs.

VI. DISCUSSION

Implications. Having large dependency graphs does not certainly imply problematic configurations as the dependency graphs of a zone are merely syntactical mappings of its delegations. For example, the TLD `.org` has a very large general dependency graph, yet all of the dependent zones belong to one organization `Afilias` thus are not really harmful for its integrity. Nevertheless, a zone's dependency graphs give an intuitive observation of its global dependencies that cannot be directly viewed from its local configuration. This information is valuable for administrators to review; it is also worthy for further inspection of potential issues if the graphs are large. As an illustration, one of surveyed zones `vau.fi.` has 6 extra



(a) $\mathbb{G}_{general}$ (b) $\mathbb{G}_{explicit}$ (c) $\mathbb{G}_{critical}$ (d) $\mathbb{G}_{essential}$

Fig. 3. Visualization of the global dependency graphs in DNS represented by surveyed zones.

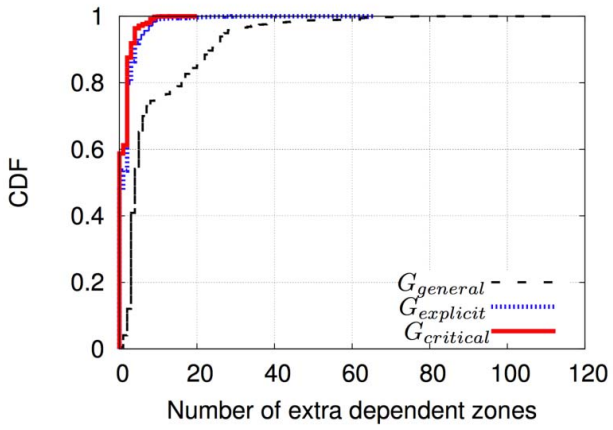


Fig. 4. CDF of the extra dependency size.

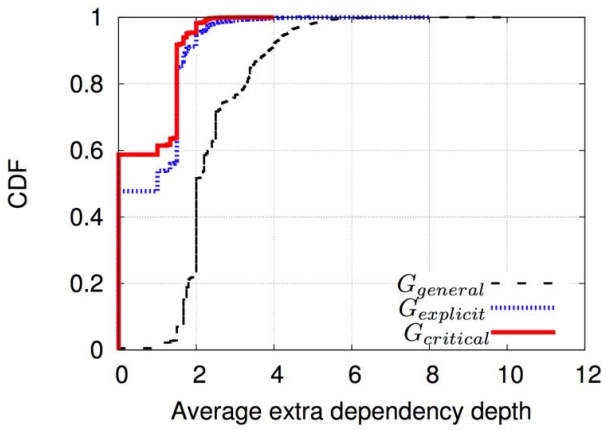


Fig. 6. CDF of the average extra dependency depth.

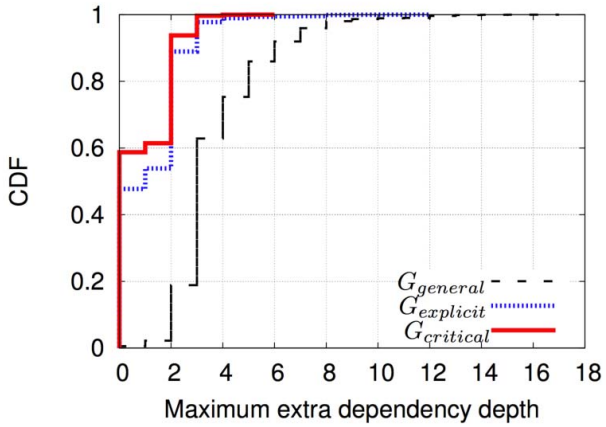
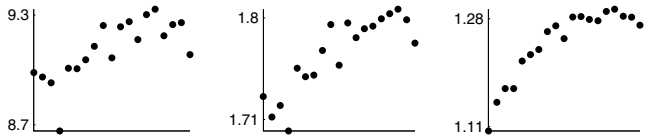


Fig. 5. CDF of the maximum extra dependency depth.



(a) $G_{general}$ (b) $G_{explicit}$ (c) $G_{critical}$

Fig. 7. The average extra dependency size of zones grouped by popularity (50,000 zones per group). The correlation is positive, yet the differences are insignificant.

critically dependent zones; outage of any could completely shut the zone down. From its dependency graphs we can also infer that a fresh resolver needs 11 steps to reach it. These are probably undesirable and unnoticed by its administrator.

Trade-off between redundancy and dependency? Ramasubramanian *et al.* [1] and Dempsky's work [8] on the integrity effects of zone dependency raised discussion in DNS operation community [3]. One argument is that sometimes adding 3rd-party delegations (which results extra dependencies) for a zone is intentional to increase its redundancy. However, as indicated by Bernstein [2], having more redundant servers does not necessarily add more zone dependencies. One can achieve this by creating in-zone alias for out-of-zone name servers and then add the alias as delegation names. In fact, the trade-off is between operational flexibility and dependency, since this approach increases operational costs as administrators need to keep the address of the in-zone alias being valid and consistent. Yet in some cases this approach might be preferable than adding more dependencies.

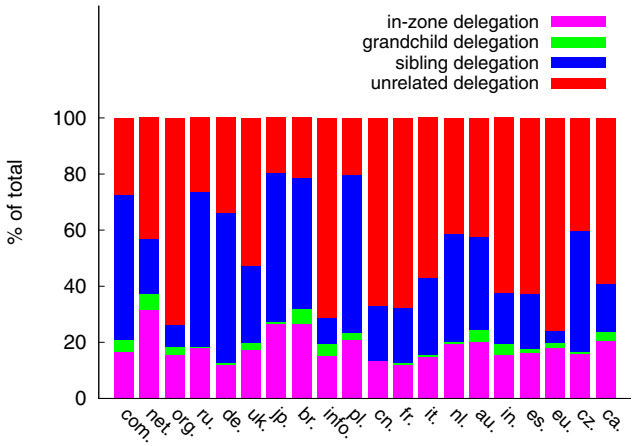


Fig. 8. Percentage of delegation categories breakdown in TLDs.

