Public Review for

The Main Name System: An Exercise in Centralized Computing

Tim Deegan, Jon Crowcroft, Andrew Warfield

This year's SIGCOMM award went to Paul V. Mockapetris, in large part for his work on the Domain Name System (DNS). Dr. Mockapetris deserves considerable credit, because DNS has been an incredibly successful component of the Internet; it hasn't required major changes despite scaling up many orders of magnitude from its original size. DNS has been recognized as achieving a remarkable balance between scalability (due to its distributed implementation) and decentralized administrative control, in which separate organizations control separate portions of the namespace.

So why consider changing it? More to the point, why should CCR publish a paper proposing a radical redesign of the DNS? This paper proposes a "recentralized" replacement for DNS in which, conceptually, the DNS is served from a single database. While the database could be replicated for fault tolerance, in a fundamental sense the distributed nature of the DNS would be abandoned. Heresy! Surely distributed solutions are always better than centralized solutions -- correct?

The authors challenge this assumption and, in the process, provide a very useful analysis of what benefits and costs accrue from the distributed nature of DNS. The paper shows that many features that would be useful in the DNS are more easily and simply provided in a centralized design; and that, despite our (perhaps aesthetic) perference for distributed solutions, many or or most of the benefits of distribution in the DNS are in fact achivable in a centralized design as well (an observation which may be more true today than at the time the DNS was designed).

CCR seeks to publish papers like this that challenge accepted wisdom in insightful ways. This paper does that by addressing an important and always timely topic; hopefully it will stimulate more discussion clarifying the strengths and weaknesses of distributed designs like that of the DNS.

Public review written by

Mark Crovella
Boston University

a c m s i g c o m m

The Main Name System: An Exercise in Centralized Computing

Tim Deegan
University of Cambridge
Computer Laboratory
15 JJ Thomson Avenue,
Cambridge CB3 0FD, U.K.

Jon Crowcroft
University of Cambridge
Computer Laboratory
15 JJ Thomson Avenue,
Cambridge CB3 0FD, U.K.

Andrew Warfield
University of Cambridge
Computer Laboratory
15 JJ Thomson Avenue,
Cambridge CB3 0FD, U.K.

ABSTRACT

Naming is a critical component of the internet architecture, and one whose complexity is often overlooked. As a global system, the DNS must satisfy millions of requests per second, while allowing distributed, delegated administration and maintenance. In this paper, we consider the design of the DNS and the widely distributed manner in which DNS records are published. We propose that the robustness and performance of the existing DNS could be dramatically improved by moving towards a *centralized* architecture while maintaining the existing client interface and delegated administration.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Design, Reliability, Performance

Keywords

DNS

1. WHY IS THE DNS DISTRIBUTED?

The domain name system (DNS) was designed as a replacement for the HOSTS.TXT file — a centrally administered list of all known machines. This database was, in a sense, distributed: every host would download a copy of the master file from an FTP server and use that copy for lookups locally.

As the internet grew, and organizations moved from time-sharing computers to networks of workstations, it was felt that both the administrative effort required to keep the HOSTS.TXT file up to date and the load on the FTP server would become too great; the hierarchical design of the DNS was intended to solve both of these problems [1]. The namespace was partitioned into administrative regions, and each organization was made responsible for providing redundant servers to publish its own section of the namespace.

The DNS was deliberately built as a simple name resolution system. Features available in directory services like X.500 [2] (e.g., searching, dynamic updates and authentication) were not included, in the hope that a simpler service would be easier to implement and therefore be widely and quickly adopted [3]. Some of these services have later been added [4–6].

As the DNS scaled to more and more domains, the tie between administrative delegation points and the distribution of the database has remained. The result is that far more nameservers are in existence than are needed for the task of publishing the database, and the benefit of having many servers is reduced by each server's publishing only the subset of the namespace that its owners and operators have administrative authority over. There are approximately 233 million hosts on the internet [7] and approximately 1.3 million authoritative nameservers listed in the .COM, .NET and .ORG zone files. However, delegation records necessary for access to any name in the DNS are published by fewer than one hundred of them[1] and a client must talk to at least two servers to look up a new name (e.g., for the first page on a new website).

