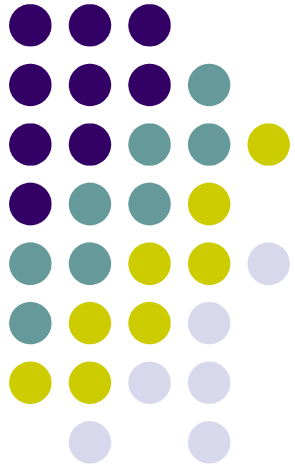
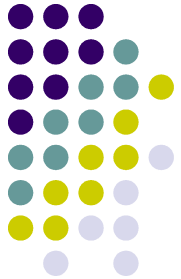
A Peer-to-Peer DNS

Ilya Sukhar
Venugopalan Ramasubramanian
Emin Gün Sirer
Cornell University



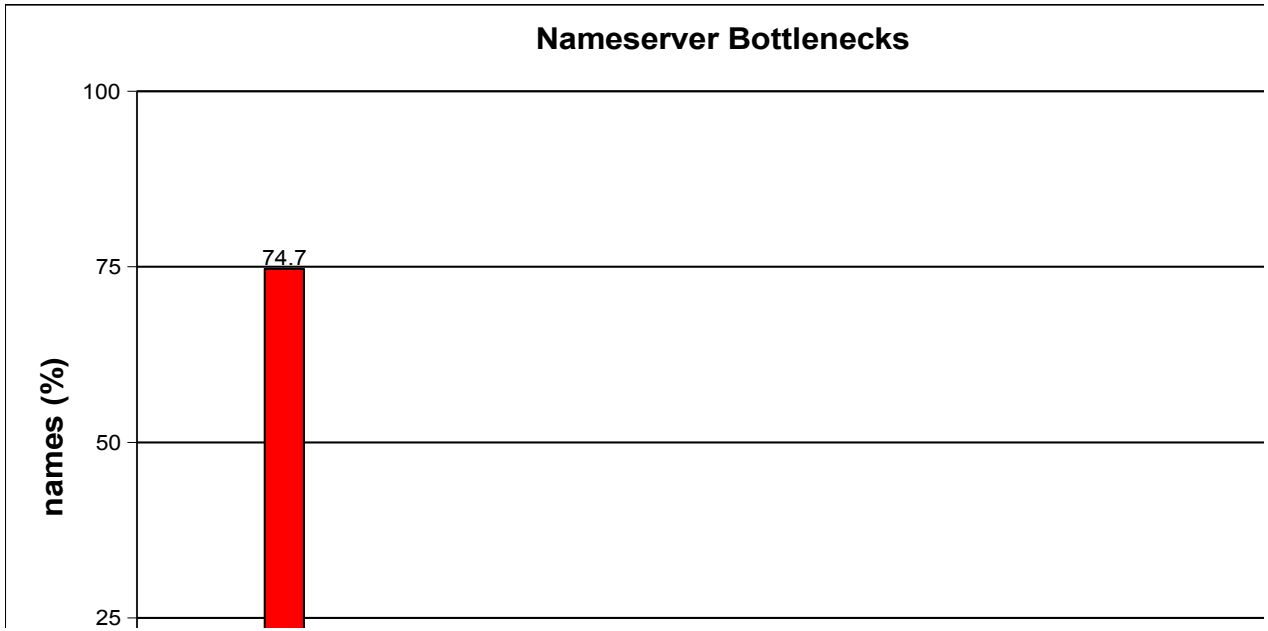
What's wrong with DNS?



