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]

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

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

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

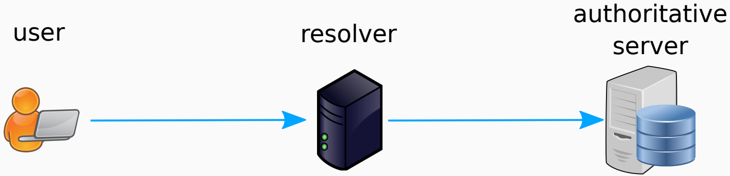
Our research on DNS over the last years

Our research on DNS security/stability:

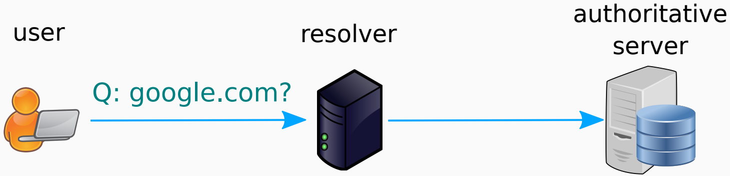
- **Anycast and DDoS**: IMC 2016 [2]
- **Resolvers**: IMC 2017 [5]
- **Anycast Engineering**: IMC 2017 [1]
- **Caching and DDoS**: IMC 2018 [4]
- **Caching and TTL, and performance:** IMC 2019 [3]
    - (this paper)

Introduction

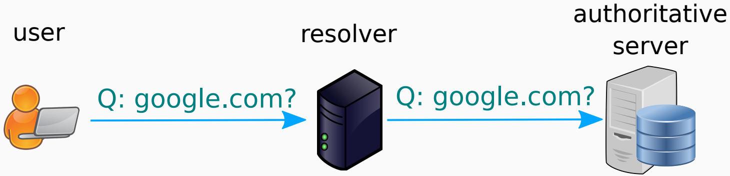
The role of TTL



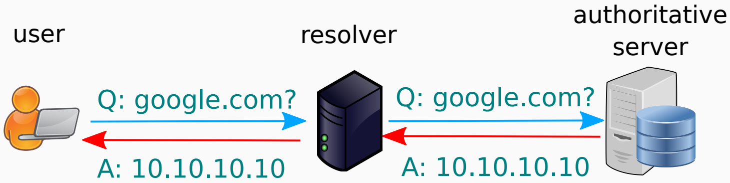
The role of TTL



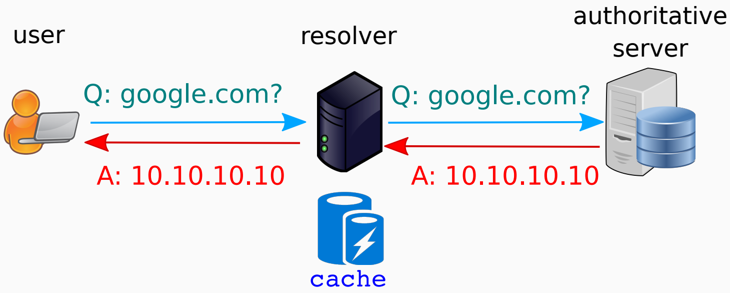
The role of TTL



The role of TTL



The role of TTL



user

resolver

authoritative server

Q: google.com?

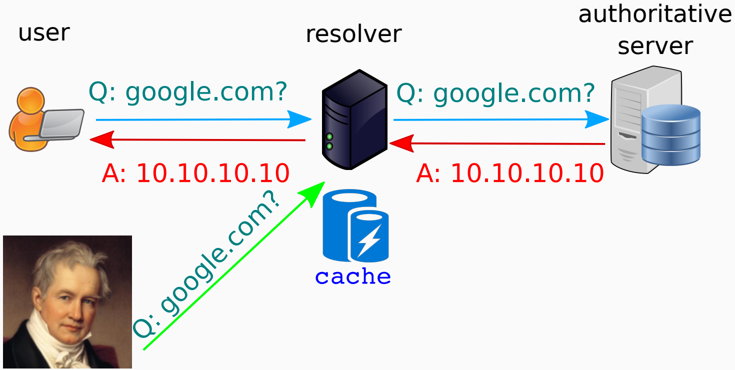
Q: google.com?

A: 10.10.10.10

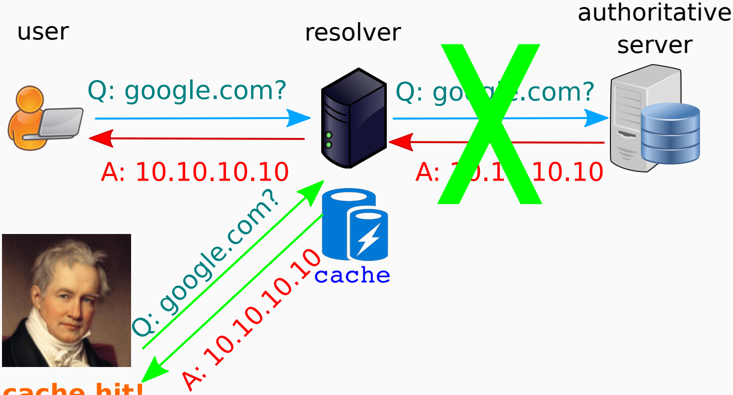
A: 10.10.10.10

cache

The role of TTL



The role of TTL



user

resolver

authoritative server

Q: google.com?

A: 10.10.10.10

Q: google.com?

A: 10.10.10.10

X

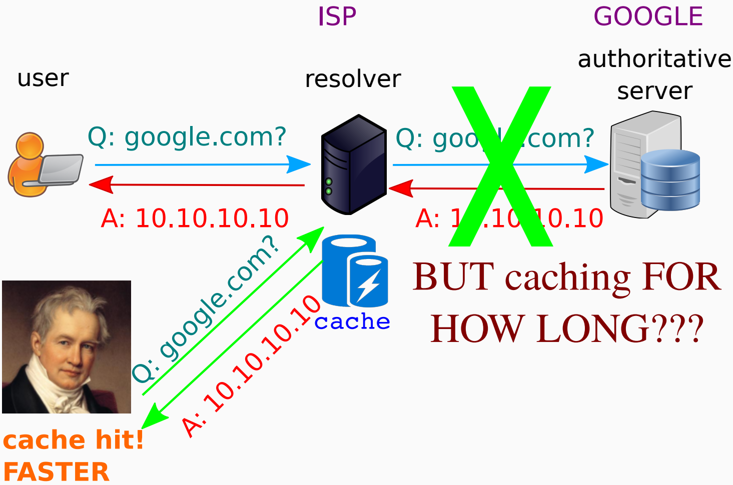
cache

Q: google.com?

A: 10.10.10.10

cache hit!
FASTER

The role of TTL



ISP

GOOGLE

user

resolver

authoritative server

Q: google.com?

A: 10.10.10.10

Q: google.com?

A: 10.10.10.10

X

cache

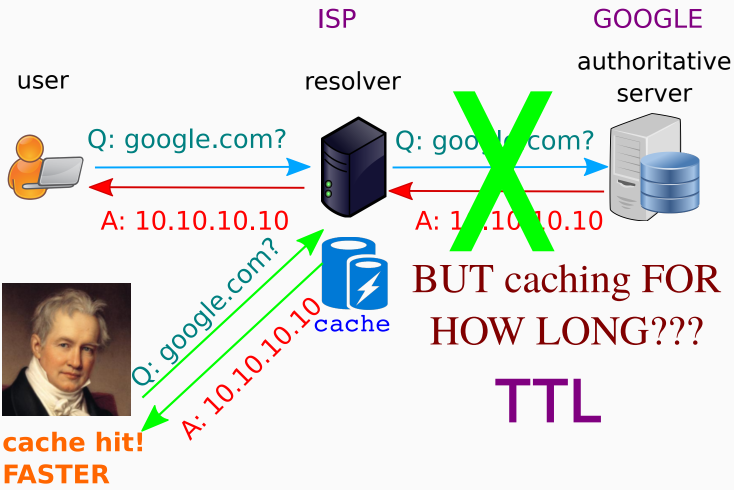
Q: google.com?

A: 10.10.10.10

cache hit!
FASTER

BUT caching FOR HOW LONG???

The role of TTL



ISP

GOOGLE

user

resolver

authoritative server

Q: google.com?

A: 10.10.10.10

Q: google.com?

