Cache Me If You Can: Effects of DNS Time-to-Live

Giovane C. M. Moura[1,2],
John Heidemann[3], Wes Hardaker[3], Ricardo de O. Schmidt[4]

IEPG – IETF 105
Montreal, CA
2019-07-21

[1]SIDN Labs, [2]TU Delft, [3]USC/ISI, [4]UPF

New paper (accepted last Thursday)

- Paper just accepted at **ACM IMC 2019**
  - Perfect timing for this meeting, and DNSOP
- Submitted version:
  `https://www.isi.edu/~johnh/PAPERS/Moura19a.html`
- Revised Version (Moura19b, camera ready) will follow

Introduction

- *Caching* is the cornerstone of DNS performance
  - 15ms query response time is good, 1ms from cache it far better
  - It also protects clients from DDoS at auth servers [1]
- TTL controls cache duration, so it affects latency, resiliency.
- There has been little evaluation of TTLs [2, 1]
- Yet no research provides *recommendations/considerations* on what values are good

Introduction

- Determining good TTLs is very challenging:
  - Short TTLs allow OPs to change services quickly
  - Long TTLs reduces latency and service load
- Given that, it's no surprise that there is no consensus on TTL choices
- This study focus on filling this gap

Research Questions

1. Are resolvers **parent** or **child**-centric?
    - e.g.: TTL for NS `google.nl` can be found at parent (`.nl`) and child (`google.nl`)
2. How different parts of a FQDN change the **effective TTL lifetime**?
    - e.g.: NS, A records are parent, child, different zones?
3. How are TTLs used **in the wild**?
    - We know that TLDs NSes are the Root zone have long TTLs (2 days)
    - CDNs tend to have short TTLs

Goal: provide **recommendations** (IETF = considerations) on choosing TTL values

Resolver's centricity

- Same query may have different responses, with diff TTLs

Q / Type	Server	Response	TTL	Sec.
.cl / NS	k.root-servers.net	a.nic.cl/NS	172800	Auth.
		a.nic.cl/A	172800	Add.
		a.nic.cl/AAAA	172800	Add.
.cl/NS	a.nic.cl	a.nic.cl/NS	3600⋆	Ans.
		a.nic.cl/A	43200	Add.
		a.nic.cl/AAAA	43200	Add.
a.nic.cl/A	a.nic.cl	190.124.27.10/A	43200⋆	Ans.

Table 1: a.nic.cl. TTL values in parent and child (⋆ indicates an authoritative answer), on 2019-02-12.

Resolver’s centricity

- We use `.uy` to address this RQ
- Why? On 2019-02-14, it had:
  - `.uy` NS/A TTL at Roots = 172800s
  - `.uy` NS TTL at child: 300s
  - `.uy` A TTL at child: 120s
  - So it’s easy to measure it with Ripe Atlas

Resolver’s centricity

	.uy-**NS**	a.nic.uy-**A**	google.co-**NS**	.uy-**NS-new**
Frequency	600s	600s	600s	600
Duration	2h	3h	1h	2h
Query	NS .uy	A a.nic.uy	NS google.co	NS .uy
TTL Parent	172800 s	172800 s	900 s	172800 s
TTL Child	300 s	120 s	345600 s	86,400
Date	20190214	20190215	20190304	20190304
Probes	8963	8974	9127	8682
valid	8863	8882	9034	8536
disc	100	92	93	96
VPs	15722	15845	16078	15325
Queries	189506	285555	97213	184243
Responses	188307	282001	96602	184243
valid	188225	281931	96589	184209
disc.	82	70	3	34

Table 2: Resolver’s centricity experiments. Datasets available at [3].

Resolver's centricity

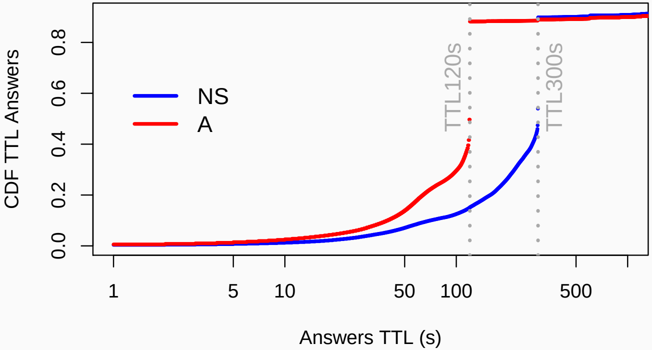

Figure 1: Observed TTLs from RIPE Atlas VPs for `.uy`-NS and `a.nic.uy`-A queries.