These problems could be solved by decoupling the distribution of DNS data from the hierarchy of authority [8–10]. So long as the delegation of authority to publish records is not altered, the mechanism used to publish them could be replaced by any system with suitable characteristics. Specifically, the publication mechanism does not need to be distributed as a matter of principle — robustness, reachability and capacity are the main requirements of the DNS, and distribution is only necessary in so far as it helps us achieve these goals.

In this paper we propose the idea of re-centralizing the publication mechanism of the DNS: replacing more than a million servers with a single centralized database, served by a small number of well-provisioned and well-placed servers, and all but eliminating the link between domain ownership and domain publication.

We will first discuss some of the difficulties that motivate such a change, and some proposed alternatives, before considering what a centralized DNS would look like and how much work it would need to do.

[1]There are only 13 "root" servers listed in the DNS, but several of those servers are further distributed among different sites. There are root servers at 62 locations at the time of writing, and more are planned.

2. WHAT'S WRONG WITH IT?

Common complaints about the DNS include:

2.1 Lookup latency

Resolving names in the DNS can take long enough to cause noticeable delays in interactive applications [11, 12]. Reasons suggested for this include the number of servers that a resolver needs to talk to, the long timeouts in resolvers and stubs when errors are encountered [13], and underprovisioned resolvers and access links. Generally, the top-level domain (TLD) servers can be expected to be well placed and geographically diverse; leaf nodes are often not so lucky. Having fewer (and better-positioned) servers involved in the resolution process would be an improvement. Caching in resolvers helps with this, but due to the heavy-tailed lookup distribution, it can only go so far.

2.2 Update latency

The latency of updates in the DNS is governed by the time-to-live (TTL) field of the record being updated. For planned updates, the TTL can be reduced in advance, allowing a speedy propagation of the update in exchange for briefly increased traffic. For unplanned updates (e.g., in response to an outage or attack) the old record may remain in caches until its TTL expires.

This timeout-based caching mechanism is not a good fit with the patterns of change in the data served. Many DNS records are stable over months, but have TTLs of a few hours. At this timescale the TTL is effectively an indicator of how quickly updates should propagate through the network, but it is used by caches as if it were an indicator of how soon the record is likely to change. This generates unnecessary queries, and indicates that a "push" mechanism for updates would be more appropriate for these parts of the DNS. Jung et al. [14] show that, based on TCP connections seen in traces, 80% of the effectiveness of TTL-based caching is gained with TTLs of only 15 minutes, even though the data may be static for much longer than that.

2.3 Administrative complexity

Administering a DNS zone is a complex task, with the potential to cause serious performance problems. Even a properly-formed zone can cause problems for its owners, and for other DNS users, if the delegation from the parent zone is not properly handled. The most common delegation errors are caused by a lack of communication between the zone's administrators, their ISPs, and the administrators of the parent zone. 15% of forward zone delegations from TLDs are "lame" in at least one server [15] (i.e., a server which does not serve the zone is announced as doing so by the parent zone). Servers publishing out-of-date zone data after a zone has been redelegated away from them also cause problems. Circular glue dependencies, which cause delays in resolving the zone from a cold cache and reduce the number of useful servers available [16], affect about 5% of zones [15]. These errors are directly connected to the way the DNS is distributed along administrative boundaries.

In addition to the risks involved in preparing the contents of a zone, the software used on nameservers is complicated and requires some expertise to properly configure, secure and maintain. This expertise is not in evidence at all nameservers [17].

Misconfigurations at clients and resolvers can cause problems too: they are responsible for a large fraction of the load on the root servers [18, 19]. This has been a problem with the DNS for some time [3] and, despite efforts to educate administrators, the situation is not improving.

2.4 Vulnerability to denial of service

Redundancy is built by replication of data, but this happens multiple times per name — an organization must rely not only on its own nameservers being operational, but also on the nameservers for every level above it in the hierarchy. Again, this is caused by the distribution model.