- Failure Resilience
  - Delegation Bottlenecks
  - Physical Bottlenecks
- Performance
  - Latency tradeoff
  - Misconfiguration & Load Imbalance

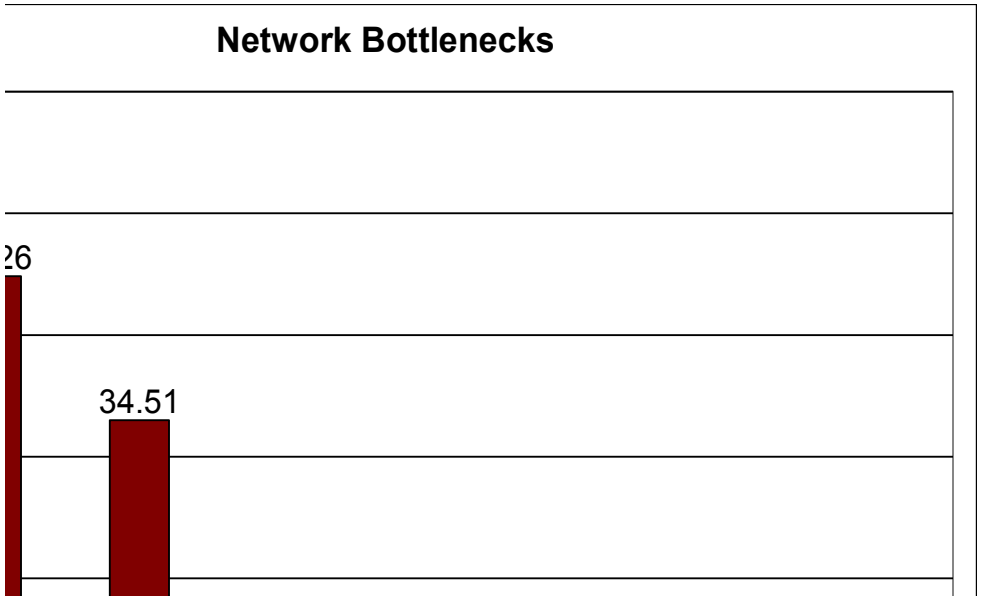
Failure Resilience – Delegation Bottlenecks

- 75% of domains are served by only two nameservers. Not a reflection of popularity – 62.8% in Top 500 have the same problem

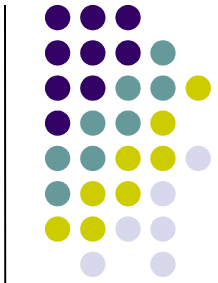


Failure Resilience – Physical Bottlenecks

- Majority of domains are physically bottlenecked at a single gateway or router
- Delegation redundancy is deceiving – many backup nameservers reside in the same subnet.

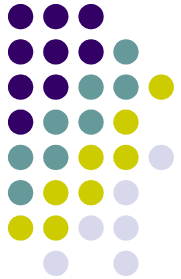


Performance – Latency



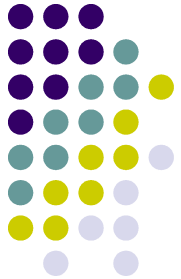
- Lookups are expensive
  - ~20-40% of web object retrieval time spent on DNS
  - ~20-30% of DNS lookups take more than 1s
    - [Jung et al. 01, Huitema et al. 00, Wills & Shang 00, Bent & Voelker 01]

The Problems



- Failure Resilience
  - Delegation and physical bottlenecks make attractive DDoS targets
- Latency
  - Dilemma of choosing between lookup performance and update propagation.
    - Timeout driven caching isn't effective. Short TTL's impose enormous overhead and drastically reduce cache hit rates.
- Static Hierarchy
  - Load imbalance, points of failure

