Surviving Attacks on DNS with Ubiquitous Replication

Daniel M. LaFlamme Brian N. Levine
Department of Computer Science
University of Massachusetts, Amherst 01003
{danl,brian}@ cs.umass.edu

Abstract

The Domain Name System is crucial to the operation of the Internet as almost every connection between computers begins with a query to the distributed DNS database, a system that maps textual names to numerical IP addresses used in routing packets. Critical to the operation of DNS are the root nameservers, which have been the target of Denial of Service attacks on several occasions—the most recent of these attacks resulted in most of the root servers being unreachable for several hours. During a more sophisticated, sustained attack, connections between computers on the entire Internet would be affected if textual address translation is needed. We propose a system to protect DNS most critical servers — the root name servers and nameservers authoritative for the Top Level Domains — by replicating their contents on machines around the Internet. The amount of data necessary to provide a replica of several TLDs could fit on in several gigabytes of storage. Updates to these replicas take place by USENET articles posted by the TLDs informing the replicas of changed DNS records. We argue that such a system is possible because storage capacity for a replica is relatively cheap. Moreover, DNS records for most domain's nameservers do not change very often. To support our claim that these records do not change very often, we collected DNS records for the authoritative name servers of a sample set of 2,266 domains. After a sixth-month period, 93% of the records did not change significantly.

1 Introduction

Any IP connection to a textual address begins with a query to the Domain Name System (DNS), which is the distributed database that maps textual hostnames to numerical IP addresses. To support these queries, DNS provides a hierarchal lookup system starting with thirteen *root-level* severs sites. Requests are then redirected to the *top level domain* (TLD) servers (e.g., `com`, `edu`, or `uk`). From there, queries reach the authoritative DNS server of each IP domain, of which there are millions.

This architecture used by DNS is prone to attack, and disregards the resourcefulness of computers connected to the Internet, which have far exceeded the now antiquated assumptions of DNS's designers. DNS has increasingly become the target of attack. The most serious and longest lasting attack to date occurred in October 2002. During one-hour period on Oct. 21, all 13 DNS root server sites were targets of a distributed denial-of-service attack [14]. Several of the root servers were unreachable from many parts of the Internet due to the attack's heavy congestion. The risk remains today: each of the most recent Internet worms — including MyDoom, so.big, and blaster — could have easily included code that performed DDOS attacks on the root-servers or top-level domains servers of DNS.

If previous disruptions to DNS have gone unnoticed, it is only because the local DNS machines that resolve queries for local users cache hostname-IP address mappings for short periods of time, typically a day or two. With a longer attack, an increasing number of Internet users would be unable to complete DNS queries and the Internet would be rendered unusable to them. A worm set loose on the Internet can survive weeks and months, and potentially pummel root-level or top-level DNS servers for just as long.

In this paper, we propose a ubiquitous caching strategy to remove DNS's status as the Achilles' heal of the Internet. Our proposal is easily and incrementally deployable — indeed, we have it operating already. It is supported by a long period of measurements we have taken to evaluate the efficacy and scalability of our approach. The main reason for having any centrality in DNS is that it greatly simplifies temporal consistency. Thus, we have devoted a large portion of this paper to examining the dynamicity of DNS record As one might guess, DNS records change very slowly. Accordingly, our proposal is based on robust and scalable push mechanism.

Specifically, in our measurements, presented in Section 4, we found that for 93% of records contained valid authorita-

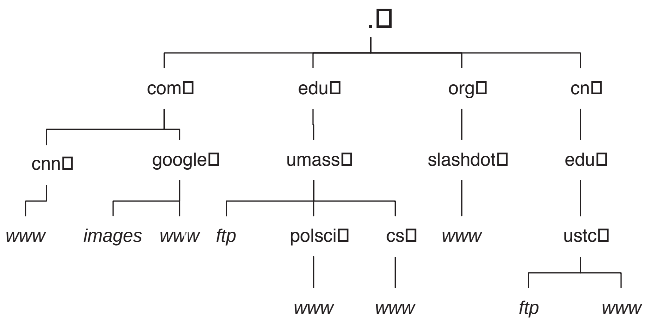

Figure 1: A Portion of the DNS tree illustrating its hierarchical structure.

tive entries even after six months without updates. Therefore, we propose that updates to authoritative records of root servers and top-level domains be pushed to clients. A push-based architecture is more robust as all lookups for authoritative servers do not require that the high-level servers be available — this would significantly reduce the possibility of DOS attacks on DNS. We calculate that ubiquitous caches can store the complete contents of a root server and ten top-level domains in less than 4 gigabytes (about $4 worth of hardware). When changes to these resource records occur, we propose that Usenet and the Network News Transfer Protocol (NNTP) can be used as a rapid, efficient, and already-deployed propagation mechanism. As a backup to NNTP, we also suggest and have deployed a peer-to-peer strategy based on existing open-source software. We have found that replication of the top-level servers has the additional benefit of decreasing lookup times and removes the two-day waiting period to propagate changes globally, an veritable eon in Internet timescales. Note our goal is not to replace the root servers, but to deploy a secondary mechanism.

The remainder of the paper is organized as follows. In Section 2, we provide an overview of DNS as it operates today. In Section 3 we outline our proposal to replicate the high-level DNS servers' databases and illustrate that the data currently stored at the root and TLD levels of the DNS hierarchy can be stored on commonly available, inexpensive storage media. We also estimate how the cost of the replicated system will scale over time as the Internet grows and storage prices drop. In Section 4 we show through real measurements that updates to domains occur infrequently enough to justify a push-based, replication approach. We also detail the feasibility of distributing updates for domains that do change to replica sites over Usenet using NNTP.