A: 10.10.10.10

X

Q: google.com?

A: 10.10.10.10

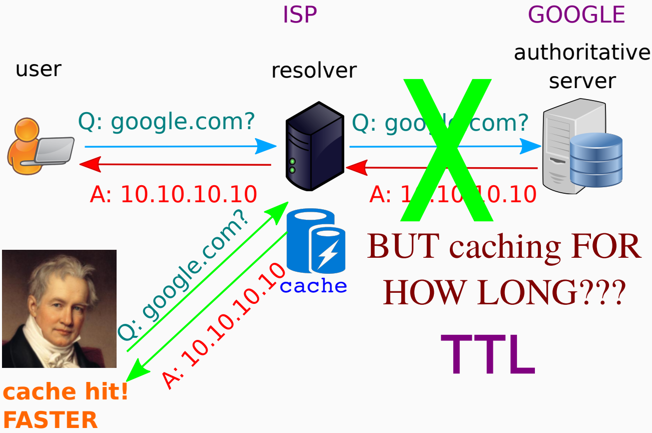
cache

cache hit!
FASTER

BUT caching FOR HOW LONG???

TTL

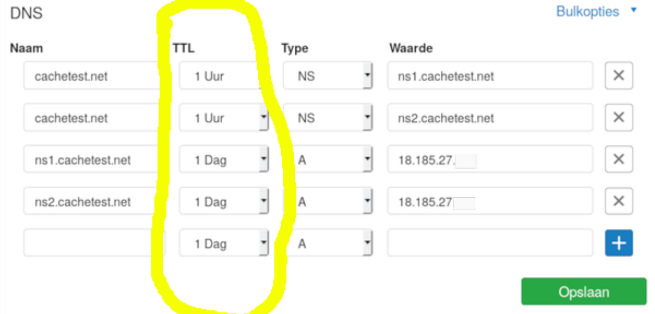
The role of TTL



- **TTL controls caching**
  - SIGNAL from auth servers to resolvers: **maximum** time length
- Caching is VERY important for performance
  - improves user experience *(aka happy eyeballs)*

And you must set TTLs

- Say you register `cachetest.net`



DNS

Naam	TTL	Type	Waarde
cachetest.net	1 Uur	NS	ns1.cachetest.net
cachetest.net	1 Uur	NS	ns2.cachetest.net
ns1.cachetest.net	1 Dag	A	18.185.27.[redacted]
ns2.cachetest.net	1 Dag	A	18.185.27[redacted]
	1 Dag	A	

What TTL values are good?

Operators:

- are given little guidance today about correct values
- and are resistant to (scared to!) make changes
  - **“if it ain’t broke don’t fix it”**

We think we can help

Figure 1: DNS ops chaging TTLs. src: trainworld.be

Our contribution

Our research contributions:

1. The *effective* TTL comes from **multiple** places
   - Parent authoritative servers
   - Child authoritative servers
   - Both NS and A records (sometimes)
2. Currently popular TTLs are unnecesssarily 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**

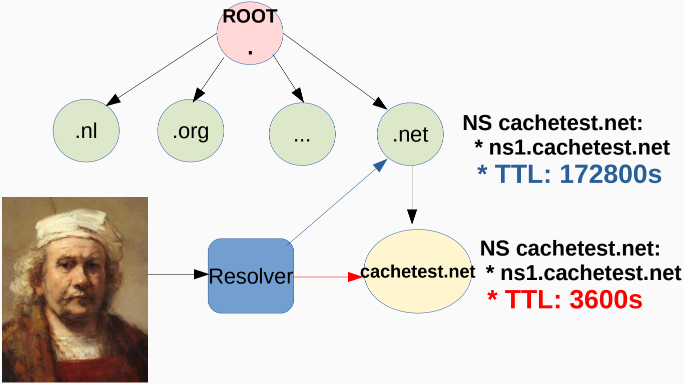
The rest of this talk

1. Parent vs Child: which TTLs to resolvers believe?
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`



NS cachetest.net:
* ns1.cachetest.net
* TTL: 172800s

NS cachetest.net:
* ns1.cachetest.net
* TTL: 3600s

Which TTL will Rembrandt use?
Parent (172800s) or child (TTL: 3600s)

Are resolvers parent- or child-centric?

Parent vs Child experiment

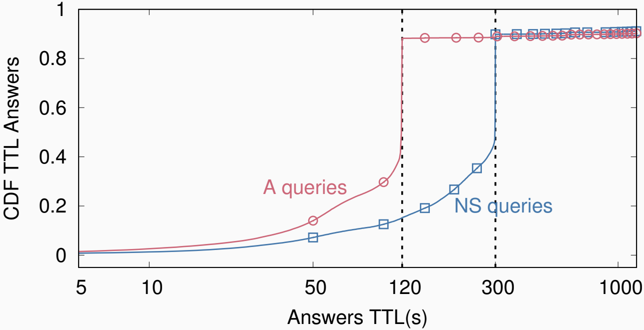
- Test with experiment on `.uy`: (2019-02-14)

Parent	NS TTL	172800s
	A TTL	172800s
Child	NS TTL	300s
	A TTL	120s

- We query with 15k Atlas VPs multiple times, every 10min
- We analyze TTL values received at VPs

Most Atlas VPs resolvers are child-centric

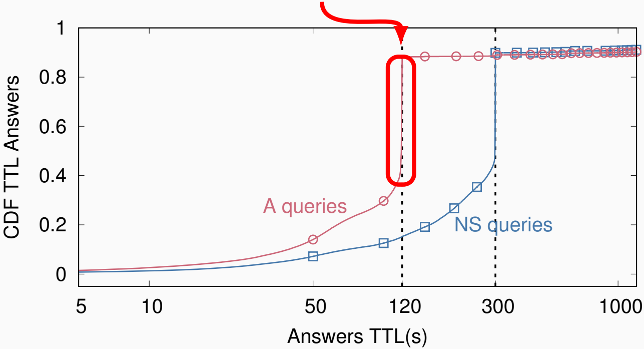
Figure 2: Observed TTLs from Atlas VPs for .uy-NS and a.nic.uy-A queries.



- 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.



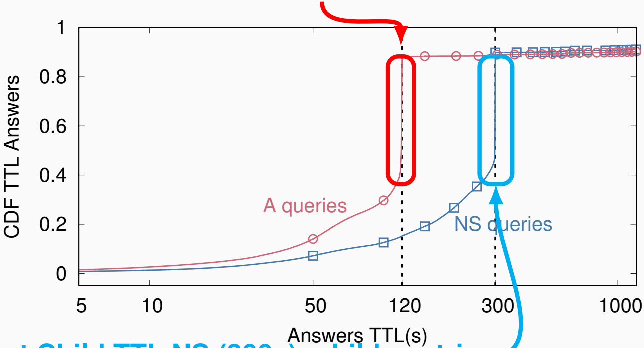
Spike at Child TTL A (120s) : most resolvers are child centric

- 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



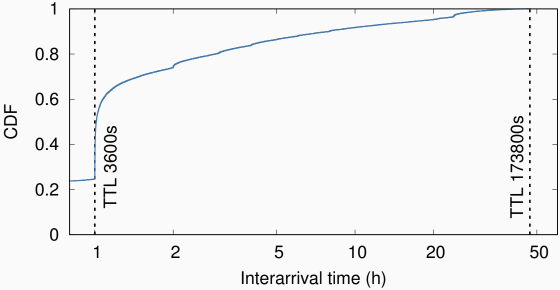
Spike at Child TTL NS (300s): child centric

- 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**

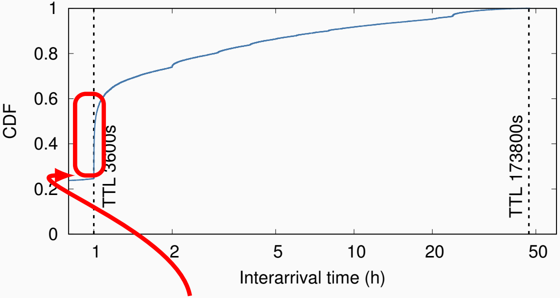


We confirmed this with a second-level domain (paper)