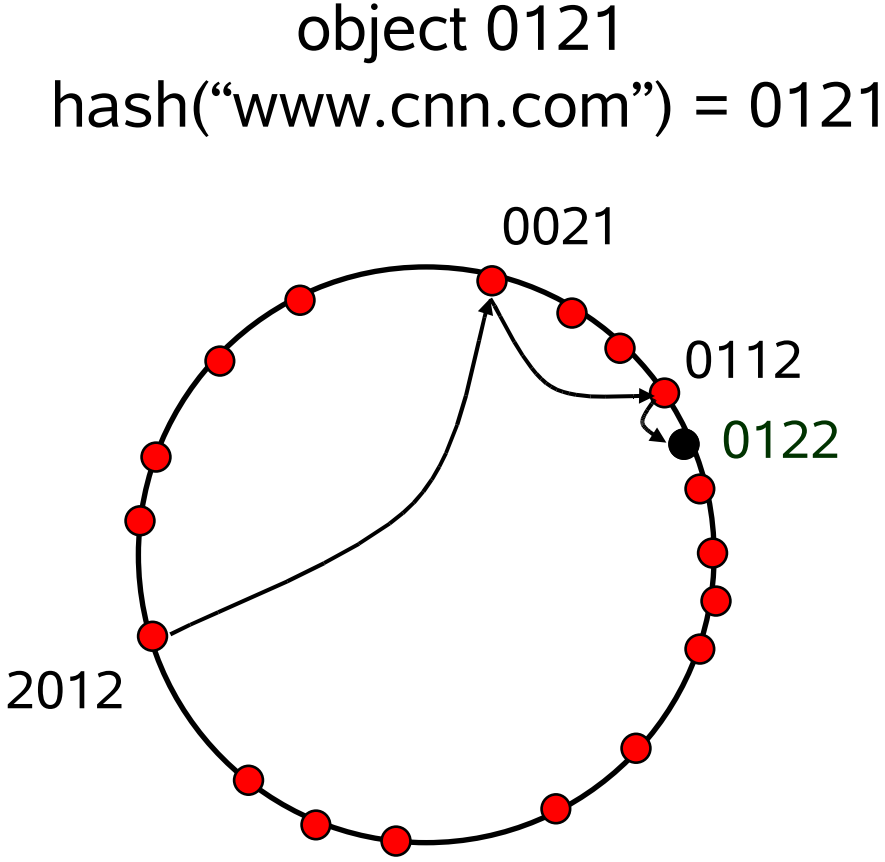
Our Approach



- Built on top of structured Distributed Hash Tables (DHTs)
  - Self organizing
  - Failure resilient
  - Scalable
  - Good performance

