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

Giovane C. M. Moura\textsuperscript{1,2}, John Heidemann\textsuperscript{3},
Wes Hardaker\textsuperscript{3}, Ricardo de O. Schmidt\textsuperscript{4}

AMC IMC 2019
Amsterdam, The Netherlands
2019-10-23

\textsuperscript{1}SIDN Labs, \textsuperscript{2}TU Delft, \textsuperscript{3}USC/ISI, \textsuperscript{4}UPF
Introduction

Parent vs Child

Zone configurations and Effective TTL

TTLs Use in the Wild
  Operators Notification

Caching (Longer TTL) vs Anycast
  Shorter vs Longer TTLs

Recommendation and Conclusions
Our research on DNS over the last years

Our research on DNS security/stability:

- **Anycast and DDoS**: IMC 2016 [2]
- **Resolvers**: IMC 2017 [5]
- **Anycast Engineering**: PAM2017 [6], IMC 2017 [1]
- **Caching and DDoS**: IMC 2018 [4]
- **Caching and TTL, and performance**: IMC 2019 [3]
  - (this paper)
Introduction
The role of TTL
The role of TTL

Q: google.com?

user → resolver → authoritative server

Q: google.com?
The role of TTL

user → resolver: Q: google.com?
resolver → authoritative server: Q: google.com?
The role of TTL

user

Q: google.com?
A: 10.10.10.10

resolver

Q: google.com?
A: 10.10.10.10

authoritative server
The role of TTL

- User queries "google.com" to the resolver.
- The resolver queries the authoritative server.
- The authoritative server provides an IP address (10.10.10.10).
- The resolver caches the response.
- The user receives the cached response.

TTL stands for Time To Live, which determines how long a record can be cached before it expires.
The role of TTL

user → resolver: Q: google.com? → A: 10.10.10.10

resolver → authoritative server: Q: google.com? → A: 10.10.10.10

resolver → cache: Q: google.com?

cache: Q: google.com?
The role of TTL

User sends a query to the resolver:
Q: google.com?

Resolver checks its cache:
Q: google.com?

Cache hit!
A: 10.10.10.10

Resolver answers user:
A: 10.10.10.10

Authoritative server responds to resolver:
A: 10.10.10.10

FASTER
The role of TTL

ISP                      GOOGLE
BUT caching FOR
HOW LONG???
A: 10.10.10.10

Q: google.com?
A: 10.10.10.10

Q: google.com?
A: 10.10.10.10

Q: google.com?
A: 10.10.10.10

cache hit!
FASTER

ISP                      GOOGLE
BUT caching FOR
HOW LONG???
A: 10.10.10.10

Q: google.com?
A: 10.10.10.10

Q: google.com?
A: 10.10.10.10

Q: google.com?
A: 10.10.10.10

Q: google.com?
A: 10.10.10.10

cache hit!
FASTER
The role of TTL

ISP
GOOGLE
BUT caching FOR
HOW LONG???
A: 10.10.10.10
authoritative
server
resolver

user

Q: google.com?
A: 10.10.10.10

cache hit!
FASTER

Q: google.com?
A: 10.10.10.10

cache

A: 10.10.10.10

BUT caching FOR HOW LONG???

TTL
The role of TTL

- TTL controls caching
  - auth servers SIGNAL to resolvers how long (TTL)
- Caching is VERY important for performance
  - improves user experience
And you must set TTLs

- Say you register cachetest.net
What TTL values are good?

Today it is unclear what an operator should do

- DNS OPs folks on TTLs: “if it ain’t broke don’t fix it”

We think we can help

Figure 1: DNS ops changing TTLs. src: trainworld.be
Our contribution

Because of conflicting and under-explained TTL advice, we show:

1. the effective TTL comes from **multiple** places
   - Parent and Child authoritative servers
   - NS and A records (sometimes)
2. TTLs are unnecessarily short
   - a. because sometimes multiple places → one is shorter and wins
   - or operators don’t realize the cost
3. We show that longer TTLs are **MUCH** faster
4. Our results were adopted by 3 ccTLD for ~20ms median latency improvement; 171ms 75%ile
1. Parent vs Child: who really sets the TTL?
2. NS and A records: are they limited? And bailiwick?
3. Real-world variation exists
4. Longer TTLs are MUCH better
5. Our recommendations
Parent vs Child
Duplicate info: which one is chosen?

