Happy Eyeballs for the DNS

Geoff Huston,
George Michaelson

APNIC Labs
October 2015

Recap: "Happy Eyeballs"

Plan A:

If you are Dual Stack and the service you are attempting to connect to is Dual Stack then try to connect using V6 first, and if the connection attempt fails then try using V4

> Which “naturally” propels the V6 transition – as more clients and services support Dual Stack then more transactions will shift to use V6

Recap: "Happy Eyeballs"

But Connection Failure took forever:

- Windows: 21 seconds
- BSD: 75 seconds
- Linux: 189 seconds

So what we wanted in the Web was a “fast fail” to keep the eyeballs on the content

Recap: "Happy Eyeballs"

Plan B:

If you are Dual Stack and the service you are attempting to connect to is Dual Stack then try to connect using both protocols, but give V6 a (small) head start

> The V6 SYN is typically given a head start of 300ms over the V4 SYN, and the first protocol to complete the TCP handshake is used for the ensuing session

Happy DNSballs?

```
00003.y.dotnxdomain.net.        IN      NS      ns1.00003.y.dotnxdomain.net.
ns1.00003.y.dotnxdomain.net.    IN      A       162.223.8.90
                                IN      AAAA    2607:fc50:1001:9500::2
```

This zone is served by an authoritative name server that has both a V4 and a V6 address

- How should a “Happy Eyeballs” DNS resolver behave?
- How do resolvers behave today?

Fast Failover in the DNS?

Plan A:

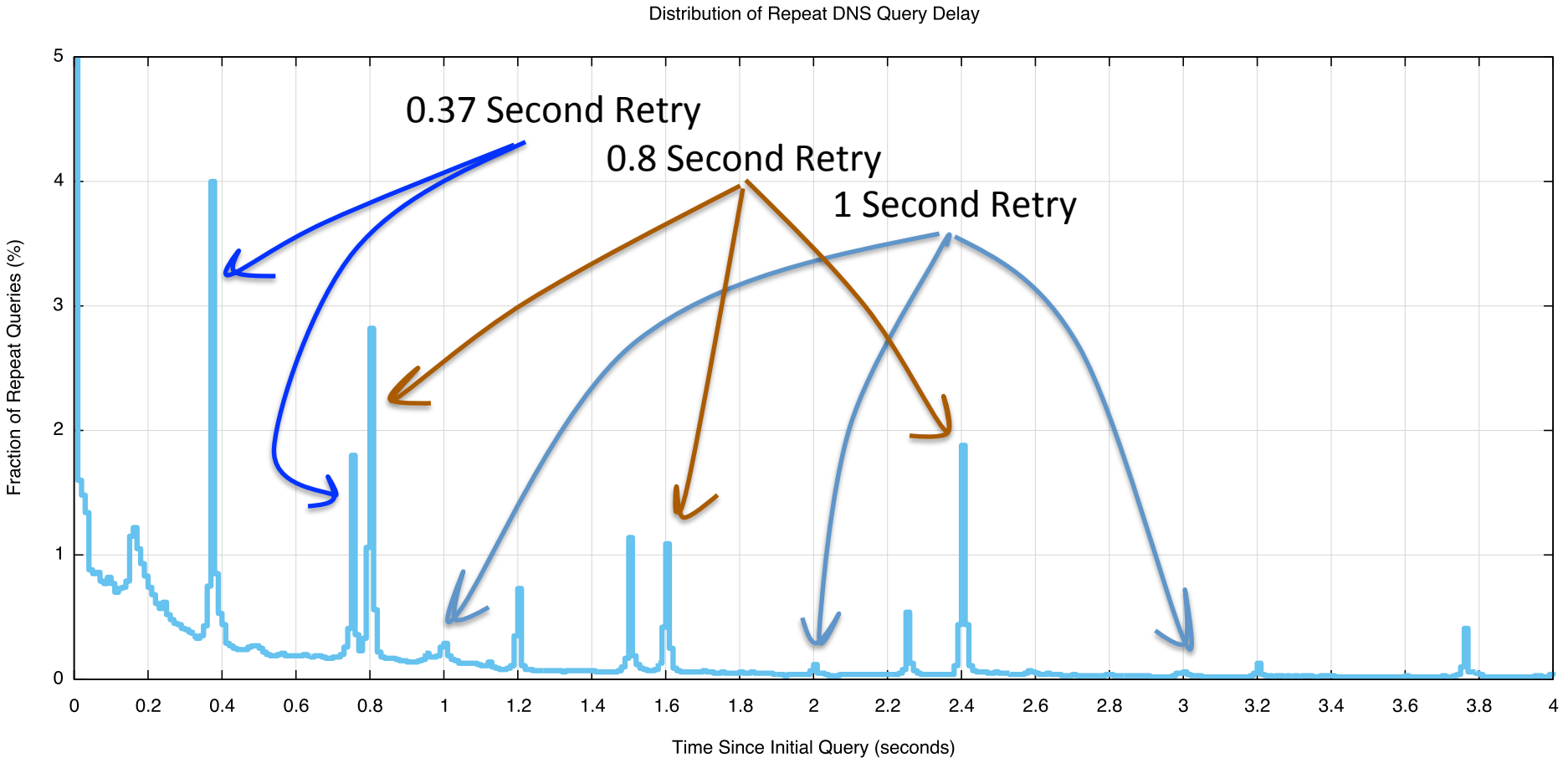
Wait for timeout?

Resolver timeout / retry algorithm is specific to the DNS resolver implementation:

RFC1034:

> “Send them queries until one returns a response.”

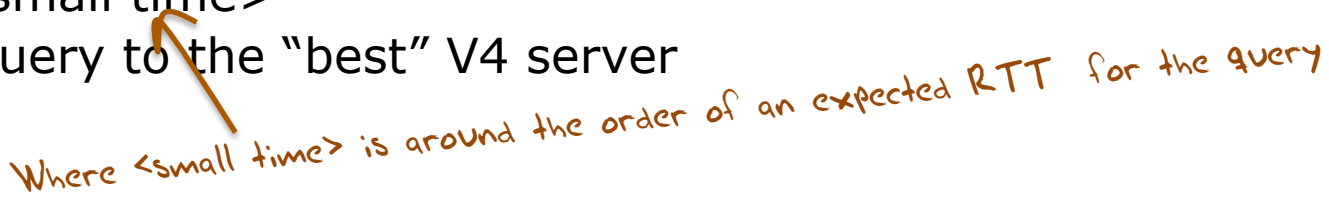
Observed DNS Resolver Re-Query Times



Fast Failover in the DNS?

