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]

RIPE 79

Rotterdam, The Netherlands

2019-10-15

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

Outline

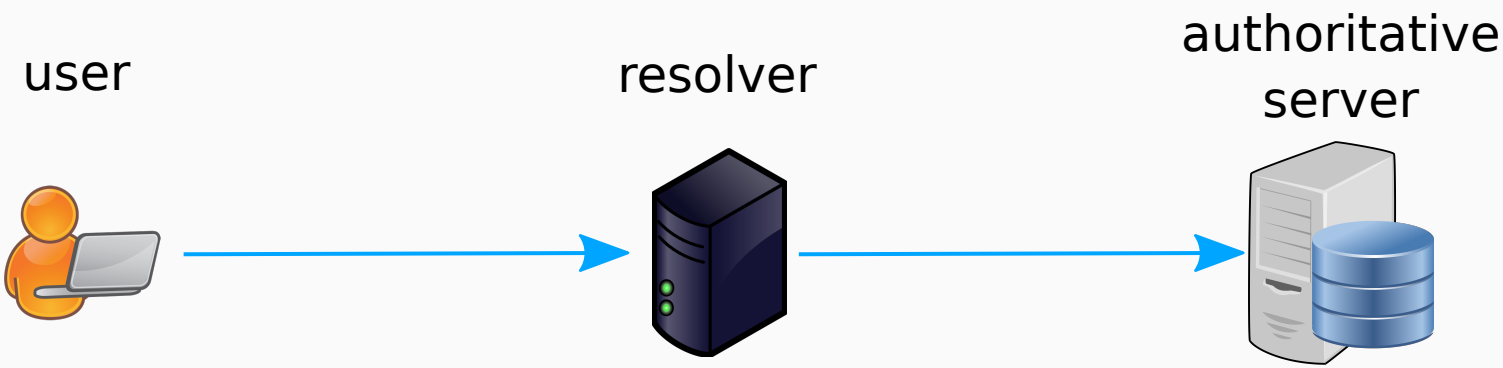
Our research on DNS over the last years

Our rearch on DNS security/stability:

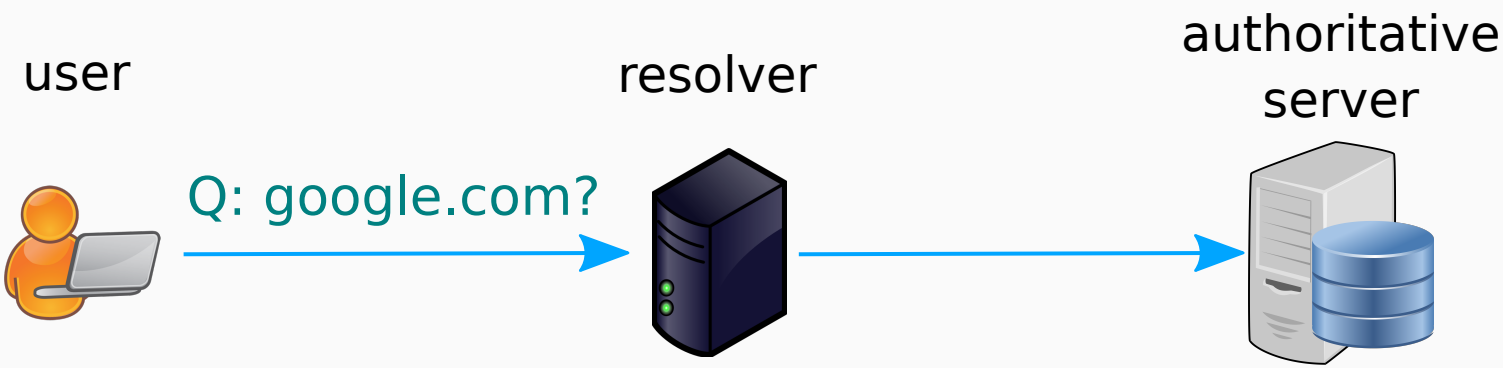
- **Anycast and DDoS**: IMC 2016 [2]
- **Resolvers**: IMC 2017 [5]
- **Anycast Engineering**: IMC 2017 [1]
- **Caching and DDoS**: IMC 2018 [4]
- **Caching and TTL, and performance:** IMC 2019 [3]
  - (this paper)
  - IMC will be next week in Amsterdam