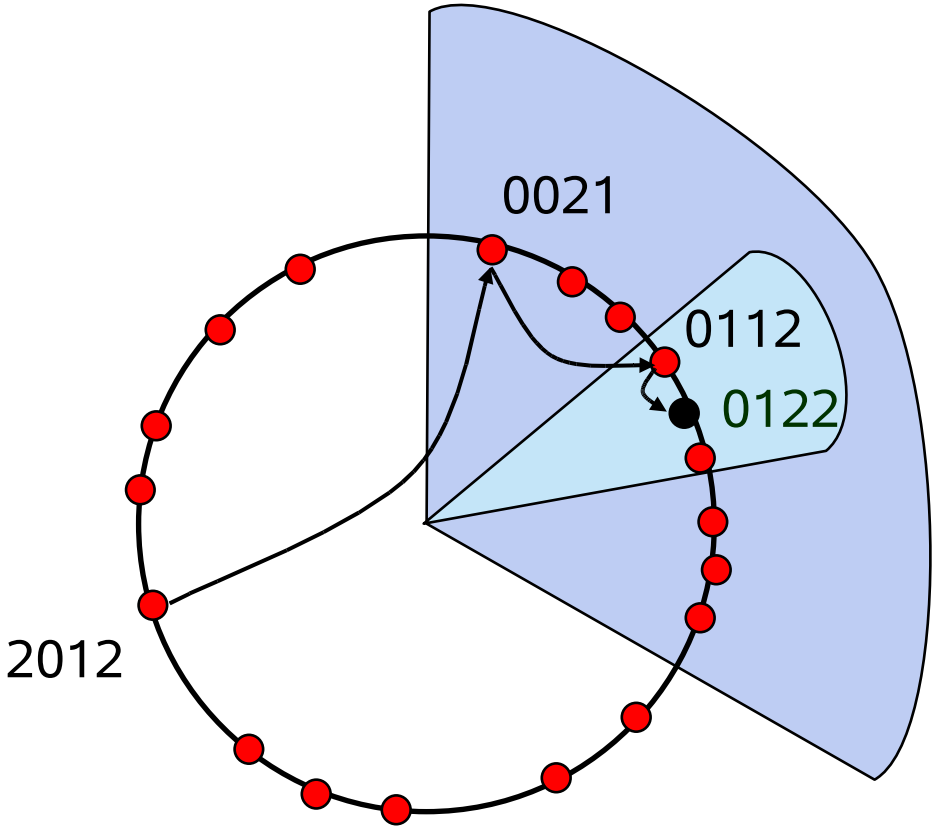
DHTs 101

- Pastry
  - Map all nodes onto common identifier space
  - Map all objects onto the same space using a key (for DNS, the name).
  - $\log_b N$ hops to travel the ring
    - Several round trips on the Internet – not so great, right?

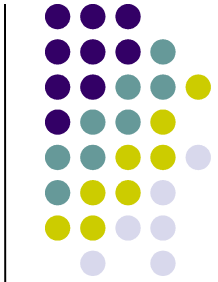


CoDoNS

- Adjusting the level of replication allows us to bound the latency of any lookup.
  - As always, must find the optimal point in the space-time tradeoff.
- How?
  - Use good mathematical properties of DNS query distribution
  - Key intuition: we can do this per-object based on popularity and properties of our distribution!

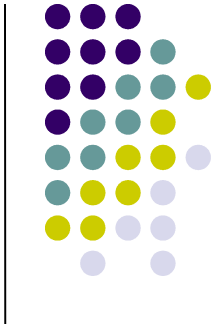


CoDoNS



- In CoDoNS, each object is replicated at some level i.
  - Object is stored on all nodes with i matching prefixes and looking it up requires at most i hops in the ring.

Performance



- Problem reduces to minimizing the level of caching such that average lookup performance remains under some constant bound C.
  - i = 1 for all objects yields O(1) lookups! Obviously not such a great idea.
- Lots of math leads us to a single, closed form solution to the optimization problem.

$$x_i = \frac{d^i(\log N - C)}{1 + d + \ldots + d^{\log N - 1}}^{\frac{1}{1-\alpha}}$$

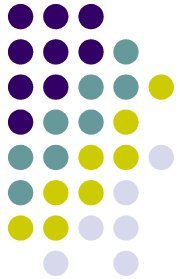
$$d = b^{\frac{1-\alpha}{\alpha}}$$

b = base of DHT

N = number of nodes in ring

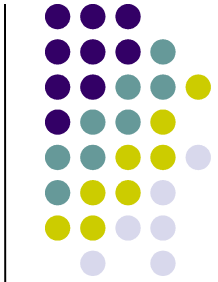
x_i = fraction of objects replicated at level i

What's the Big Deal?



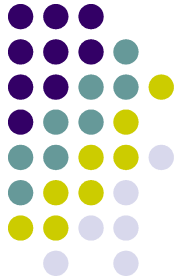
1. Provides strict guarantee of average lookup latency
   - Can achieve desired cache hit rate. C = .5 is perfectly feasible.
2. Utilizes as little bandwidth and space as possible despite constant time lookups.
3. Balances load – objects are replicated based on popularity.
4. Resilient against failures.
5. Update propagation is easy when each object's location is described by a single level i.

Implementation Details



- Namespace management and query resolution are two different things.
  - We improve the latter and don't touch the former.
  - For name owners, CoDoNS is insert, delete, and update and nothing more. For end users, CoDoNS is a resolver.
  - Name hierarchy, administrative policies, politics, domain sales? We're agnostic.
- CoDoNS serves the exact same namespace with the exact same interface.