- Parent and child TTLs may vary: `dig NS cachetest.net`

Which TTL will Rembrandt use? Parent (172800s) or child (TTL: 3600s)
Are resolvers parent- or child-centric?

Parent vs Child experiment

- Test with experiment on .uy: (2019-02-14)
  - **Parent**: NS/A TTL: 172800s
  - **Child**: NS TTL: 300s ; A: 120s
- We query with 15k VPs (Ripe Atlas) multiple times, every 10min
- We analyze TTL values received at VPs
Most Atlas VPs resolvers are child-centric

Figure 2: Observed TTLs from Atlas VPs for .uy-NS and a.nic.uy-A queries.

- Spike at Child TTL A (120s): most resolvers are child centric
- Spike at Child TTL NS (300s): child centric

Remember: TTL parents: 2 days
Most Atlas VPs resolvers are child-centric

Figure 2: Observed TTLs from Atlas VPs for .uy-NS and a.nic.uy-A queries.

Spike at Child TTL A (120s) : most resolvers are child centric

• Remember: TTL parents: 2 days
Most Atlas VPs resolvers are child-centric

**Figure 2:** Observed TTLs from Atlas VPs for .uy-NS and a.nic.uy-A queries.

Spike at Child TTL A (120s) : most resolvers are child centric

Spike at Child TTL NS (300s): child centric

• Remember: TTL parents: 2 days
Is centricity true for TLDs and SLDs?

- Test with .nl TLD A records (ns*.dns.nl)
- TTLs are 3600s (child) vs. 17800s (parent)

Figure 3: Minimum interarrival time of A queries for TLD

We confirmed this with a second-level domain (paper)
Is centricity true for TLDs and SLDs?

- Test with `.nl` TLD A records (ns*.dns.nl)
- TTLs are 3600s (child) vs. 17800s (parent)

**Figure 3:** Minimum interarrival time of A queries for TLD

Spike at Child TTL A (3600s): confirm child centric for TLD

We confirmed this with a second-level domain (paper)
Is centricity true for TLDs and SLDs?

- Test with `.nl` TLD A records (ns*.dns.nl)
- TTLs are 3600s (child) vs. 17800s (parent)

**Figure 3:** Minimum interarrival time of A queries for TLD

Spike at Child TTL A (3600s): confirm child centric for TLD

We confirmed this with a second-level domain (paper)
Most resolvers will use child TTLs

- Rembrant (and users) mostly use child TTLs
- **Child TTL** controls caching (most times)

Which TTL will Rembrandt use?
Parent (172800s) or child (TTL: 3600s)
Zone configurations and Effective TTL
Are there dependencies between A and NS TTLs?

To resolve *.sub.cachetest.net, you need both NS and A.

Are NS and A cached independently?

1. $t=0$: all Atlas VPs query (fills cache with NS and A)
2. $t=4800$: what happens? NS is expired; A is still in cache: do resolvers use the "cached A" or refresh it again?

Trick: at $t=540$, we renumber A to 10.10.10.2 (different answer)

Will Marcus Aurelius receive cached or new answer?
Are there dependencies between A and NS TTLs?

sub.cachetest.net

In zone

Out of zone
Are there dependencies between A and NS TTLs?

- **sub.cachetest.net**
  - **In zone**
  - **Out of zone**

**NS:** ns1.sub.cachetest.net

To resolve *.sub.cachetest.net, you need both NS and A.

Are NS and A cached independently?

1. **t=0:** all Atlas VPs query (fills cache with NS and A)
2. **t=4800:** what happens? NS is expired; A is still in cache: do resolvers use the "cached A" or refresh it again?

