Fewer nameservers, more addresses

Dave Knight

RIPE 85 DNS Working Group

Always-on, *Ultra* Secure.

neustar Security Services

More nameservers

- Recently launched a second edge DNS platform, UltraDNS2
- UltraDNS users typically have up to 6 Ultra nameservers for a domain, e.g.

```
example.com.  IN  NS  pdns196.ultradns.biz.
example.com.  IN  NS  pdns196.ultradns.co.uk.
example.com.  IN  NS  pdns196.ultradns.com.
example.com.  IN  NS  pdns196.ultradns.org.
example.com.  IN  NS  pdns196.ultradns.net.
example.com.  IN  NS  pdns196.ultradns.info.
```

- Users adopting UltraDNS2 add 2 more

```
example.com.  IN  NS  ns1.ultradns2.com.
example.com.  IN  NS  ns1.ultradns2.org.
```

- Concern that in some TLDs 8 is just too many

TLD registry delegation size constraints

- How many TLDs allow fewer than 8 nameservers in a delegation?
- It was surprisingly hard to find information on this ☺
- A public list: https://dm.realtimeregister.com/docs/api/tlds/metadata
- Also got a list from a registrar
- They don't always agree, merged them in favor of the lower limit
- 1595 TLD *extensions* constrain the number of nameservers in a delegation as follows

13	12	10	9	8	7	6	5	4	3	2
107	4	819	26	63	**25**	**126**	**245**	**104**	**41**	**35**

- **576 / 1595** (36%) of TLD *extensions* don't allow 8 nameservers in a delegation

A simple solution?

- **Just use fewer nameservers, we do anycast!**
  - All of the nodes can answer for any nameserver
  - You go to the closest node anyway
  - Why would we need lots of nameservers?
- **Our anycast deployment employs varied announcement strategies**
  - Pursuing a balance between performance and resiliency
  - Each node deliberately de-preferences BGP advertisements of the prefixes for some nameservers
  - Goal is that no resolver goes to the same node for every nameserver
  - We want resolvers to exercise all those strategies
  - **We want resolvers to use all our IP addresses, do we really care about names?**

Can we give fewer nameservers more addresses?

```
example.com.          NS    ns.ultradns.biz
example.com.          NS    ns.ultradns.net
example.com.          NS    ns.ultradns2.com
example.com.          NS    ns.ultradns2.org

ns.ultradns.biz.      A     192.0.2.1
                      AAAA  2001:db8:1::1
ns.ultradns.net.      A     198.51.100.1
                      AAAA  2001:db8:2::1
ns.ultradns2.com.     A     203.0.113.1
                      AAAA  2001:db8:3::1
ns.ultradns2.org.     A     172.16.0.1
                      AAAA  2001:db8:4::1
```

vs

```
example.com.          NS    n1.ultradns.com
example.com.          NS    n1.ultradns2.org

ns.ultradns.com.      A     192.0.2.1
                      A     198.51.100.1
                      AAAA  2001:db8:1::1
                      AAAA  2001:db8:2::1

ns.ultradns2.org.     A     203.0.113.1
                      A     172.16.0.1
                      AAAA  2001:db8:3::1
                      AAAA  2001:db8:4::1
```

- Does it work ?
  - Will resolvers use lots of addresses from one name?
  - Do they penalize all the addresses for a name if one of them is bad?

Domain with 2 nameservers with many addresses

```
daveknight.ca.      NS    u1ns.knig.ht.
                    NS    u2ns.knig.ht.

u1ns.knig.ht.       A     156.154.34.14
                    A     156.154.34.15
                    A     156.154.34.16
                    A     156.154.34.13
                    AAAA  2610:a1:3000::13
                    AAAA  2610:a1:3000::16
                    AAAA  2610:a1:3000::14
                    AAAA  2610:a1:3000::15

u2ns.knig.ht.       A     156.154.34.23
                    A     156.154.34.24
                    AAAA  2610:a1:3000::23
                    AAAA  2610:a1:3000::24
```

- Two nameservers, one for each of our platforms.
- Each nameserver configured with a list of IP addresses representative of what is announced by each platform.