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**



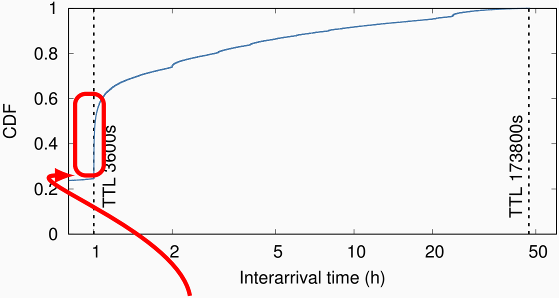
Spike at Child TTL A (3600s): confirm child centric for TLD

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**

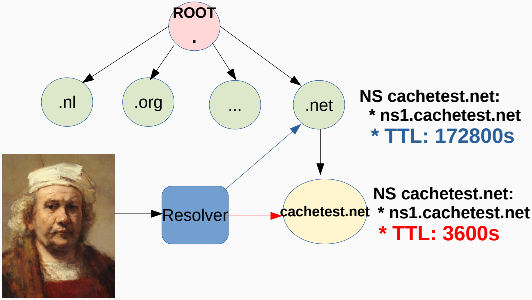


Spike at Child TTL A (3600s): confirm child centric for TLD

We confirmed this with a second-level domain (paper)

Most resolvers will use child TTLs

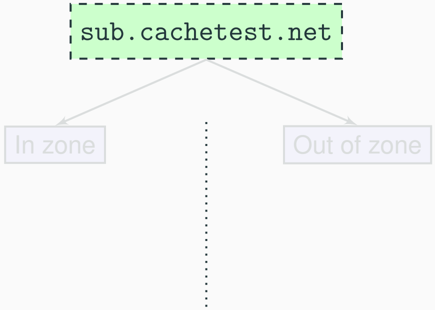
- Rembrant (and users) mostly use child TTLs
- The **Child TTL** controls caching (most times)



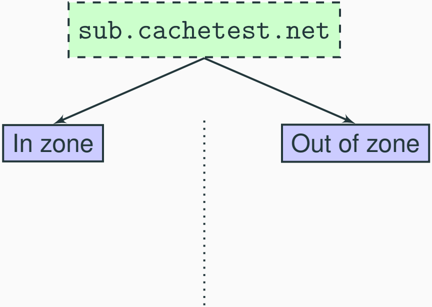
Which TTL will Rembrandt use?
Parent (172800s) or child (TTL: 3600s)

Zone configurations and Effective TTL

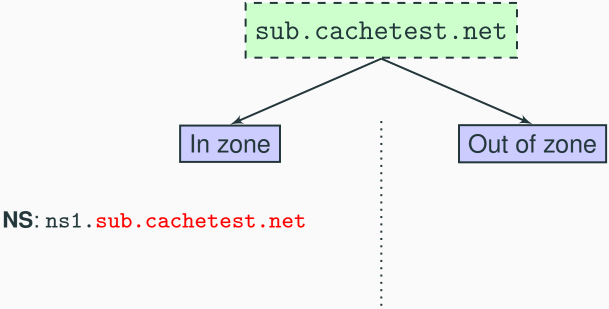
Are there dependencies between A and NS TTLs?



Are there dependencies between A and NS TTLs?



Are there dependencies between A and NS TTLs?



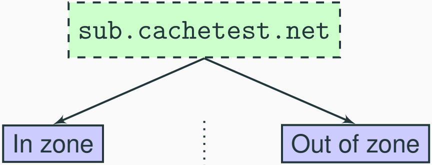
sub.cachetest.net

In zone

Out of zone

NS: ns1.sub.cachetest.net

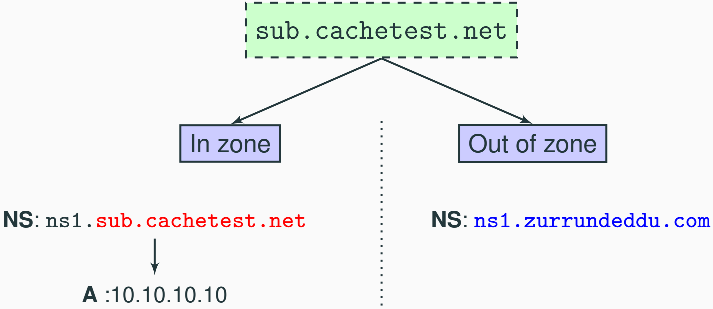
Are there dependencies between A and NS TTLs?



NS: ns1.sub.cachetest.net

NS: ns1.zurrundeddu.com

Are there dependencies between A and NS TTLs?



sub.cachetest.net

In zone

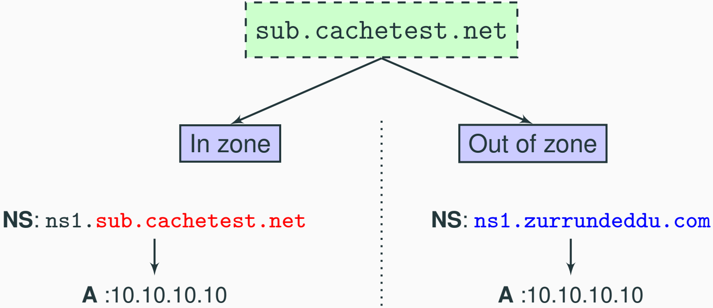
Out of zone

NS: ns1.sub.cachetest.net

A :10.10.10.10

NS: ns1.zurrundeddu.com

Are there dependencies between A and NS TTLs?



sub.cachetest.net

In zone

Out of zone

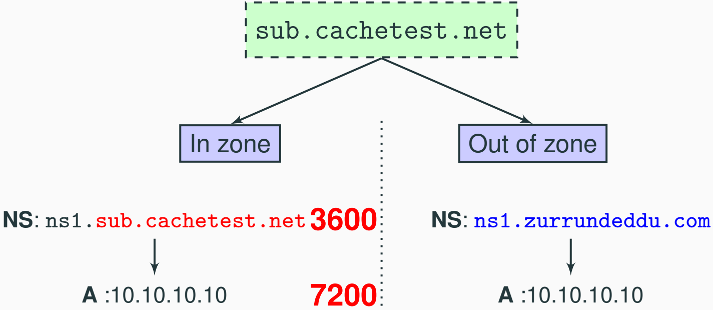
NS: ns1.sub.cachetest.net

A :10.10.10.10

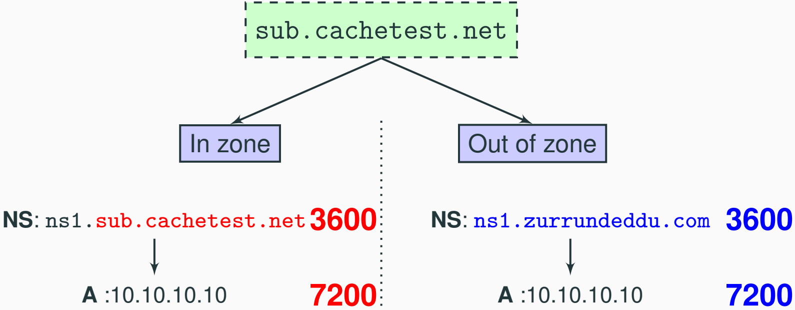
NS: ns1.zurrundeddu.com

A :10.10.10.10

Are there dependencies between A and NS TTLs?



Are there dependencies between A and NS TTLs?