There have been distributed denial-of-service attacks on the root and TLD servers in the past. Some of them have been successful in causing delays and losses [20], although anecdotal evidence suggests that to date, human error has been much more effective than malice at causing outages at the upper levels of the DNS, and care is taken in deploying root servers that they can survive large spikes in load [21]. Lower-level zones, which typically do not have the same levels of funding and expertise available to them, are more vulnerable to attack. Many zones are served by two nameservers that are on the same LAN, or even the same machine [15], although the standard requires each zone to have redundant servers.

2.5 Lack of authentication

The current DNS relies almost entirely on IP addresses to authenticate responses: any attacker capable of intercepting traffic between a client and a server can inject false information. Mechanisms have been developed to use shared-key and public-key cryptography to provide stronger authentication of replies [4, 5], although they are not yet widely deployed for client-server queries.

Any proposed replacement or upgrade of the DNS should address as many of these problems as possible, while avoiding the introduction of new ones. Next, we consider some possible improvements in the light of these criteria.

3. CURES, TONICS AND ELIXIRS

We will now look at some possible changes to the architecture of the DNS. They mostly address the first four problems from the previous section; authentication of records is being addressed by the DNSSEC community and is largely unaffected by these proposals.

3.1 Replicating the entire DNS

Kangasharju and Ross [8] advocate replacing per-zone "secondary" authoritative nameservers with a set of replica servers, each holding a full copy of the DNS. Each server would be responsible for managing updates to a subset of the zones in the DNS: it would receive updates for its zones from their "primary" servers and propagate them to the other replicas via multicast or satellite channels. Lookup latency is improved: clients only need to talk to their local replica. Update latency is impaired because updates must wait their turn on the replication channel (although unplanned updates may be faster because the system allows shorter TTLs). Replicating the full DNS at every server makes it easier to find, if not to avoid, administrative errors. They give no particular attention to denial-of-service attacks, and retain the current IP-based authentication mechanism.

3.2 A single central service

The entire DNS could be brought into a single database, eliminating the current wide distribution. This is in spirit a return to the central HOSTS.TXT file, but without the administrative overhead; all naming data would be gathered together and redistributed. Authority to change naming data would remain with the zone administrators. Read latency is improved because all authoritative records now come from well-placed, well-provisioned servers, and client queries are always answered immediately without deferring them to other servers: this reduces the number of network links and servers where failure or congestion would cause long delays to a lookup. Update latency is slightly increased, as with Kangasharju and Ross's scheme, because updates must be handled by the central service. Administrative complexity is reduced, because the delegation-related pitfalls can be entirely removed, and each zone's administrator now only configures client software for submitting updates, not server software. The database is less vulnerable to denial-of-service attacks (providing it can be provisioned properly) because there are fewer points of failure to defend: many of the DNS's vulnerable points are removed. The nameservice would have to be eliminated entirely by an attacker who wished to deny service to a particular record or zone. The authentication of records is the same as the current IP-based scheme. This is the solution we propose in this paper.

3.3 A distributed hash table (DHT)

The DNS could be served from a distributed hash table [9, 10]: servers would be organized in a peer-to-peer network, routing client requests to a server which holds the relevant data. The load is shared between many relatively lightweight servers, and new servers can be added as needed. Robustness is provided by replication and overlay routing. Administrative complexity is reduced for the domain owners, who only have to inject their data into the DHT, and not run their own servers. Using a DHT gives a marked increase in read latency, as requests must travel $O(\log n)$ hops between servers in an n-server DHT. Replicating popular (frequently-read) records more widely through the DHT counteracts this by reducing the *average* latency of lookups [10, 22]. Less popular data are left with longer read times — but less popular data are also less likely to be in a client's cache so have more need of low-latency service. Update latency is higher for more popular records, because the extra replicas must be reached. Unplanned updates could made faster by replacing TTL-based caching with the caching provided by the underlying DHT.