- Remember: TTL parents: 2 days
- Most resolvers are child centric, preferring TTLs of **AA answers**, as in §in 5.4.1 of RFC2181 [4]

Resolver's centricity

- We confirmed this finding with a second-level domain (google.com)
- And with passive data from `.nl`: see paper for more

How different parts of FQDN change TTL lifetime?

- We use a test domain: `sub.cachetest.net`
- Two scenarios:
  - In-bailiwick: NS: `ns3.sub.cachetest.net`
  - Out-of-bailiwick: NS: `ns1.zurrundeddu.com`
- Intentionally set TTL of NS to be shorter thant TTL A (3600 vs 7200)
- Question: **if TTL(NS) < TTL(A)**, what happens when NS expires?
  - Are records cached independently or both of them expire at the same time?

How different parts of FQDN change TTL lifetime?

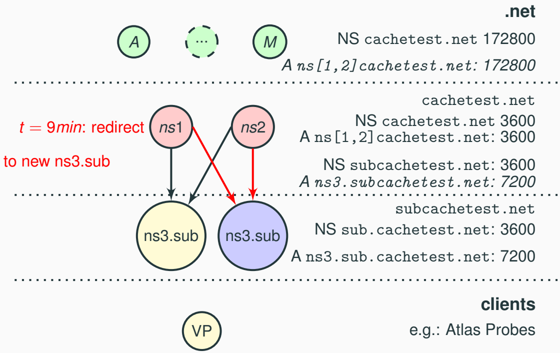

Figure 2: TTLs and domains for in-bailiwick experiment [3]. Italics indicate glue records.

- To control that, we *change the records at T=9min*
- New servers **gives a different answer** to the same AAAA query (probeID.sub.cachetest.net)

How different parts of FQDN change TTL lifetime?

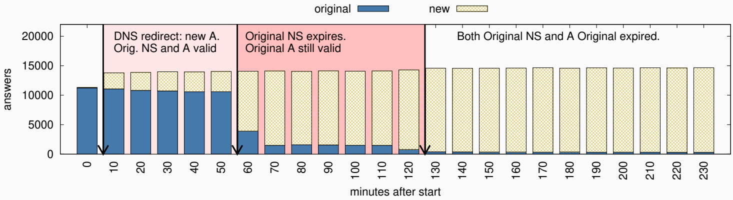

Figure 3: Timeseries of answers for **in-bailiwick** experiment

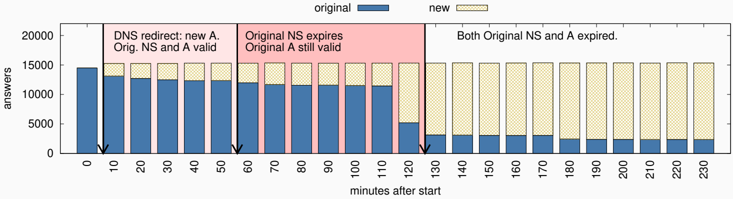

Figure 4: Timeseries of answers for **out-of-bailiwick** experiment

How different parts of FQDN change TTL lifetime?

In-bailiwick after NS expires:

```
dig ns sidn.nl @ns1.dns.nl

;; AUTHORITY SECTION:
sidn.nl.          3600   IN   NS     ns1.sidn.nl.
sidn.nl.          3600   IN   NS     ns2.sidn.nl.
sidn.nl.          3600   IN   NS     ns3.sidn.nl.
;; ADDITIONAL SECTION:
ns1.sidn.nl.      3600   IN   A      213.154.241.88
ns1.sidn.nl.      3600   IN   AAAA   2001:7b8:606::88
ns2.sidn.nl.      3600   IN   A      194.171.17.5
ns2.sidn.nl.      3600   IN   AAAA   2001:610:0:800d::5
ns3.sidn.nl.      3600   IN   A      194.0.30.2
ns3.sidn.nl.      3600   IN   AAAA   2001:678:34:0:194
```

How different parts of FQDN change TTL lifetime?

