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

CoDoNS

Venugopal Ramasubramanian
Emin Gün Sirer

SIGCOMM ’04

Presenter: Saurabh Kadekodi
Agenda

- DNS overview
- Current DNS Problems
- Beehive
- Cooperative DNS (CoDoNS)
  - Overview
  - Architecture
  - Implementation
  - Issues and Implications
- Evaluation
Hierarchically partitioned namespace
Top-down lookup
No bounds on lookup - hence expensive
DNS Workarounds

- Replicate root and intermediate servers
- Use caching wherever possible
- TTL based cache updates
DNS Problems

- Bottlenecks
- Implementation Errors
- Latency Issues
- Misconfigurations
- Load Imbalance
- Cache coherence and updates
## Bottlenecks

### Delegation Bottleneck

<table>
<thead>
<tr>
<th>Bottlenecks</th>
<th>All Domains</th>
<th>Top 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.82%</td>
<td>0.80%</td>
</tr>
<tr>
<td>2</td>
<td>78.44%</td>
<td>62.80%</td>
</tr>
<tr>
<td>3</td>
<td>9.96%</td>
<td>13.20%</td>
</tr>
<tr>
<td>4</td>
<td>4.64%</td>
<td>13.00%</td>
</tr>
<tr>
<td>5</td>
<td>1.43%</td>
<td>6.40%</td>
</tr>
<tr>
<td>13</td>
<td>4.12%</td>
<td>0%</td>
</tr>
</tbody>
</table>
Bottlenecks

- Physical Bottleneck

- Several nameservers belong to same subnet!
Latency

- Low performance due to low cache hit-rates - reason heavy-tailed distribution

- Short TTLs eliminates caching and overloads nameservers
Hierarchical structure imposes large loads on interior nodes towards the root

A DoS attack in November 2002 resulted in 9 of the 13 root servers unresponsive!
Update Propagation

- Servers do not maintain cache mappings

- Updates occur due to TTL expiry

- 40% of domain names have TTL of a day or higher
Properties of New System

- High Performance
- Resilience to Attacks
- Fast Update Propagation
Q: What is Cooperative DNS (CoDoNS)?
A: A global cooperative shared cache formed using peer-to-peer networks

Q: How should it be structured?
A: DHT, duh!

Q: What is the average lookup in DHT?
A: $O(\log n)$
Q: How do make it faster?
A: Replicate!

Q: What is the cost of replication?
A: Space Overhead

Q: How much do you replicate?
A: Beehive
Beehive

- Proactive replication framework

- Intelligent choice of different levels of replication for different objects tunes average lookup of system to a **constant** i.e. $O(1)$
Exploits DNS Zipf-like behavior to pose the following optimization question:

How do we minimize the total number of replicas subject to the constraint that the aggregate lookup latency is less than a desired constant $C$?

\[ x_i = \left[ \frac{d^i (\log N - C)}{1 + d + \ldots + d^{\log N-1}} \right]^{\frac{1}{1-\alpha}} \text{ where } d = b^{\frac{1-\alpha}{\alpha}} \]
Beehive in CoDoNS

- $C = 0.5$ results in a large percentage of queries answered from first hit itself
- It minimizes number of replicas for target lookup performance
- Replication improves resilience
- Level of replication can be used to quickly identify replicas for fast update
Feedback system to replicate as per need

Decentralized replication - each node for itself

On-the-fly adjustment of replication for flash crowds

Proactive update propagation makes TTL based mechanism unnecessary
CoDoNS Architecture

- Globally distributed nodes forming peer-to-peer network
- No change to existing client resolvers
- Decoupling of namespace management from query resolution
- API: insert, delete, update
- To survive home node failure, it replicates all its records to adjacent nodes
CoDoNS Peer Network

Case 1: Most of the time

Home Node
Working

CoDoNS Peer Network

Case 2: Reply from caches

Home Node

legacy DNS
Working

CoDoNS Peer Network

Home Node

Case 3: Reply from home node

legacy DNS
Working

CoDoNS Peer Network

Home Node

Case 4: Use legacy DNS
CoDoNS support inverse mappings viz. IP to name

All updates pushed to all replicas and they hold them until replication level is downgraded / update is received

TTL rounded to 30 secs. Lower TTL not stored as they indicate dynamic server selection

CoDoNS supports negative caching
Issues and Implications

- DNSSEC used instead of physical delegation to trusted nodes

- CoDoNS authenticates owners for every insert, update and delete

- Removal of authority from location of server provides greater flexibility and freedom of namespace management
Issues and Implications

- Enables multiple operators to compete for same part of name hierarchy

- Self-verifying records prohibit dynamic resolution using complex functionality at runtime

- CoDoNS relies on participants to contribute resources on behalf of others
Evaluation

- 75 globally distributed PlanetLab nodes
- Workload MIT trace 4 - 11 Dec 2000
- Zipf-like distribution with parameter 0.91

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pastry: base</td>
<td>16</td>
</tr>
<tr>
<td>leaf-set size</td>
<td>24</td>
</tr>
<tr>
<td>Beehive: 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>
Lookup Performance

Cumulative Distribution of Latency

Median Latency vs Time
Flash Crowd Effect

Median Latency vs Time for Flash Crowds
Load Balance vs Time
For a million node CoDoNS, updating 99% of replicas would take < 1 min even if most popular domains were replicated throughout.
Thus, CoDoNS aims at achieving a much more scalable, resilient and efficient domain name management eliminating most of DNS problems.

Questions???