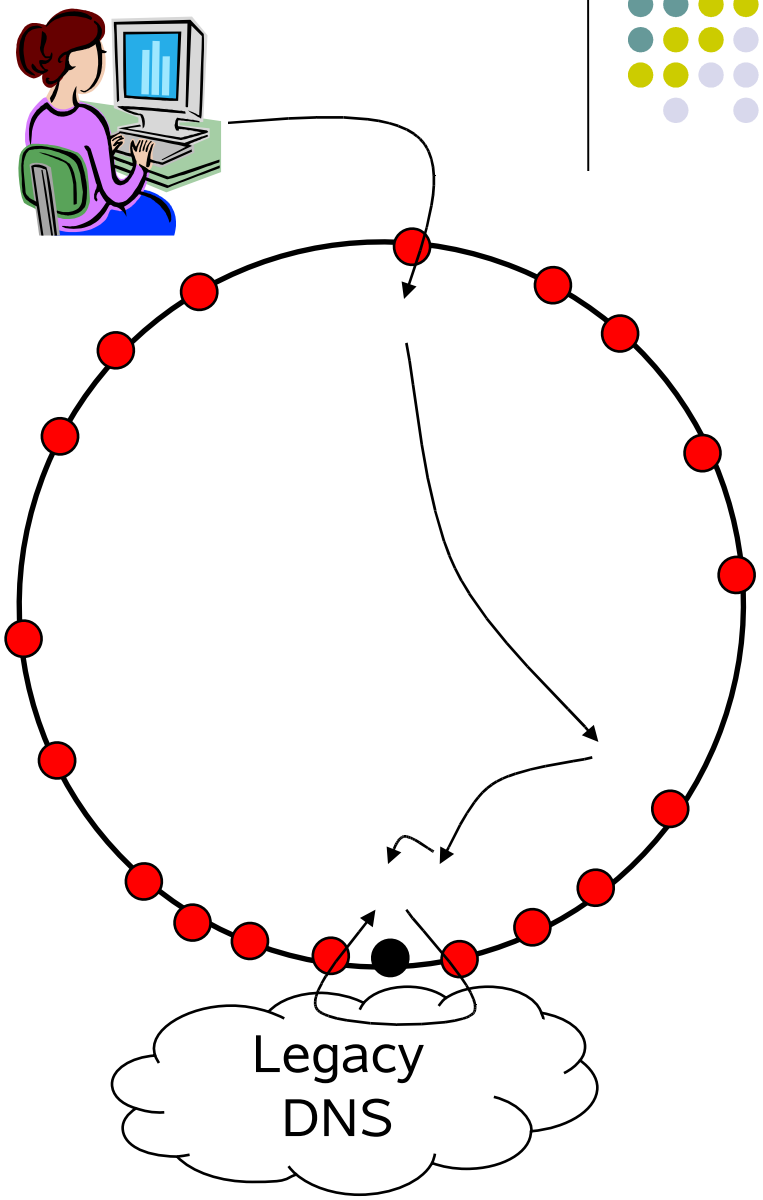
Implementation Details



- Caching and authoritative services
  - Caching: All names not explicitly inserted are resolved via traditional methods. Once inserted, only a single home node polls legacy DNS for updates. No undue stress is put upon existing systems.
    - Initial insertion is checked for validity at multiple locations and then signed by our private key.
  - Authoritative: Domain is delegated to nsXX.codons.net
- Security Model
  - If you believe in DNSSEC, you (should) believe in CoDoNS.
  - Malicious or compromised nodes are not an issue unless the private key is stolen in which case you probably have bigger problems.