If the DNS were to behave like the Web:

- assemble a sorted list of V4 and V6 addresses
- launch a query to the “best” V6 server
- wait for <small time>
- launch a query to the “best” V4 server



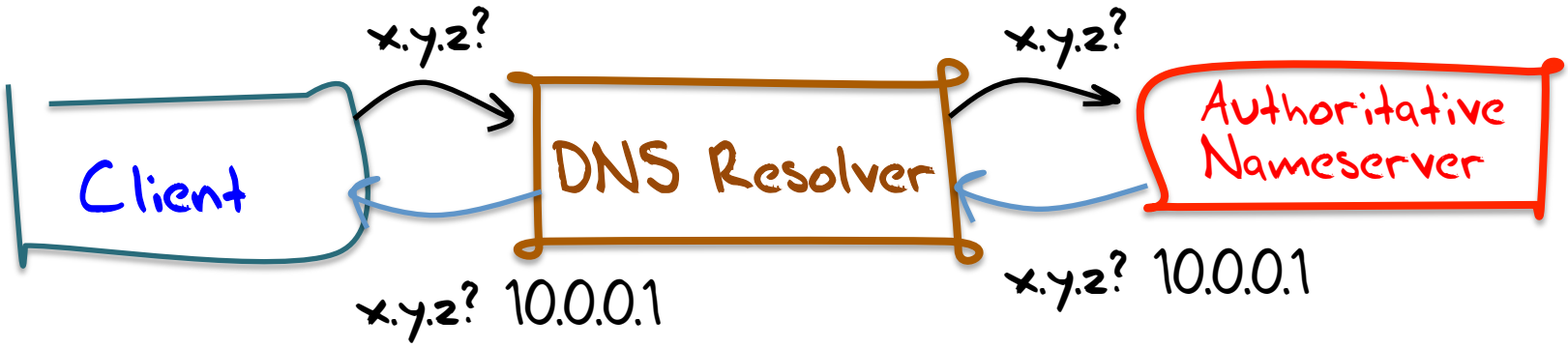
But this is not what typically happens today.

What does happen?

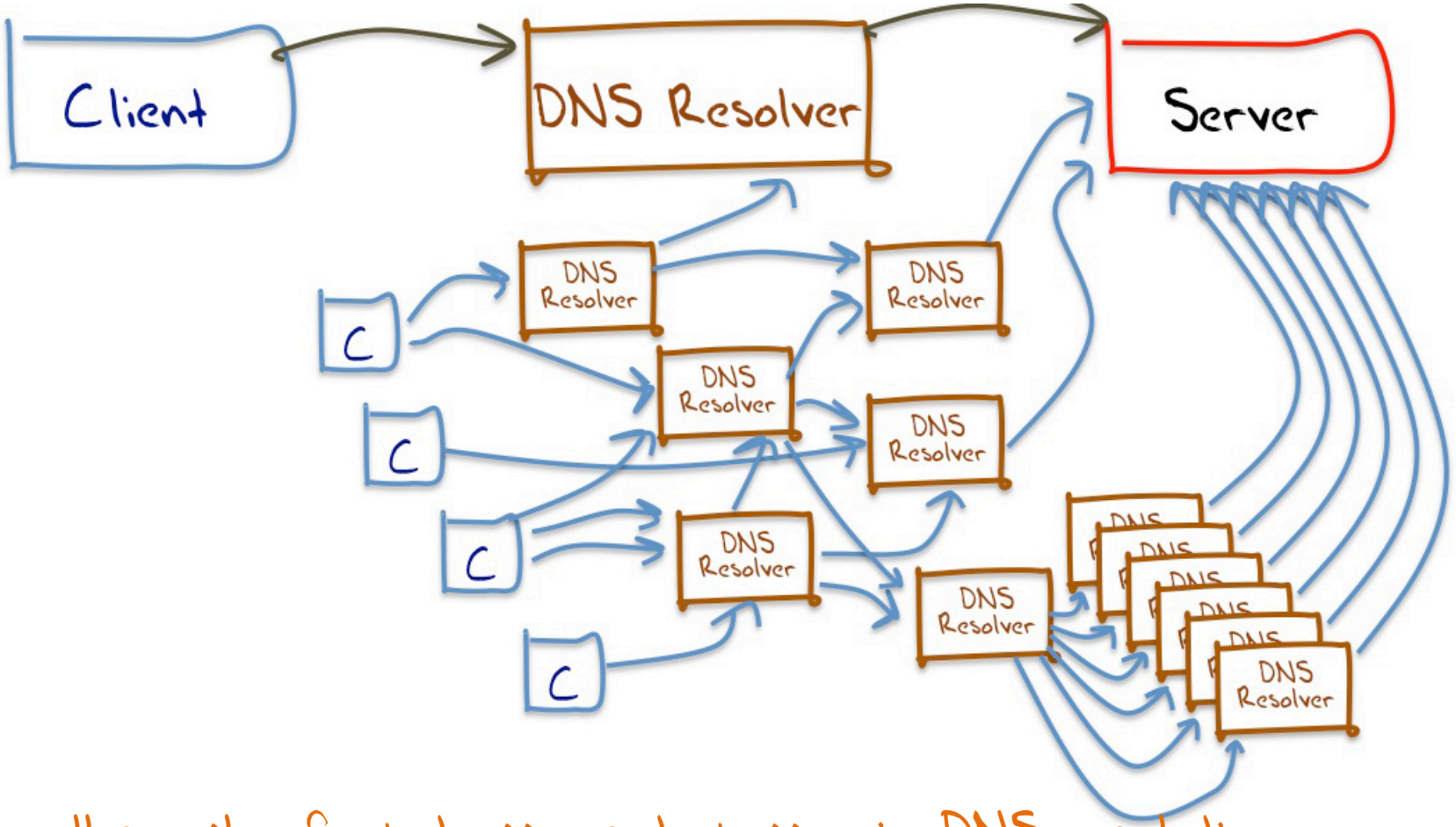
Measuring DNS Resolver Behaviour

Aside: Understanding DNS Resolvers is "tricky"

What we would like to think happens in DNS resolution!