Introduction

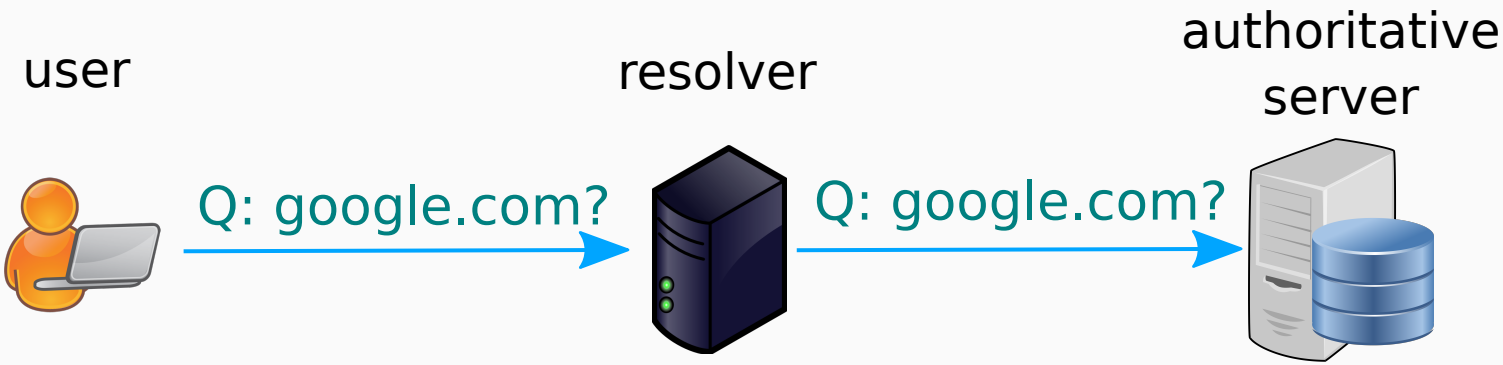
The role of TTL



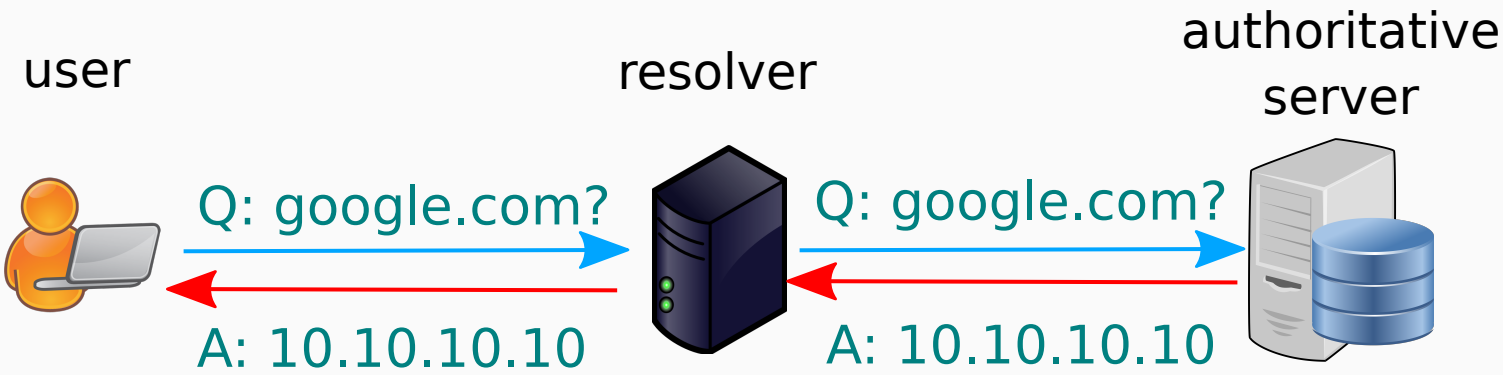
The role of TTL