3.4 A different caching scheme

Changing the caching rules and keeping the hierarchy of authoritative servers would provide some benefit. If local caches were allowed to keep records after they had expired, they could serve the stale data to clients immediately when a request arrived, and then asynchronously refresh the records. This would reduce the client-perceived latency at the cost of introducing about 1-2% error rate in those queries where stale cache entries are used [23]. It would increase the latency of updates, of course, and slightly decrease the effectiveness of denial-of-service attacks. This scheme would require changes to the client side, but would be easy to roll out incrementally.

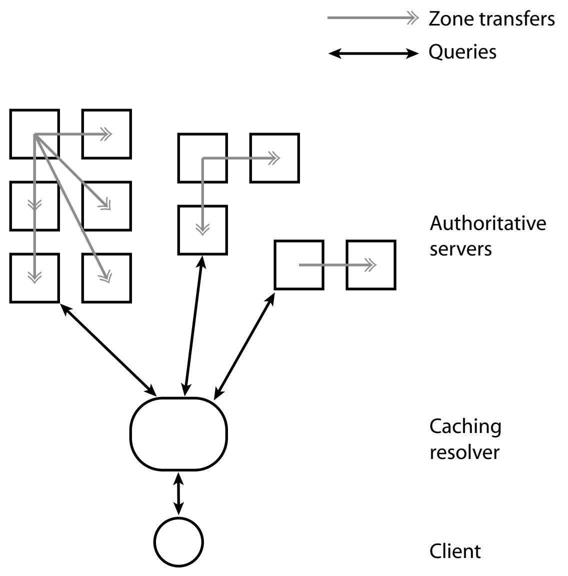

Figure 1: The current DNS

3.5 Replacing the DNS entirely

Building a new nameservice from scratch would allow a lot of freedom to design a service that is suited to the needs of the DNS's users. Unfortunately, the DNS is too far entwined in the software at every point on the internet to hope to extract it now. Any plausible strategy will have to interoperate cleanly with the clients of the current DNS.

Any one of these options could provide an improvement over the current DNS, although we have not addressed the difficulties of implementation yet. In the next section, we discuss the most attractive option, if possibly the hardest to engineer: the central service.

4. RECENTRALIZING

A centralized DNS would merge all the data that are currently spread among the many servers into one coherent database, which would be queried by caches just as the authoritative servers are now. The centralized service could be made available from the same IP addresses that the root zone is served from at the moment: thus, no changes would be necessary to client software. The database would be replicated at a small number of geographically distributed sites to provide availability and fault-tolerance. Updates would be received at any of these sites, authenticated and distributed among replicas, using a voting protocol such as the Paxon Parliament protocol [24] or UFP [25] to ensure consistency without requiring all sites to be working and reachable simultaneously. This would effectively replace the distributed lookup mechanism of the DNS (Figure 1) with a content distribution network and a very simple lookup (Figure 2).

The service must be available and responsive even in times of attack and failure. This is the main reason for distribution and replication, and must be balanced against the update latency incurred. It must, of course, also be scaleable — a single server capable of handling the total load on the DNS today would be very impressive — but wide-area replication is not the only solution to that. We suggest that replica-

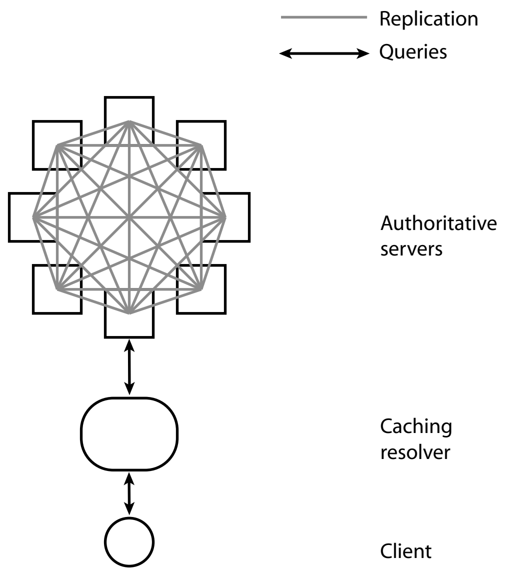

Figure 2: A centralised service

tion to fewer than one hundred sites should suffice. This would be similar to the current root servers or TLD servers for large TLDs, and would give similar levels of protection against denial-of-service attacks against the entire system, while giving much better protection to individual zones.

