Using Multiple Authoritative Vendors Does Not Work Like You Thought

DNS-OARC 42
2024-02-08
Charlotte, North Carolina, USA

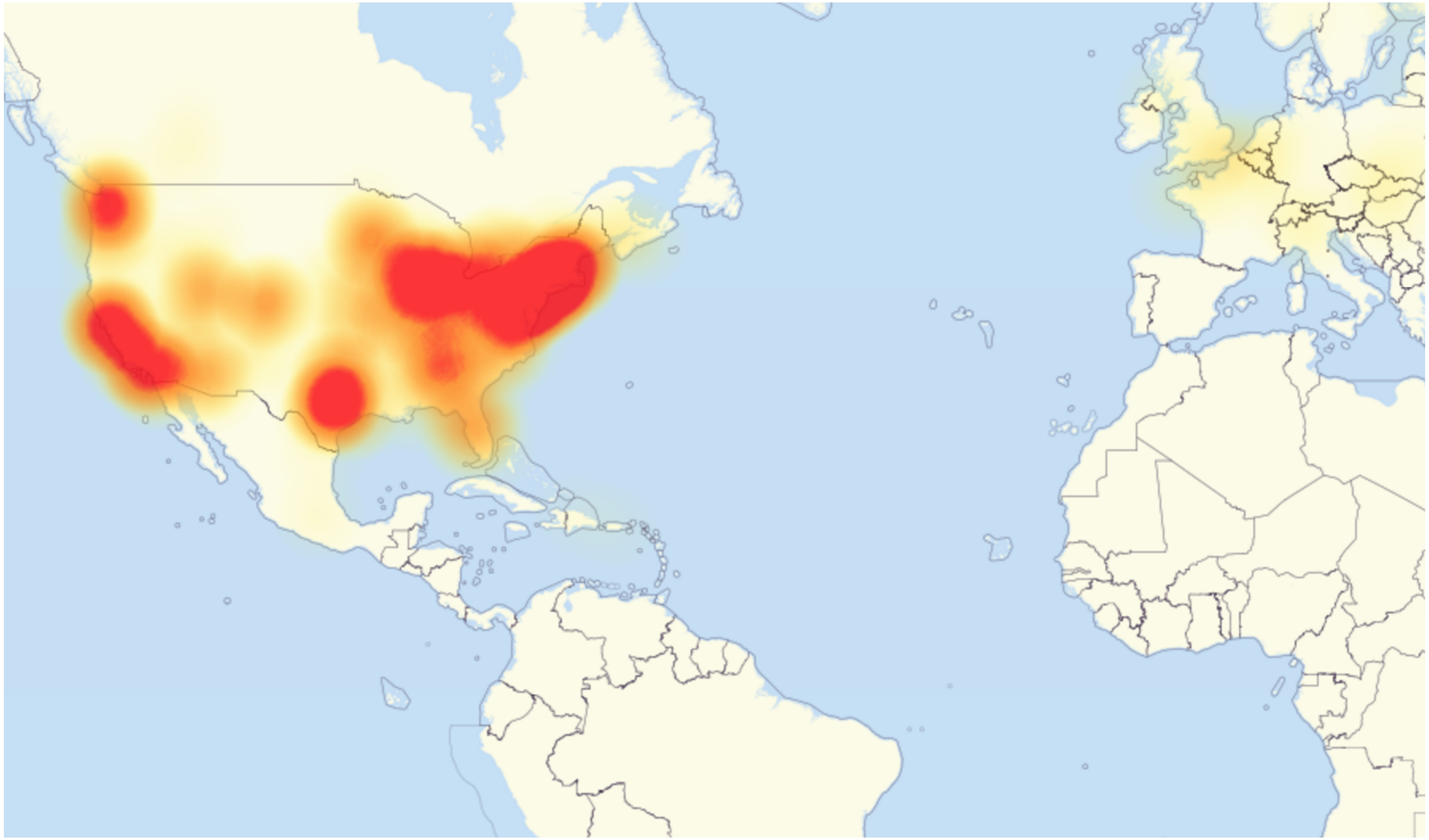
Shane Kerr <shane.kerr@ibm.com>
Back-end Engineer

NS1, an IBM Company

The Trouble with Using a Single Authoritative Edge Provider

Everyone can have a bad day.