Query some resolvers while breaking things

```
# Configure all IP addresses on a test nameserver

# Break one IP address at a time, send 1000 queries to each resolver
for ip (.13 .14 .15 .16 ::13 ::14 ::15 ::16 .23 .24 ::23 ::24)
    disable ip
    for iteration (0..999)
        for resolver (bind knot unbound cloudflare google opendns quad9 ultradns)
            query @resolver {iteration}-{uuid}-{af}-{resolver}.daveknight.ca IN A
    enable ip

# Capture the packets arriving at the nameserver
# Count how many queries are sent to a broken address
            ( all resolvers always got an answer after retrying )
```


neustar Security Services

My eyes!!

- Apologies for what follows, I should learn how to visualize data
- The tables of numbers show counts of initial queries to each IP
  - Columns are the nameserver IP address where the query was sent
  - Rows are the broken IP address
- We expect to see a low number of queries to a broken address
  - The nameserver should quickly stop trying to query a broken address
- We want to see it rise again quickly in subsequent rows
  - The nameserver should detect recovery and send queries again
- We don't want blocks of addresses with low numbers of queries
  - The nameserver is grouping the addresses for a nameserver name
- Oh, .13 is an IPv4 address, ::13 is an IPv6 address

u1ns

	.13	.14	.15	.16	::13
.13	6	137	126	127	97
.14	85	4	99	94	75
.15	121	110	4	119	114
.16	70	85	83	7	92
::13	116	102	106	101	6
::14	115	98	111	115	97
::15	91	97	89	96	62
::16	119	119	122	99	109
.23	92	87	92	37	89
.24	76	80	90	75	100
::23	96	91	98	112	94
::24	83	79	72	98	85

neustar Security Services

BIND9

		u1ns									u2ns		
		.13	.14	.15	.16	::13	::14	::15	::16	.23	.24	::23	::24
broken	.13	0	0	0	0	108	189	350	148	0	1	114	90
ips	.14	2	0	0	0	267	177	2	107	0	0	218	227
	.15	1	0	0	0	156	302	163	98	0	0	228	52
	.16	1	0	0	0	59	196	1	321	1	0	212	209
	::13	1	0	0	0	1	0	0	0	0	0	0	998
	::14	1	0	0	0	282	1	0	0	0	0	0	716
	::15	0	0	0	0	510	1	1	0	0	0	0	488
	::16	0	0	0	0	380	53	350	1	0	0	0	216
	.23	0	0	0	0	266	119	12	223	0	0	257	123
	.24	2	0	0	0	125	270	67	131	0	0	78	327
	::23	1	0	0	0	266	137	356	129	0	0	2	109
	::24	1	0	0	0	76	234	95	576	0	0	17	1

- Strongly prefers IPv6
- Weird behavior!

If one address is broken, penalize all higher numbered addresses until piling onto the last one?

- Slow to get an answer when retrying

neustar Security Services

Knot Recursor

		u1ns							u2ns			
		.13	**.14**	**.15**	**.16**	**::13**	**::14**	**::15**	**::16**	**.23**	**.24**	**::23**	**::24**
broken	**.13**	**6**	137	126	127	97	75	93	95	135	107	0	2
ips	**.14**	85	**4**	99	94	75	52	50	47	75	94	169	156
	.15	121	110	**4**	119	114	95	73	92	108	112	30	22
	.16	70	85	83	**7**	92	86	81	87	78	71	140	120
	::13	116	102	106	101	**6**	7	6	5	119	120	164	148
	::14	115	98	111	115	97	**6**	76	93	126	120	20	23
	::15	91	97	89	96	62	62	**2**	50	98	100	116	137
	::16	119	119	122	99	109	92	101	**8**	108	108	0	15
	.23	92	87	92	37	89	67	87	70	**7**	87	134	151
	.24	76	80	90	75	100	81	88	75	63	**4**	123	145
	::23	96	91	98	112	94	94	97	100	98	101	**2**	17
	::24	83	79	72	98	85	89	111	72	83	58	167	**3**