`sub.cachetest.net`

In zone

NS: `ns1.sub.cachetest.net` **3600**

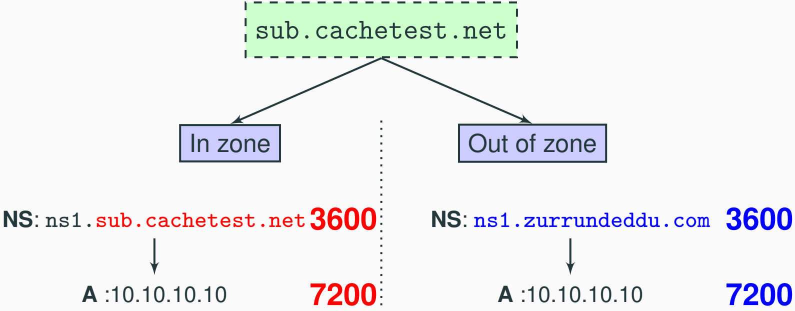
A :10.10.10.10 **7200**

Out of zone

NS: `ns1.zurrundeddu.com` **3600**

A :10.10.10.10 **7200**

Are there dependencies between A and NS TTLs?



sub.cachetest.net

In zone

Out of zone

NS: `ns1.sub.cachetest.net` **3600**

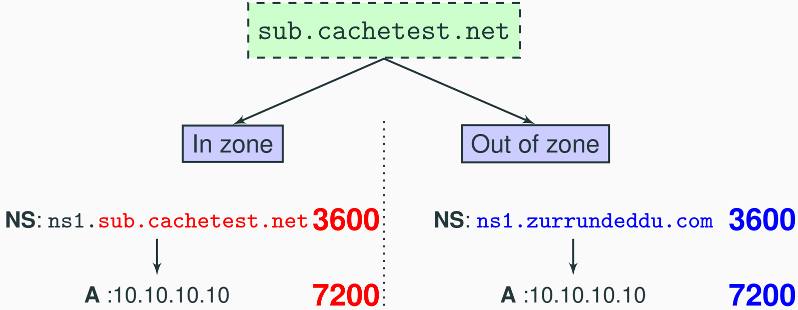
A :10.10.10.10 **7200**

NS: `ns1.zurrundeddu.com` **3600**

A :10.10.10.10 **7200**

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

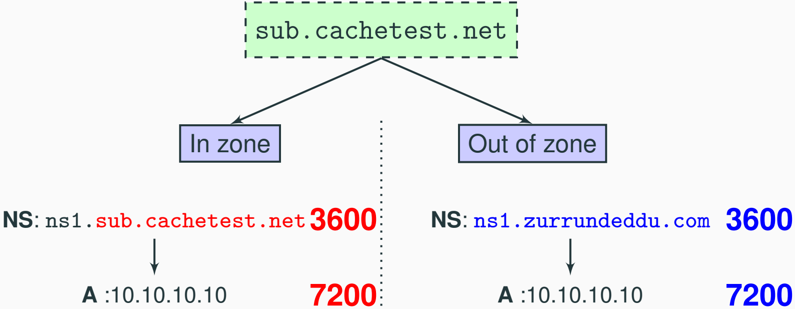
Are there dependencies between A and NS TTLs?



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

Are NS and A cached independently?

Are there dependencies between A and NS TTLs?



sub.cachetest.net

In zone

Out of zone

NS: ns1.sub.cachetest.net **3600**

A :10.10.10.10 **7200**

NS: ns1.zurrundeddu.com **3600**

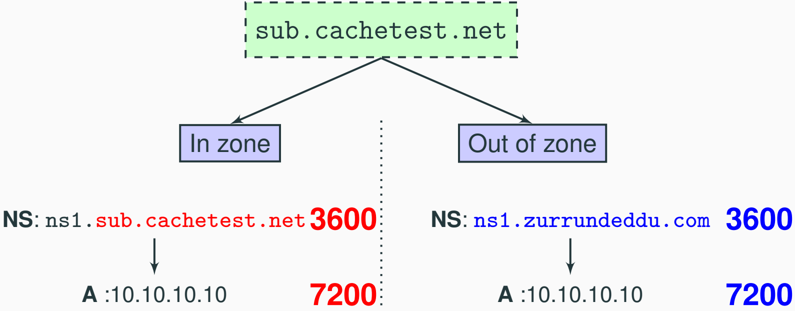
A :10.10.10.10 **7200**

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?



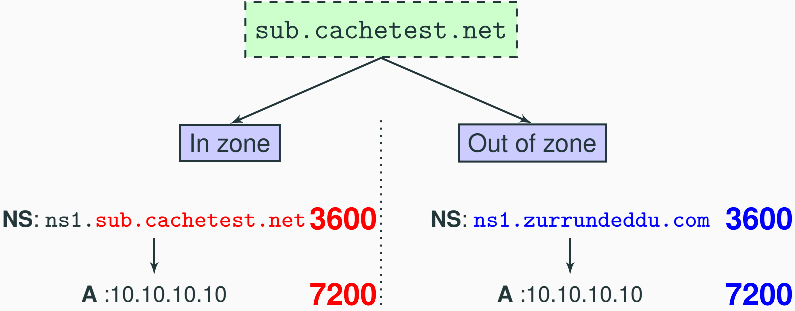
To resolve *.sub.cachetest.net, you need both **NS** and **A**

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?



`sub.cachetest.net`

In zone

NS: `ns1.sub.cachetest.net` **3600**

A :10.10.10.10 **7200**

Out of zone

NS: `ns1.zurrundeddu.com` **3600**

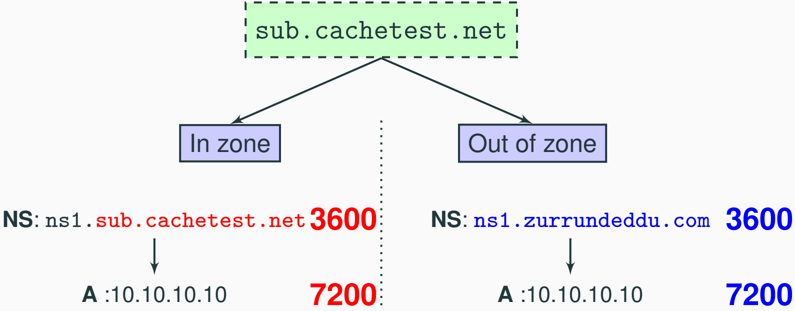
A :10.10.10.10 **7200**

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?



sub.cachetest.net

In zone

Out of zone

NS: `ns1.sub.cachetest.net` **3600**

A :10.10.10.10 **7200**

NS: `ns1.zurrundeddu.com` **3600**

A :10.10.10.10 **7200**

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

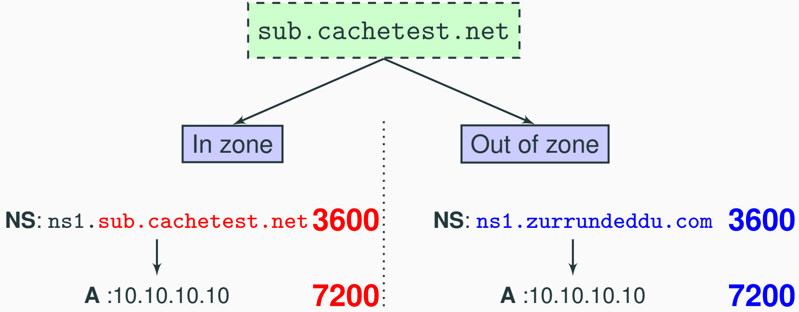
Are NS and A cached independently?

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?



sub.cachetest.net

In zone

NS: ns1.sub.cachetest.net 3600

A :10.10.10.10 7200

Out of zone

NS: ns1.zurrundeddu.com 3600

A :10.10.10.10 7200

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

Are NS and A cached independently?

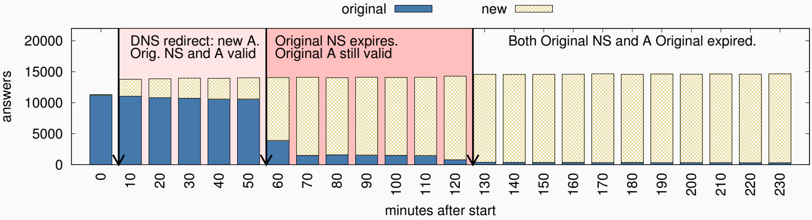
1. t=0: all Atlas VPs query (fills cache with NS and A)
2. t=4800: what happens ? NS is expired; A is still in cache:

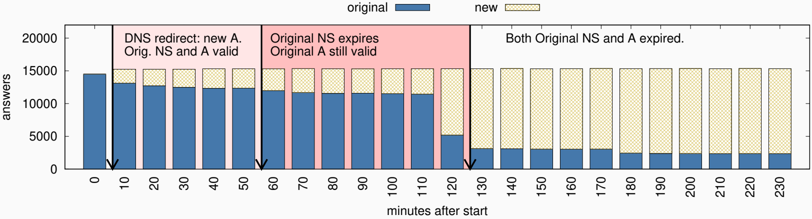
do resolvers use the “cached A” or refresh it again?

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

Will Marcus Aurelius receive cached or new answer?