On 2016-10-26 there was a massive Distributed Denial of Service (DDoS) against Dyn, causing widespread Internet outages.



Engineering Principle: Remove Single Points of Failure (SPoF)

Redundancy makes total failure less likely.

This has to be done properly in order to actually actually reduce chance of failure.

Redundancy is a commonly used approach.

For example RAID, server clustering, and of course within the DNS itself.

Redundancy increases complexity.

This leads to fun new kinds of system failure.

DNS Authoritative Server Redundancy: Theory

Each zone should have multiple NS records.

Each NS record has one or more addresses.

If one address does not work, a DNS resolver will use another.

Typically a resolver will try to go to the address that answers the most quickly. Methods for this vary wildly, but basically slower addresses should get fewer queries.

If one address does not work, a DNS resolver will use another.

A server that does not answer at all should get very few queries.

SPoF and Authoritative DNS Vendors

To remove a DNS vendor as a SPoF, add redundancy by adding one or more vendors.

“Common sense”, right?

Definitely recommended since at least 2016.

But Does It Work?

Meten is weten. - Dutch saying

”To measure is to know.”
”To measure it to know about.”

Let’s Experiment! 🧑‍🔬🔬📡

Experiment: Authoritative Setup

Create two separate edges.

The first is an NS1 dedicated network. Uses a route controlled by NS1.

The second is on Route53. This is because we already had an account there for other reasons, not due to any careful analysis or other motivation.

Setup zone.

No DNSSEC.

All infrastructure records have a long time-to-live (TTL).

This includes SOA, NS, A, and AAAA records.

No changes were made to these during experiment.

They should mostly be cached.

Target is a single wildcard TXT record.

Short TTL, so not cached.

Replies with different answer for NS1 edge and Route53 edge, for debugging.

Experiment: Client Setup

Used RIPE Atlas with around 200 probes.

Probes chosen by downloading list of all available probes and picking at random before experiment started.

80% of the probes were in the USA, the remaining 20% from around the world. This was used to roughly emulate traffic for an American-based company.

Make a DNS new query every minute.

Using network resolver for Atlas probe.

This usually means that whatever is configured in DHCP for the network hosting the Atlas probe is used.

All answers stored in RIPE Atlas.

This is how RIPE Atlas always works, but it is good for this type of research.

Experiment: Execution

Withdraw BGP for one of the edges.

We used the NS1 edge, because we control that.

Simulates the simplest failure mode.

See what happens.

Experimental Results: Expectation

Short period with lots of failures.

Half of queries should fail after BGP withdrawal, since about half of resolvers should use the now-invalid address prefix.

Steady state with some small amount of failure.

Resolvers will stop querying the failing servers.

Some periodic checking will cause failures.