Out-of-bailiwick after NS expires:

```
dig ns google.nl @ns1.dns.nl

;; AUTHORITY SECTION:
google.nl.          3600    IN    NS    ns1.google.com.
google.nl.          3600    IN    NS    ns2.google.com.
google.nl.          3600    IN    NS    ns3.google.com.
google.nl.          3600    IN    NS    ns4.google.com.
```

How different parts of FQDN change TTL lifetime?

- Most recursives trust cached A records when served from different zones (out-of-bailiwick)
- They do not trust, however, when served from the same zone
- Why?
    - When NS expires, resolvers has to ask it again
        - In-bailiwick responses contain *additional* records with the *new* renumbered address
        - out-of-bailiwick contain only the NS records

How are TTLs used in the wild?

- We crawl different lists of domains
  - Alexa
  - Majestic
  - Umbrella
  - `.nl`
  - Root (TLDs)
- We retrieve: NS, A, AAAA, MX, and DNSKEY
- We analyze **child TTL** values (as most resolvers are child centric)
- And discuss results with some operators

How are TTLs used in the wild?

	Alexa	Majestic	Umbre.	.nl	Root
responsive	988654	928299	783343	5454833	1535
CNAME	50981	7017	452711	9436	0
SOA	12741	8352	59083	12268	0
responsive NS	924932	912930	271549	5433129	1535
Out only	878402	873447	244656	5417599	748
ratio	95.0%	95.7%	90.1	99.7%	48.7%
In only	37552	28577	20070	12586	654
Mixed	8978	10906	6823	2941	133

Table 3: Bailiwick distribution in the wild.

How are TTLs used in the wild?

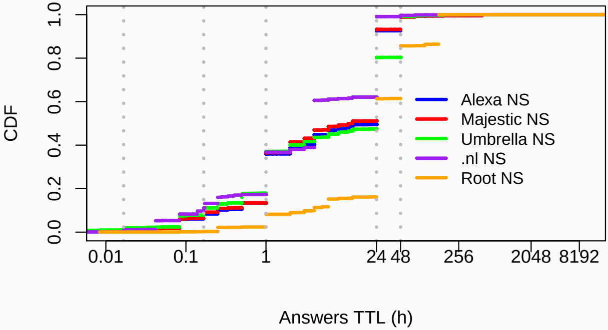

Figure 5: CDF of TTLs for NS records

How are TTLs used in the wild?

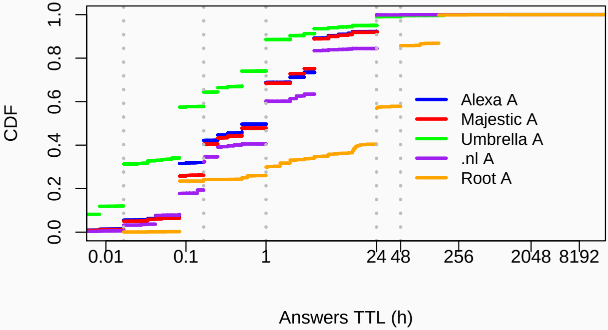

Figure 6: CDF of TTLs for A records

How are TTLs used in the wild?

Discussion with Operators:

- We found **34 TLDs** with TTLs (NS) $<$ 30min; 122 under 120min
- We reached out to 8 ccTLDs ops, 6 responded:
  - 3 had not considered it
  - 2 said it was intentional (temporary infra change)
  - 1 said it was this way since they took it over
- 3 TLDs *increased their TTL* after our notification
  - To 1 day, from 300s, 1800s, 30s

How are TTLs used in the wild?

Feedback from `.uy`: TTL from 300s to 86400

- Improved response times:
    - median RTT 28ms vs 8ms;
    - 75%ile from 183ms to 21ms

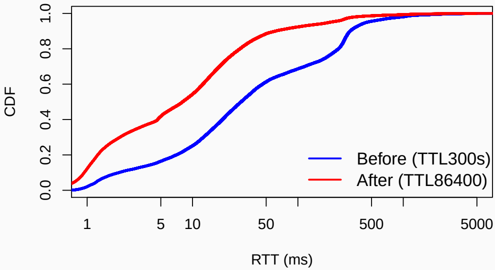

Figure 7: RTT from RIPE Atlas VPs for NS `.uy` queries before and after changing TTL NS records.