A distributed denial of service attack against such a system would be comparable to an attack on the gTLDs, and have a similar effect – flooding network links badly enough to cause collateral problems in many other systems as well as causing widespread failure of name resolution. For larger scale replication, a two-class infrastructure would allow more copies while avoiding excessive update latency [25].

Instead of running one or more nameservers, a zone administrator would submit updates to the central service, which would verify that the changes were consistent and authorized before publishing the new records. By removing server delegations (but not the delegation of authority) the administrative pitfalls associated with them are eliminated. Other problematic areas (e.g., aliasing, and the matching of forward and reverse mappings) would be made easier by the availability of the entire DNS at upload time, supporting automatic detection of many configuration errors.

We will now discuss the benefits of a centralized design in more detail, looking at some of the features which we consider to be desirable in any replacement for the DNS.

4.1 Interoperability

There are a large number of DNS clients deployed already; a scheme which required changes to client code would not be reasonable. Centralizing the DNS does not involve any changes to the protocol or to the resolution algorithm; some changes to the caching algorithm might be made possible by the centralization, but they are certainly not necessary for the success of the scheme.

The new system must retain the DNS's hierarchical delegation of authority, by accepting updates only from authoritative publishers. This does not provide the publisher with any guarantees about the behaviour of the servers, but for most organizations, which are happy to trust their secondary DNS servers and the TLD and root zone operators at the moment, this should be acceptable. Clients are left with the current IP-based scheme for the moment: there is definitely room for improvement here.

DNSSEC uses per-record digital signatures which could support this delegation of authority, but it is a complex system and global deployment is still some way off. An interim scheme using PKI for authentication and authorization of data publishers could be used for those who do not want to deploy DNSSEC in their zones.

4.2 Low latency

The latency of reads would be reduced: almost all reads would be satisfied by a single request to a fast and well-connected server. This would not only give faster lookups under normal conditions, but also reduce the impact of network congestion and link failures because fewer links are involved. The service is intended to handle many more reads than writes, so read latency is more important than update latency. If it is distributed for reliability and reachability, a full copy of the database should be available at each site: redirecting requests between servers increases latency and decreases reliability.

The latency of planned updates (where the records being changed have very short TTLs to avoid caching old versions) would be slightly increased: updates must be handled by the central service instead of by the domain owner's machines. So long as this latency increase is no more than a few minutes, it is an acceptable price to pay for fast reads and high availability. Luckily, naming data are largely static: the typical DNS entry is static over a timescale of months [26]. Also, updates are relatively tolerant of delay – a few seconds would not be noticed, and a few minutes would be entirely acceptable for almost all DNS updates. Commercial secondary DNS services often limit the batching interval for updates to much longer than that.

Update latency scales poorly with the number of copies of each record that must be reached; because we want to have all records available from all sites, this is a limiting factor on how many sites we can have.

The latency of unplanned updates can only be countered by a change in the TTL-based caching mechanism: eliminating it altogether would require changes in too many client systems, but a fast and highly available service might lessen the need for long TTLs. (For example, TTLs of no more than 15 minutes could be used, as mentioned in Section 2.) More ambitious schemes, allowing caches to keep closer coherence with the central servers, can be imagined, though their feasibility would depend on the rate of updates.

4.3 Simplicity

The system should be as simple as possible, both to configure and to implement. We agree with the designers of the DNS that a simple system is more likely to be widely (and correctly) implemented and deployed.

The design of the system can be described by two interfaces: one describing how records are submitted to the service for publication (probably based largely on the standard zone file format); and one describing the distribution of updates between sites. In order to encourage multiple implementations, a third interface can be drawn between the servers that maintain consistency of the data between sites and the servers that publish the data.

