Serving DNS using Chord (or Pastry, or ...) Good idea, bad idea?

Russ Cox
Athicha Muthitacharoen
Robert Morris

rsc,athicha,rtm@pdos.lcs.mit.edu

IPTPS ’02
March 8, 2002

Motivation

Before DNS, there was a global `hosts.txt`.

DNS is an attempt to distribute `hosts.txt` but:

- Configuring DNS properly is tedious and time-consuming. Everyone has to be a DNS admin.
- I can’t have a domain without finding someone to serve my names.
- Easy to have locally-correct-but-globally-wrong configurations.

Peer-to-peer lookup services solve some of these problems:

- Organization, replication, and much configuration done by P2P layer.
- I don’t need to run my own 24/7 DNS server.
- Lack of hierarchy avoids half-broken configurations.

Motivation, II

‘‘Distributing authority for a database does not distribute corresponding amounts of expertise. Maintainers fix things until they work, rather than until they work well, and want to use, not understand, the systems they are provided. Systems designers should anticipate this, and try to compensate by technical means.’’

Mockapetris and Dunlap, 1988

DNS

IP-based DNS authentication

- Trust IP layer; believe what you hear from those you trust.
- Root servers are known.
- Servers delegate authority to other IP addresses.
- Verification induces lookup algorithm.

DNSSEC-based DNS authentication

- Trust public key crypto; believe what is signed by those you trust.
- Root server keys are known.
- Servers delegate authority to other keys.
- Verification leaves lookup completely unspecified.

With DNSSEC, can explore other lookup methods.

Peer-to-peer!

DNS with P2P Hash Table

Look up SHA1(name, query-type).

(Maybe walk key hierarchy, maybe store records with all relevant keys.)

Perfect match for ‘‘distributed `hosts.txt`.’’

Prototype implemented using Chord.

Evaluation: Latency

Uncached latency is enormous.

- $O(\log_2 n)$ RPCs per lookup.
- Compare to conventional DNS: typically 1 or 2 RPCs per lookup.

Cached latency a little better, not much.

Evaluation: Functionality

We have all the functionality of a distributed `hosts.txt`.

And nothing more.

No support for ‘‘ANY’’ queries.

No dynamically-generated records.

No DNS-server-side load balancing (randomization).

No DNS-server-side proximity routing (Akamai).

Evaluation: Ease of Administration

O’Reilly BIND+DNS book lists 13 common problems.

9 are arguably software deficiencies (needing to restart the server after config changes, ...).

4 are actually protocol-specific problems; we just stir them around.

> Missing subdomain server delegation → missing public key delegation
>
> Bad subdomain delegation → bad public key delegation
>
> Network outage → different network outage

1 is gone:

> Slave server can’t load zone → caching is transparent

But why trust random servers rather than run my own?

> Chicken and egg problem or fundamental worry?
>
> Comparison with IP routing? [XXX]

Evaluation: Robustness

Robustness inherited from P2P layer:

- Better fault tolerance against DoS attacks.
- Hard to target specific records — network routes around damage.
- No central points of failure.

Then again:

- DNS DoS attacks are so 1998.
- The central DNS servers have very good reliability.
- New DoS: inserting lots of bogus records.
- How do we tell DoS from normal usage?
  Why can’t I have a billion records for my personal domain?

Conclusions

If it were 25 years ago, P2P DNS might be worth considering.

Current DNS is more than just distributed `hosts.txt`.

Some benefits, mostly drawbacks.

Addressing a non-issue: more benefit from fixing BIND’s UI.

Better the devil you know than the devil you don’t.