**Trick:** at **t=540**, we renumber A to 10.10.10.2 (diff answer)

Will Marcus Aurelius receive cached or new answer?
Are there dependencies between A and NS TTLs?

**sub.cachetest.net**

- **In zone**
  - **NS**: ns1.sub.cachetest.net
  - **A**: 10.10.10.10
  - TTL: 7200

- **Out of zone**
  - **NS**: ns1.zurrundeddu.com
  - **A**: 10.10.10.10
  - TTL: 3600

To resolve *.sub.cachetest.net, you need both NS and A.

1. **t=0**: all Atlas VPs query (fills cache with NS and A)
2. **t=4800**: what happens? NS is expired; A is still in cache: do resolvers use the "cached A" or refresh it again?
3. **trick**: at **t=540**, we renumber A to 10.10.10.2 (diff answer)

Will Marcus Aurelius receive cached or new answer?
Are there dependencies between A and NS TTLs?

In zone:

- **NS:** ns1.sub.cachetest.net
- **A:** 10.10.10.10

Out of zone:

- **NS:** ns1.zurrundeddu.com

To resolve *.sub.cachetest.net, you need both NS and A. Are NS and A cached independently?

1. **t=0:** all Atlas VPs query (fills cache with NS and A)
2. **t=4800:** what happens? NS is expired; A is still in cache: do resolvers use the "cached A" or refresh it again?
   - Trick: at **t=540**, we renumber A to 10.10.10.2 (diff answer)

Will Marcus Aurelius receive cached or new answer?
Are there dependencies between A and NS TTLs?

```
sub.cachetest.net

In zone

NS: ns1.sub.cachetest.net
A: 10.10.10.10

Out of zone

NS: ns1.zurrundeddu.com
A: 10.10.10.10
```
Are there dependencies between A and NS TTLs?

To resolve *.sub.cachetest.net, you need both NS and A.

Are NS and A cached independently?

1. t=0: all Atlas VPs query (fills cache with NS and A)
2. t=4800: what happens? NS is expired; A is still in cache: do resolvers use the "cached A" or refresh it again?

trick: at t=540, we renumber A to 10.10.10.2 (diff answer)

Will Marcus Aurelius receive cached or new answer?

NS: ns1.sub.cachetest.net 3600
NS: ns1.zurrundeddu.com

A: 10.10.10.10
A: 10.10.10.10
A: 10.10.10.10
Are there dependencies between A and NS TTLs?

**In zone**
- **NS**: ns1.sub.cachetest.net
- **A**: 10.10.10.10
- TTL: 3600 seconds

**Out of zone**
- **NS**: ns1.zurruneddu.com
- **A**: 10.10.10.10
- TTL: 3600 seconds

To resolve *.sub.cachetest.net, you need both NS and A records. Are NS and A records cached independently?

1. **t=0**: all Atlas VPs query (fills cache with NS and A)
2. **t=4800**: what happens? NS is expired; A is still in cache: do resolvers use the "cached A" or refresh it again?

**Trick**: at **t=5400**, we renumber A to 10.10.10.2 (diff answer)

Will Marcus Aurelius receive cached or new answer?
Are there dependencies between A and NS TTLs?

To resolve *.sub.cachetest.net, you need both NS and A.
Are there dependencies between A and NS TTLs?

To resolve *.sub.cachetest.net, you need both NS and A

Are NS and A cached independently?

In zone

- **NS**: `ns1.sub.cachetest.net` 3600
- **A**: `10.10.10.10` 7200

Out of zone

- **NS**: `ns1.zurrunddeddu.com` 3600
- **A**: `10.10.10.10` 7200
Are there dependencies between A and NS TTLs?

To resolve *.sub.cachetest.net, you need both NS and A

Are NS and A cached independently?

1. $t=0$: all Atlas VPs query (fills cache with NS and A)
Are there dependencies between A and NS TTLs?

To resolve *.sub.cachetest.net, you need both NS and A

Are NS and A cached independently?

1. $t=0$: all Atlas VPs query (fills cache with NS and A)
2. $t=4800$: what happens? NS is expired; A is still in cache: do resolvers use the “cached A” or refresh it again?
Are there dependencies between A and NS TTLs?