Experiment: Actual Results

Everything works! 🎉

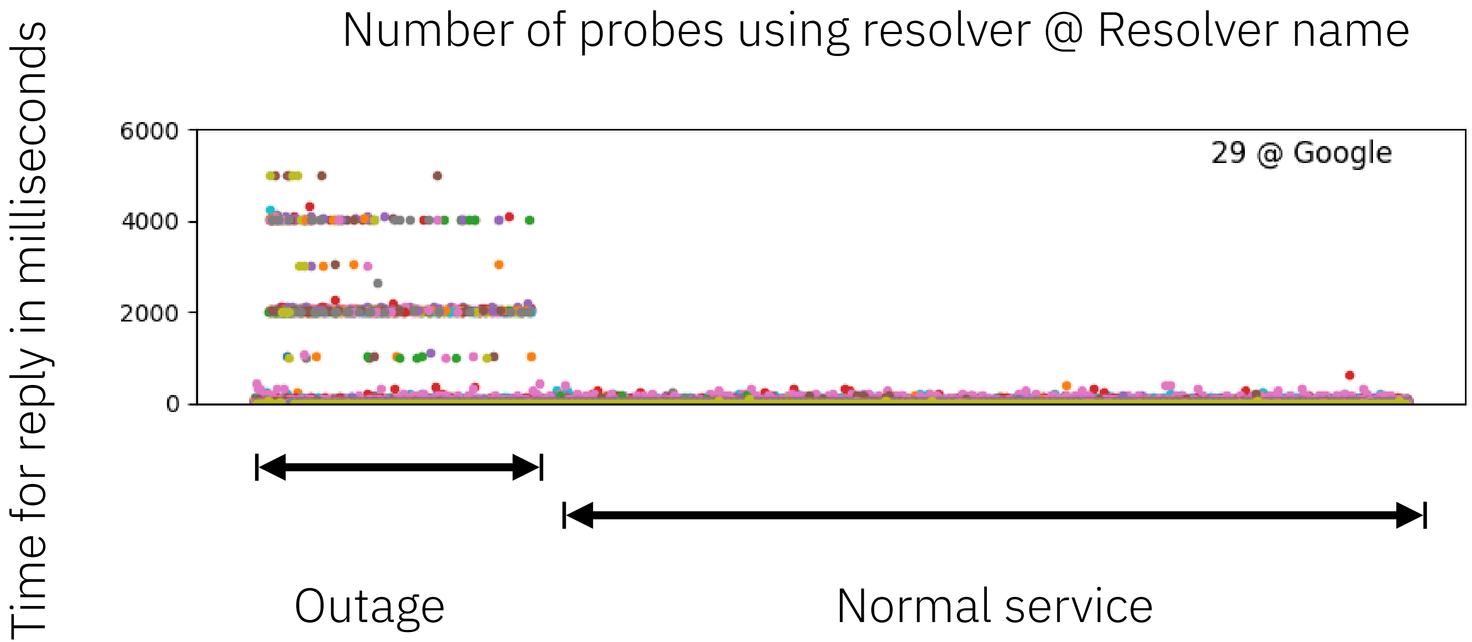
Lots of stuff really slow!!! 😖

Looking at the data, it is pretty clearly due to dropped packets and then retries.

Banding of times around 500 milliseconds, 1000 milliseconds, and so on, depending on the particular resolver.

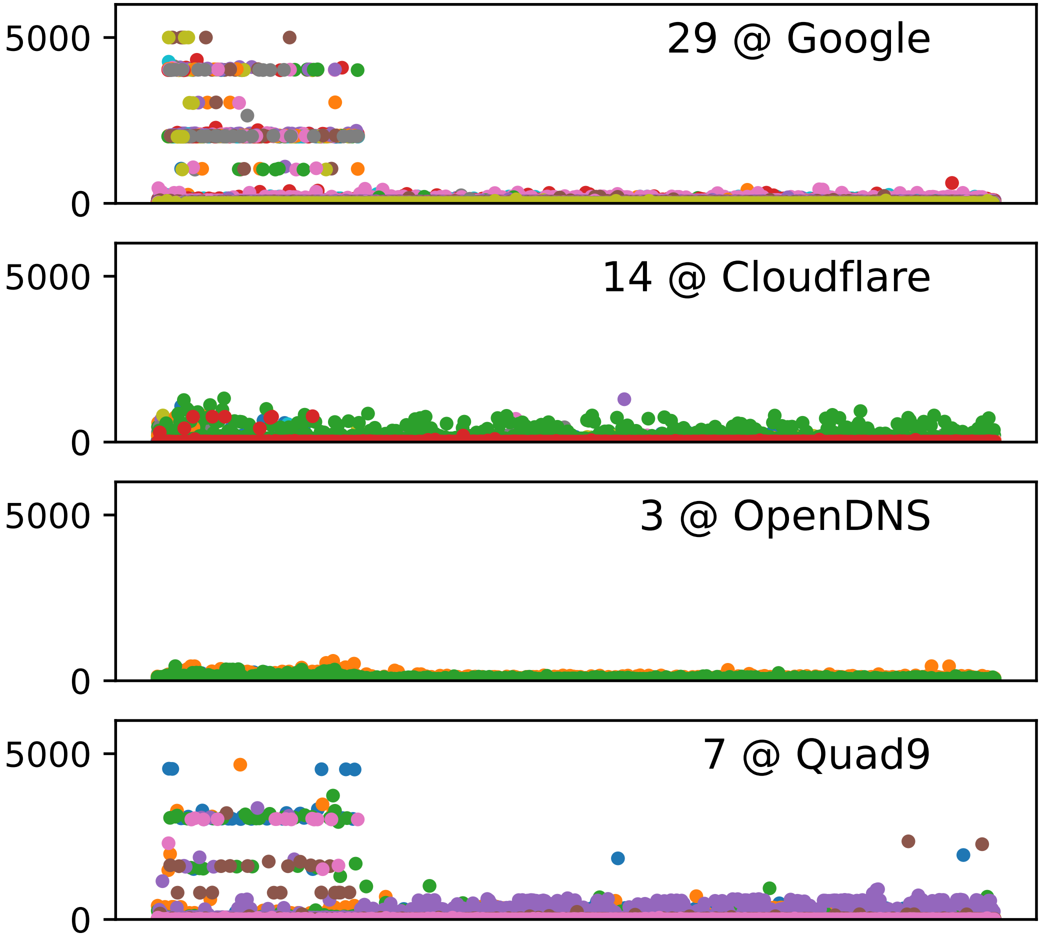
Slowness continued as long as one edge was down.