2 Background

This section provides an overview of the DNS protocol and the overall operation of the DNS database. Additional information explaining the operation of DNS can be found in the Appendix.

2.1 Structure

The DNS [11] is a distributed, hierarchal, and globally available database whose primary purpose is to map textual addresses to the numerical addresses used for routing in the Internet. The system is organized as a tree that establishes a consistent naming scheme for hosts on the Internet. Each internal node of the tree is called *domain. Leaf* nodes in the DNS tree are hosts.

A decentralized architecture was one of the driving design goals of the DNS. An example illustrating the hierarchical structure of DNS is shown in Figure 1. The domains that are immediate children of the root servers are called *Top Level Domains* (TLDs). Each domain in the DNS system must serviced by at least two *authoritative* name servers. These name servers store information about the domain, such as *resource records.* The mapping from a hostname to its IP address, or from a domain to its mail server, are both examples of resource records.

2.2 Root and TLD Servers

The root of the DNS system is serviced by name servers distributed at thirteen sites around the world. Each site is named with a letter of the alphabet from A to M. Most of the root server sites are located on east and west coasts of the United States, and there are two in Europe and one in Japan. The root servers serve as an entry point to the DNS system, meaning that any client that knows the IP address of at least one of the root server sites can resolve any textual address in the DNS system: first querying a root server and then issuing further queries to other name servers based on the response from the root.

Each of the thirteen root servers has a copy of a database that is agreed upon by Internet Assigned Numbers Authority (IANA) and the US Dept. of Commerce [10]. For each TLD in DNS, the root server database contains a list of nameserver names and IP addresses that are authoritative for that TLD. Since there are only 13 gTLDs and 244 ccTLDs, the root database is fairly small and does not change often. However, the root name servers are heavily used; the organizations operating the root servers report loads exceeding 100 million queries per day [15]. Interesting studies examining the performance of the root servers as well as the nature of the queries they receive can be found in several sources [4, 15, 5, 12].

Each TLD is operated by a set of authoritative servers that maintain a list of nameservers authoritative for each domain registered in the TLD. For example, currently, the com domain has approximately 22.6 million registered domains. Not surprisingly, the authoritative nameservers for the TLDs maintain a database much larger than the database stored at the root level.

2.3 DNS Robustness in Practice

In practice, several techniques are employed to defend the root against DDoS attacks at root servers [10, 14, 3]. Since there are not thirteen root server machines, but thirteen root server sites, IP Anycast is used to route request to a given DNS root server to one of the several hosts behind that root server's address. In addition to anycast, host and connection resources are over-provisioned to make the system more robust: during the Oct. 21 attack, the root servers were able to answer all queries that they received. The problem was that many queries were not received by the root servers at all due to the network congestion caused by the DDoS attack.

3 Ubiquitous Replicas

This section provides an overview of our scheme and provides some insight into what would be required to implement it in today's Internet. Our goal is a practical scheme that enables institutions, companies, and individuals to easily create a replica server.

3.1 Overview

To become a replica, DNS servers first obtain a recent snapshot of the high level DNS server contents for the TLDs they are most interested in (replicating all TLDs is not necessary). Downloading a complete snapshot with a high-speed connection to the Internet would take minutes or seconds: all of com is 622Mb compressed. Table 1 shows the storage size of complete replicas for various TLDs. Alternatively, operating systems could come with a copy of replica on a CD or DVD; as we show in Section 4, the rate of change of records is slow enough that differences could be downloaded.

When resolving DNS queries, the DNS replica server can query the root and TLD servers; upon a timeout, instead of returning an error, the results from the cache are used. Once the authoritative server for a domain is resolved, the users's DNS query is sent to that authoritative nameserver and resolved.

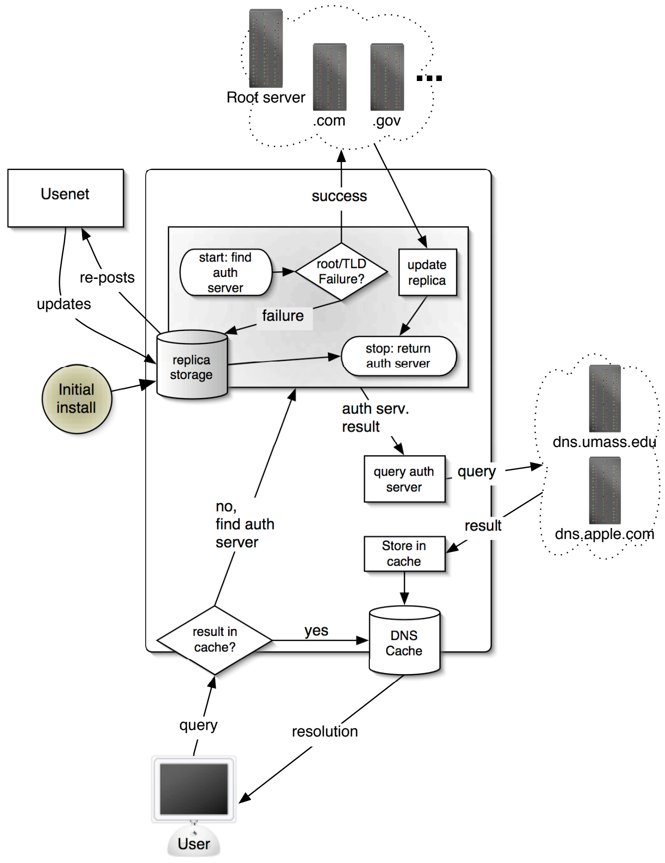

Figure 2: Overview of our proposed replica architecture.

To receive updates to stored data, each replica subscribes to the USENET newsgroup specific to the TLD, which we describe in detail below. Additionally, updates from successful TLD lookups (signed by the TLD) are posted to USENET for other replicas to download.