To resolve *.sub.cachetest.net, you need both **NS** and **A**

**Are NS and A cached independently?**

1. t=0: all Atlas VPs query (fills cache with NS and A)
2. t=4800: what happens? NS is expired; A is still in cache: do resolvers use the "cached A" or refresh it again?
Are there dependencies between A and NS TTLs?

To resolve *.sub.cachetest.net, you need both NS and A

Are NS and A cached independently?

1. t=0: all Atlas VPs query (fills cache with NS and A)
2. t=4800: what happens? NS is expired; A is still in cache: do resolvers use the “cached A” or refresh it again?

trick: at t=540, we renumber A to 10.10.10.2 (diff answer)
Are there dependencies between A and NS TTLs?

To resolve *.sub.cachetest.net, you need both NS and A

Are NS and A cached independently?

1. t=0: all Atlas VPs query (fills cache with NS and A)
2. t=4800: what happens? NS is expired; A is still in cache: do resolvers use the “cached A” or refresh it again?

trick: at t=540, we renumber A to 10.10.10.2 (diff answer)

Will Marcus Aurelius receive cached or new answer?
Are they dependent? Yes, for in zone

Cache warms
NS Expires, A Valid (3600 < t < 7200)
Are they dependent? Yes, for in zone

Cache warms

DNS redirect: new A. Orig. NS and A valid
Original NS expires. Original A still valid
Both Original NS and A Original expired.
Orig. NS and A valid
DNS redirect: new A.

Why? Glues cause cache refresh
Are they dependent? Yes, for in zone

Cache warms

NS Expires, A Valid (3600 < t < 7200)

Are they dependent? Yes, for in zone

Cache warms

NS Expires, A Valid (3600 < t < 7200)
Are they dependent? Yes, for in zone

Cache warms

NS Expires, A Valid (3600 < t < 7200)

in zone: A refreshed (new server): dependent caching?
Are they dependent? Yes, for in zone

Cache warms

NS Expires, A Valid (3600 < t < 7200)

out-of-zone: cached A (old server): independent caching?
in zone: A refreshed (new server): dependent caching?
Are they dependent? Yes, for in zone

Why? Glues cause cache refresh

out-of-zone: cached A (old server): independent caching?
in zone: A refreshed (new server): dependent caching?
Are they dependent? Yes, for in zone

Cache warms

NS Expires, A Valid (3600 < t < 7200)

out-of-zone: cached A (old server): independent caching?
in zone: A refreshed (new server): dependent caching?
Are there dependencies between A and NS TTLs?

- Marcus Aurelius will notice “early” refreshed A for in-zone (in bailiwick)
- The way you configure your zone impacts caching, not only TTLs

src:
https://en.wikipedia.org/wiki/Marcus_Aurelius
CC BY-SA 3.0
Outline

Introduction

Parent vs Child

Zone configurations and Effective TTL

TTLs Use in the Wild
  Operators Notification

Caching (Longer TTL) vs Anycast
  Shorter vs Longer TTLs

Recommendation and Conclusions
TTLs Use in the Wild
How are TTLs used in the wild?

- There is no consensus how to choose TTLs
- But folks have to choose them anyway
- We use 5 lists:
  - Alexa
  - Majestic
  - Umbrella
  - .nl
  - Root (TLDs)
- We probe several records types
- We analyze child TTL values
- And discuss results with some operators
Most domains are out-of-bailiwick