- No significant IP version preference
- Fast to get an answer when retrying
- Doesn't penalize broken addresses for long

neustar Security Services

Unbound

		u1ns								u2ns			
		.13	**.14**	**.15**	**.16**	**::13**	**::14**	**::15**	**::16**	**.23**	**.24**	**::23**	**::24**
broken	**.13**	**4**	78	82	88	105	86	97	83	83	95	97	102
ips	**.14**	65	**4**	98	94	100	103	86	91	97	95	81	86
	.15	97	35	**5**	101	96	104	94	80	99	109	92	88
	.16	111	100	0	**4**	91	112	87	88	89	107	102	109
	::13	86	102	103	77	**1**	101	102	91	91	83	85	78
	::14	116	104	105	125	109	**3**	0	0	110	116	100	112
	::15	101	95	114	84	89	94	**4**	0	95	109	118	97
	::16	100	124	87	84	92	102	90	**5**	93	99	49	75
	.23	104	87	58	42	102	102	97	101	**4**	99	90	114
	.24	112	105	98	89	109	106	98	78	0	**4**	103	98
	::23	87	102	110	95	118	99	99	90	87	109	**4**	0
	::24	84	84	86	75	77	91	99	99	99	92	113	**1**

- No significant IP version preference
- Doesn't significantly penalize a broken address

neustar Security Services

Cloudflare

		u1ns								u2ns			
		.13	.14	.15	.16	::13	::14	::15	::16	.23	.24	::23	::24
broken	.13	**153**	285	234	120	0	0	0	0	78	130	0	0
ips	.14	82	**131**	307	145	0	0	0	0	102	233	0	0
	.15	161	81	**158**	193	0	0	0	0	181	226	0	0
	.16	216	187	106	**95**	0	0	0	0	183	213	0	0
	::13	315	235	194	75	**0**	0	0	0	119	62	0	0
	::14	286	232	235	80	0	**0**	0	0	84	83	0	0
	::15	313	248	174	98	0	0	**0**	0	74	93	0	0
	::16	350	241	156	45	0	0	0	**0**	123	85	0	0
	.23	232	195	139	53	0	0	0	0	**83**	298	0	0
	.24	261	188	205	63	0	0	0	0	113	**169**	0	0
	::23	321	241	168	84	0	0	0	0	119	67	**0**	0
	::24	319	210	233	59	0	0	0	0	86	93	0	**0**

- Strongly refers IPv4
- Does an initial retry using the same socket, so higher number of initial queries to broken Ips
- Doesn't significantly penalize a broken address

neustar Security Services

Google

		u1ns								u2ns			
		.13	**.14**	**.15**	**.16**	**::13**	**::14**	**::15**	**::16**	**.23**	**.24**	**::23**	**::24**
broken	**.13**	**72**	95	85	73	96	79	82	95	101	67	71	84
ips	**.14**	80	**81**	79	85	92	66	78	97	89	94	72	87
	.15	88	63	**75**	73	109	93	92	86	89	71	80	81
	.16	73	66	69	**63**	94	89	88	95	80	95	95	93
	::13	66	101	73	72	**89**	87	81	78	103	94	79	77
	::14	75	87	88	62	101	**86**	73	83	103	72	90	80
	::15	76	81	65	79	102	90	**99**	77	89	88	77	77
	::16	67	90	72	81	96	97	84	**81**	90	82	79	81
	.23	67	61	84	87	118	100	83	90	**68**	71	91	80
	.24	67	72	87	82	89	87	92	96	67	**76**	84	101
	::23	65	77	83	82	106	91	96	83	72	76	**91**	78
	::24	72	80	80	83	88	92	105	88	78	79	75	**80**

- No obvious IP version preference
- Always retries the broken address once, so higher number of queries to a broken IP
- Doesn't significantly penalize a broken address

neustar Security Services

OpenDNS

		u1ns								u2ns			
		.13	**.14**	**.15**	**.16**	**::13**	**::14**	**::15**	**::16**	**.23**	**.24**	**::23**	**::24**
broken	**.13**	**0**	0	0	0	180	173	168	158	0	0	156	165
ips	**.14**	0	**0**	0	0	166	190	169	150	0	0	162	163
	.15	0	0	**0**	0	166	176	163	157	0	0	179	159
	.16	0	0	0	**0**	153	185	171	177	0	0	158	156
	::13	33	36	45	45	**61**	68	121	187	19	21	186	178
	::14	50	43	32	35	184	**51**	79	114	21	30	171	190
	::15	33	53	39	47	201	192	**48**	43	23	25	96	200
	::16	32	34	39	48	174	190	188	**50**	14	15	40	176
	.23	0	0	0	0	181	175	165	162	**0**	0	171	146
	.24	0	0	0	0	175	161	160	176	0	**0**	153	175
	::23	24	32	24	29	188	187	197	171	13	14	**53**	68
	::24	13	20	14	19	180	152	184	191	6	6	161	**54**