3.2 Pushing Updates to Replicas

We propose that updates to a replica can be propagated by the TLD registries to interested replicas over Usenet [7] and the Network News Transfer Protocol (NNTP) [9]. We hope to leverage the existing USENET infrastructure for the distribution of DNS updates to replicas.

In our scheme, TLD registries post updates to a newsgroup that is appropriately named. We have set up our own USENET server to carry and publish three groups: `alt.replica.dns.com` ,`alt.replica.dns.edu` ,and `alt.replica.dns.net`. We have written a perl script that allows replicas to periodically query a USENET server and download signed updates.

One of the advantages of a system like USENET is that each replica does not need to run it's own NNTP server to have access to the news groups. A relatively small number of USENET servers could potentially serve a large number of replicas.

3.2.1 Fast propagation over USENET

The propagation of articles on USENET is based on periodic updates between pairs of servers. As a server learns of new postings from one peer, it is pushed to other peers during subsequent periodic transfers. Accordingly, propagation from the TLD to all replicas can be relatively slow (perhaps on the order of hours or days).

Propagation of updates can take place faster if replicas post updates received from the TLDs during successful, normal DNS lookups. Specifically, when a domain updates its records with a TLD, the TLD creates a new replica record. The replica record is posted to USENET. When a DNS server requests the record from the TLD, included in the result is the replica record and USENET article number. The DNS server can then re-post the article to its local USENET server. Updates to popular DNS records are therefore propagated through USENET more quickly by being posted multiple times with the same article number.

3.2.2 Security

It is easy for a malicious user to spoof a message from a registry with false update information. A simple way around this problem would be for the update messages to be signed by the registries and verified at the replicas using public key cryptography. Since the registries are the only entities that should post update messages for their TLDs, each replica would need to keep the public key of each TLD registry in order to verify the signatures of the update messages. There are currently 13 gTLDs and 244 ccTLDs. 263 keys in not an unmanageable number of keys to expect a replica to keep; moreover, the number of keys will only increase as the number of TLDs increase. These keys could be published widely in much the same way as CA certificates are distributed in web browsers. If DNSSEC is ever deployed widely, those keys could be used.

3.3 Pulling Updates

It is very difficult, if not impossible, to bring down USENET. However, as an alternative mechanism to allow clients to pull updates and complete replicas, we've set a bittorrent tracker site at `http://prisms.cs.umass.edu/dns`. Bittorrent is a widely-used protocol designed for peer-to-peer sharing of files that are larger than complete DNS replicas; e.g., movies, games, and other software. Bittorrent is designed around the notion of parallel downloads: files are available from the *seed* site and any other host that has completed the download, as well as any host that is in the process of downloading the file. While it would be easy to attack the tracker site, we note that we intend the use of bittorrent only as an alternative; On the URL above we have posted software to allow replicas to automatically pull updates from the bittorrent site, and if they wish, then post to a different tracker.

3.4 Pushing Updates Improves DNS Performance

g There is no mechanism for pushing changes to name-servers across the Internet. This is because DNS entries are cached from the latest query, which accounts partially for the scalability of the DNS system. Most domains allow caching of their DNS records for a period of two days during which time updates are not pulled by clients.

Currently, when a domain changes the name or address of one of its name servers, it notifies the next highest entity in the DNS tree through the *registrar* where the domain was originally registered. These changes are then propagated to the TLD registries by the registrars. The new nameserver information for the changed domain will then be available after this transaction between the registrar and the registry take place. This time is usually on the order of hours. DNS changes will be be propagated more quickly under our scheme than simply waiting for a stale entry to expire.

We could not find any studies that explore article propagation delay on USENET. We ran a small experiment to test Usenet propagation. We posted ten articles and observed how long it took for them to appear at various planetlab computers around the country. For six sites out of 10 we checked, the posts appeared in less than one minute: Berkeley, UPenn, MIT, UCLA, UT Austin. At UMaryland, eight of the posts were received in less than 10 minutes. At UCalgary, none of the posts appeared after ten minutes. Some sites that run news servers post statistics comparing the timestamp on the article with the time the article is received at their news server. One such site suggests that 90% of articles received by their news server arrive 7 minutes after they were posted [1].

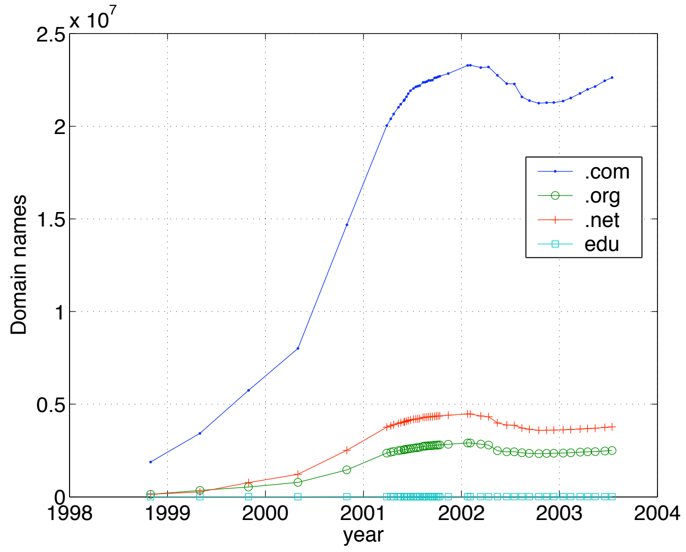

Figure 3: The number of registered domains for four TLDs from July 1998 to January 2004 [2].

