The Design and Implementation of a Next Generation Name Service for the Internet

Venugopalan Ramasubramanian Emin Gün Sirer

Dept. of Computer Science, Cornell University,
Ithaca, NY 14853
{ramasv,egs}@cs.cornell.edu

ABSTRACT

Name services are critical for mapping logical resource names to physical resources in large-scale distributed systems. The Domain Name System (DNS) used on the Internet, however, is slow, vulnerable to denial of service attacks, and does not support fast updates. These problems stem fundamentally from the structure of the legacy DNS.

This paper describes the design and implementation of the Cooperative Domain Name System (CoDoNS), a novel name service, which provides high lookup performance through proactive caching, resilience to denial of service attacks through automatic load-balancing, and fast propagation of updates. CoDoNS derives its scalability, decentralization, self-organization, and failure resilience from peer-to-peer overlays, while it achieves high performance using the Beehive replication framework. Cryptographic delegation, instead of host-based physical delegation, limits potential malfeasance by namespace operators and creates a competitive market for namespace management. Backwards compatibility with existing protocols and wire formats enables CoDoNS to serve as a backup for legacy DNS, as well as a complete replacement. Performance measurements from a real-life deployment of the system in PlanetLab shows that CoDoNS provides fast lookups, automatically reconfigures around faults without manual involvement and thwarts distributed denial of service attacks by promptly redistributing load across nodes.

Categories and Subject Descriptors: C.2.4 [Computer-Communication Networks]: Domain Name System

General Terms: Design, Performance, Reliability.

Keywords: DNS, peer to peer, proactive caching.

1. INTRODUCTION

Translation of names to network addresses is an essential predecessor to communication in networked systems. The Domain Name System (DNS) performs this translation on the Internet and constitutes a critical component of the Internet infrastructure. While the DNS has sustained the growth of the Internet through static, hierarchical partitioning of the namespace and wide-spread caching, recent increases in malicious behavior, explosion in client population, and the need for fast reconfiguration pose difficult problems. The existing DNS architecture is fundamentally unsuitable for addressing these issues.

The foremost problem with DNS is that it is susceptible to denial of service (DoS) attacks. This vulnerability stems from limited redundancy in nameservers, which provide name-address mappings and whose overload, failure or compromise can lead to low performance, failed lookups and misdirected clients. Approximately 80% of the domain names are served by just two nameservers, and a surprising 0.8% by only one. At the network level, all servers for 32% of the domain names are connected to the Internet through a single gateway, and can thus be compromised by a single failure. The top levels of the hierarchy are served by a relatively small number of servers, which serve as easy targets for denial of service attacks [4]. A recent DoS attack [28] on the DNS crippled nine of the thirteen root servers at that time, while another recent DoS attack on Microsoft's DNS servers severely affected the availability of Microsoft's web services for several hours [38]. DNS nameservers are easy targets for malicious agents, partly because approximately 20% of nameserver implementations contain security flaws that can be exploited to take over the nameservers.

Second, name-address translation in the DNS incurs long delays. Recent studies [41, 16, 18] have shown that DNS lookup time contributes more than one second for up to 30% of web object retrievals. The explosive growth of the namespace has decreased the effectiveness of DNS caching. The skewed distribution of names under popular domains, such as *.com*, has flattened the name hierarchy and increased load imbalance. The use of short timeouts for popular mappings, as is commonly employed by content distribution networks, further reduces DNS cache hit rates. Further, manual configuration errors, such as lame delegations [29, 27], can introduce latent performance problems.

Finally, widespread caching of mappings in the DNS prohibits fast propagation of unanticipated changes. Since the DNS does not keep track of the locations of cached mappings, but relies on timeout-based invalidations of stale copies, it cannot guarantee cache coherency. Lack of cache coherency in the DNS implies that changes may not be visible to clients for long durations, effectively preventing quick service relocation in response to attacks or emergencies.

SIGCOMM'04, Aug. 30–Sept. 3, 20034, Portland, Oregon, USA.

Fresh design of the legacy DNS provides an opportunity to address these shortcomings. A replacement for the DNS should exhibit the following properties.