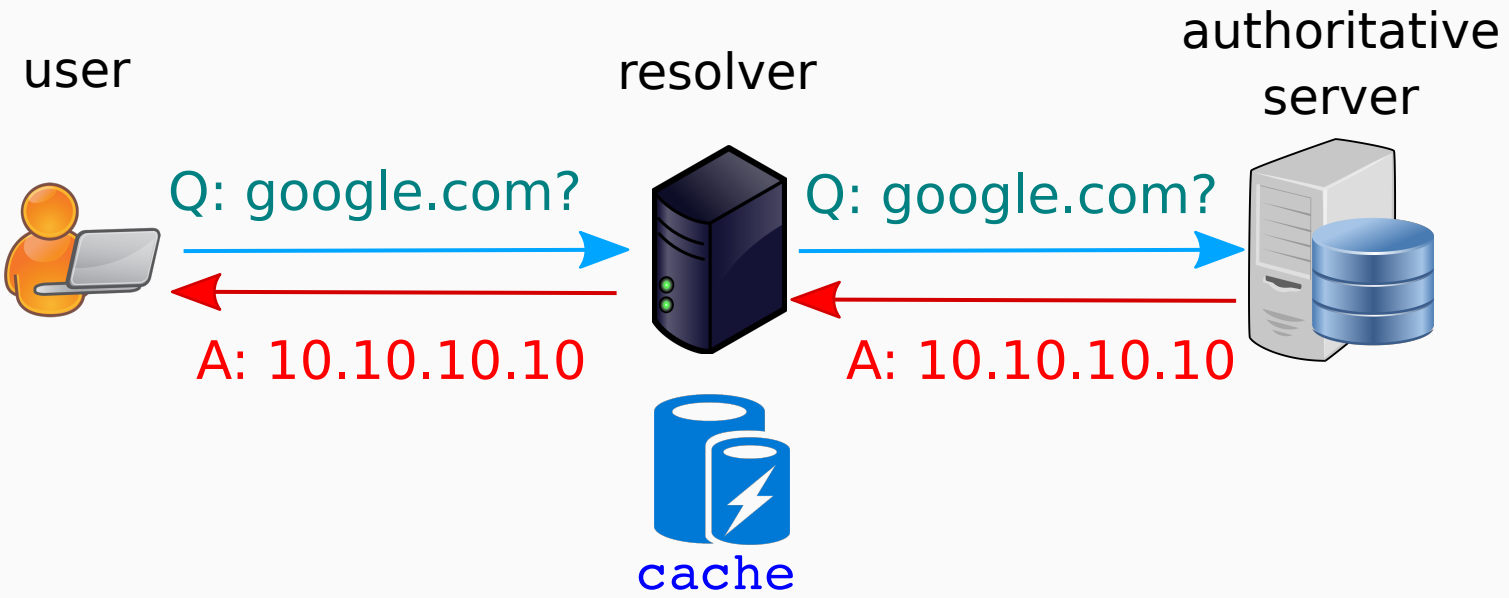
The role of TTL



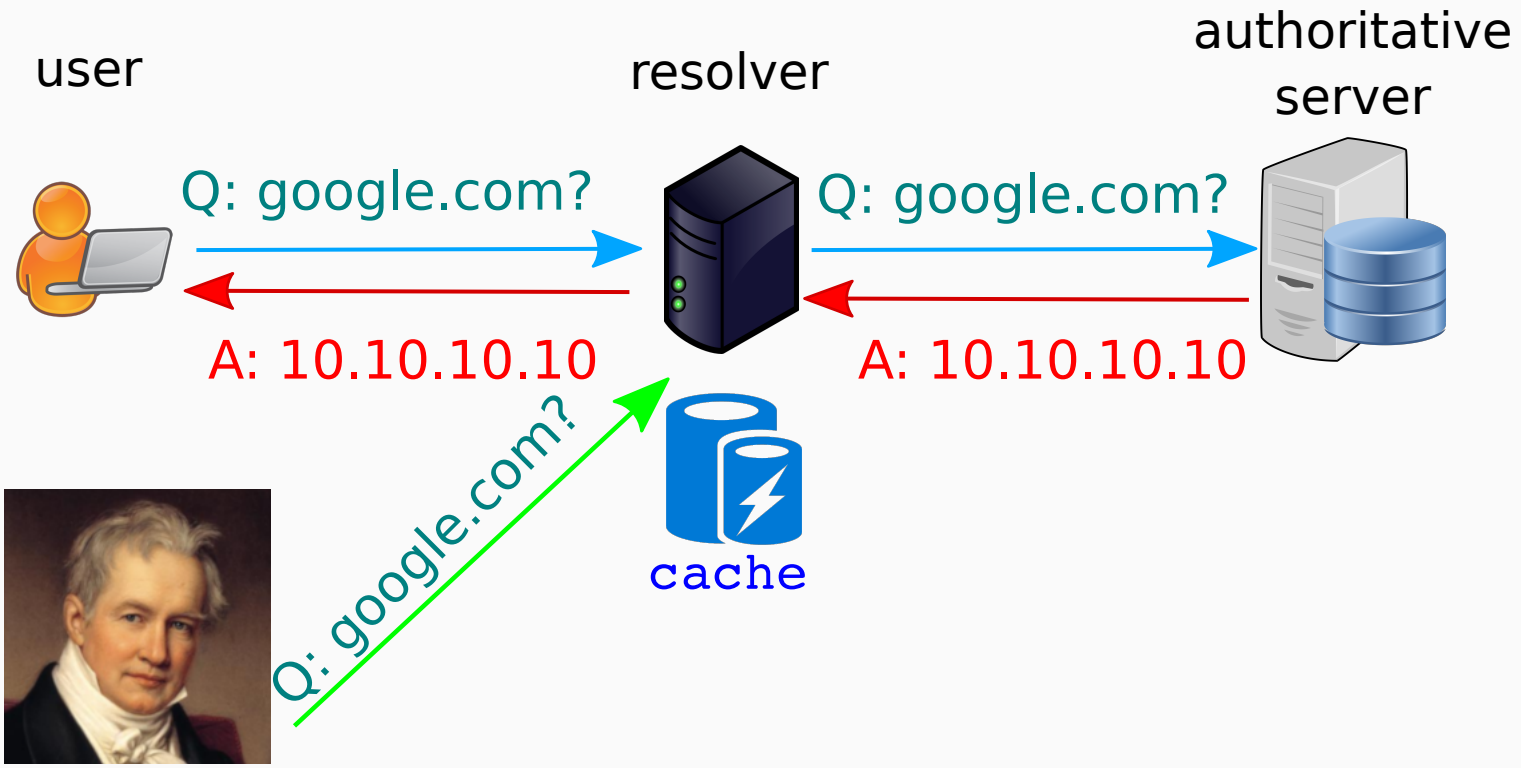
The role of TTL



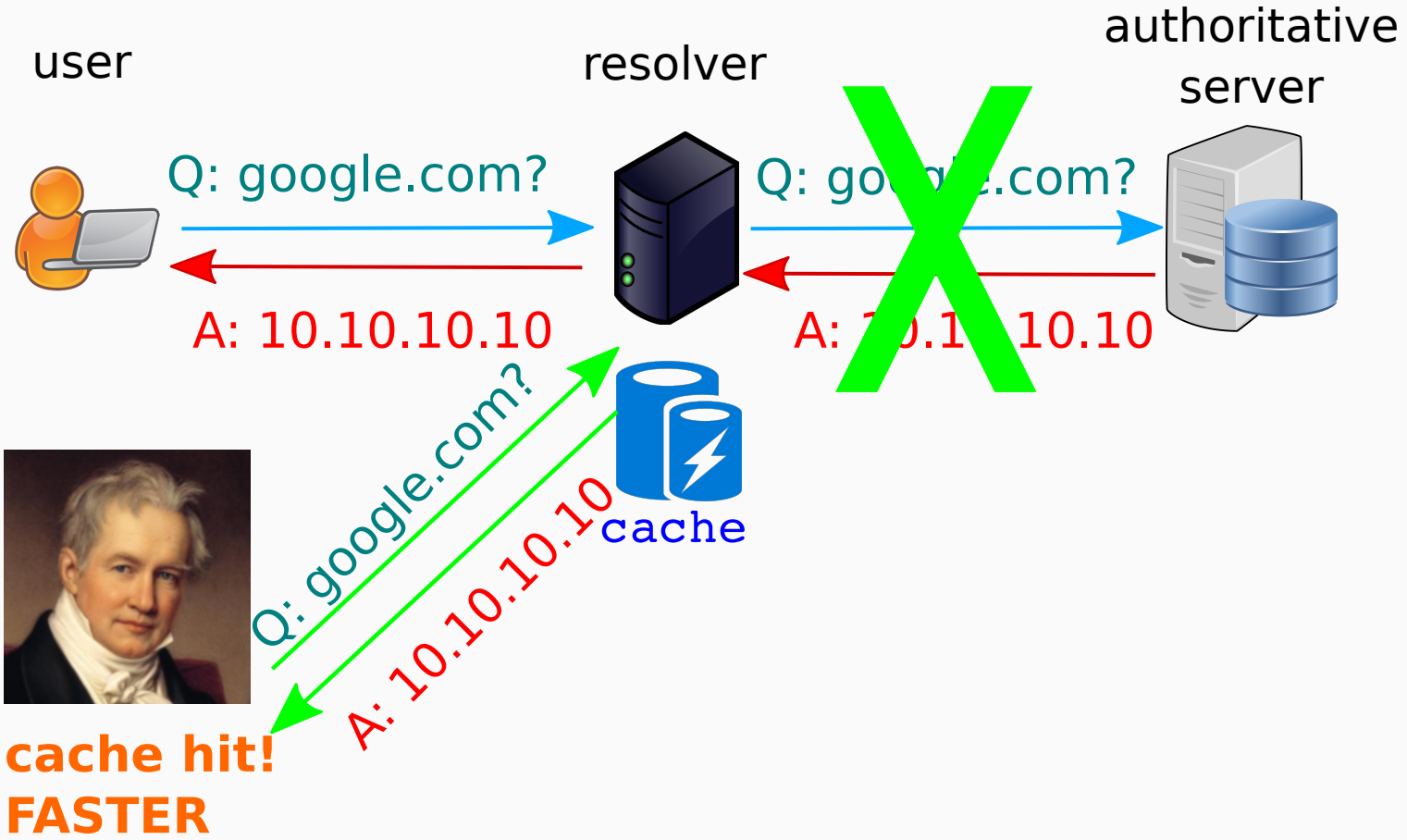