TLD	Domains	Exact Replica size	
		uncompressed	compressed
`.com`	26,425,704	1.9 GB	358 MB
`.net`	4,381,339	316 MB	60 MB
`.edu`	7,298	609 KB	163 KB

Table 1: Number of domain name registrations for various TLDs in January 2004.

3.5 Estimating Replica Growth

As the number of domain name entries in each TLD increases, so will the size of the replica. Therefore, it is important to estimate how the storage required to replicate a TLD database will grow over time.

Table 1 shows the number of entries currently in the `com`, `net`, and `edu` domains according to the latest files available for download from Verisign, the maintainer of these TLDs. Figure 3 shows the growth of those TLDs from July 1998 to January 2004 [2]. The historical trend for these TLDs has been rapid growth from July 1998 until the middle of 2001; much of 2002 was marked by a decline, though from November 2002 to the latest measurement have shown a flat growth for all but the `com` TLD, which is linear and moderate. If the current trends hold, then it will take ten years for the number of `com` domains to double.

Fortunately, storage space has historically decreased in cost at an exponential rate according to Thompson and Best [13]. Figure 3 shows a graph of prices per megabyte as hard storage products have been introduced by IBM. From the slope of the line in the figure, we were able to estimate how fast the total number of domain entries can grow in `com`, `net`, and `edu` TLDs in order to not increase the cost of storing a replica beyond \$4. Figure 4 shows this estimation. Any growth in domain name entries lower than the lower parallel line will lower the cost of replication. Any growth in domain names lower than the top parallel line will put replication cost below \$400. The diamond-marked line is a projection of the number of domain names if the Internet were suddenly to experience growth at a rate equal to the power-law growth of the late 1990s. If a 10-year period of such growth occurred until 2014 (and hard drive costs continued dropping as they have for the last twenty years), the cost of a replica for an organization would still not cost more than \$400. Over that 10-year period, the number of domains could increase from 10^7 to 10^{12} entries.

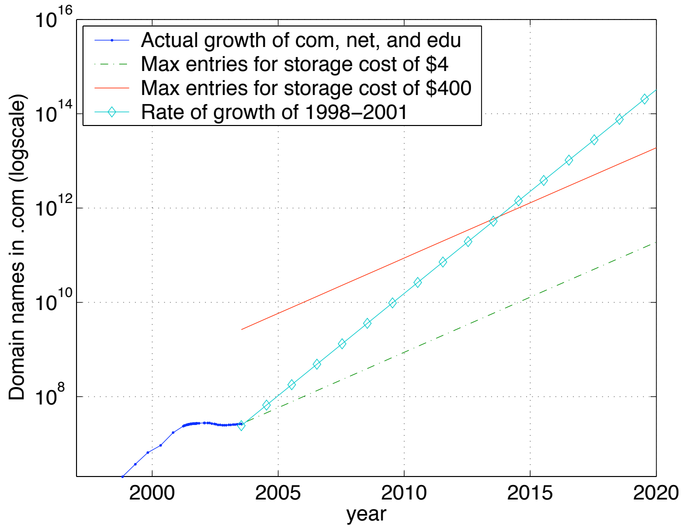

Figure 4: (Logscale) The parallel lines represent the number of entries that can fit on a drive as costs drop according to Thompson and Best [13]. The diamond-marked line represents a rate of growth equal to the late 1990s.

4 Measurements of DNS Record Stability

In the previous section we addressed several issues related to our proposal: its push and pull architecture; security; and replica cost over time. In this section, we present a close examination of how often changes to TLD databases occur as this determines how much data will be pushed to clients.

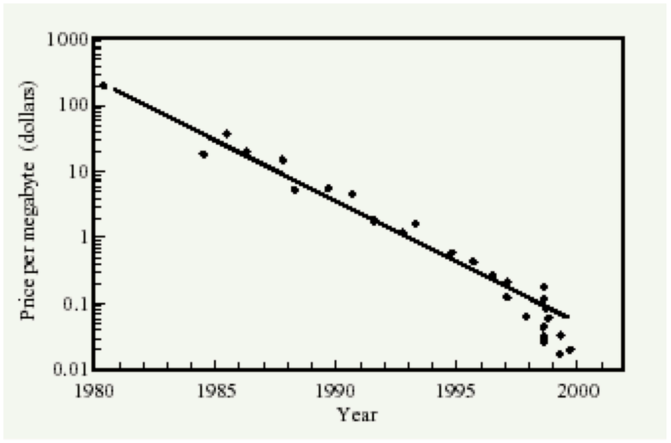

Figure 5: Taken from Thompson and Best [13]: "The price history of hard disk products versus year of product introduction." (Republished with permission)

Initially, we did not have access to the complete `com`, `net`, and `edu` databases from Verisign. We have had access to daily updates of these databases since late December 2003. Therefore, we show both our analysis of one-month of complete records, and our measurements of a sampling of 2,266 domains over a six-month period.

The 2,266 domains we monitored were composed of every domain that users of the UMass Department of Computer Science network queried during a two-week period. The breakdown by TLD for our 2,266 member list is shown in Table 2. We wrote a script to perform the following operations for each host in the list: (1) pick a random DNS root server and query it for the current host's TLD. The root server would return a list of nameservers authoritative for that host's TLD. (2) We then pick a random nameserver from this list of servers and query it for the host. This returned a list of authoritative nameservers for that host's domain, which we then logged.

We ran the aforementioned script every 4 hours on our hostname list. We collected data from May 27, 2003 to November 24, 2003. There was a four-day break in measurements starting on July 15. This translates to 540 data points over 2,160 hours.

4.1 Messages and Bytes Posted Per Day

Figure 6 answers the most pertinent question: how many messages and how many bytes are transmitted per day with ubiquitous caching of DNS high-level servers? From the short duration of our measurements, it seems that less than one message under 500kb would be posted per day for each TLD. The rest of this section evaluates a sampling of 2,266 domain records that we measure over a six-month period.