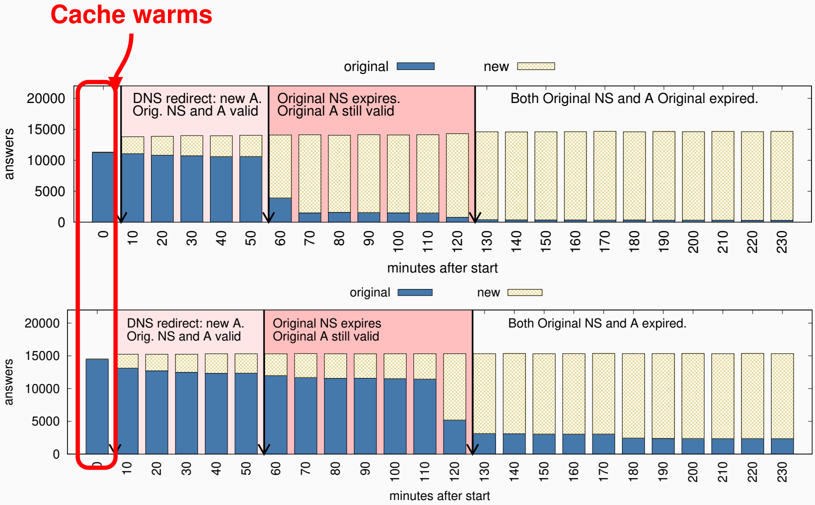
Are they dependent? Yes, for in zone





Are they dependent? Yes, for in zone

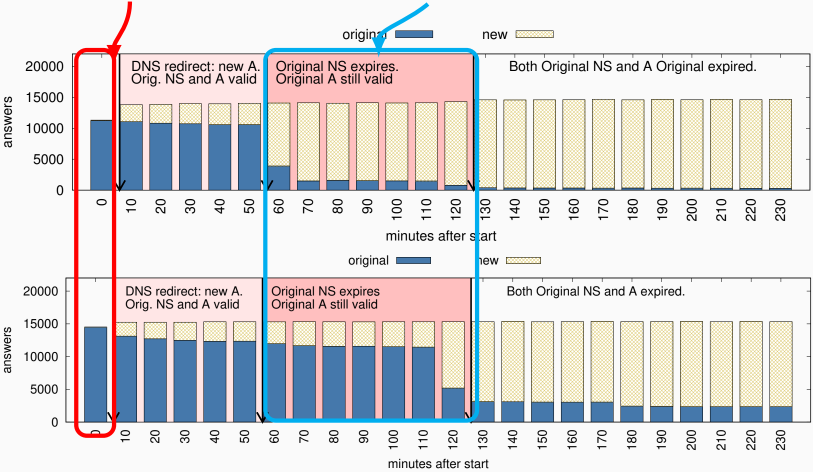
Cache warms



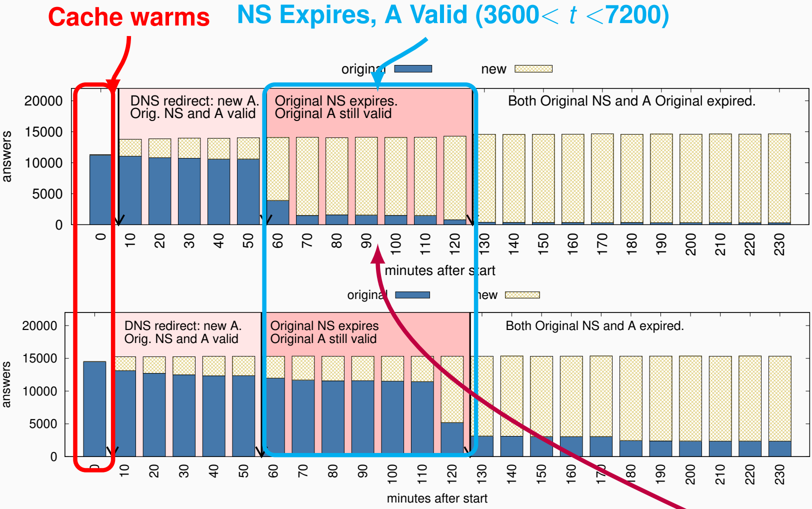
Are they dependent? Yes, for in zone

Cache warms

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



Are they dependent? Yes, for in zone



Cache warms

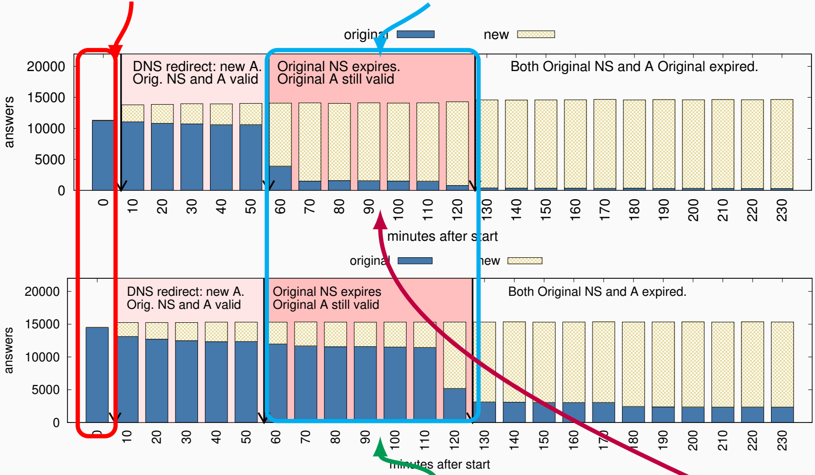
NS Expires, A Valid (3600$< t <$7200)

in zone: refreshed A (new server): dependent caching?

Are they dependent? Yes, for in zone

Cache warms

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



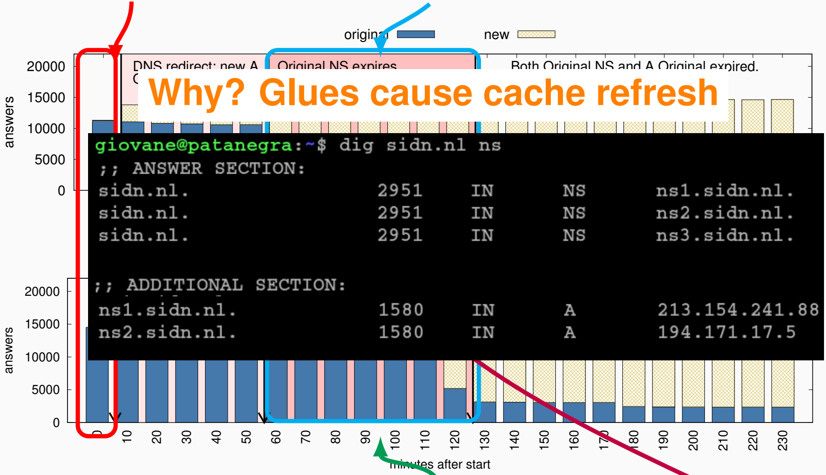
out-of- zone: cached A (old server): independent caching?

in zone: refreshed A (new server): dependent caching?

Are they dependent? Yes, for in zone

Cache warms

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



Why? Glues cause cache refresh

```
giovane@patanegra:~$ dig sidn.nl ns
;; ANSWER SECTION:
sidn.nl.                2951    IN      NS      ns1.sidn.nl.
sidn.nl.                2951    IN      NS      ns2.sidn.nl.
sidn.nl.                2951    IN      NS      ns3.sidn.nl.

;; ADDITIONAL SECTION:
ns1.sidn.nl.            1580    IN      A       213.154.241.88
ns2.sidn.nl.            1580    IN      A       194.171.17.5
```

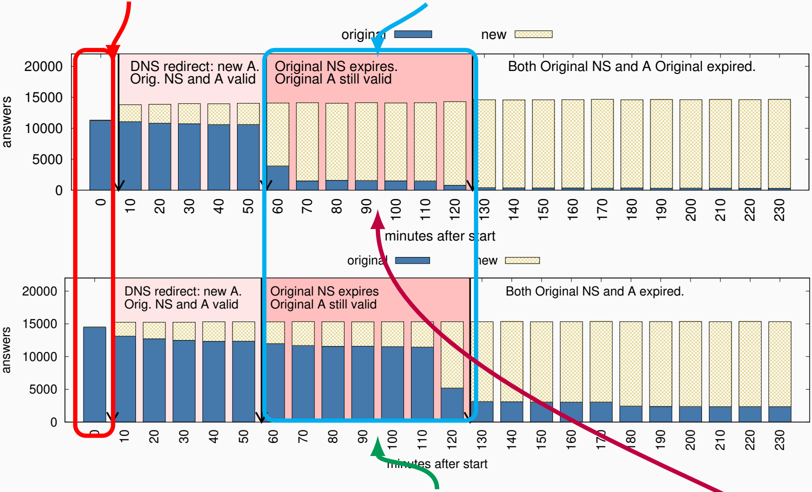
out-of- zone: cached A (old server): independent caching?

in zone: refreshed A (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: refreshed A (new server): dependent caching?