The role of TTL



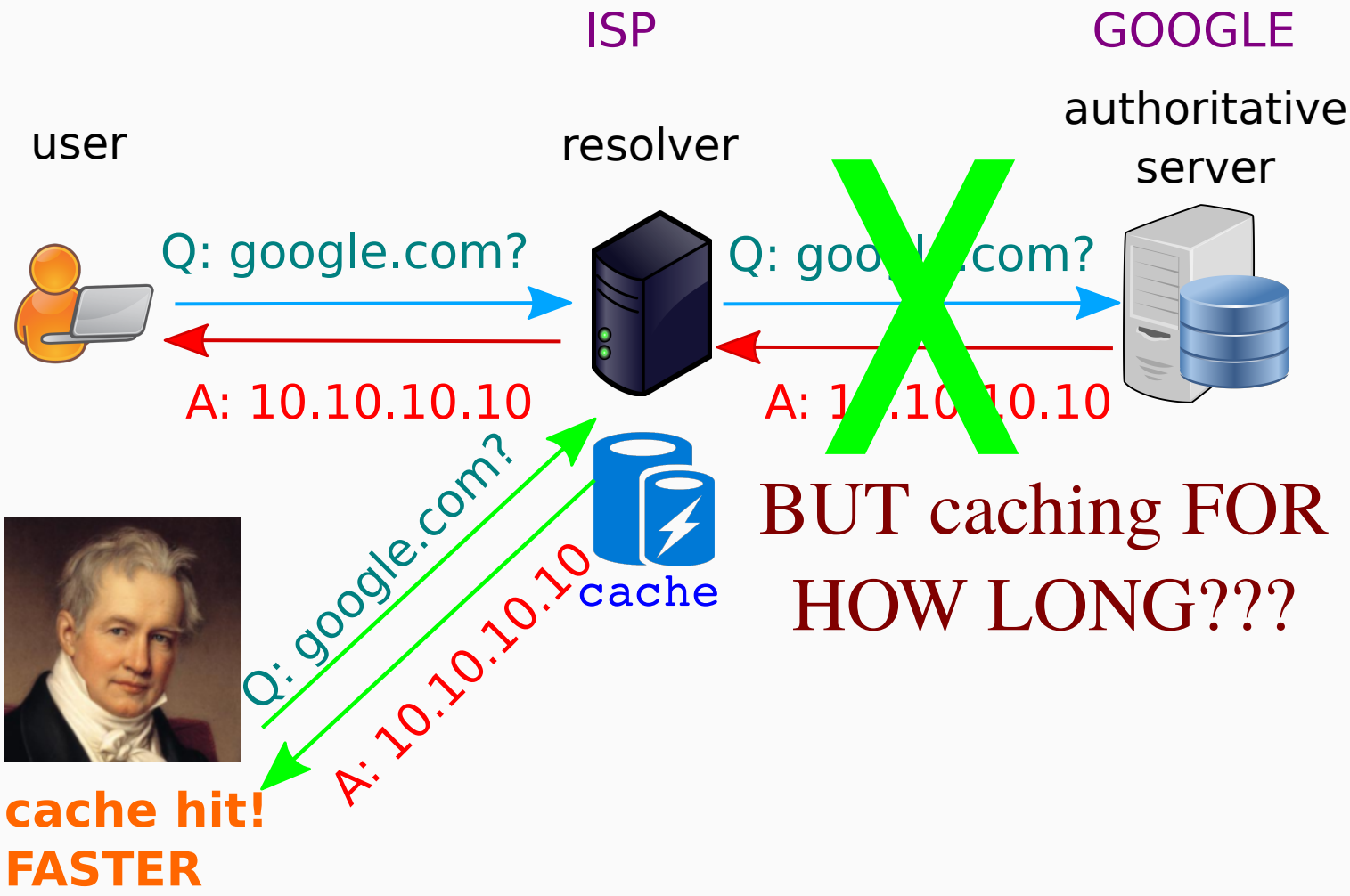
The role of TTL



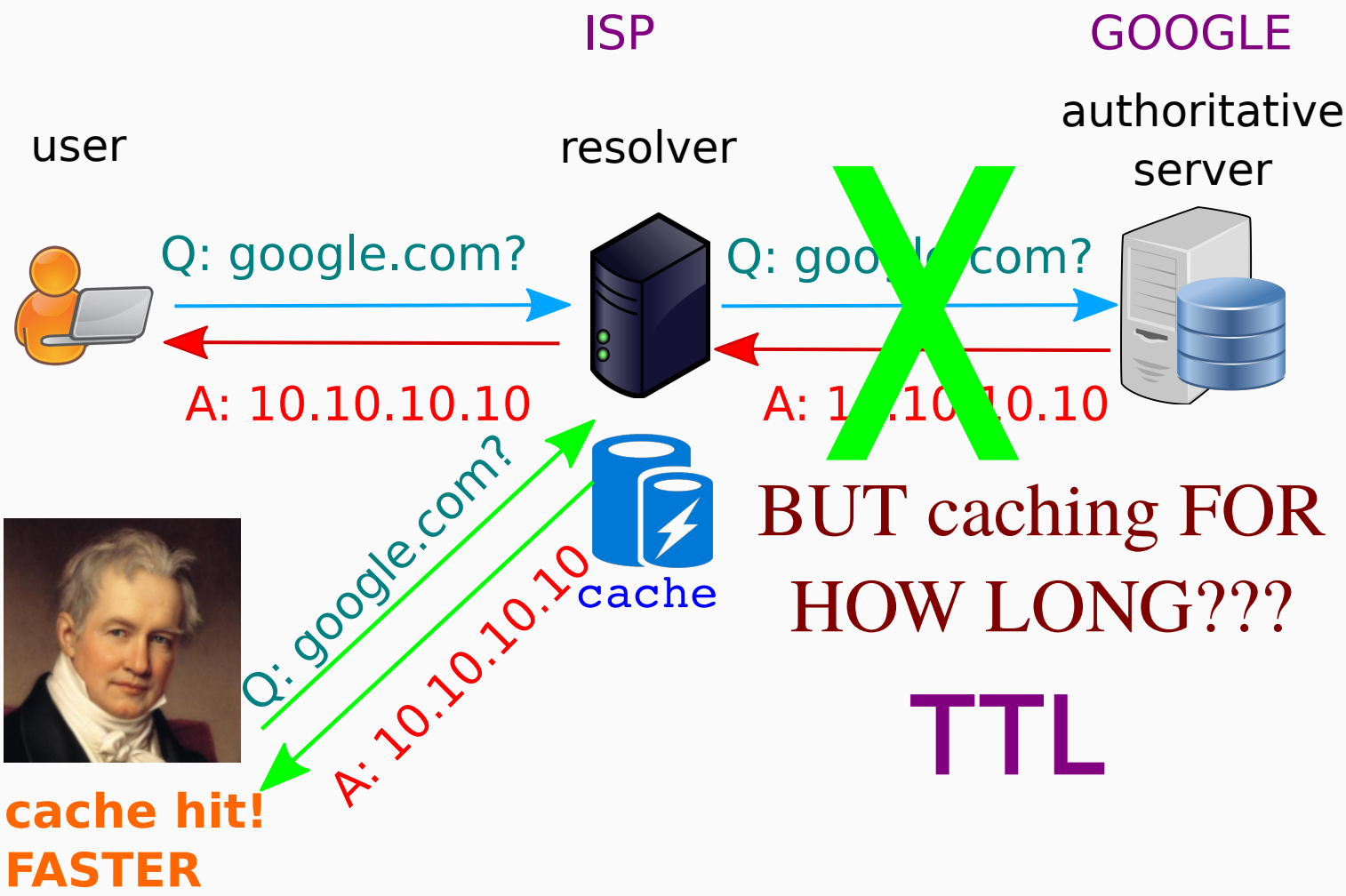
The role of TTL