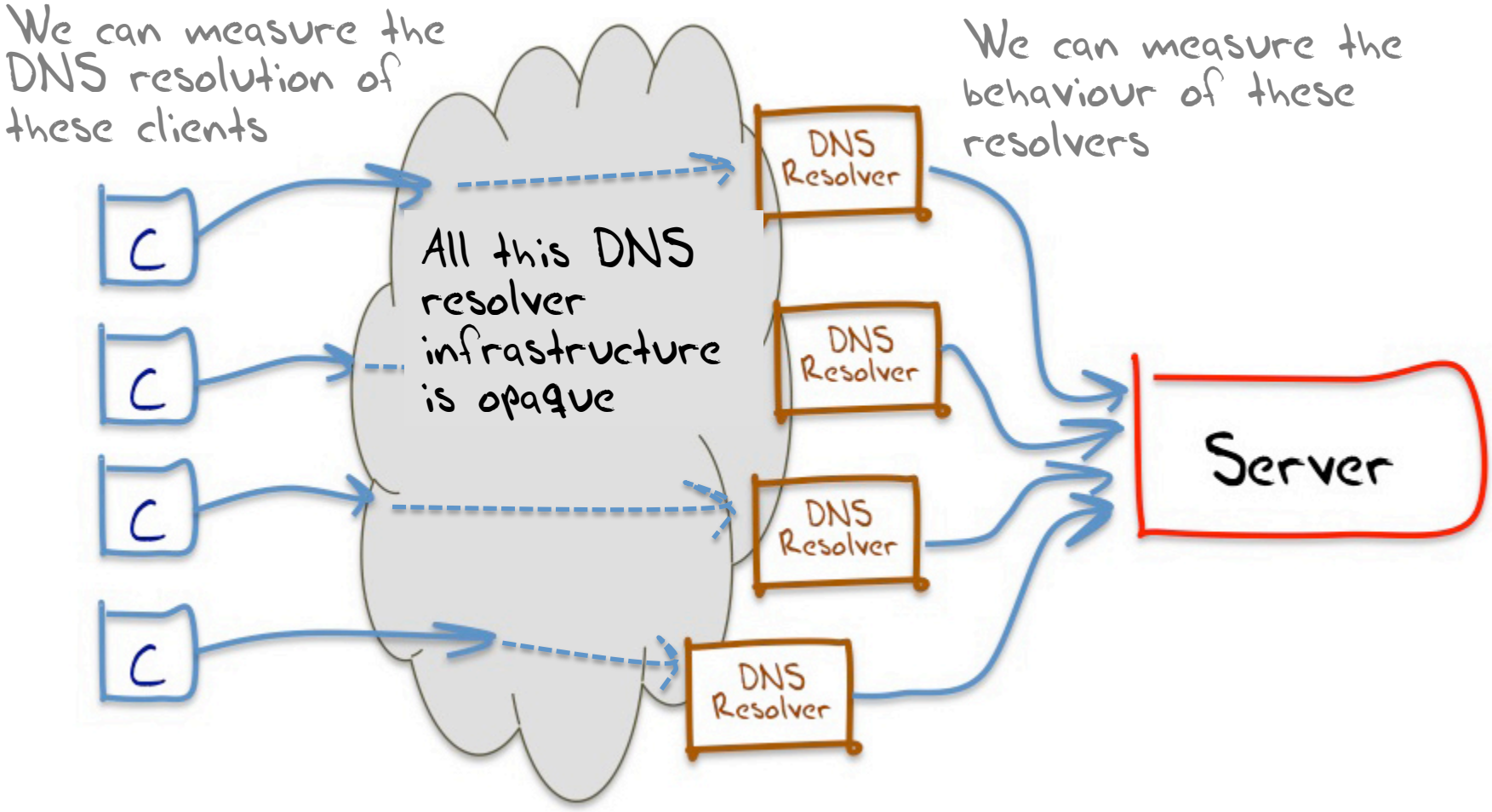


Aside: Understanding DNS Resolvers is "tricky"