Performance quickly recovered when the edge brought up again by advertising the BGP route.

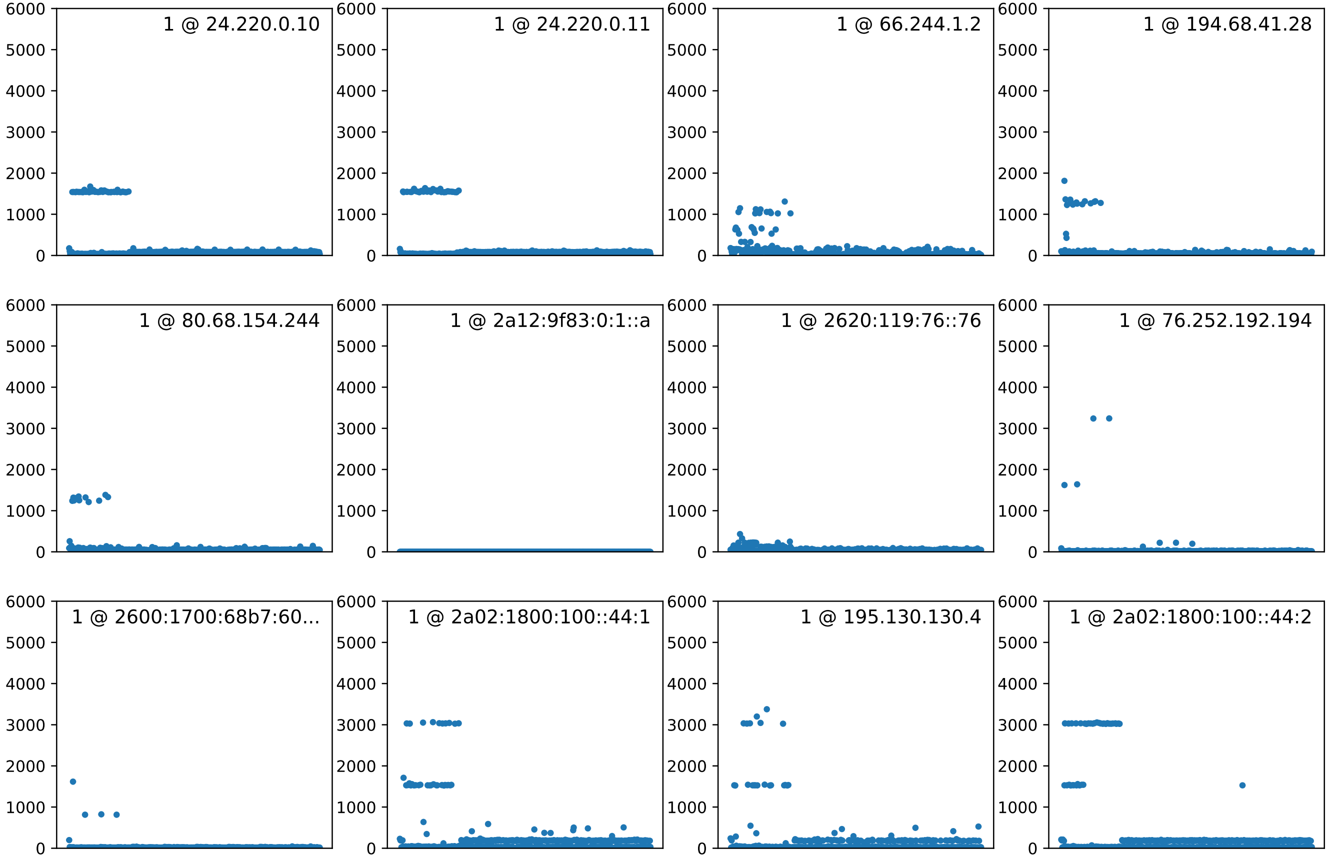


Each colored dot is a query.

The color of the dot represents a specific RIPE Atlas probe.


29 @ Google
14 @ Cloudflare
3 @ OpenDNS
7 @ Quad9
5000
0
5000
0
5000
0
5000
0

Other Resolvers in Public Space



Experimental Results: Why?

Some public DNS do not share cache.

Difference resolver instances behind a load balancer will not learn about unresponsive authoritative servers.

Shared cache not absolutely necessary. They could also use IP-based hashing to direct clients to a consistent resolver.

Possibly there is no server round-trip-time (RTT) tracking at all in some clients?

Maybe there is some other authoritative server RTT selection algorithm issues?

Possibly no caching of authoritative server timeouts? Or only remember this for a very short time?

Experimental Results: Don’t Panic

🎊🎆🎉

Everything works.

Resolvers *do* get answers, just with a delay.

🆗✔️👌

If you have a mostly-static zone, you are probably fine using multiple authoritative edge networks.

Pick a reasonable TTL and records will be cached, masking the impact of failures.

Note: Untested hypothesis!

☠️☣️☢️

For zones doing traffic steering or other dynamic replies… you are probably not fine.

Zones using these techniques are often used by applications where latency is a problem.

In this case, taking between ½ and 3 seconds is ***terrible***!

Analysis:
Can We Fix It?

We see problems in lots of resolvers, not just one or two public resolvers.

There are lots of non-public resolvers with poor behavior in this failure mode.

These appear to be diverse, rather than exactly the same problems. This implies multiple different code bases.

No, we cannot fix it.

At least, not easily.

And also not quickly.