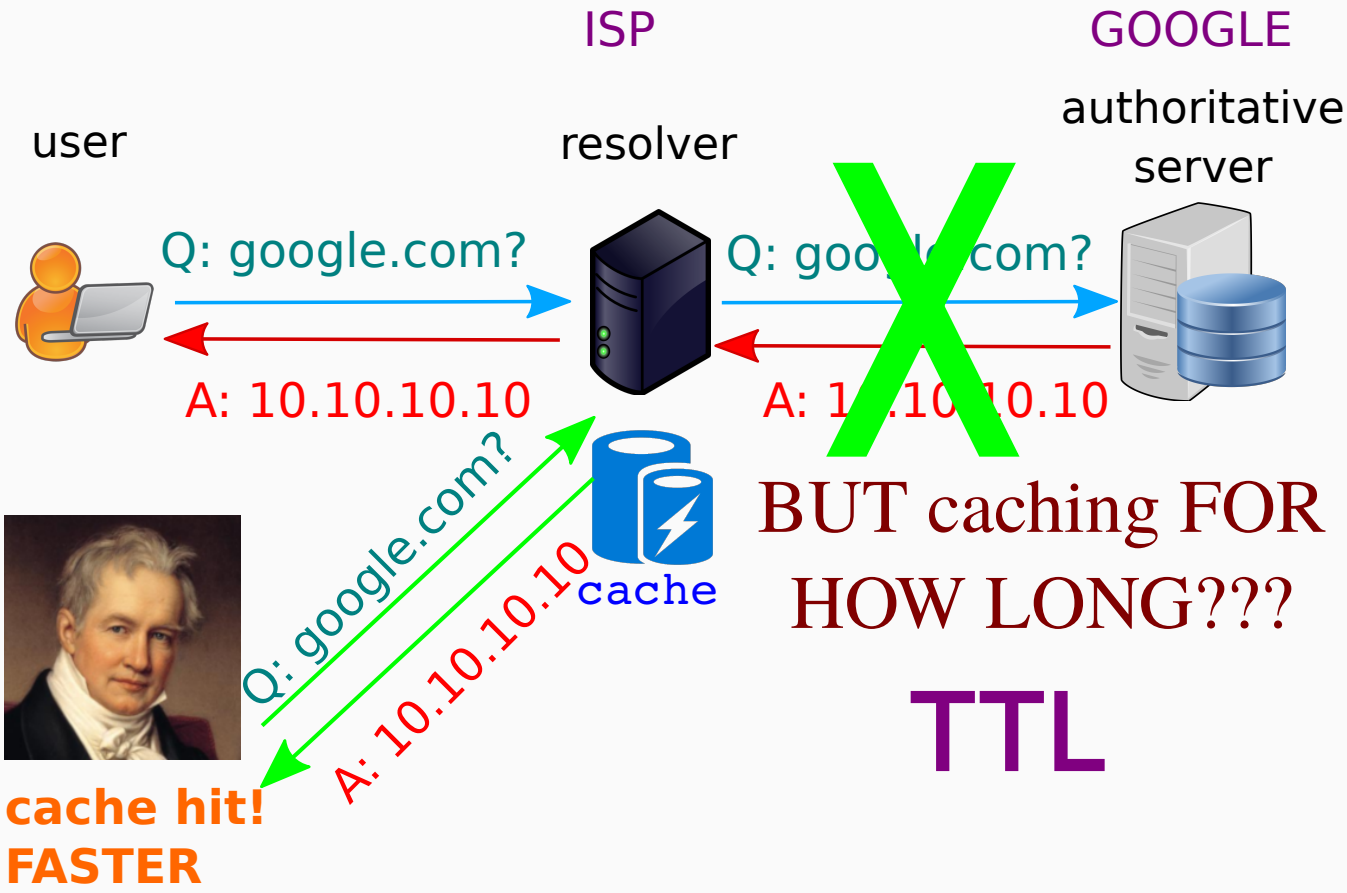
The role of TTL



The role of TTL



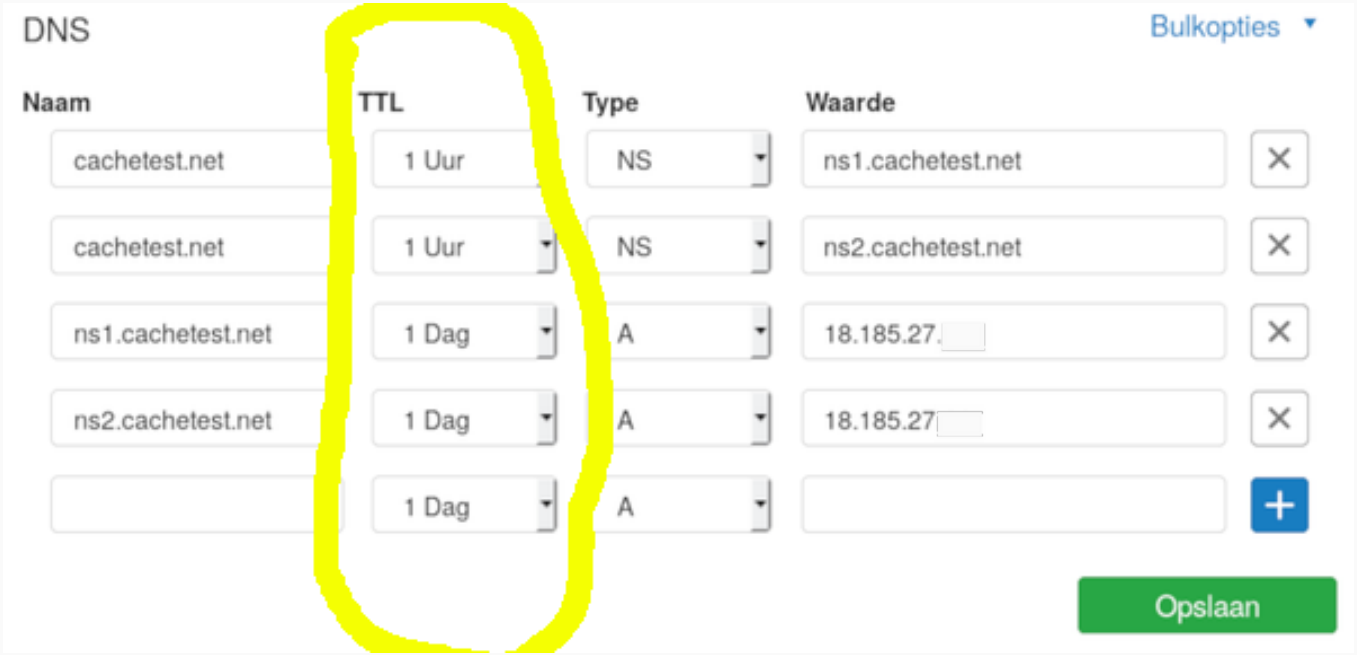
The role of TTL