Are there dependencies between A and NS TTLs?

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

- Marcus Aurelius will notice“early” refreshed A for in-zone (in bailiwick)
- Zone configuration impacts caching too, not just TTLs

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?

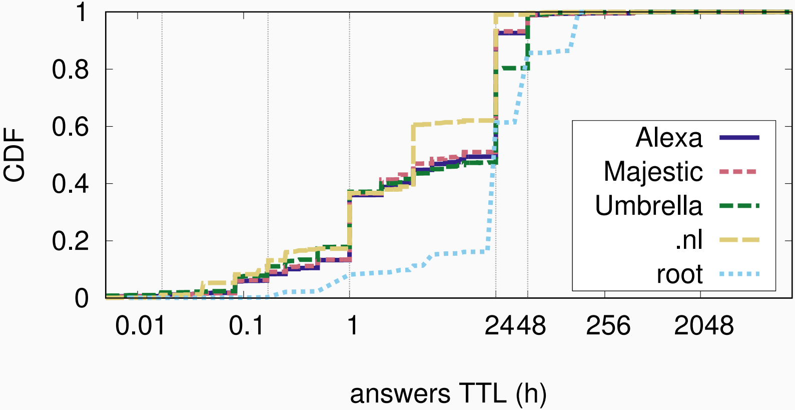
- There is no consensus how to choose TTLs
- But folks have to choose them anyway
- We use 5 lists:
  - Alexa
  - Majestic
  - Umbrella
  - `.nl` Zone
  - Root Zone (TLDs)
- We probe several records types
- We analyze **child TTL** values
- We discussed results with some operators

Most domains are out-of-bailiwick

	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 out only	95.0%	95.7%	90.1	99.7%	48.7%
In only	37552	28577	20070	12586	654
Mixed	8978	10906	6823	2941	133

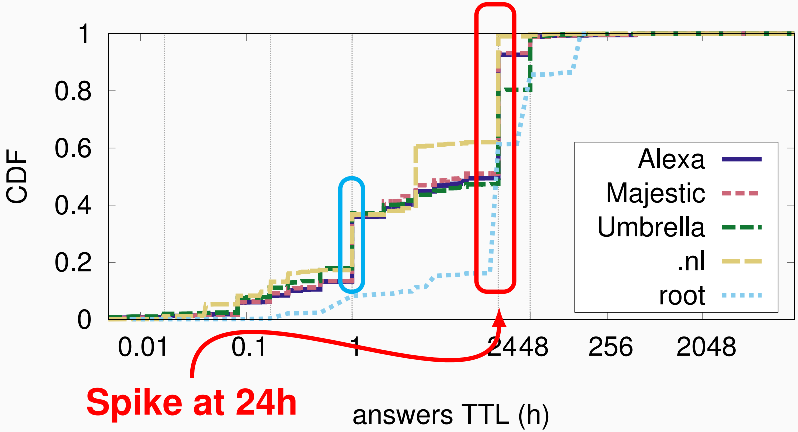
- Out of bailiwick (out-of-zone):
    - **records are cached independently** (no glues)
- Chosen TTLs values for different records will be respected

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)

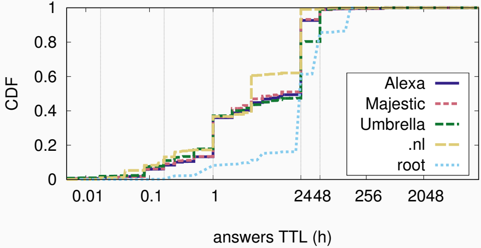
NS records have longer TTLs (>24h)

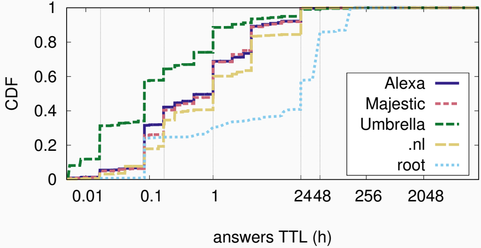


Spike at 24h

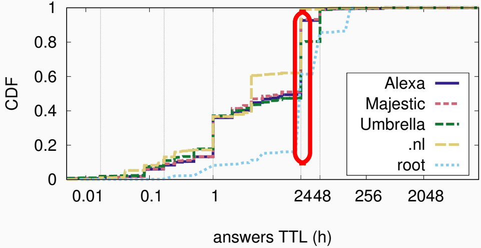
- > 60% NS records are long
    - (**Good** for caching and performance)
- But 40% are one hour or less (not so good)

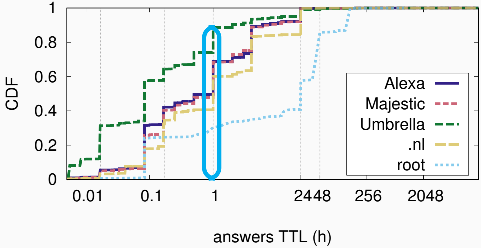
A records have far shorter TTLs than NS



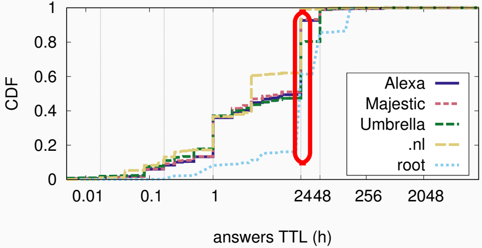


A records have far shorter TTLs than NS

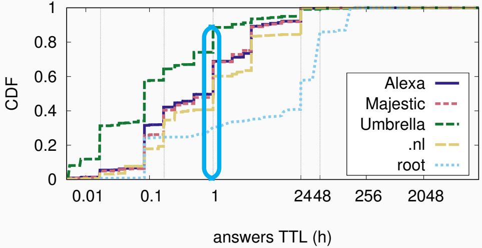




A records have far shorter TTLs than NS



Shorter A records TTLs leads to poor caching



Operators Notification: 3 changed their TTLs

- We found **34 TLDs** with short NS TTL ($<=$30min)
  - **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 changed: 300s to 86400s

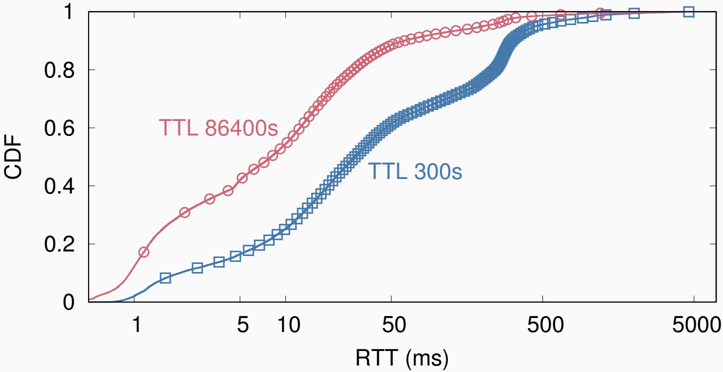


Figure 4: RTT from RIPE Atlas VPs for NS .uy queries (NS)

.uy latency reduced a lot!

- .uy NS TTL changed: 300s to 86400s

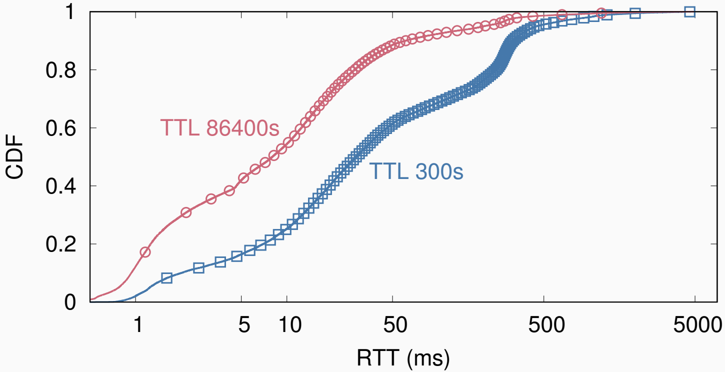


Figure 4: RTT from RIPE Atlas VPs for NS .uy queries (NS)

.uy latency reduced a lot!