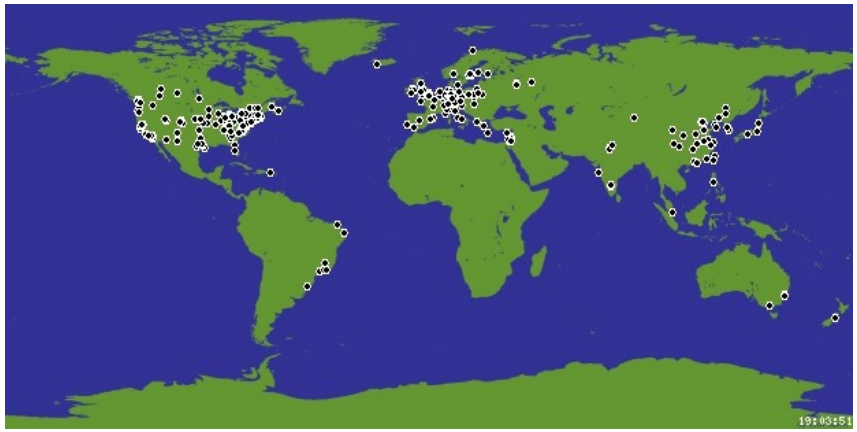
Bottom Line

- Serves two functions
  - Caching / safety net for legacy DNS
  - Authoritative name service
- Name hierarchy independent of server heirarchy
- Name delegations independent of server requirements
- Fully transparent and compatible with legacy DNS

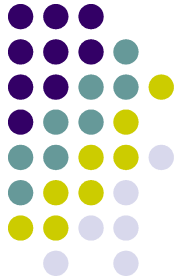


Our Current Deployment

- PlanetLab
  - Global Consortium for “developing, deploying, and accessing planetary-scale services.”
  - Translation: Access to many high powered boxes on fat pipes at universities and research labs.
- 700+ nodes at 300+ sites.