- **High Performance**: Decouple the performance of DNS from the number of nameservers. Achieve lower latencies than legacy DNS and improve lookup performance in the presence of high loads and unexpected changes in popularity ("the slashdot effect").
- **Resilience to Attacks**: Remove vulnerabilities in the system and provide resistance against denial of service attacks through decentralization and dynamic load balancing. Self-organize automatically in response to host and network failures.
- **Fast Update Propagation**: Enable changes in name-address mappings to quickly propagate to clients. Support secure delegation to preserve integrity of DNS records, and prohibit rogue nodes from corrupting the system.

This paper describes Cooperative Domain Name System (CoDoNS), a backwards-compatible replacement for the legacy DNS that achieves these properties. CoDoNS combines two recent advances, namely, structured peer-to-peer overlays and analytically informed proactive caching. Structured peer-to-peer overlays, which create and maintain a mesh of cooperating nodes, have been used previously to implement wide-area distributed hash tables (DHTs). While their self-organization, scalability, and failure resilience provide a strong foundation for robust large-scale distributed services, their high lookup costs render them inadequate for demanding, latency-sensitive applications such as DNS [18]. CoDoNS achieves high lookup performance on a structured overlay through an analytically-driven proactive caching layer. This layer, called Beehive [32], automatically replicates the DNS mappings throughout the network to match anticipated demand and provides a strong performance guarantee. Specifically, Beehive achieves a targeted average lookup latency with a minimum number of replicas. Overall, the combination of Beehive and structured overlays provides the requisite properties for a large scale name service, suitable for deployment over the Internet.

Our vision is that globally distributed CoDoNS servers self-organize to form a flat peer-to-peer network, essentially behaving like a large, cooperative, shared cache. Clients contact CoDoNS through a local participant in the CoDoNS network, akin to a legacy DNS resolver. Since a complete takeover from DNS is an unrealistic undertaking, we have designed CoDoNS for an incremental deployment path. At the wire protocol level, CoDoNS provides full compatibility with existing DNS clients. No changes to client-side resolver libraries, besides changing the identities of the nameservers in the system configuration (e.g. modifying */etc/resolv.conf* or updating DHCP servers), are required to switch over to CoDoNS. At the back end, CoDoNS transparently builds on the existing DNS namespace. Domain names can be explicitly added to CoDoNS and securely managed by their owners. For names that have not been explicitly added, CoDoNS uses legacy DNS to acquire the mappings. CoDoNS subsequently maintains the consistency of these mappings by proactively checking with legacy DNS for updates. CoDoNS can thus grow as a layer on top of legacy DNS and act as a safety net in case of failures in the legacy DNS.

Measurements from a deployment of the system in Planet Lab [2] using real DNS workloads show that CoDoNS can substantially decrease the lookup latency, handle large flash-crowds, and quickly disseminate updates. CoDoNS can be deployed either as a complete replacement for DNS, where each node operates in a separate administrative and trust domain, or as an infrastructure service within an ISP, where all nodes are in the same administrative and trust domain.

The peer-to-peer architecture of CoDoNS securely decouples namespace management from a server's location in the network and enables a qualitatively different kind of name service. Legacy DNS relies fundamentally on physical delegations, that is, query handoffs from host to host until the query reaches a set of designated servers considered authoritative for a certain portion of the namespace owned by a namespace operator. Since all queries that involve that portion of the namespace are routed to these designated servers, the namespace operator is in a unique position of power. An unscrupulous namespace operator may abuse this monopoly by modifying records on the fly, providing differentiated services, or even creating synthetic responses that redirect clients to their own servers. Nameowners that are bound to that namespace have no other recourse. In contrast, name records in CoDoNS are tamper-proof and self-validating, and delegations are cryptographic. Any peer with a valid response can authoritatively answer any matching query. This decoupling of namespace management from the physical location and ownership of nameservers enables CoDoNS to delegate the same portion of the namespace, say *.com*, to multiple competing namespace operators. These operators, which are each provided with signing authority over the same space, assign names from a shared, coordinated pool, and issue self-validating name bindings into the system. Since CoDoNS eliminates physical delegations and designated nameservers, it breaks the monopoly of namespace operators and creates an even playing field where namespace operators need to compete with each other on service.