TLD	count
com	1,175
edu	385
org	201
net	117
uk	57
de	43
tr	39
ca, gov	29
kr	24
jp	20
cn	18
au	16
nl	12
ch, us	7
tw, se	6
fr, fi, it	5
no, dk, co, be, to	4
cc, sg, at, pl, nz	3
il, pt, es, ru, br, gr, tv, info, sk	2
cx, lt, mil, int, do, sm, pe	1

Table 2: Composition of the 2,266-member hostname list by TLD.

4.2 Our Classification of Record Changes

For our long-term collection of 2,266 sampled hosts, we analyzed our data by looking for changes at each time-step; that is, we looked for changes that occurred between two consecutive data collection periods (in our case, every 4 hours). If we could not directly compare the data at two consecutive timesteps because a TLD name server did not return nameservers authoritative for the domain we queried, we look for changes between the the current timestep and the next later comparable timestep.

Figure 7 shows two example queries for the authoritative nameservers for www.convera.com: the first one at 7:00pm on Wed May 28, and the second one 4 hours later at 11:00 pm. This example illustrates what is meant by a change between timesteps. The IP address for the nameserver ns.excalib.com changed in the four hours between the two queries from 208.152.92.252 to 67.129.159.161. We classify this change as an ipAddress change as an IP address for one of the authoritative nameservers for convera.com changed. We elaborate on the kinds of changes that can occur below.

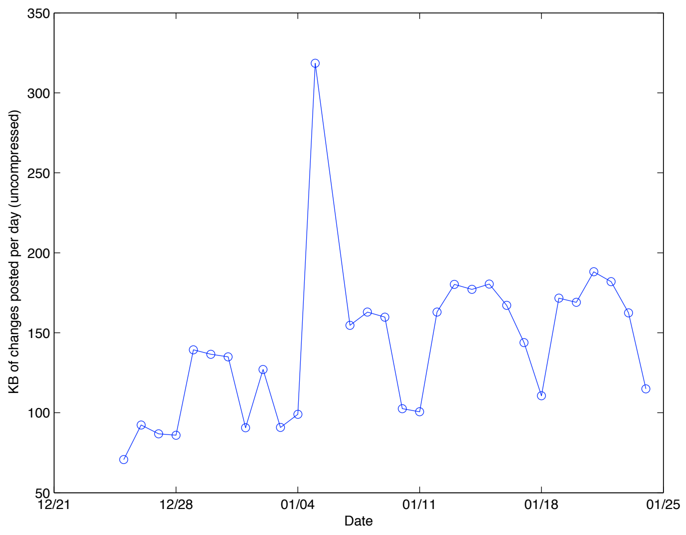

Figure 6: The size in kilobytes of the list of changes to the com, edu, and net TLDs over a one month period (uncompressed); from our measurements of Verisign data. Posts to Usenet can be up to 1 megabyte.

```
name: www.convera.com
nameserver: 192.52.178.30
time: 1054162801.94 (Wed May 2819:00:01 2003)
    Authority:
         convera.com                 svl-ans-01.inet.qwest.net
         convera.com                 dca-ans-01.inet.qwest.net
         convera.com                 ns.excalib.com
    Additional:
         svl-ans-01.inet.qwest.net   205.171.14.195
         dca-ans-01.inet.qwest.net   205.171.9.242
         ns.excalib.com              208.152.92.252

--------------------------------------------------

name: www.convera.com
nameserver: 192.42.93.30
time: 1054177201.43 (Wed May 28 23:00:01 2003)
    Authority:
         convera.com                 svl-ans-01.inet.qwest.net
         convera.com                 dca-ans-01.inet.qwest.net
         convera.com                 ns.excalib.com
    Additional:
         svl-ans-01.inet.qwest.net   205.171.14.195
         dca-ans-01.inet.qwest.net   205.171.9.242
         ns.excalib.com              67.129.159.161
```

Figure 7: Two DNS records as reported by `dig`.

We collected statistics on the number and types of changes that occur. Naturally, there are several ways in which we can count changes. We have defined the following taxonomy of six change types to classify each change we see between timesteps. Each is discussed in terms of the *current* DNS response and the *previous* DNS response, which is usually the DNS response 4 hours previous to the current response.

1. **ipChange:** at least one authoritative name server for domain had an IP address that changed between the previous and current. The textual address of the nameserver must have existed in the previous response.

2. **allAuthDifferent:** all of the authoritative name servers for the domain have changed from previous to current.

3. **addAuthority:** a new name server was added to the list of authoritative name servers for the domain from previous to current; all previous name servers are listed in current.

4. **delAuthority:** a name server was removed from the list of authoritative name servers for D between previous and current.

5. **addAdditional:** a new name server IP address entry was added to the additional section of the response for D between previous and current; all previous name servers are listed in current. This type of difference is of almost no significance as long as there are other entries besides the new one in the "additional section" of the response. Nameservers are not required to return anything in the "additional section" but often do to save the querying client the additional DNS query for the IP address of an authoritative nameserver. As long as there is at least one other entry in the "additional section", the querying client will not have to perform the second query anyway.

6. **delAdditional:** a new name server-IP address entry was removed from the additional section of the response for D between previous and current. This type of difference is of almost no significance as long as there are other entries besides the deleted one in the "additional section" of the response for the same reason given for **addAdditional** changes.

4.3 Resource Record Stability over Six Months

Recall that we run a DNS query asking for the authoritative nameservers for each host on the list and store the result. We later use this collected data to determine how and how often authoritative nameservers for hosts change. For the

2,266-member list of hostnames, we gathered statistics by counting changes in three different ways. We did this to show that even though the set of authoritative nameservers for a domain can change, that change doesn't necessarily mean the domain would become unreachable to a querying client because of that change. In fact, that data will show that the the majority of the changes that do occur are not critical and thus will not prevent an given client from contacting one of the authoritative nameservers for a domain given that it has cached data for that domain.