4.4 Flexibility

Under the current delegation scheme, zone administrators can choose to implement new and interesting policies which are not part of the original design of the DNS. In some content distribution networks (CDNs) the DNS response is tailored based on the source of the query, in the hope that this will direct a client to a nearby server. Although this is not a particularly good method of choosing a server in a CDN [27,28], it is commonly enough used that a replacement scheme should be able to cope with it. Another common requirement is on-the-fly generation of records, often used for reverse DNS. This sort of functionality could be provided in a centralized DNS by allowing table lookups and simple regular-expression rules.

For more exotic requirements, some zones could be delegated to their own servers using "NS" records as usual, with the associated costs (increased latency, cost of equipment, additional points of failure) to the zone owner. Alternatively, providing a fully programmable extension along the lines of Active Names [29] would allow zone owners to customize their own records' behaviour without excluding them from the centralized scheme, but this conflicts with the goals of simplicity and low latency: it would require more powerful servers, as well as mechanisms for resource partitioning, scheduling and sandboxing.

Future changes to the DNS would of course be easier to roll out on a centralized service managed by a small number of organizations than across more than a million independently-administered servers.

4.5 Upgrade path

An important practical question is how the service would be rolled out and paid for. One possible scenario is this: a group of TLD operators co-operate to build the service, and use it first to publish their own zone files. A co-operative venture of this kind would be an opportunity to reduce running costs, and also to offer a new service to zone owners — migrating your zone to the same servers as its parent would give faster resolution, and there is already a market for highly available and well-placed nameservice [30]. Once the service is established, more customers (leaf nodes and TLDs) could be solicited. The operating costs would come from the yearly fees already paid for domain registration and DNS hosting, and the control of the system would be in the hands of the TLD operators (who are, we hope, trustworthy, and if not are already in a position to do great damage to their clients).

5. CHARACTERIZING THE LOAD

Informed decisions about the degree of replication needed cannot be taken without detailed information on the workload: the amount and type of data present, the request patterns, the rate of change, etc. This is not given clearly in existing work; it would be a worthwhile exercise just to produce a plausible and detailed set of requirements for any DNS-replacement scheme. There has been detailed analysis of traffic at the root servers [18,31] and of the distribution of DNS requests in captured traffic [14], but work remains to be done, particularly in characterizing the update patterns. We are forced at the moment to make informed estimates based on the limited information that is available.

Each root server handles about 4000–8000 queries per second [19, 32], so at a rough estimate the total load across the root servers is of the order of 100,000 queries per second.[2] The load at the gtld-servers, which serve .COM and .NET, is about the same. The total load across all authoritative servers is harder to estimate; in 1992, 1/3 of long-range DNS queries were to the root servers [33], but that number will certainly have changed, not least because the generic top-level domains (gTLDs) are now served from their own servers.

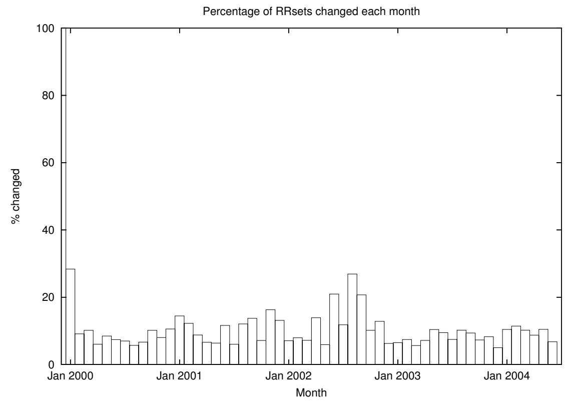

Figure 3: Changes to .IE RRsets over time.

In June and July of 2004, we gathered data on the contents of the DNS by recursively requesting zone transfers of all zones under some common gTLDs. We found that 23.4% of zones had at least one server which allowed AXFR requests, giving us something over 8 million zones. Similar data for 90 European country-code top-level domains (ccTLDs) were provided by the RIPE NCC, from their regional hostcount project [34]. The RIPE hostcount is an established project, and major ISPs have been lobbied to allow zone transfers to RIPE's local collection points: they successfully transferred 50.1% of zones (a little over 10 million) from 90 ccTLDs in April 2004.