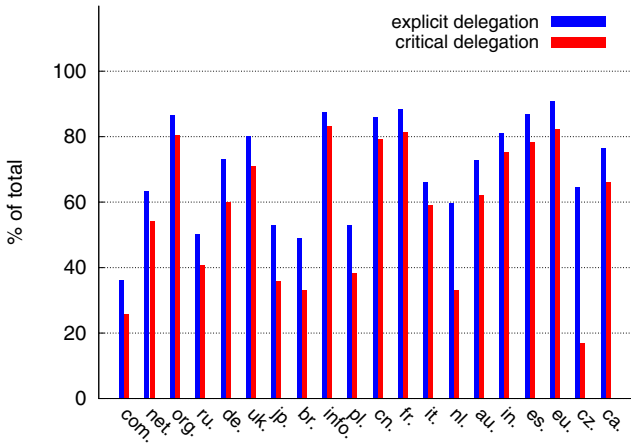


Fig. 9. Non-zero percentage of extra dependency size breakdown in TLDs.

Is the indirect delegation necessary? The root of the zone dependency problem is the indirect delegation. Arguably this design choice is not necessary. Comparing to the current design, directly giving addresses for sub-zone delegation could not only eliminate extra zone dependencies, but also greatly simplify the iteration process of resolver. Delegation by name does gain some operational flexibilities. For example, it reduces possible cross-organizational coordination for renumbering delegation addresses. However we argue that this benefit could be achieved with application layer utilities, not necessarily be included in the core protocol semantics. While it is impractical to really change the current delegation mechanism of DNS, future protocol design should take this into account when facing similar design choices.

VII. Conclusion

The indirect delegation in DNS embraces extra zone dependencies, which could have some negative effects on a zone's resolution. We have presented a graph-based model that captures different dependency relations between DNS zones and constructs various dependency graphs to describe how the dependencies are introduced. Our survey with over 1 million DNS zones presented a big picture view of zone dependencies in DNS.

Although the zone dependency problem has not been a major cause of DNS operational errors and incidents of integrity compromise, we believe its negative effects might get worse quickly if the dependencies in DNS increase and exceed some thresholds. DNS administrators should be more aware of this problem and reduce unnecessary dependencies of their managed zones. It is also of interest to deeply analyze its possible effects and continuously monitor its long-term trend. We believe this work provides some valuable information for further study of this problem.

Acknowledgements

This work was supported by National Key Research and Development Program (Grant No.2017YFB0803202) and National Natural Science Foundation of China (Grant No. 61472215). The Corresponding Author of this paper is Dr. Jia Zhang (Email: zhangjia2017@mail.tsinghua.edu.cn).

References

[1] V. Ramasubramanian and E. G. Sirer, "Perils of transitive trust in the domain name system," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, ser. IMC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 35–35.

[2] D. J. Bernstein, "Notes on the Domain Name System," accessed Jan, 2013. [Online]. Available: http://cr.yp.to/djbdns/notes.html

[3] [dns operations], "DNS trust dependencies for TLDs," accessed Jan, 2013. [Online]. Available: https://lists.dns-oarc.net/pipermail/dns-operations/2009-June/thread.html#3976

[4] V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis, and L. Zhang, "Impact of Configuration Errors on DNS Robustness," in *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4. ACM, 2004, pp. 319–330.

[5] C. Deccio, C.-C. Chen, P. Mohapatra, J. Sedayao, and K. Kant, "Quality of Name Resolution in the Domain Name System," in *Proceedings of ICNP 2009*, oct. 2009, pp. 113 –122.

[6] C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra, "Measuring Availability in the Domain Name System," in *Proceedings of INFOCOM 2010*, march 2010, pp. 1 –5.

[7] E. Osterweil, D. McPherson, and L. Zhang, "Operational implications of the DNS control plane," *IEEE Reliability Society Newsletter*, 2011.

[8] M. Dempsky, "DNSTrust TLD dependency graphs," accessed Jan, 2013. [Online]. Available: http://shinobi.dempsky.org/~matthew/dnstrust/graphs/

[9] [dns operations], "Perils of Transitive Trust, followup," accessed Jan, 2013. [Online]. Available: https://lists.dns-oarc.net/pipermail/dns-operations/2006-May/thread.html#537

[10] A. Phokeer, A. Aina, and D. Johnson, "Dns lame delegations: A case-study of public reverse dns records in the african region," in *Africomm, Eai International Conference on Einfrastructure and Eservices for Developing Countries*, 2016.

[11] P. Koch, "DNS Glue RR Survey and Terminology Clarification," 2007, accessed Jan, 2013. [Online]. Available: http://omniplex.om.funpic.de/home/test/draft-koch-dns-glue-clarifications-03.txt

[12] B. Hubert and R. Mook, "Measures for making DNS more resilient against forged answers," *RFC5452*, 2009.

[13] S. M. Bellovin, "Using the Domain Name System for System Break-ins," in *Proceedings of the 5th conference on USENIX UNIX Security Symposium - Volume 5*. Berkeley, CA, USA: USENIX Association, 1995, pp. 18–18.

[14] P. Pearce, B. Jones, F. Li, R. Ensafi, N. Feamster, N. Weaver, and V. Paxson, "Global Measurement of DNS Manipulation," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 307–323.