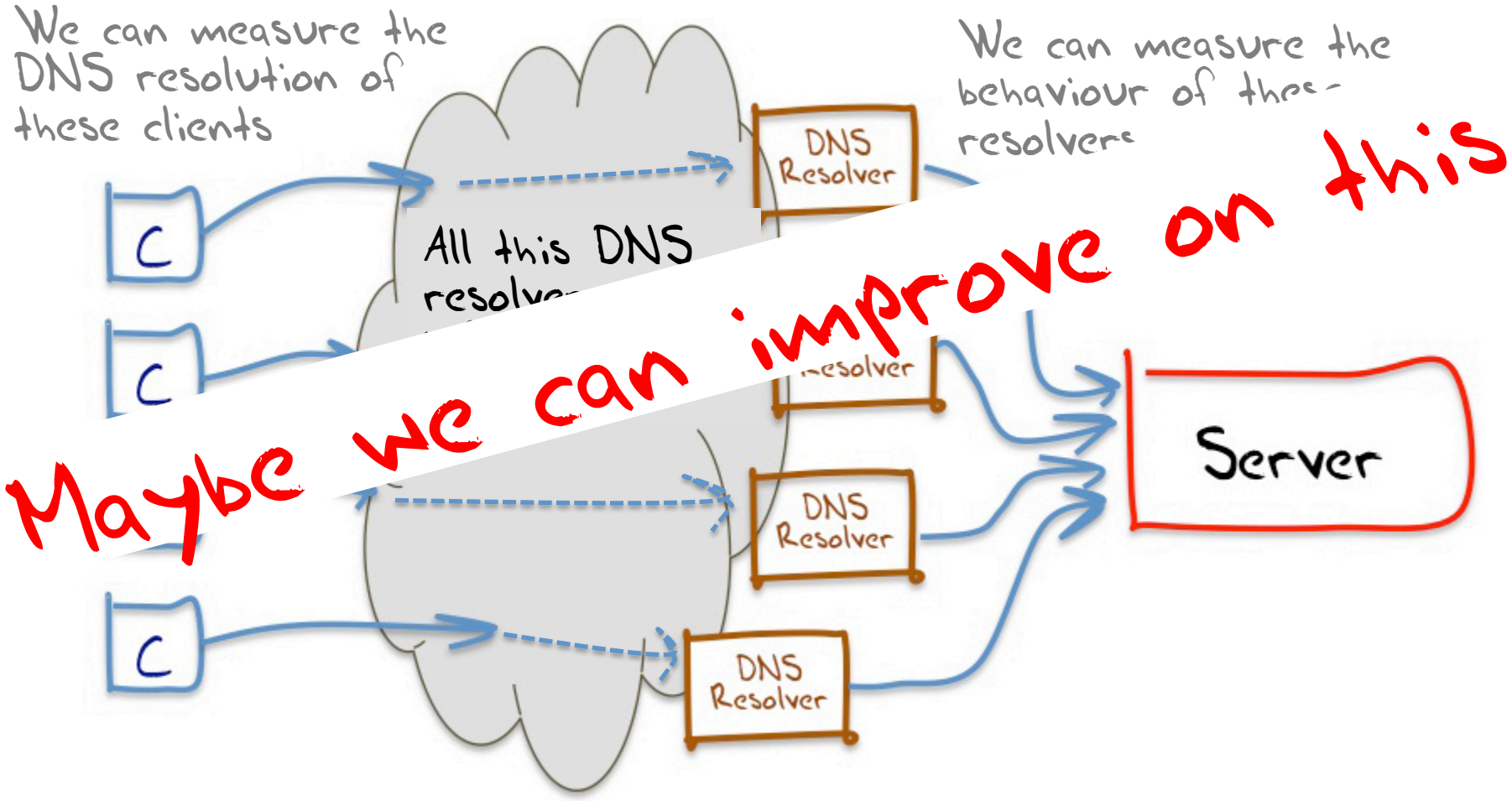
A small sample of what appears to happen in DNS resolution

Aside: Understanding DNS Resolvers is "tricky"



The best model we can use for DNS resolution in these experiments

Aside: Understanding DNS Resolvers is "tricky"



The best model we can use for DNS resolution in these experiments

Glueless Delegation

Glueless Delegation

Servers (iDNS) Attack

Florian Maury, ANSSI

May 10, 2015

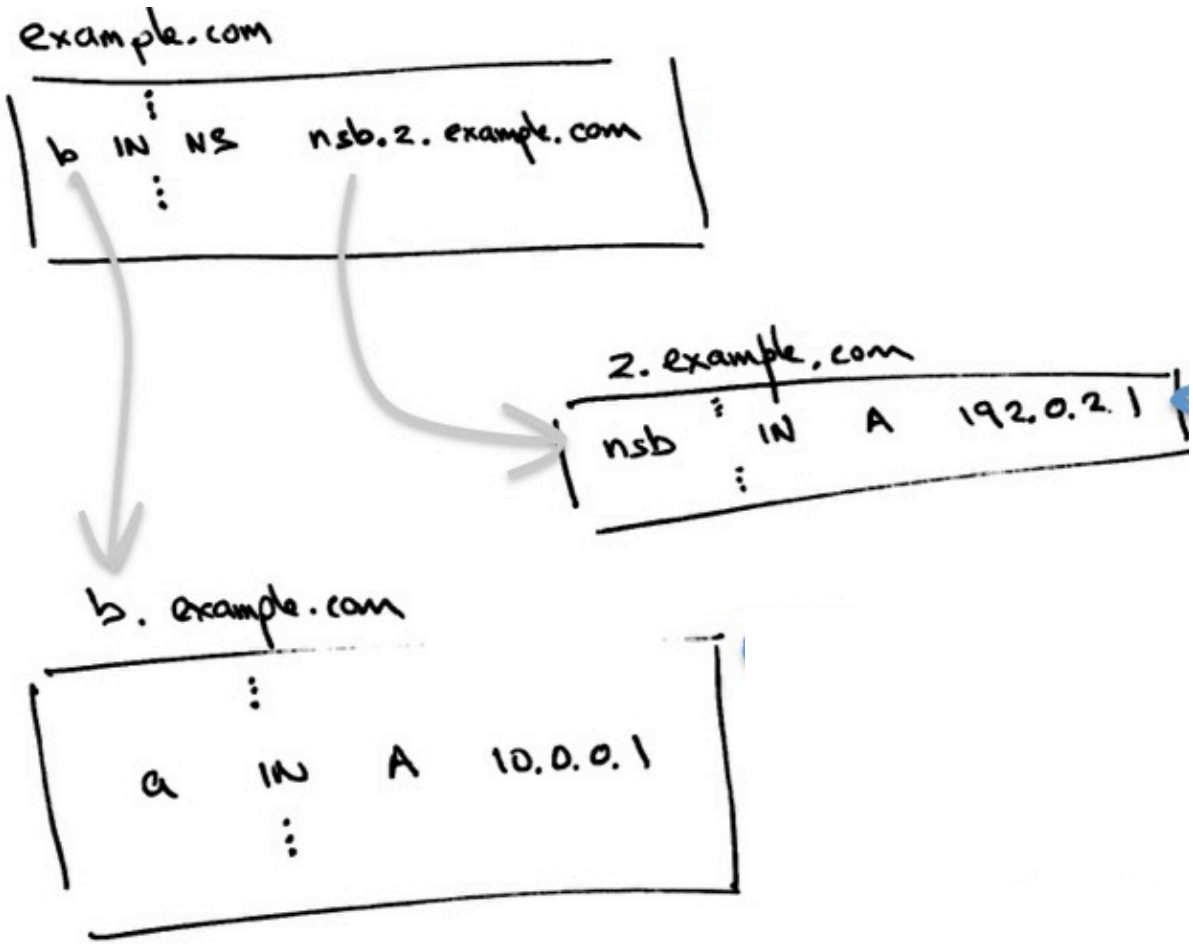
Reminders about Delegations Inner-working