If these zones are representative of the DNS as a whole, we can calculate that the gTLDs contain about 17GB of data, and all the RIPE-area ccTLDs about 15GB. Assuming the non-RIPE ccTLDs are as busy as the RIPE ones, that makes about 55GB of forward data. Reverse-DNS entries for 233 million hosts should fit comfortably in another 10GB.

Therefore it should be possible to build a single server which holds the entire DNS in RAM, and certainly it is possible to hold it on disk. A single server running BIND can handle 10,000 queries per second; NSD can handle 15,000, and some commercial offerings claim even higher throughput. With clustering and load-balancing, very high query rates could be handled; between 60 or 70 sites, a load of a few million queries per second is not unmanageable.

The update traffic is also hard to characterize, and very little has been published about it. Figure 3 shows the change history of the RRsets seen under the .IE TLD by the RIPE recursive-transfer tool. For each month it gives the percentage of all RRsets that were updated (including being created or deleted). It shows that the majority of records seen by the RIPE probe stay the same over a timescale of several months.

[2]Up to 70% of those queries are repeat queries caused by misconfigured clients, caches, firewalls and routers.

This graph does not include reverse zones or those dynamic zones which do not allow zone transfers: the RIPE probe sees about half of the forward zones under .IE. Also it does not show, for the RRsets that do change, how often they change over a month. A more fine-grained survey of the RRset rate of change is being undertaken to fill in some of these gaps. In general, a lot of work remains to be done to produce a full description of the load on the DNS.

6. CONCLUSION

We have considered a number of common problems with the DNS, and some of the possible solutions. The particular solution we propose is to remove the unnecessary and inefficient distribution and recentralize the publication of data in the DNS.

The wide distribution of the current DNS does not provide the resilience to attack or fast response which might be hoped for from a distributed system; rather it introduces higher latency and multiple points of failure in the resolution of any name.

We have highlighted the lack of knowledge about the contents and functioning of the current DNS: designing any replacement for such a large and vital piece of internet infrastructure requires a much more detailed specification of the requirements than is currently available. However, we argue from some initial estimates that it is not infeasible to build such a system, and that it is worthwhile trying to specify it in more detail.

Some other problems remain to be solved. What would the incentives be for competition and diversity in a centralized DNS? Could a market in data management be built on top of a uniform underlying database? Most importantly, is it reasonable to construct a single, centralized DNS given the load that the system must support? We believe that these issues are all not only interesting, but entirely tractable research issues, and look forward to exploring them in greater depth.

7. ACKNOWLEDGMENTS

The authors would like to thank the RIPE NCC and HEAnet for providing the output of their hostcount probes.

8. REFERENCES

[1] P. V. Mockapetris. Domain names — concepts and facilities. RFC 1034, November 1987.

[2] *Open Systems Interconnection — The Directory.* Number X.500–586 in ITU–T Recommendations. ITU, 2001. (Also ISO/IEC Standard 9594:2001).

[3] Paul V. Mockapetris and Kevin J. Dunlap. Development of the domain name system. In *Proc. ACM SIGCOMM 1988*, pages 123–133, August 1988.

[4] D. Eastlake. Domain name system security extensions. RFC 2535, March 1999.

[5] P. Vixie, O. Gudmundsson, D. Eastlake 3rd, and B. Wellington. Secret key transaction authentication for DNS (TSIG). RFC 2845, May 2000.

[6] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. Dynamic updates in the domain name system (DNS UPDATE). RFC 2136, April 1997.

[7] Internet Systems Consortium. *Internet Domain Survey.* `http://www.isc.org/ops/ds/`.

[8] Jussi Kangasharju and Keith W. Ross. A replicated architecture for the domain name system. In *Proc. 19th IEEE INFOCOM*, volume 2, pages 660–669, March 2000.

[9] Russ Cox, Athicha Muthitacharoen, and Robert T. Morris. Serving DNS using a peer-to-peer lookup service. In *Proc. 1st IPTPS*, volume 2429 of *LNCS*, pages 155–165, March 2002.

[10] Venugopalan Ramasubramanian and Emin Gun Sirer. The design and implementation of a next generation name service for the internet. In *Proc. ACM SIGCOMM 2004*, August 2004.

