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

Venugopalan Ramasubramanian   Emin Gün Sirer

Presented By: Kamalakar Kambhatla

* Slides adapted from the paper - The Design and Implementation of a Next Generation Name Service for the Internet by Venugopalan Ramasubramanian and Emin Gün Sirer – SIGCOMM 2004
Overview of DNS

- Static, hierarchical partitioning of the namespace, top down lookup through delegations
- TTL based caching, expensive, no bounds lookups
Problems

* Performance bottlenecks
  - Slow, cold misses lead to high latencies
* Vulnerable to DoS attacks
  - 80% of the domain names are served by just two nameservers and 0.8% by only one
  - Top levels of hierarchy are served by a relatively smaller number of servers
  - Recent DoS attack on Microsoft’s DNS servers severely affected the availability of Microsoft’s web services for several hours
Problems

- Latency issues
- Load imbalance
- Manual configuration, implementation errors
  - Large number of broken or inconsistent delegations
- Cache coherency and updates
Bottlenecks

* Resolvers in legacy DNS operate on behalf of clients to map queries to matching resource records
* Several nameservers belong to same subnet
* Name resolution latency is a significant component of the time required to access web services
* Short TTLs eliminate caching and overload nameservers
* Low performance mainly due to low cache hit rates, Zipf-like query distribution in DNS
* Root and TLD handle a large load and are frequently subjected to DoS attacks
* A massive distributed DoS attack in Nov 2002 rendered 9 of 13 root servers unresponsive – now root is more than 60 nameservers and is served through a special-case support for BGP-level anycast!!
* Large scale caching poses problems for maintaining the consistency of cached records in presence of dynamic changes.

* Selection of a suitable TTL value is tough – short TTLs affect lookup performance and increase network load, while long TTLs interfere with service relocation.

* Nearly 40% of domain names use TTLs of one day or higher, which prohibits fast dissemination of unanticipated changes to records.
Properties of Replacement DNS

* High Performance
  - Achieve lower latencies than legacy DNS and improve lookup performance in presence of high loads
* Resilience to Attacks
  - Remove vulnerabilities in the system and provide resistance against DoS attacks through decentralization and dynamic load balancing
* Fast Update Propagation
  - Enable changes in name-address mappings to quickly propagate to clients
CoDoNS - Overview

* Derives scalability, decentralization, self-organization and failure resilience from structured p2p overlays
* High lookup performance through proactive caching
* Resilience to DoS attacks through automatic load balancing
* Fast propagation of updates
CoDoNS - Overview

* Achieves high performance through Beehive replication framework
* Cryptographic delegation instead of host based physical delegation limits potential malfeasance by namespace operators
* Backwards compatibility enables CoDoNS to serve as a backup for legacy DNS or even complete replacement
Globally distributed CoDoNS servers self-organize to form a flat p2p network, behaving like a large, cooperative, shared cache.

Clients contact CoDoNS through a local participant in the CoDoNS network, akin to the legacy DNS resolver.

No changes to client-side resolver libraries, besides changing the ids of nameservers in the system configuration.
Proactive replication framework that enables prefix-matching DHTs to achieve $O(1)$ lookup performance – examples are Pastry and Tapestry

Objects and nodes have randomly assigned Ids from the same circular space, and each object is stored at the nearest node in the Id space, called the home node

Each node routes a request for an object, by successively matching prefixes; that is, by routing the request to a node that matches one more digit with the object until the home node, is reached
* Query issued for object 2101
* Pastry incurs 3 hops to find a copy of the object
* By placing copies of the object at all nodes one hop prior to the home node in the request path, the lookup latency can be reduced by one hop
* O(logN) hops in the worst case to reach the home node
* By choosing levels of replication, average lookup performance can be tuned to any desired constant
* Can achieve O(1) lookups by replicating every object at every node, would incur excessive space overhead, consume significant bandwidth and lead to large update latencies
Beehive - Overview

* Minimizes bandwidth and space consumption by posing an optimization problem
* Minimize the total number of replicas subject to the constraint that the average lookup latency is less than a desired constant C
* For Zipf-like query distributions, Beehive analytically derives the optimal closed-form solution to this problem
* Expression for closed-form solution that minimizes total number of replicas for Zipf-like query distributions with parameter $\alpha<1$ is

$$x_i = \left[ \frac{d^i (\log N - C)}{1 + d + \cdots + d^\log N - 1} \right]^{\frac{1}{1-\alpha}} , \text{where } d = b^{\frac{1-\alpha}{\alpha}}$$
Beehive - Overview

* Any targeted average lookup latency can be achieved by selecting an appropriate C
* CoDoNS - C=0.5 hops, which means that a large percentage of requests are answered immediately
* Minimal bandwidth and storage overhead
* Replicating objects across several nodes balances the load, improves resilience against DoS attacks
* Level of replication can be used to quickly determine the location of all object replicas, and update them when necessary
* Globally distributed nodes that self organize to form a p2p network
* Provides query resolution services to clients using the same wire format and protocol as legacy DNS, and thus requires no changes to client resolvers
* Decouples namespace management from query resolution
* No restrictions on hierarchical structure of the namespace
* API: insert, delete and update
CoDoNS - Architecture