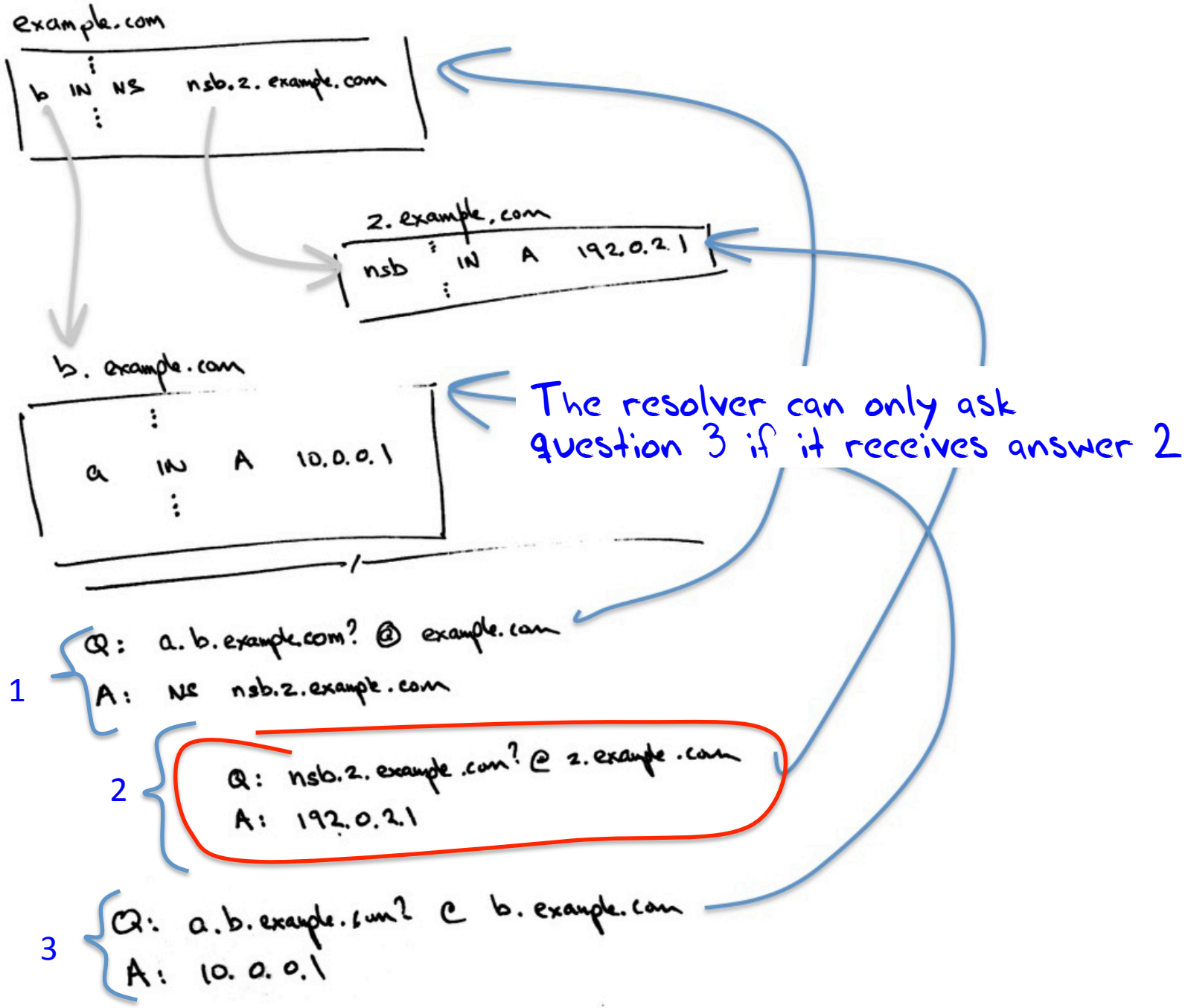
Glueless delegation example

```
;; AUTHORITY SECTION
france.fr.  IN NS ns2.produhost.net.
france.fr.  IN NS ns33.produhost.net.
```

Glued delegation example

```
;; AUTHORITY SECTION
ssi.gouv.fr.  IN NS dns1.certa.ssi.gouv.fr.
ssi.gouv.fr.  IN NS dns1.ssi.gouv.fr.
;; ADDITIONAL SECTION
dns1.ssi.gouv.fr.  IN A 213.56.166.96
dns1.certa.ssi.gouv.fr.  IN A 213.56.176.3
```


example.com
b IN NS nsb.2.example.com
2.example.com
nsb IN A 192.0.2.1
b.example.com
a IN A 10.0.0.1


example.com
b IN NS nsb.2.example.com
2.example.com
nsb IN A 192.0.2.1
b.example.com
a IN A 10.0.0.1
The resolver can only ask question 3 if it receives answer 2
1
Q: a.b.example.com? @ example.com
A: NS nsb.2.example.com
2
Q: nsb.2.example.com? @ 2.example.com
A: 192.0.2.1
3
Q: a.b.example.com? @ b.example.com
A: 10.0.0.1

Glueless Delegation

We can change the behaviour of the DNS response to the NS domain query

And we observe that the resolver has received the response by the subsequent query to the child domain

Testing V6 Preference in the DNS

We set up three domain structures:

- Glueless V4 only - NS name has only an A RR
- Glueless V6 only – NS name has only a AAAA RR
- Glueless Dual Stack – NA name has both A and AAAA RRs

And tested this in an online Ad campaign using a pool of unique names to circumvent DNS name caching in resolvers

The Experiment

25 July 2015 – 31 August 2015

43,679,222 completed experiments

Web results:

DNS V4 Only	42,515,729	97%
DNS V6 Only	16,605,301	38%
DNS Dual Stack	41,653,531	95%

38% of tests involved using DNS resolvers that were able to perform DNS queries over IPv6

The Experiment

25 July 2015 – 31 August 2015
43,679,222 completed experiments

Web results:

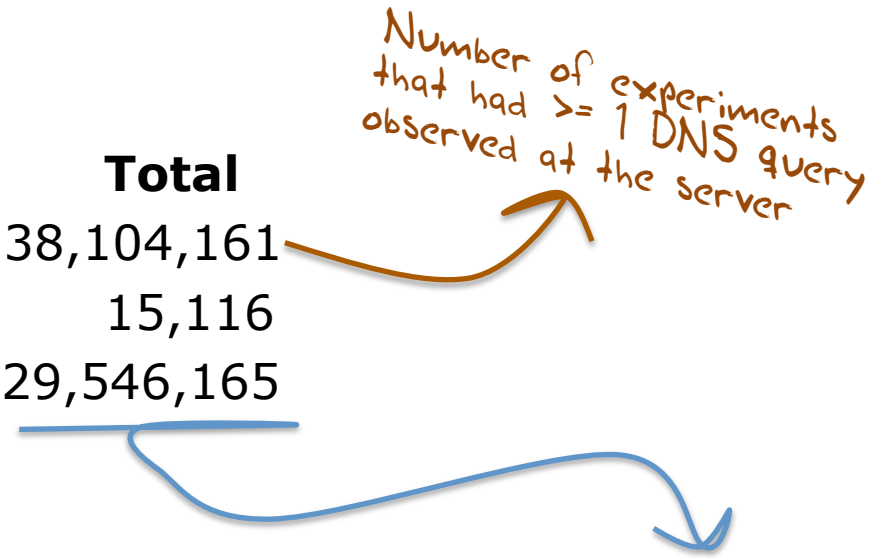
DNS V4 Only	42,515,729	97%
DNS V6 Only	16,605,301	38%
DNS Dual Stack	41,653,531	95%

38% of tests involved using DNS re
queries over IPv6

So if resolvers were "neutral" with respect to A and AAAA RRs in name servers, then we would see approximately 19÷ of queries to the Dual Stack structure take place over IPv6 – yes?

DNS Query Behaviours per Experiment

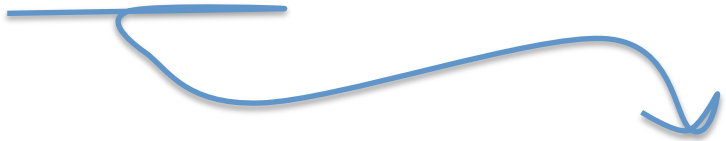
Experiment Behaviour	Total
V4 only	38,104,161
V6 only	15,116
V4 and V6	29,546,165



Yes, that's a total of 67,665,443 experiments in the DNS, while only 43,679, 222 completed the web fetch cycle (64% completion rate)

DNS Query Behaviours per Experiment

Experiment Behaviour	Total
V4 only	38,104,161
V6 only	15,116
V4 and V6	29,546,165

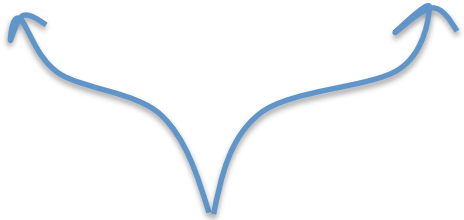


Dual Stack DNS object fetch behaviour

Exclusively used V4:	24,257,143	82%
Exclusively used V6:	1,982,312	7%
Used V4 and v6:	3,193,945	11%

DNS Queries

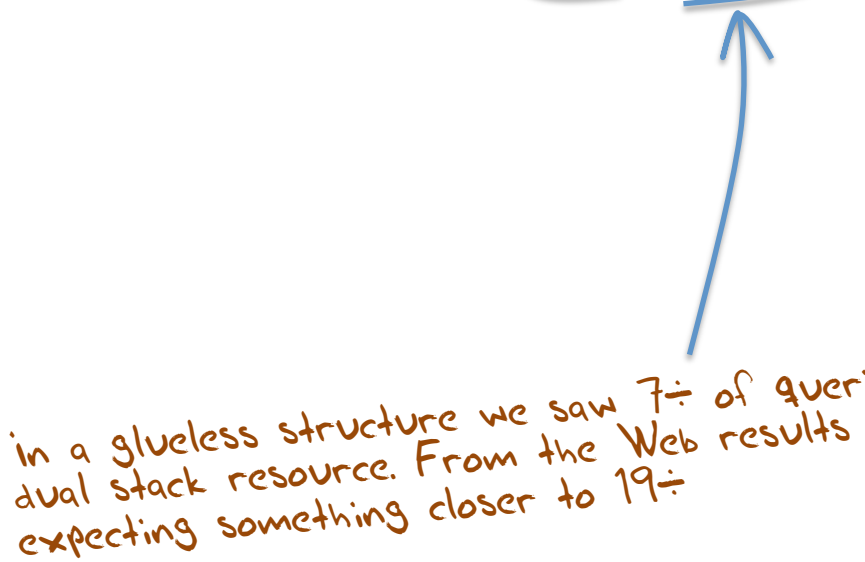
Resource	v4 Queries	v6 Queries
4-only	110,265,765	0
6-only	0	61,601,964
Dual-stack	101,897,693	7,346,050



Total number of DNS queries for A and AAAA RRs seen at the authoritative server for The DNS name

DNS Queries

Resource	v4 Queries	v6 Queries
4-only	110,265,765	0
6-only	0	61,601,964
Dual-stack	101,897,693	7,346,050



DNS Resolvers

Let’s switch from the queries make by resolvers to the visible resolvers themselves

Resolvers seen:

IPv4-Only Resolvers:

IPv6-Only Resolvers:

Dual Stack Resolvers:

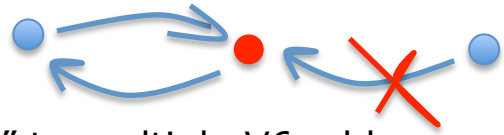
Aside: Identifying DNS Dual Stack Resolvers

Identifying a resolver as a dual stack resolver involves some assumptions, as the logged queries do not implicitly reveal that a V4 and a V6 address are actually addresses of the same resolver:

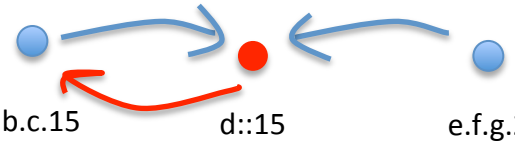
- If a test query set involved a single V4 and single V6 address then I tentatively "join them" to a single resolver

- 6-to-4 addresses are "joined" to each other

- Loops are preferred



- If a v4 address is "joined" to multiple V6 addresses in this way (or vv) then I undo the join except in those cases where the V4 and V6 addresses share a common final octet/nibble



DNS resolvers

Let's switch from the queries make by resolvers to the visible resolvers themselves

Resolvers seen:	464,950	
IPv4-Only Resolvers:	446,173	(96%)
IPv6-Only Resolvers*:	11,377	(2%)
Dual Stack Resolvers:	7,040	(2%)

* Could not uniquely associate the IPv6 address with a single IPv4 address

DNS Dual Stack Resolvers

282 dual stack resolvers use 6-to-4 for their IPv6 connections

- None of these resolvers prefer IPv6 when querying a dual stack auth server

4 dual stack resolvers used Teredo (!)

- They made a mix of V4 and V6 queries (63% v4)

6,759 dual stack resolvers used non-mapped V6 addresses

- 58% of queries using V4, 42% using V6

DNS Dual Stack resolvers

Lets look the queries made by the visible dual stack resolvers:

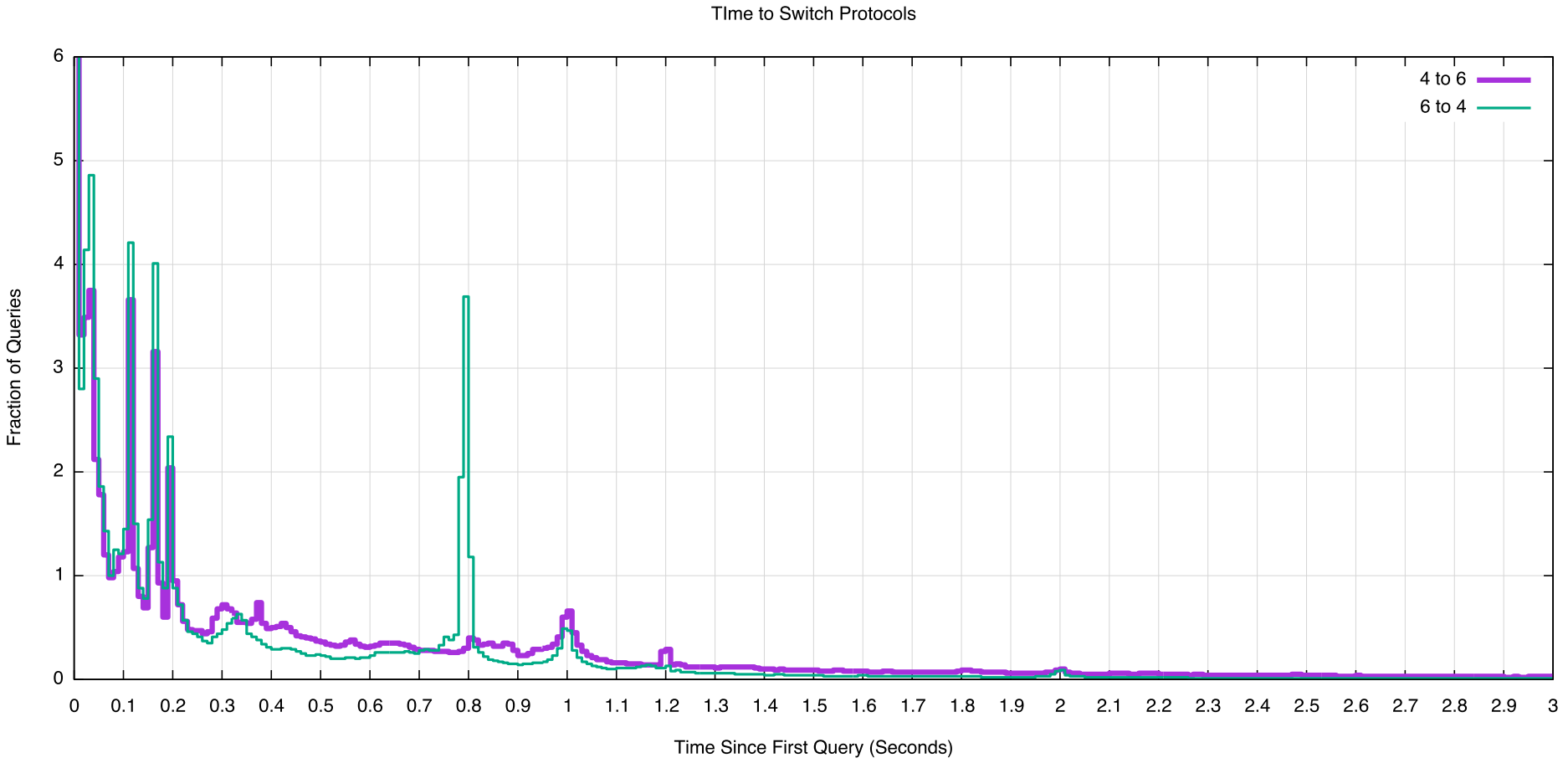
Dual Stack Resolvers:	7,290	
Always Prefer 4:	1,074	(15%)
Always Prefer 6:	197	(3%)
Mixed:	6,001	(82%)
Did not query DS name:	18	(0%)