Research Questions

1. Are resolvers **parent** or **child**-centric?
   - most child-centric
2. How different parts of a FQDN change the **effective TTL lifetime**?
   - bailiwick impacts caching significantly
3. How are TTLs used **in the wild**?
   - all over the place, longer NS than A/AAAA
   - mostly out-of-bailiwick

Shorter vs Longer TTLs

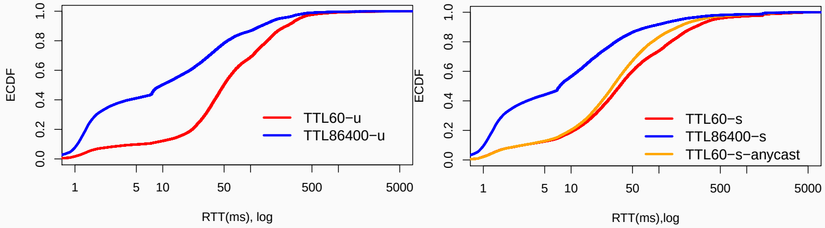

Figure 8: Distribution of client latency from Atlas VPs to controlled DNS with different TTLs.

- Longer TTLs leads to better response times (if cached) than **anycast** with short TTLs
- Also, reduced the query load in 77% on authoritative servers

Reasons for Longer or shorter TTLs

- **Longer caching**:
  - faster responses
  - lower DNS traffic
  - more robust to DDoS attacks on DNS
- **Shorter caching**:
  - supports operational changes
  - can help with a DNS-based response to DDoS attacks
  - can cope better with DNS-based load balancing

Organizations must weight these trade-offs to find a good balance; we propose two recommendations next.

So, recommendations

- There is no single optimal TTL for all users. But:
  - for general users, **longer TTLs**, as well as for TLD ops
  - exception: if you're use DNS-based DDoS protection
- A/AAAA records, and NS:
  - For out-of-bailiwick, records are cached *independently*
  - For in-bailiwick, TTL of A/AAAA **should be shorter or equal to NS**
  - (short A/AAAA may be desired if DDoS mitigation services are an option)
- Location: at least one out-of-bailiwick NS, in case zone becomes unreachable

Conclusions

- TTLs on DNS are a complex topic
- We carefully design many experiments to evaluate how factors interact
- We show that, in the wild, there is little consensus on TTL values
- Discussions with OPs lead to improve latencies to users (.uy)
- In short: Longer TTLs if you can
- **DNSOP Meeting**: consideration #5 on our draft based on this study
  - `https://tools.ietf.org/html/draft-moura-dnsop-authoritative-recommendations-04`

References I

[1] G. C. M. Moura, J. Heidemann, M. Müller, R. de O. Schmidt, and M. Davids, “When the dike breaks: Dissecting DNS defenses during DDoS,” in *Proceedings of the ACM Internet Measurement Conference*, Oct. 2018. [Online]. Available: https://www.isi.edu/%7ejohnh/PAPERS/Moura18b.html

[2] J. Jung, A. W. Berger, and H. Balakrishnan, “Modeling TTL-based internet caches,” in *Proceedings of the IEEE Infocom*. San Francisco, CA, USA: IEEE, Apr. 2003. [Online]. Available: http://www.ieee-infocom.org/2003/papers/11_01.PDF

References II

[3] RIPE NCC, “RIPE Atlas measurement ids,” https://atlas.ripe.net/measurements/ID, Mar. 2019, iD is the experiment ID: uy-NS: 19544918, a.nic.uy-A: 19581585, google.co-NS: 19927577, mapache-de-madrid.co-NS: 19584842, in-bailiwick: 20199814, out-of-bailiwick: 20181892, TTL60-u:19862830, TTL86400-u:19863763, TTL60-s:19871393, TTL86400-s:19871498, TTL60-s-anycast:19875360, uy-NS2: 19925152.

[4] R. Elz and R. Bush, “Clarifications to the DNS Specification,” IETF, RFC 2181, Jul. 1997. [Online]. Available: http://tools.ietf.org/rfc/rfc2181.txt