Analysis: What Can Be Done?

Can we work around it in DNS?

We probably cannot use the DNS itself.

DNS will ultimately always fall back on resolver behavior.

Can we work around it at the network layer?

We might be able to use routing tricks, for example advertising covering prefixes answering the addresses by another vendor.

These add a lot of complexity, and cooperation between vendors, so seems likely to decrease reliability.

Specific applications could use multiple names.

You can use different target domains, each run by a different vendor. This prevents any shared-fate in resolution of any given name.

Science Always Yields: Ideas for Further Research

This is literally the simplest failure scenario.

Anywhere further up the tree can also fail in exciting ways!

Partial failures are more likely than total failures.

Failures of specific functionality are also possible (IPv4 outages, DNSSEC bugs, zone truncation, and so on).

Deep dive into specific software behavior may yield interesting results.

Poorly-performing software can be identified, and improvements proposed.

Resolver developers and operators may be convinced to change their systems.

Further work into optimizing number and layout of name servers and addresses would be useful.

This experiment indicates that too many addresses would result in some failures.

Conclusions

DNS remains surprising.

A naive approach to multi-vendor authoritative DNS may not be ideal.

It may be possible to improve redundancy, both for domain holders and for the DNS system overall.

More research on multi-vendor failures will likely yield more surprises.

References

Heatmap of Dyn outage from Wikipedia:
https://en.wikipedia.org/wiki/DDoS_attacks_on_Dyn#/media/File:Level3_Outage_Map_(US)_-_21_October_2016.png

RIPE Atlas measurement:
https://atlas.ripe.net/measurements/62455258/