The rest of this paper is organized as follows. In the next section, we describe the basic operation of the legacy DNS and highlight its drawbacks. Section 3 describes the design and implementation of CoDoNS in detail. In Section 4, we present performance results from the PlanetLab deployment of CoDoNS. We summarize related work in Section 5, and conclude in Section 6.

2. DNS: OPERATION AND PROBLEMS

The Domain Name System (DNS) is a general-purpose database for managing host information on the Internet. It supports any kind of data, including network address, ownership, and service configuration, to be associated with hierarchically structured names. It is primarily used to translate human-readable names of Internet resources to their corresponding IP addresses. In this section, we provide a brief overview of the structure and operation of the legacy DNS, identify its major drawbacks, and motivate a new design.

2.1 Overview of Legacy DNS

The legacy DNS [25, 26] is organized as a static, distributed tree. The namespace is hierarchically partitioned into non-overlapping regions called *domains*. For example, *cs.cornell.edu* is a sub-domain of the domain *cornell.edu*, which in turn is a sub-domain of the top-level domain *edu*. Top-level domains are sub-domains of a global root domain.

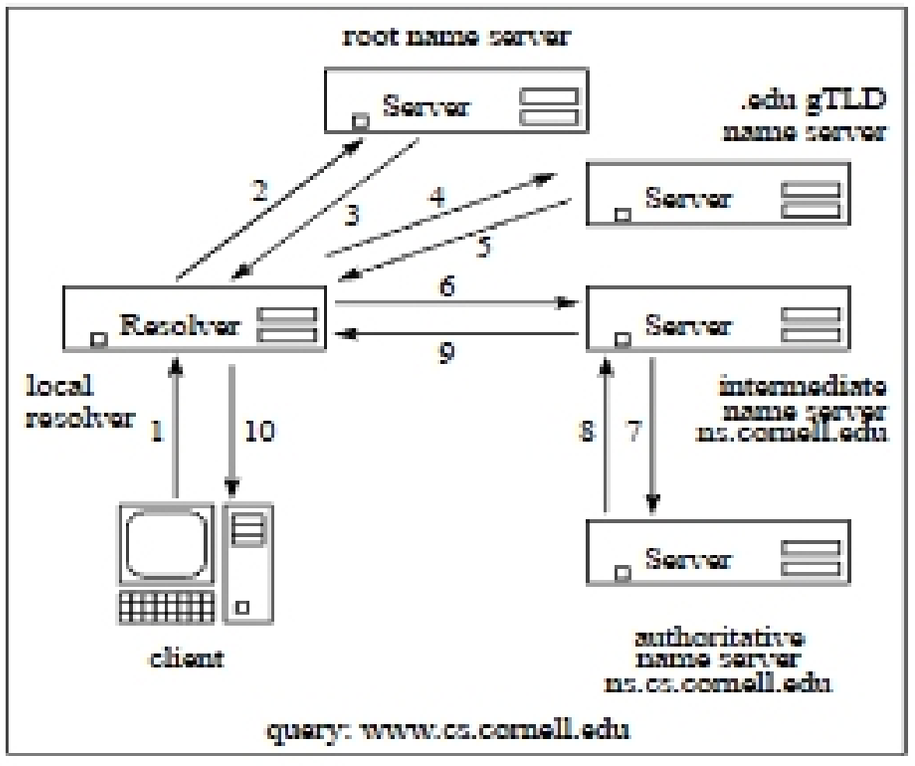

Figure 1: Name Resolution in Legacy DNS: Resolvers translate names to addresses by following a chain of delegations iteratively (2-5) or recursively (6-9).

Domain names, such as *www.cs.cornell.edu*, belong to *name-owners*.

Extensible data structures, called *resource records*, are used to associate values of different types with domain names. These values may include the corresponding IP address, mail host, owner name and the like. The DNS query interface allows these records to be retrieved by a query containing a domain name and a type.