DNS V6 Capable resolvers

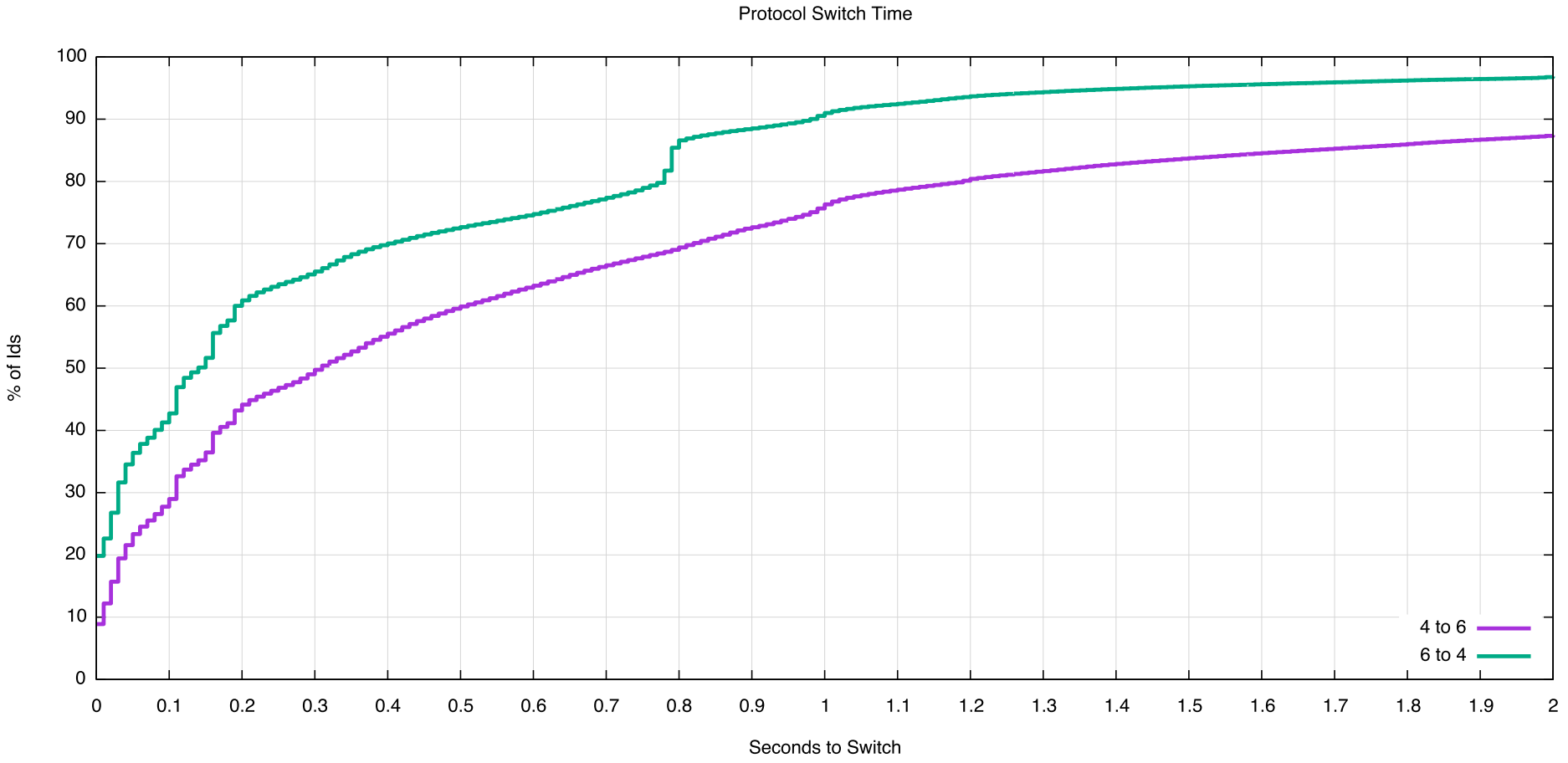
V6 Capable Resolvers:	18,421	
Did not use V6 for Dual Stack:	5,088	(28%)
Always Preferred V6:	1,458	(8%)
Mixed V6/V4 for Dual Stack:	11,875	(64%)

Of the mixed V4/V6 situation V6 was used to resolve the dual stack glue record for 5,651,796 identifiers of a total of 38,782,137 identifiers, or 15÷ of the time

DNS Protocol Switch Times



DNS Protocol Switch Times



What Does Google's Public DNS Do?

Observed V6 resolver addresses for Google PDNS: 566

Observed preference for V6 dual stack: 0

(using glueless delegation)

What does Bind Do?

Can we see Bind?

- Well, as far as I am aware (please correct me) Bind is the only resolver that will not follow a CNAME in a NS record
- So lets use that as a working definition for Bind and see what Bind does

What does Bind do?

Experiments using dual stack BIND resolvers:

Asked for Dual Stack using V4:	4,075,246 (52%)
Asked for Dual Stack using V6:	690,566 (17%)
Asked for Dual Stack using V4 and V6:	1,263,312 (31%)

What does Bind do?

Number of resolvers: 264,501 of 479,468 (55%)

(These are the resolvers who do not follow a CNAME RR)

Compare V4 only to V4 Dual Stack

Used IPv4 to query a dual stack resource: 123,339 / 136,946 (90%)

Compare V6 only to V6 Dual Stack

Used IPv6 to query a dual stack resource: 9,402 / 11,950 (79%)

What does NON-Bind do?

Experiments using dual stack NON-BIND resolvers:

Asked for Dual Stack using V4:	22,135,775	(87%)
Asked for Dual Stack using V6:	1,291,746	(5%)
Asked for Dual Stack using V4 and V6:	1,930,633	(8%)

What does NON-Bind do?

Number of resolvers: 214,967 of 479,468 (45%)

(These are the resolvers who do follow a CNAME RR)

Compare V4 only to V4 Dual Stack

Used IPv4 to query a dual stack resource: 136,039 / 139,834 (98%)

Compare V6 only to V6 Dual Stack

Used IPv6 to query a dual stack resource: 2,554 / 2,693 (94%)

Happy DNS Eyeballs?

Not really.

Only 4% of resolvers appear to be dual stack capable ☹

And of those that do, they are not favoring IPv6 over IPv4 ☹

And there is not clear evidence of the use of a fast failover approach from IPv6 to IPv4 ☹

Does it matter?

How can you tell when you no longer need to keep running IPv4 on an authoritative name server?

When there are no longer any queries made using IPv4

But this answer assumes that dual stack resolvers have a clear preference to use IPv6 first and perform a fast failover to IPv4

Which is not happening today in the DNS ☹

That's it!