We present the results for each list using three *change count methods*. These counting types are listed below from least restrictive to most restrictive:

1. any: This is most conservative test for change. Given any two responses for a domain, a change is counted for d if *any* change type occurs between those two timesteps. For example, a record could have changed only by the addition of a name server (`addAuthority`).

2. auth_or_ip : Given any two responses for some domain, a change is counted if either an `ipChange` or an `allAuthDifferent` change occurs between those two timesteps.

3. auth_only: This is the least conservative test for count changes. Given any two responses for a domain, a change is counted only if an `allAuthDifferent` change occurs between those two timesteps. In other words, with this type of change, a previous record is completely invalid.

In other words, method "any" provides an lower bound on the number of changes that could be counted. The "auth_only" method provides an upper bound on the number of changes that could be considered.

We present three types of analysis that reveal the characteristics of record changes that we observed.

4.3.1 Record lifetime

Figure 8 shows the the percentage of the total number of domains that have not changed from the start of collection over time. The figure shows our observation that 29% of domains had at least one change in their records during the six-month experiment; i.e., 71% of domains had no changes at all (using counting method any).

The steep initial drop for change count method anysuggest that at there is a group of domains that change often since they are accounted for so early in the graph. Indeed, the two change types that occur with by far the most frequency under method any are `addAdditional` (51.6%)

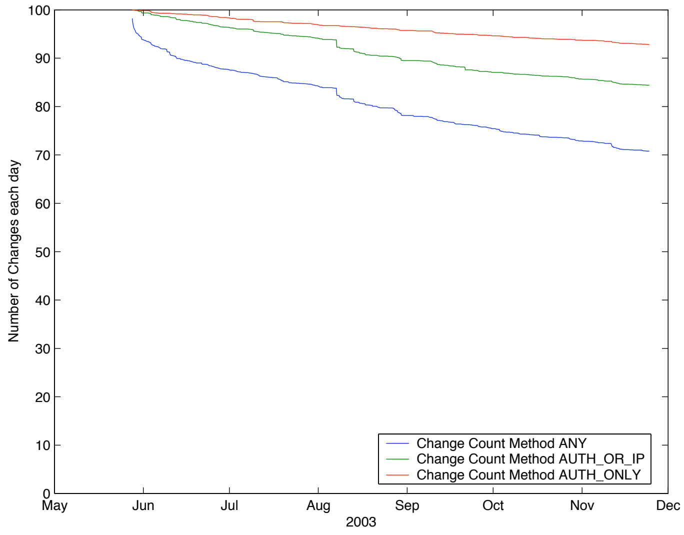

Figure 8: Percentage of total (2266) domains remaining unchanged at each timestep for all three change count methods.

and `delAdditional` (45.0%). These changes are not significant. In all cases when the change between two DNS responses was `addAdditional` or `delAdditional` there are other entries in the additional section that the two responses have in common so it is likely a querying client would simply use one of those if it were to actually contact an authoritative name server for that domain.

The slopes of the lines representing change count method auth_or_ip and auth_only are much less steep than that of change count method any. The types of changes that are counted using these two methods are more critical and thankfully, much more rare than those counted under any. Changes that are counted auth_only are the only ones that would prevent reachability of the authoritative domain servers for the domain that changed. As can be seen, the vast majority of the domains in the list do not change in that way.

4.3.2 Burstiness of Changes

The burstiness of changes that occur time are also of interest to us in calculating how much work our system incurs. Figure 9 plots the number of changes occurred between collection periods over time. (A four-day break in measurements occurred in July.) This graph allows us to see if changes are distributed evenly over time or if they exhibit spikes at certain times over period of time we collected data. Figure 10 shows the CDF of the number of changes per four-hour measurement period. The plots show that changes are not bursty and workloads would be low. For auth_or_ip and auth_only, the amount of traffic per four hour period for USENET would be very low. Method any shows some larger amount of traffic; this suggests each TLD could have two newsgroups: one with critical changes (corresponding to auth_or_ip or auth_only `allAuthDifferent` type changes); and a second with more frequent changes perhaps delayed and bulk processed in one message.

There are three large spikes visible in Figure 9. These spikes occur at August 7 at 11am EST, August 12 at 11 am EST, and on August 16 at 11am EST. After examining the data to account for these spikes, we found that all of the sites responsible for these changes were related to Microsoft. As our list of hostnames is composed of hosts that users have queried for, the list has hosts with domain names of online services offered by Microsoft. These include `mapblast.com`, `research.microsoft.com`, `passport.com`, `hotmail.com`, `expedia.com`, `msn.com`, and `microsoft.com`. On August 7, Microsoft changed the IP address of its authoritative nameserver, dns1.dc.msft.net. On August 12, it changed dns1.dc.msft.net's IP back to what it was on August 7. On August 16, Microsoft changed the IP of another one of its authoritative name servers. Such an event illustrates how the number of changes, and therefore the number of update messages required, can increase drastically when a record of an nameserver that is authoritative for many domains is changed.

5 Related Research

The only other work we are aware of that specifically addresses availability of the DNS system is by Kangasharju and Ross [8]. They have proposed a replica-based mechanism, however, their proposal has significant differences from ours. Each replica in their system stores a copy of the entire DNS database; in other words, every host registered in DNS. The Kangasharju and Ross proposal relies on the availability of either IP multicast or broadcast satellites. Under the first approach, each replica would belong to an IP multicast group. When a replica receives updates from a child nameserver it is servicing, it sends this update to all other replicas via the group's shared IP multicast address. With the second approach, each replica site is equipped with satellite transmitter and receiver equipment. Any time one replica receives an update from a child name server, it transmits this update over the satellite channel. In this way, the updates sent to other replicas are delivered out-of-band and each replica receives the update instantaneously.