[11] Md Ahsan Habib and Marc Abrams. Analysis of sources of latency in downloading web pages. In *Proc. 5th WebNet*, October 2000.

[12] Edith Cohen and Haim Kaplan. Prefetching the means for document transfer: A new approach for reducing web latency. In *Proc. 19th IEEE INFOCOM*, volume 2, pages 854–863, March 2000.

[13] KyoungSoo Park, Vivek S. Pai, Larry Peterson, and Zhe Wang. CoDNS: Improving DNS performance and reliability via cooperative lookups. In *Proc. 6th OSDI*, pages 199–214, December 2004.

[14] Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. DNS performance and the effectiveness of caching. In *Proc. 1st IMW*, pages 55–67, November 2001.

[15] Vasileios Pappas, Zhiguo Xu, Songwu Lu, Daniel Massey, Andreas Terzis, and Lixia Zhang. Impact of configuration errors on DNS robustness. In *Proc. ACM SIGCOMM 2004*, August 2004.

[16] D. J. Bernstein. *Notes on the Domain Name System.* `http://cr.yp.to/djbdns/notes.html`.

[17] Men & Mice. *BIND DNS Surveys.* `http://www.menandmice.com/6000/6200_bind_research.html`.

[18] Duane Wessels and Marina Fomenkov. Wow, that's a lot of packets. In *Proc. 4th PAM*, April 2003.

[19] Duane Wessels. Is your caching resolver polluting the internet? In *Proc. ACM SIGCOMM 2004*, August 2004.

[20] Paul Vixie, Gerry Sneeringer, and Mark Schleifer. *Events of 21–Oct–2002.* `http://d.root-servers.org/october21.txt`.

[21] R. Bush, D. Karrenberg, M. Kosters, and R. Plzak. Root name server operational requirements. RFC 2870, June 2000.

[22] Marvin Theimer and Michael B. Jones. Overlook: Scalable name service on an overlay network. In *Proc. 22nd ICDCS*, pages 52–61, July 2002.

[23] Edith Cohen and Haim Kaplan. Proactive caching of DNS records: Addressing a performance bottleneck. In *Proc. SAINT 2001*, pages 85–94, January 2001.

[24] Leslie Lamport. The part-time parliament. Digital SRC Research report 49, September 1989.

[25] Chaoying Ma. *Designing a universal name service.* PhD thesis, University of Cambridge, 1992.

[26] Richard Liston, Sridhar Srinivasan, and Ellen Zegura. Diversity in DNS performance measures. In *Proc. 2nd IMW*, pages 19–31, November 2002.

[27] Jianping Pan, Y. Thomas Hou, and Bo Li. An

overview of DNS-based server selections in content distribution networks. *Computer Networks*, 43:695–711, December 2003.

[28] Jeffrey Pang, Aditya Akella, Anees Shaikh, Balachander Krishnamurthy, and Srinivasan Seshan. On the responsiveness of DNS-based network control. In *Proc. 2nd IMC*, pages 21–26, October 2004.

[29] Amin Vahdat, Michael Dahlin, Thomas Anderson, and Amit Agarwal. Active names: Flexible location and transport of wide-area resources. In *Proc. 2nd USITS*, October 1999.

[30] Verisign, Inc. *Ensuring your company's online presence*. June 2005. White paper. `http://www.verisign.com/Resources/`.

[31] Andre Broido, Evi Nemeth, and kc claffy. Spectroscopy of DNS update traffic. In *Proc. ACM SIGMETRICS 2003*, pages 320–321, June 2003.

[32] Tony Lee, Brad Huffaker, Marina Fomenkov, and kc claffy. On the problem of optimization of DNS root servers' placement. In *Proc. 4th PAM*, April 2003.

[33] P. Danzig, K. Obrackza, and A. Kumar. An analysis of wide-area name server traffic. In *Proc. ACM SIGCOMM 1992*, 1992.

[34] RIPE NCC. *The RIPE Region Hostcount*. `http://ripe.net/info/stats/hostcount/`.