<table>
<thead>
<tr>
<th></th>
<th>Alexa</th>
<th>Majestic</th>
<th>Umbre.</th>
<th>.nl</th>
<th>Root</th>
</tr>
</thead>
<tbody>
<tr>
<td>responsive</td>
<td>988654</td>
<td>928299</td>
<td>783343</td>
<td>5454833</td>
<td>1535</td>
</tr>
<tr>
<td>CNAME</td>
<td>50981</td>
<td>7017</td>
<td>452711</td>
<td>9436</td>
<td>0</td>
</tr>
<tr>
<td>SOA</td>
<td>12741</td>
<td>8352</td>
<td>59083</td>
<td>12268</td>
<td>0</td>
</tr>
<tr>
<td>responsive NS</td>
<td>924932</td>
<td>912930</td>
<td>271549</td>
<td>5433129</td>
<td>1535</td>
</tr>
<tr>
<td>Out only</td>
<td>878402</td>
<td>873447</td>
<td>244656</td>
<td>5417599</td>
<td>748</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>ratio out only</th>
<th>95.0%</th>
<th>95.7%</th>
<th>90.1</th>
<th>99.7%</th>
<th>48.7%</th>
</tr>
</thead>
<tbody>
<tr>
<td>In only</td>
<td>37552</td>
<td>28577</td>
<td>20070</td>
<td>12586</td>
<td>654</td>
<td></td>
</tr>
<tr>
<td>Mixed</td>
<td>8978</td>
<td>10906</td>
<td>6823</td>
<td>2941</td>
<td>133</td>
<td></td>
</tr>
</tbody>
</table>

- Out of bailiwick (out-of-zone): **records are cached independently** (no glues)
- Your chosen TTLs values for different records will be respected
NS records have longer TTLs (>24h)

- > 60% NS records are long (Good for caching and performance)
- But 40% are one hour or less (not so good)
NS records have longer TTLs (>24h)

- > 60% NS records are long (Good for caching and performance)
- But 40% are one hour or less (not so good)
A records TTLs far shorter than NS

The CDF graphs show the distribution of answers TTL (in hours) for various domains, including Alexa, Majestic, Umbrella, .nl, and root. The graphs indicate that shorter A records TTLs lead to poor caching.
A records TTLs far shorter than NS

Shorter A records TTLs leads to poor caching.
A records TTLs far shorter than NS

Shorter A records TTLs leads to poor caching
Operators Notification: 3 changed their TTLs

- We found **34 TLDs** with short TTL for NSes ($\leq 30$min)
- **We notified** 8 ccTLDs
- 3 TLDs *increased their TTL to 1 day* after our notification
  - .uy, and
  - another in Africa
  - and another in the Middle-East
.uy latency reduced a lot!

- .uy NS TTL from 300s to 86400s

![Graph showing RTT distribution for TTL 300s and TTL 86400s](image)

**Figure 4:** RTT from RIPE Atlas VPs for NS .uy queries (NS)
.uy latency reduced a lot!

- .uy NS TTL from 300s to 86400s

Figure 4: RTT from RIPE Atlas VPs for NS .uy queries (NS)
.uy latency reduced a lot!

- .uy NS TTL from 300s to 86400s: lower latency for clients

-----

Figure 5: RTT from RIPE Atlas VPs for NS .uy queries (NS)

Median RTT improves by 20ms; 75%ile by 152ms
uy latency reduced for all regions

Check for Atlas location bias

Figure 6: Median RTT as seen by RIPE Atlas VPs per region

Longer TTL → longer caching → faster answers
Check for Atlas location bias

Figure 7: Median RTT as seen by RIPE Atlas VPs per region

Longer TTL → longer caching → faster answers

Up to 150ms median latency reduction (AF)
We are no Luiz Suárez... but

- We still helped Uruguayan .uy users
- And two other countries:
  - One in East Africa
  - Another one in the Middle East
- Experiment proving how TTLs are important for performance

src: https://commons.wikimedia.org/wiki/File:Luis_Su%C3%A1rez_2018.jpg CC BY-SA 3.0
Longer TTLs are like the old Turbo button

- Some DNS OPs spend 1000s too reduce latency
- Longer TTLs improve latency at **zero cost**

*src: wikipedia.org*
Caching (Longer TTL) vs Anycast
Caching vs Anycast

- People and CDNs spend lots on huge anycast deployments
- OPs could say: “I’ll have short TTL since I use anycast”, because anycast can make it up for it.
- Does anycast really beats caching?
Caching vs Anycast: experiment

Probes + Resolver

[Diagram of Probes + Resolver]