Our intention is to lessen the importance of the high level DNS servers and therefore mitigate the impact of successful attacks. In comparison, replicating records for every root server, TLD server, and host on the Internet has a number of drawbacks.

First, the Kangasharju and Ross method requires much more space at each replica.

Second, it raises more severe consistency issues — e.g., the IP address for www.cnn.com changes much more often than the authoritative name server for the cnn.com domain.

Third, replicating information for every host in DNS may not be desirable from a privacy standpoint. Many organizations may not want the hostname-IP mapping for each of their hosts replicated across the Internet.

Fourth, in our approach, the authoritative server for a domain remains the gateway for accessing resource records of hosts in that domain; e.g., the IP address of a web server is often rotated by administrators to thwart DOS attacks; Akamai.com resolves host IP addresses depending on the location of the querier. In other words, we leave it to each domain to determine the resources required to protect their authoritative servers.

Fifth, their update mechanism is tenuous. IP multicast is still not widely deployed and faces many commercial and security challenges [6]. The satellite approach is attractive for its out-of-band and instantaneous update availability features; however, requiring satellite transmitter equipment places the ability to own a replica out of the hands of most users. Even with a scheme in which each replica is not required to have transmission equipment, having only receiver equipment could still cause problems.

6 Conclusion

We have proposed a mechanism by which the effects of Distributed Denial of Service attacks on important nameservers in the DNS, those authoritative for the root and TLDs can be mitigated by creating replicas of their databases at sites across the Internet. Using the already existing infrastructure that the Usenet system provides, these replicas can be sent updates as the resource records at these high-level DNS servers change. Our measurements of record size, rate of change and burstiness all show that our method is incrementally deployable, scalable over time, effective, and manageable by every users.

References

[1] Article propagation delay graph, August 2003. http://news.panservice.it/inndelay.

[2] History of gtld domain name growth, August 2003. http://www.zooknic.com/Domains/counts.html.

[3] J. Abley. Hierarchical anycast for global service distribution. Internet Software Consortium technical note, 2003.

[4] N. Brownlee, K. Claffy, and E. Nemeth. Dns measurements at a root server. In *Globecom*, 2001.

[5] N. Brownlee, K.C. Claffy, and E. Nemeth. Dns root/gtld performance measurements. In *Globecom*, 2001.

[6] C. Diot, B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *IEEE Network*, 14(1):78–88, / 2000.

[7] M. Hornton and R. Adams. Standard for interchange of usenet message. Technical report, IETF, 1987. http://www.ietf.org/rfc/rfc1036.txt.

[8] J. Kangasharju and K. W. Ross. A replicated architecture for the domain name system. In *IEEE Infocom*, pages 660–669, 2000.

[9] Brian Kantor and Phil Lapsley. Network news transfer protocol. Technical report, IETF, 1986. http://www.ietf.org/rfc/rfc0977.txt.

[10] L.-J. Liman, J. Crain, S. Woolf, B. Manning, A. Pawlik, and S. Hollenbeck. Operation of the root name servers, 2003. http://www.root-servers.org/presentations/rootops-gac-rio.pdf.

[11] P. Mockapetris. Domain names–concepts and facilities. Technical report, IETF, 1983. http://www.ietf.org/rfc/rfc0882.txt.

[12] Y. Sekiya, K. Cho, A. Kato, R. Somegawa, T. Jinmei, and J. Murai. Root and cctld dns server observation from worldwide locations. In *Passive and Active Measurement Workshop (PAM)*, 2003.

[13] D.A. Thompson and J.S. Best. The future of magnetic data storage technology. *IBM Journal of Research and Development*, 44(3), 2000.

[14] P. Vixie, G. Sneeringer, and M. Schleifer. Events of 21-oct-2002, 2002. http://f.root-servers.org/october21.txt.

[15] D. Wessels and M. Fomenkov. Wow, that's a lot of packets. In *Passive and Active Measurement Workshop (PAM)*, 2003.

A Appendix: Additional information about DNS

Prior to its creation, hostname to address mappings were stored in a central file called `HOSTS.txt` located at the Information Sciences Institute. As the number of hosts on the Internet grew, maintenance and distribution of this master list became unwieldy.

The two major groups of players are the TLD registries and the the entities that create and maintain replicas. The registries are organizations that are granted authority through ICANN to maintain the database of DNS information for a TLD. A list of registries for the thirteen gTLDs can be found at http://www.icann.org/tlds. The registries interact with accredited registrars, who in turn sell individual domain names to individuals and organizations. It is the content of these registries we wish to replicated. Both the registries and the creators/maintainers of the replicas around the Internet play an important role in the viability of the plan.

Each TLD can be classified as either a *generic TLD* (gTLD) or a *country code TLD* (ccTLD). `com`, `edu`, and `org` are all gTLDs. Each country in the world is assigned a two letter country code to use for hosts within that country. In the diagram, `cn` represents the ccTLD for the People's Republic of China. Other examples include `us, uk, jp,` and `th` (not shown). In general, the main difference between gTLDs and ccTLDs are the restrictions on who can register domains as children nodes. Besides that administrative difference, gTLDs and ccTLDs occupy the same level of the DNS namespace and can be considered functionally equivalent.

Each host on the Internet is named by appending the domains it is a part of onto its name, with each domain being separated by a dot: ".". For example, the fully qualified name for the Political Science department's web server at UMass is `www.polsci.umass.edu`. This name was derived starting at the host name, `www`, and following the parent links all the way back to the root. The trailing dot separating the TLD from the null root label is usually omitted when writing or typing a hostname.