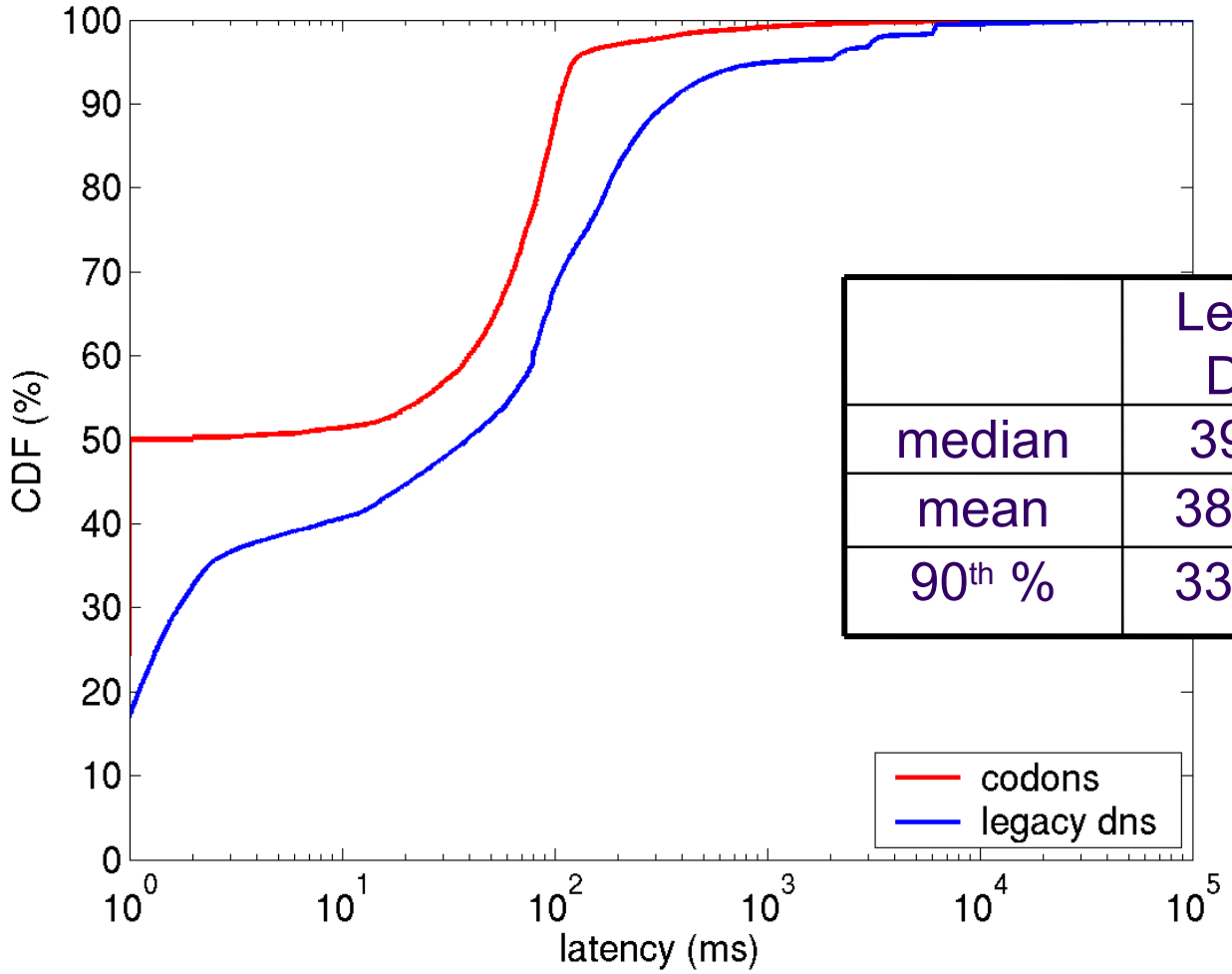


Real World Performance



- Trace from MIT nameservers.
  - 12 hours, December 2000.
  - ~300k queries, ~50k domains.
- Most significant result: very fast average time for lookups!

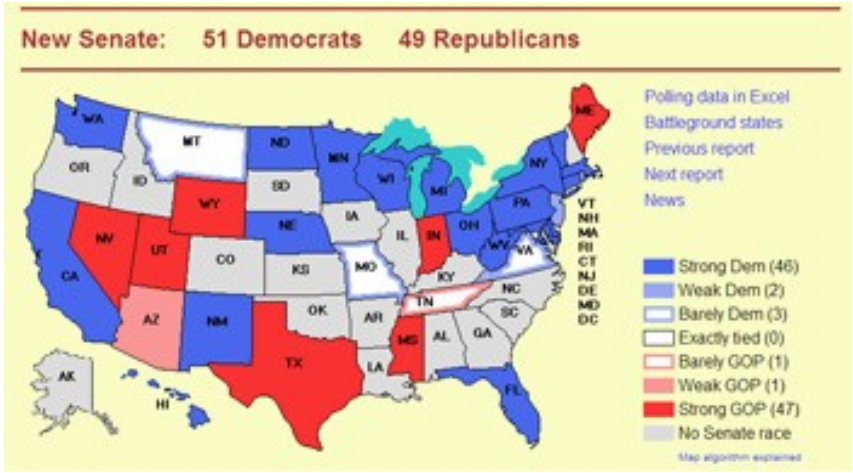
Fast Lookups



	Legacy DNS	CoDoNS
median	39 ms	2 ms
mean	382 ms	199 ms
90th %	337 ms	213 ms

Electoral-Vote.com

Andy Tanenbaum's side project – a demanding first customer. In 2004, experienced malicious DDoS on nameservers. Back for revenge.



- Peak: Nov 1st-8th
  - Over 1 mil queries per day.
  - Nobody bothered/dared to DDoS.
  - No downtime.

Conclusion

- Proactive caching based on analytical models derived from query distribution leads to strong bounds on lookup times.
  - Low latency, efficient updates, self-configuring, real redundancy, etc.
- We're looking to partner with ISPs and DNS providers to host CoDoNS nodes.
- We're willing to host or backup your DDoS prone names.
- Any questions?
  - http://www.cs.cornell.edu/people/egs/beehive/
  - {ilya, egs}@systems.cs.cornell.edu