- Prefers IPv6
- Will sometimes try IPv4 when an IPv6 address is broken
- Fast to get an answer when retrying
- Doesn't significantly penalize a broken address

neustar Security Services

Quad9

		u1ns								u2ns			
		.13	**.14**	**.15**	**.16**	**::13**	**::14**	**::15**	**::16**	**.23**	**.24**	**::23**	**::24**
broken	**.13**	**69**	199	112	257	0	3	1	23	139	129	49	1
ips	**.14**	107	**81**	138	152	11	44	9	59	165	123	55	19
	.15	171	82	**75**	102	15	9	18	51	257	120	56	19
	.16	235	164	64	**71**	40	31	58	0	121	168	2	18
	::13	101	79	140	129	**3**	0	3	71	210	127	100	0
	::14	119	80	114	80	157	**3**	0	40	133	85	134	29
	::15	137	80	106	88	80	122	**5**	0	184	108	72	1
	::16	135	128	103	84	36	15	118	**3**	219	126	18	0
	.23	163	193	197	65	14	72	57	26	**74**	104	10	0
	.24	94	340	153	102	0	33	68	46	52	**73**	10	0
	::23	118	244	122	93	0	7	53	11	202	112	**2**	2
	::24	137	287	119	92	0	10	134	26	111	55	10	**0**

- Slight IPv4 preference
- Looking at the captures there's much less obvious pattern of behavior than other resolvers.

neustar Security Services

UltraDNS

		u1ns							u2ns			
		.13	**.14**	**.15**	**.16**	**::13**	**::14**	**::15**	**::16**	**.23**	**.24**	**::23**	**::24**
broken	**.13**	**15**	101	92	81	86	93	79	109	80	91	88	85
ips	**.14**	89	**22**	104	73	86	83	100	90	91	86	105	70
	.15	112	100	**13**	84	96	92	96	82	72	84	95	74
	.16	104	91	85	**14**	96	96	103	81	104	64	85	77
	::13	96	80	79	86	**13**	96	84	81	93	114	86	92
	::14	89	102	81	85	85	**8**	85	89	83	105	100	87
	::15	89	85	98	92	87	96	**6**	87	88	95	79	98
	::16	80	94	83	87	88	91	92	**6**	91	97	102	89
	.23	80	95	99	71	77	86	94	93	**21**	93	109	82
	.24	95	101	79	75	83	90	96	101	89	**19**	77	83
	::23	76	85	89	73	94	102	104	92	86	98	**9**	92
	::24	84	90	79	97	91	83	92	100	91	94	91	**8**

- No obvious IP version preference
- Doesn't significantly penalize a broken address

neustar Security Services

So, does it work?

- Will resolvers use lots of addresses from one name?
  - Address Family biases aside…
  - **All resolvers tested use all the addresses!**

- Do they penalize all the addresses for a name if one of them is bad?
  - Only BIND9 observed to aggressively penalize any address
  - **See no evidence of a resolver penalizing groups of addresses for a name!**

- Resolvers use nameserver names to populate a list of IP addresses where they can find authority data, otherwise nameserver names don't matter.
- **It works!**

Oh, what about glue?

- In my zone I can give a nameserver name many addresses
- Can I do the same in the parent zone?
  - Once more we venture into the perplexing world of registry constraints
- EPP allows “one or more” address attributes on a host record
  - What do registries actually allow?
- Tried to add 12 addresses to a host object in a few TLDs
  - CA allows a maximum of 1 address
  - UK allows a maximum of 2 addresses
  - COM allows at least 12 addresses
  - ORG allows at least 12 addresses
- If we’re going to have glue, we want it in COM, or ORG (other TLD extensions with a generous policy are probably available)