* Associates the node whose ID is closest to consistent hash of the domain name as the home node for that domain name
* Home node stores a permanent copy of the resource records owned by that domain name and manages their replication
* If home node fails, next closest node in ID space automatically becomes the new home node.
* Replicates all records on several nodes adjacent to home node in ID space to avoid data loss due to node failures
CoDoNS – Query Resolution
CoDoNS - Implementation

- Layered on top of Pastry and Beehive
- Each CoDoNS server implements a complete, recursive, caching DNS resolver
- Supports inverse queries that map IP addresses to a domain name
- Domain names have unique 128-bit IDs
- Home node, the closest node in the ID space, stores permanent copies of resource records of the domain name and maintains their consistency in the system
- Home node uses the TTL specified by legacy DNS as the duration to store the records. Refetches records from legacy DNS after TTL duration, and propagates updated records to all replicas if records change
Issues and Implications

* Decouples namespace management from the physical location of nameservers
* Uses cryptographic delegations and self-verifying records based on DNSSEC standard
* Every namespace operator has a public-private key pair; private key used to digitally sign DNS records managed by operator
* Chain of certificates terminating at a small number of well-known public keys for globally trusted authorities
* Signature and public key stored in DNS as resource records of type sig and key
* Clients can verify the authenticity of a resource record by fetching the sig record and the key record from the DNS
Issues and Implications

* DNSSEC not yet widely deployed, hence cannot rely on legacy DNS to provide certificates for resource records
* CoDoNS uses own centralized authority to sign resource records fetched from legacy DNS
* Queries to legacy DNS directed to a small pool of certifying resolvers, which fetch authoritative resource records from legacy DNS, sign them, and append the sig records to the legacy DNS response
* Requires trust to be placed in the certifying resolvers
* Malicious participants may disrupt the system by corrupting the routing tables of peers and misrouting or dropping queries
* Can use secure routing tables to limit impact of corrupt nodes, but may increase lookup delay
* P2p network of servers of globally distributed PlanetLab nodes

* Results show that CoDoNS provides low latency resolution service, resists flash-crowds and provides support for fast update propagation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pastry</td>
<td></td>
</tr>
<tr>
<td>base</td>
<td>16</td>
</tr>
<tr>
<td>leaf-set size</td>
<td>24</td>
</tr>
<tr>
<td>Beehive</td>
<td></td>
</tr>
<tr>
<td>target C</td>
<td>0.5 hops</td>
</tr>
<tr>
<td>aggregation interval</td>
<td>6 min</td>
</tr>
<tr>
<td>analysis interval</td>
<td>60 min</td>
</tr>
</tbody>
</table>
Setup

- Started with no initial DNS records
- After some time, issue DNS requests from a real workload to the server at each node
- Measure lookup latency and periodically record the load handled and overhead incurred by each node
- Workload obtained from traces collected at MIT
- Workload comprises of 281,943 queries for 47,230 unique domain names.
- Popularity of domain names closely follows a Zipf-like distribution with parameter 0.91.
- Measurements taken from a deployment on 75 distributed PlanetLab nodes.
Lookup Performance

Cumulative Distribution of Latency

Median Latency vs Time

Query Resolution Latency

<table>
<thead>
<tr>
<th>Latency</th>
<th>Mean</th>
<th>Median</th>
<th>90th %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoDoNS</td>
<td>106 ms</td>
<td>1 ms</td>
<td>105 ms</td>
</tr>
<tr>
<td>CoDoNS + DNS</td>
<td>199 ms</td>
<td>2 ms</td>
<td>213 ms</td>
</tr>
<tr>
<td>Legacy DNS</td>
<td>382 ms</td>
<td>39 ms</td>
<td>337 ms</td>
</tr>
<tr>
<td>PlanetLab RTT</td>
<td>121 ms</td>
<td>82 ms</td>
<td>202 ms</td>
</tr>
</tbody>
</table>
Flash-crowd Effect

Median Latency vs Time with introduction of a Flash-crowd at the 6th hour
Load Balance

Load Balance vs Time

*Coefficient of variation defined as the ratio of standard deviation of the load across all the nodes to the mean nodes
Update Propagation

- 98% of the replicas are updated within 1 sec
- Latency to update 99% of replicas one hop from the home node is about 1 second
- Overall, update propagation latency depends on the extent of replication
- Worst case – logN hops to update all nodes
CoDoNS provides low latencies for query resolution
Massive replication for the most popular records, but a modest number of replicas, achieves high performance with low overhead
Eliminating static query processing hierarchy and scheduling load dynamically onto peer nodes greatly decreases the vulnerability to DoS attacks
Self-organization and continuous replication avoids bottlenecks in presence of flash crowds
Proactive update propagation ensures that unanticipated changes can be quickly disseminated and cached in the system