DNS and Distributed Hash Tables
Not Quite Perfect Together

Russ Cox, Athicha Muthitacharoen, Robert Morris
rsc,athicha,rtm@lcs.mit.edu

Massachusetts Institute of Technology
Laboratory for Computer Science

Overview

The experiment: redo DNS in a peer−to−peer manner.

The result: not as good as conventional DNS.

The talk: what we expected, what we learned.

- *Draw general conclusions about peer−to−peer systems.*
- *Directions for future research.*
- *Or guidelines for selecting peer−to−peer apps.*

Motivation

Before DNS there was a global *hosts.txt*.

DNS is an attempt to distribute *hosts.txt*, but:

- *Everyone has to be a DNS admin.*
- *I can't have a domain without a 24/7 DNS server.*
- *Locally correct, globally wrong configurations.*

P2P lookup systems might fix these:

- *Organization, replication, much configuration handled by the P2P layer.*
- *I don't need to keep a 24/7 server up.*
- *Lack of hierarchy avoids half−broken configs (?)*

Motivation, II

Mockapetris and Dunlap, 1988:

‘‘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.’’

P2P as technical compensation.

DNS: Authentication and Lookup

Original DNS uses IP−based authentication:

- *Trust IP layer; believe what you hear from trusted sources.*
- *Root server IP addresses are known.*
- *Servers delegate to other IP addresses.*
- *Verification induces lookup algorithm.*

DNSSEC uses crypto−based authentication:

- *Trust public key crypto; believe what is signed by trusted sources.*
- *Root server public keys are known.*
- *Servers delegate to other public keys.*
- *Verification leaves lookup unspecified.*

DNSSEC lets us explore alternate lookup methods:
peer−to−peer dynamic hash tables

DNS using P2P Hash Table

Look up SHA1(*name*, *query−type*).

- *Maybe walk key hierarchy to verify.*
- *Maybe store records with proof chain.*

Exactly a distributed *hosts.txt*.

Prototype implemented using Chord.

Evaluation: Latency

Uncached latency is too big: *O*(log *n*) RPCs

- *Chord: log base 2*
- *Pastry, Kademlia: log base 16*
- *DNS: log base 1,000,000(?)*

If we are going to run P2P apps that need low latency, we need a better story.

Evaluation: Robustness

Robustness inherited from distributed hash table:

- *Better tolerance of DoS attacks.*
- *Hard to target specific records.*
- *Network routes around damage.*
- *No central points of failure.*

DNS is fairly robust already.

- *Root servers highly replicated.*
- *Root servers have very good uptime.*
- *DNS DoS attacks are so 1998.*

New DoS: insert lots of records.

- *Why can't I have a billion records for my personal domain?*
- *Quotas require authentication.*
- *Maybe everyone will play well with others.*

Evaluation: Network Outages

Suppose MIT gets cut off from the Internet.

In old DNS:

- *MIT can still look up and connect to MIT hosts.*
- *MIT cannot look up nor connect to Internet hosts.*
- *Internet cannot look up nor connect to MIT hosts.*

In a P2P DNS:

- *MIT can't look up but can connect to MIT hosts.*
- *MIT can look up but can't connect to Internet hosts.*
- *Internet can look up but can't connect to MIT hosts.*

Can we fix this without another layer?

Evaluation: Functionality

All the functionality of a distributed *hosts.txt*, but:

- *No dynamically−generated records.*
- *No support for ‘‘ANY’’ queries.*
- *No server−side load balancing (randomization).*
- *No server−side proximity routing (Akamai).*

Distributed hash tables aren’t enough.

- *Don’t want to assume client−side functionality.*
- *Don’t want to deploy new servers for each application.*
- *Active peer−to−peer networks?*

Evaluation: Administration

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

- *9 are arguably software deficiencies.*
- *3 are protocol specific, but we just stir them.*
  missing subdomain 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

We're addressing a non−issue: fix BIND's UI instead.

Evaluation: Administration, II

Don’t have to run my own 24/7 servers.

- *Have to trust servers run by others.*
- *No one to blame when things go wrong.*

Why trust servers run by others?

- *Fundamental P2P problem?*
- *Chicken and egg problem?*

Can we arrange incentives for running servers?

- *IP routers run by ISPs: financial incentives.*
- *Tarzan: plausible deniability for anonymity.*

Conclusions

20 years ago this might have been worth considering.

- *DNS has evolved into more than just a hosts.txt*.

In order for generic peer−to−peer technology to replace customized apps like DNS, it needs:

- *Lower latency.*
- *Protection against insertion DoS.*
- *Choice of functionality for network outages.*
- *More than just distributed hash tables.*
- *High confidence in the network.*
- *Generic incentives for people to run servers.*

Alternate Ending

Peer−to−peer systems have fundamental limitations and simply aren’t appropriate for apps that need:

- *Lower latency.*
- *Protection against insertion DoS.*
- *Choice of functionality for network outages.*
- *More than just distributed hash tables.*
- *High confidence in the network.*
- *Generic incentives for people to run servers.*