Optimum configuration

```
fewnamesmanyips.com.      NS    ns1.fewnamesmanyips.com.
                          NS    ns1.fewnamesmanyips.org.

fewnamesmanyips.org.      NS    ns1.fewnamesmanyips.com.
                          NS    ns1.fewnamesmanyips.org.

fewnamesmanyips.ca.       NS    ns1.fewnamesmanyips.com.
                          NS    ns1.fewnamesmanyips.org.
```

```
ns1.fewnamesmanyips.com.   A      156.154.34.13
                           A      156.154.34.14
                           A      156.154.34.15
                           A      156.154.34.16
                           A      156.154.34.23
                           A      156.154.34.24
                           AAAA   2610:a1:3004::13
                           AAAA   2610:a1:3004::14
                           AAAA   2610:a1:3004::15
                           AAAA   2610:a1:3004::16
                           AAAA   2610:a1:3004::23
                           AAAA   2610:a1:3004::24
```

```
ns1.fewnamesmanyips.org.   A      156.154.34.13
                           A      156.154.34.14
                           A      156.154.34.15
                           A      156.154.34.16
                           A      156.154.34.23
                           A      156.154.34.24
                           AAAA   2610:a1:3004::13
                           AAAA   2610:a1:3004::14
                           AAAA   2610:a1:3004::15
                           AAAA   2610:a1:3004::16
                           AAAA   2610:a1:3004::23
                           AAAA   2610:a1:3004::24
```

- Announcement strategies
  - UltraDNS has 4
  - UltraDNS2 has 2
  - Want resolvers to exercise 6 x (IPv4+IPv6)
- Care about getting all my addresses into a resolver
  - Give every nameserver **EVERY** address
  - Resolver gets everything it needs in the first lookup
  - Often as glue in the additional section
- TLD diversity
  - Could theoretically use just one nameserver
  - Don't want all my eggs in one TLD basket
  - Provision nameservers in many TLDs, customers use as appropriate

neustar Security Services

Signed referrals aren't too big

```
$ dig +noall +auth +add +stat @a.gtld-servers.net. fewnamesmanyips.com. in ns +dnssec | cut -c1-80

fewnamesmanyips.com. 172800      IN    NS    ns1.fewnamesmanyips.com.
fewnamesmanyips.com. 172800      IN    NS    ns1.fewnamesmanyips.org.
fewnamesmanyips.com. 86400 IN    DS    13512 13 2 322041B3332E9BF01E6D5424F590B8FA0C8A
fewnamesmanyips.com. 86400 IN    RRSIG DS 8 2 86400 20221102152424 20221026141424 5
ns1.fewnamesmanyips.com. 172800 IN    A     156.154.34.13
ns1.fewnamesmanyips.com. 172800 IN    A     156.154.34.14
ns1.fewnamesmanyips.com. 172800 IN    A     156.154.34.15
ns1.fewnamesmanyips.com. 172800 IN    A     156.154.34.16
ns1.fewnamesmanyips.com. 172800 IN    A     156.154.34.23
ns1.fewnamesmanyips.com. 172800 IN    A     156.154.34.24
ns1.fewnamesmanyips.com. 172800 IN    AAAA  2610:a1:3004::13
ns1.fewnamesmanyips.com. 172800 IN    AAAA  2610:a1:3004::14
ns1.fewnamesmanyips.com. 172800 IN    AAAA  2610:a1:3004::15
ns1.fewnamesmanyips.com. 172800 IN    AAAA  2610:a1:3004::16
ns1.fewnamesmanyips.com. 172800 IN    AAAA  2610:a1:3004::23
ns1.fewnamesmanyips.com. 172800 IN    AAAA  2610:a1:3004::24
;; Query time: 48 msec
;; SERVER: 192.5.6.30#53(192.5.6.30)
;; WHEN: Wed Oct 26 20:24:23 CEST 2022
;; MSG SIZE  rcvd: 610
```

COM 610 bytes
ORG 578 bytes
CA 331 bytes (no glue)

When glue is present a resolver gets every IP it will ever need in the referral.

neustar Security Services

Conclusions

- Not proposing a new best practice
  - This seems to be a good workaround for our particular constraints
  - Not everyone needs this
  - **It works!**
- It doesn’t seem to change resolution behavior
  - Slight optimization as fewer queries are needed to learn all IP addresses
- Customers should probably still use more nameservers if they can
  - For TLD diversity
  - Can **Just Use Fewer Nameservers** when required, with no performance hit!
- It is surprising, this may have impact for monitoring and measurement
  - Tools probably don’t all expect to have to walk a list of IP addresses for a name
  - We can adapt our own, but there are many naive tests out there
- Next steps
  - Further experiments trialing this with production domains

Thank you

Always-on, *Ultra* Secure.

neustar
Security Services