- .uy NS TTL changed: 300s to 86400s: lowered client latency

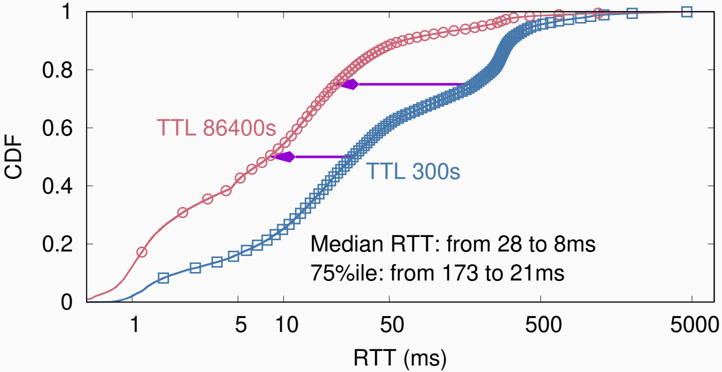


Figure 5: RTT from RIPE Atlas VPs for NS .uy queries (NS)

Median RTT improves by 20ms; 75%ile by 152ms

.uy latency reduced for all regions

Check for Atlas location bias

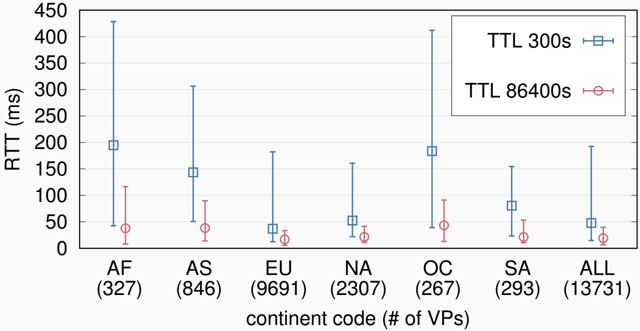


Figure 6: Median RTT as seen by RIPE Atlas VPs per region

Longer TTL → longer caching → faster answers

.uy latency reduced for all regions

Check for Atlas location bias

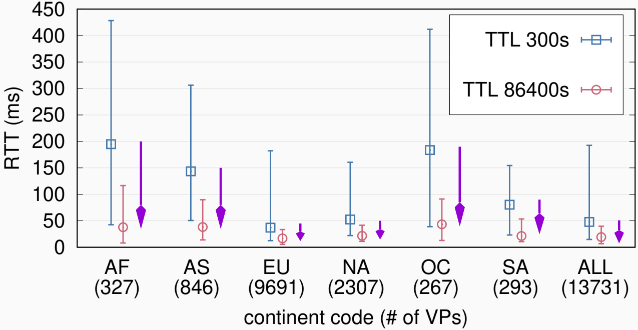


Figure 7: Median RTT as seen by RIPE Atlas VPs per region

Longer TTL → longer caching → faster answers

Up to 150ms median latency reduction (AF)

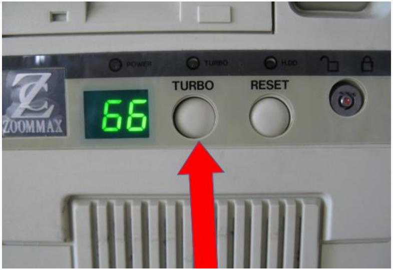
We are no Luiz Suárez... but

- We still helped Uruguayan `.uy` users
- And two other countries:
  - One in East Africa
  - Another one in the Middle East
- Experiment proved 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 of *(your currency here)* too reduce latency
- Longer TTLs improve latency at **zero cost**



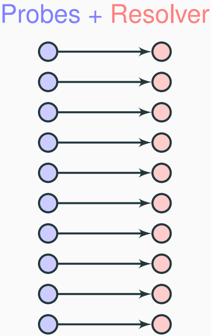
src: wikipedia.org

Caching (Longer TTL) vs Anycast

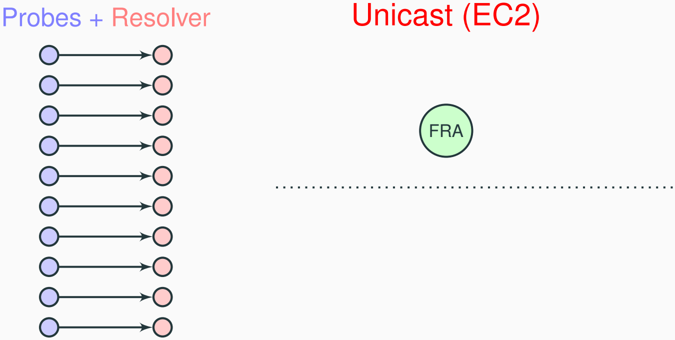
Caching vs Anycast

- There are many large, expensive anycast deployments
- OPs could say:
  - “*I’ll have short TTL since I use anycast*”,
  - because anycast can make it up for it.
- **Does anycast actually beat caching?**

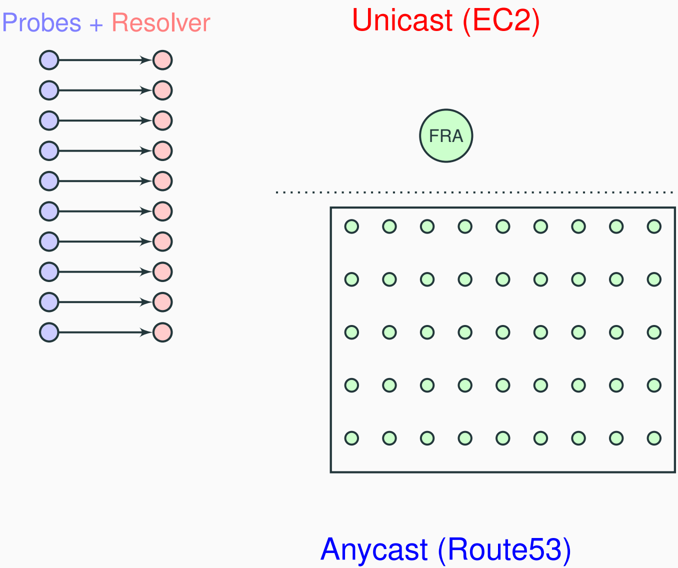
Caching vs Anycast: experiment



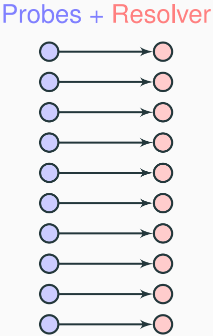
Caching vs Anycast: experiment



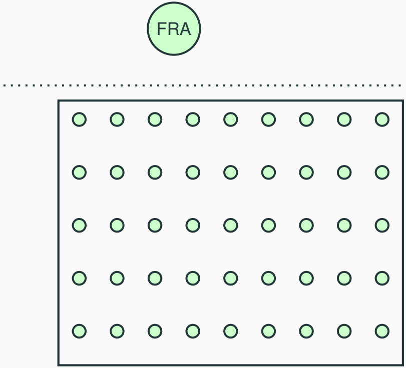
Caching vs Anycast: experiment



Caching vs Anycast: experiment

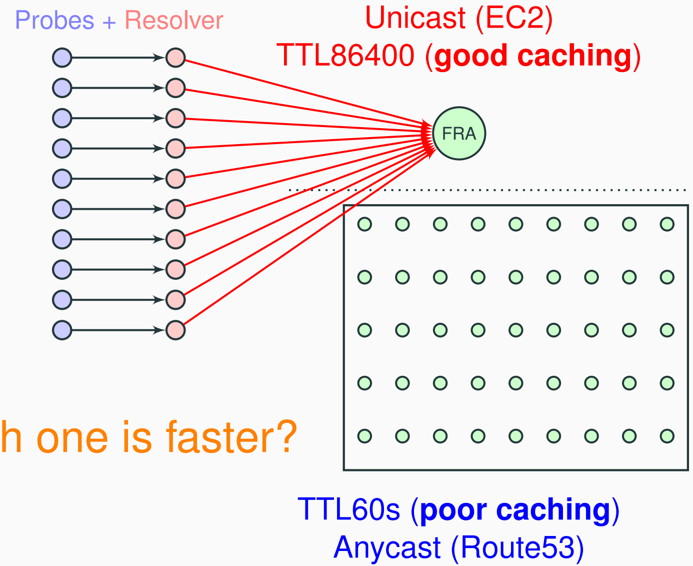


Unicast (EC2)
TTL86400 (**good caching**)