The legacy DNS delegates the responsibility for each domain to a set of replicated *nameservers* called *authoritative nameservers*. The authoritative nameservers of a domain manage all information for names in that domain, keep track of authoritative nameservers of the sub-domains rooted at their domain, and are administered by *namespace operators*. At the top of the legacy DNS hierarchy are *root nameservers*, which keep track of the authoritative nameservers for the *top-level domains* (TLDs). The top-level domain namespace consists of generic TLDs (gTLD), such as *.com*, *.edu, and .net*, and country-code TLDs (ccTLD), such as *.uk*, *.tr, and .in*. Nameservers are statically configured with thirteen IP addresses for the root servers. BGP-level anycast is used in parts of the Internet to reroute queries destined for these thirteen IP addresses to a local root server.

Resolvers in the legacy DNS operate on behalf of clients to map queries to matching resource records. Clients typically issue DNS queries to local resolvers within their own administrative domain. Resolvers follow a chain of authoritative nameservers in order to resolve the query. The local resolver contacts a root nameserver to find the top-level domain nameserver. It then issues the query to the TLD nameserver and obtains the authoritative nameserver of the next sub-domain. The authoritative nameserver of the sub-domain replies with the response for the query. This process continues recursively or iteratively until the authoritative nameserver of the queried domain is reached. Figure 1 illustrates the different stages in the resolution of an example domain name *www.cs.cornell.edu*. While this figure provides a simple overview of the communication involved in name resolution, in practice, each query may trigger additional lookups to resolve intermediate nameservers [25, 26].

Bottlenecks	All Domains	Top 500
1	0.82 %	0.80 %
2	78.44 %	62.80 %
3	9.96 %	13.20 %
4	4.64 %	13.00 %
5	1.43 %	6.40 %
13	4.12 %	0 %

Table 1: Delegation Bottlenecks in Name Resolution: A significant number of names are served by two or fewer nameservers, even for the most popular 500 sites.

Pursuing a chain of delegations to resolve a query naturally incurs significant delay. The legacy DNS incorporates aggressive caching in order to reduce the latency of query resolution. The resolvers cache responses to queries they issue, and use the cached responses to answer future queries. Since records may change dynamically, legacy DNS provides a weak form of cache coherency through a *time-to-live* (TTL) field. Each record carries a TTL assigned by the authoritative nameserver, and may be cached until TTL expires.

2.2 Problems with Legacy DNS

The current use and scale of the Internet has exposed several shortcomings in the functioning of the legacy DNS. We performed a large scale survey to analyze and quantify its vulnerabilities. Our survey explored the delegation chains of 593160 unique domain names collected by crawling the Yahoo! and the DMOZ.ORG web directories. These domain names belong to 535088 unique domains and are served by 164089 different nameservers. We also separately examined the 500 most popular domains as determined by the Alexa ranking service. In this section, we describe the findings of our survey, which highlight the problems in failure resilience, performance, and update propagation in the legacy DNS.

Failure Resilience - Bottlenecks

The legacy DNS is highly vulnerable to network failures, compromise by malicious agents, and denial of service attacks, because domains are typically served by a very small number of nameservers. We first examine the *delegation bottlenecks* in DNS; a delegation bottleneck is the minimum number of nameservers in the delegation chain of each domain that need to be compromised in order to control that domain. Table 1 shows the percentage of domains that are bottlenecked on different numbers of nameservers. 78.63% of domains are restricted by two nameservers, the minimum recommended by the standard [25]. Surprisingly, 0.82% of domains are served by only one nameserver. Even the highly popular domains are not exempt from severe bottlenecks in their delegation chains. Some domains (0.43%) spoof the minimum requirement by having two nameservers map to the same IP address. Overall, over 90% of domain names are served by three or fewer nameservers and can be disabled by relatively small-scale DoS attacks.

Failure and attack resilience of the legacy DNS is even more limited at the network level. We examined *physical bottlenecks*, that is, the minimum number of network gateways or routers between clients and nameservers that need to be compromised in order to control that domain. We measured the physical bottlenecks by performing traceroutes to 10,000 different nameservers, which serve about 5,000 ran-