Name servers can also delegate responsibility for part of its name space to its children. This is often useful in large organizations that have subdivisions. In the figure, the UMass name servers which are authoritative for the UMass zone delegate responsibility for parts of that name space to two subdomains, polsci and cs. Under this arrangement, the polsci nameservers would contain resource records for hosts in the polsci.umass.edu domain. Responsibility for hosts in the cs.umass.edu domain are delegated by the umass nameservers to the cs name servers in the same way.

A.1 A Typical DNS Query

In addition to the structure of the distributed database, the DNS system specifies a protocol by which computers can issue queries to name servers in order to resolve an IP address, obtain the name and IP of a domain's mail server, etc. A typical DNS query for an IP address of a server is shown below. In this diagram, we are assuming that there are no records cached at any of the nameservers. We also assume that iterative DNS queries are used at every step.

In this example, a client C1 in the cs.umass.edu domain wants the IP address of www.cnn.com. C1 is configured to have "ns.cs.umass.edu" as its default name server so in (1), C1 sends the query for www.cnn.com to cs.umass.edu's nameserver. cs.umass.edu's namesever looks in its cache, but finds nothing. It then chooses one of the DNS root servers and sends the query (2) to the A root server. A root server responds (3) with a list of name servers that are authoritative for the .com top level domain (TLD). cs.umass.edu's nameserver then chooses one of the authoritative .com servers and sends the query (4) for www.cnn.com to the server it chose. The authoritative .com server examines the query and responds (5) with a list of name servers that are authoritative for the cnn.com domain. ns.cs.umass.edu then chooses one of the name servers that are authoritative for cnn.com and sends the query for www.cnn.com to the server it chose (6). The authoritative server for cnn.com then responds (7) with the A record for www.cnn.com which includes the IP address of www.cnn.com. The cs.umass.edu nameserver can then cache the response and return the result to C1 (8). C1 can then contact www.cnn.com directly using the IP address it resolved.

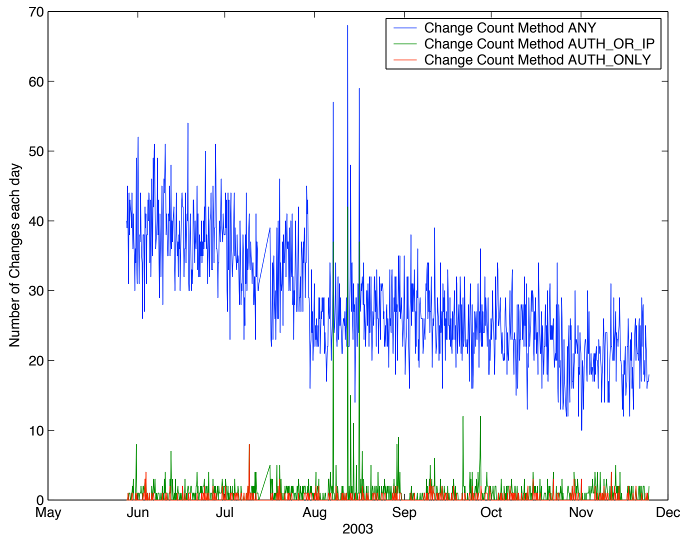

Figure 9: Number of changes occurring at each 4-hour interval.

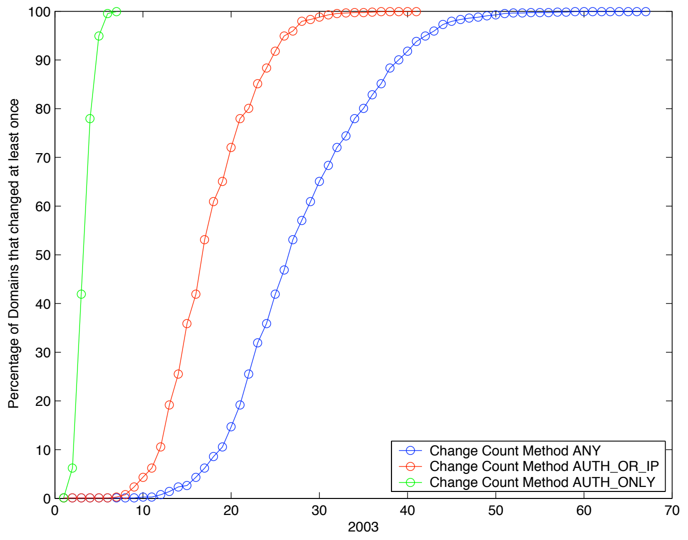

Figure 10: The CDF of changes in Figure 9.

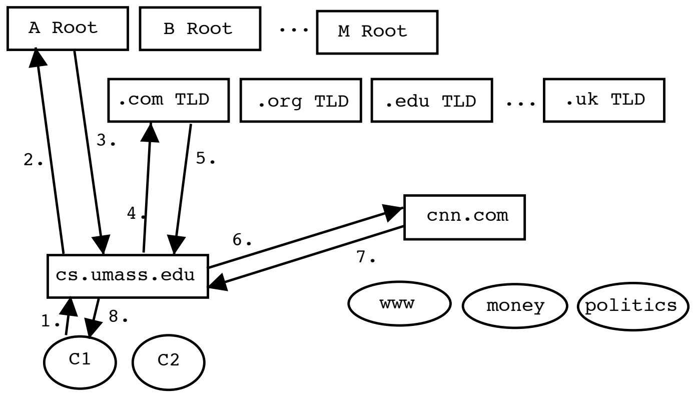

Figure 11: A typical DNS query.