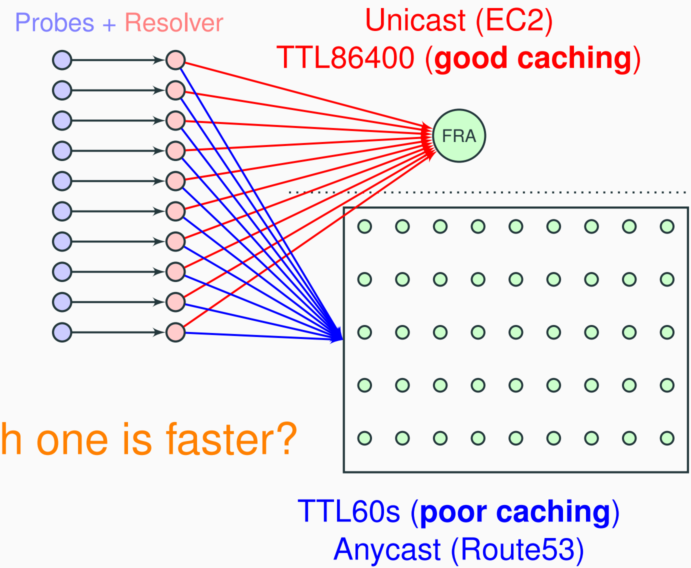
TTL60s (**poor caching**)
Anycast (Route53)

Caching vs Anycast: experiment



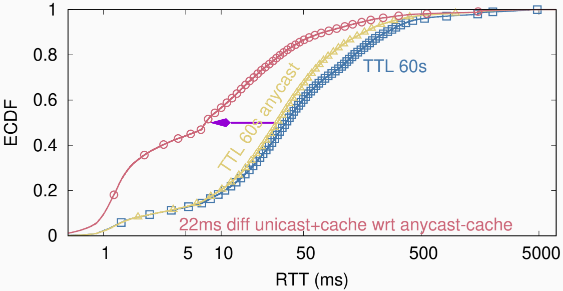
Which one is faster?

Caching vs Anycast: experiment



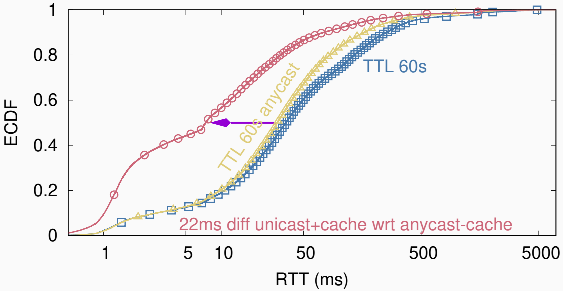
Which one is faster?

TTLs (caching) matter more than anycast



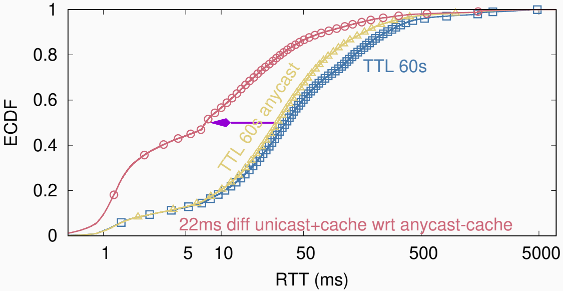
- Near-client caching beats great infrastructure!
  - Anycast TTL60 (no cache): **29.96ms** (median)
  - Unicast TTL86400 (cache): **7.38ms** (median):
  - **22ms median latency reduction**
- Query load: **77% down** with caching
- Conclusion: TTLs matter more for performance
  - (anycast is needed for other things too, e.g. DDoS [2])
  - **We still strongly recommend using anycast** [5]

TTLs (caching) matter more than anycast



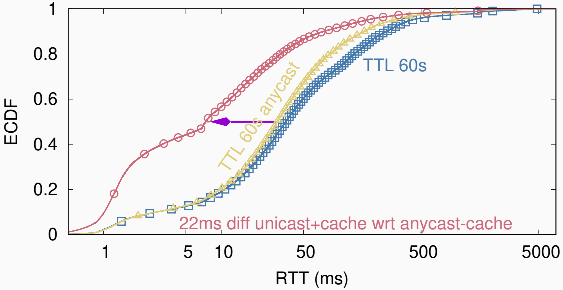
- Near-client caching beats great infrastructure!
  - Anycast TTL60 (no cache): **29.96ms** (median)
  - Unicast TTL86400 (cache): **7.38ms** (median):
  - **22ms median latency reduction**
- Query load: **77% down** with caching
- Conclusion: TTLs matter more for performance
  - (anycast is needed for other things too, e.g. DDoS [2])
  - **We still strongly recommend using anycast** [5]

TTLs (caching) matter more than anycast



- Near-client caching beats great infrastructure!
    - Anycast TTL60 (no cache): **29.96ms** (median)
    - Unicast TTL86400 (cache): **7.38ms** (median):
    - **22ms median latency reduction**
- Query load: **77% down** with caching
- Conclusion: TTLs matter more for performance
    - (anycast is needed for other things too, e.g. DDoS [2])
    - **We still strongly recommend using anycast** [5]

TTLs (caching) matter more than anycast



- Near-client caching beats great infrastructure!
  - Anycast TTL60 (no cache): **29.96ms** (median)
  - Unicast TTL86400 (cache): **7.38ms** (median):
  - **22ms median latency reduction**
- Query load: **77% down** with caching
- Conclusion: TTLs matter more for performance
  - (anycast is needed for other things too, e.g. DDoS [2])
  - **We still strongly recommend using anycast** [5]

Reasons for Longer or shorter TTLs

- **Longer caching**:
  - faster responses to clients
  - lowers DNS traffic levels
  - more robust to DDoS attacks [4]
- **Shorter caching**:
  - faster operational value changes
  - useful for DNS redirect based DDoS scrubbing services
  - DNS-load balancing

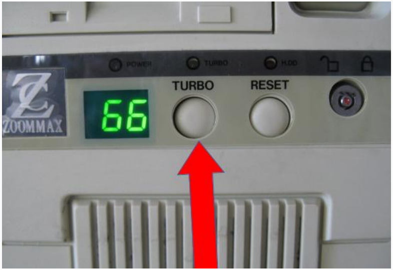
Organizations must weight these trade-offs to find a good balance

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 50ms or more
  - But **keep on using anycast** too [2, 5]
- **Should you reconsider your TTLs as well?**

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



References i

[1] de Vries, W. B., de O. Schmidt, R., Haraker, W., Heidemann, J., de Boer, P.-T., and Pras, A.

Verfploeter: Broad and load-aware anycast mapping.

In *Proceedings of the ACM Internet Measurement Conference* (London, UK, 2017).

[2] Moura, G. C. M., de O. Schmidt, R., Heidemann, J., de Vries, W. B., Müller, M., Wei, L., and Hesselman, C.

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

References ii

In *Proceedings of the ACM Internet Measurement Conference* (Santa Monica, California, USA, Nov. 2016), ACM, pp. 255–270.

[3] MOURA, G. C. M., HEIDEMANN, J., DE O. SCHMIDT, R., AND HARDAKER, W.

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

In *Proceedings of the ACM Internet Measurement Conference* (Amsterdam, the Netherlands, Oct. 2019), ACM, p. to appear.

References iii

[4] Moura, G. C. M., Heidemann, J., Müller, M., de O. Schmidt, R., and Davids, M.

When the dike breaks: Dissecting DNS defenses during DDoS.

In *Proceedings of the ACM Internet Measurement Conference* (Boston, MA, USA, Oct. 2018), pp. 8–21.

[5] Müller, M., Moura, G. C. M., de O. Schmidt, R., and Heidemann, J.

Recursives in the wild: Engineering authoritative DNS servers.

In *Proceedings of the ACM Internet Measurement Conference* (London, UK, 2017), ACM, pp. 489–495.