Which one is faster?
Caching vs Anycast: experiment

Probes + Resolver

Unicast (EC2)

TTL 60s (no caching)

TTL 86400 (good caching)

Which one is faster?
Caching vs Anycast: experiment

Probes + Resolver

Unicast (EC2)

TTL 86400 (good caching)

Anycast (Route53)

TTL 60s (no caching)

Which one is faster?
Caching vs Anycast: experiment

Probes + Resolver

Unicast (EC2)
TTL86400 (good caching)

Anycast (Route53)
TTL60s (no caching)

Which one is faster?
Caching vs Anycast: experiment

Which one is faster?

Probes + Resolver

Unicast (EC2)
TTL86400 (good caching)

FRA

TTL60s (no caching)
Anycast (Route53)
Caching vs Anycast: experiment

Probes + Resolver

Unicast (EC2)
TTL86400 (good caching)

Which one is faster?

TTL60s (no caching)
Anycast (Route53)
Caching near client beats even great server infrastructure!

- Anycast TTL60 (no cache): \textbf{29.96ms} (median)
- Unicast TTL86400 (cache): \textbf{7.38ms} (median):
  - \textbf{22ms median latency reduction}

- Query load: \textbf{77\% down} with caching
- so TTLs matter more for performance
  - (anycast is great to many things too, DDoS for example \cite{2})
  - We strongly recommend anycast \cite{5}
TTLs (caching) matter more than anycast

- Caching near client beats even great server infrastructure!
  - Anycast TTL60 (no cache): **29.96ms** (median)
  - Unicast TTL86400 (cache): **7.38ms** (median):
    - 22ms median latency reduction
- Query load: 77% down with caching
- so TTLs matter more for performance
  - (anycast is great to many things too, DDoS for example [2])
  - We strongly recommend anycast [5]
TTLs (caching) matter more than anycast

- Caching near client beats even great server infrastructure!
  - Anycast TTL60 (no cache): **29.96ms** (median)
  - Unicast TTL86400 (cache): **7.38ms** (median):
  - **22ms median latency reduction**

- Query load: **77% down** with caching
- so TTLs matter more for performance
  - (anycast is great to many things too, DDoS for example [2])
  - We strongly recommend anycast [5]
TTLs (caching) matter more than anycast

- Caching near client beats even great server infrastructure!
  - Anycast TTL60 (no cache): **29.96ms** (median)
  - Unicast TTL86400 (cache): **7.38ms** (median):
    - **22ms median latency reduction**
- Query load: **77% down** with caching
- so TTLs matter more for performance
  - (anycast is great to many things too, DDoS for example [2])
  - **We strongly recommend anycast** [5]
Reasons for Longer or shorter TTLs

- **Longer caching:**
  - faster responses
  - lower DNS traffic
  - more robust to DDoS attack [4]

- **Shorter caching:**
  - faster operational changes
  - useful for DNS redirect based DDoS
  - DNS-load balance

Organizations must weight these trade-offs to find a good balance
Recommendation and Conclusions
Conclusions

- **Recommendation:** longer TTLs (1 day) if you can
  - unless using CDN load-balancing or DNS-redir DDoS
- Why? Because it can save you more than 50ms or more
  - But keep on using anycast too [2, 5]
- People have designed caches; use them wisely
- **Should you reconsider your TTLs as well?**

- Paper: [https://www.isi.edu/~johnh/PAPERS/Moura19b.html](https://www.isi.edu/~johnh/PAPERS/Moura19b.html)
- IETF draft: [draft-moura-dnsop-authoritative-recommendations](https://www.isi.edu/~johnh/PAPERS/Moura19b.html)

Verfploeter: Broad and load-aware anycast mapping.

Anycast vs. DDoS: Evaluating the November 2015 root DNS event.

Cache me if you can: Effects of DNS Time-to-Live (extended).


When the dike breaks: Dissecting DNS defenses during DDoS.


Recursives in the wild: Engineering authoritative DNS servers.


Anycast latency: How many sites are enough?