- **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 chaging 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 unnecesssarily short
    - a. because sometimes multiple places $\rightarrow$ 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 $\sim$**20ms median latency improvement; 171ms 75%ile**

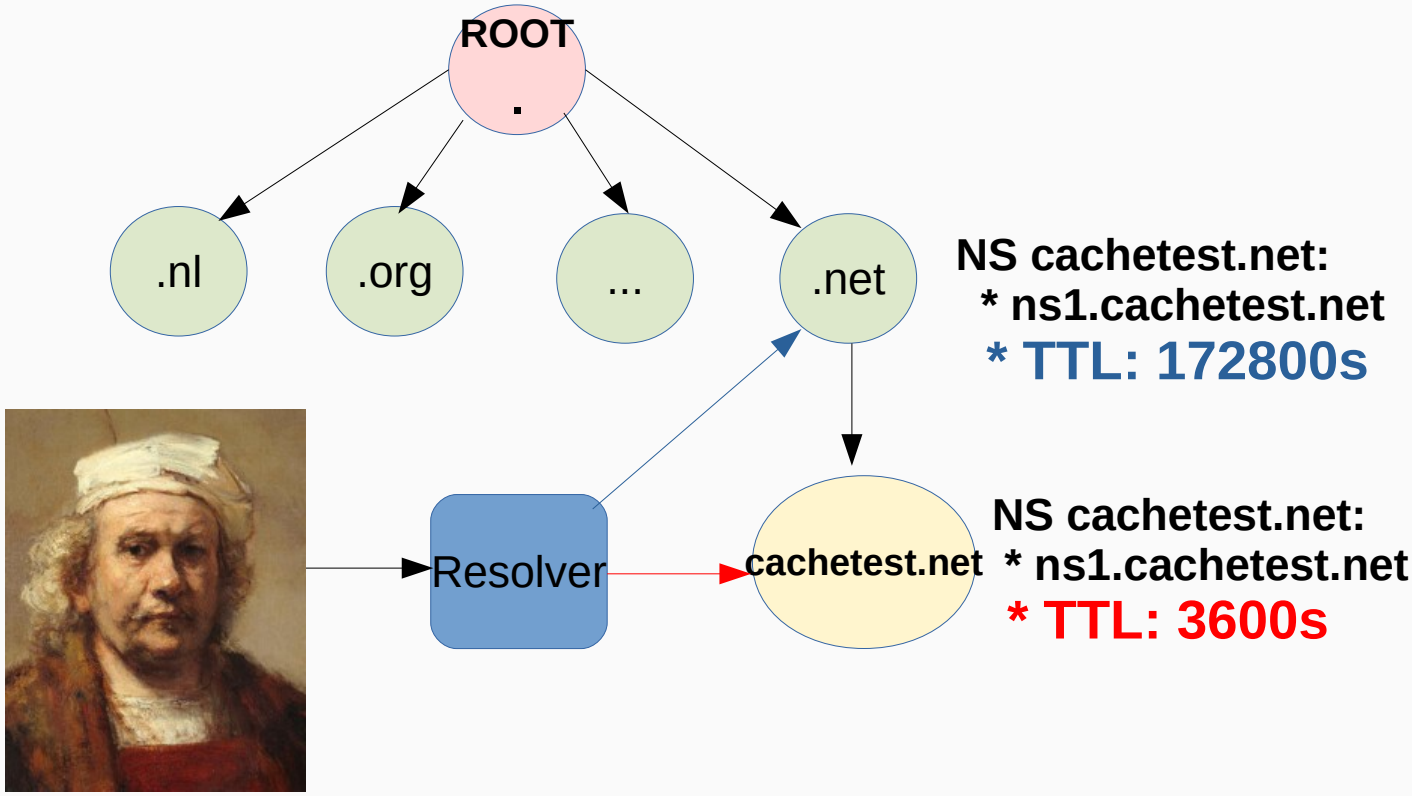
The rest of this talk

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)

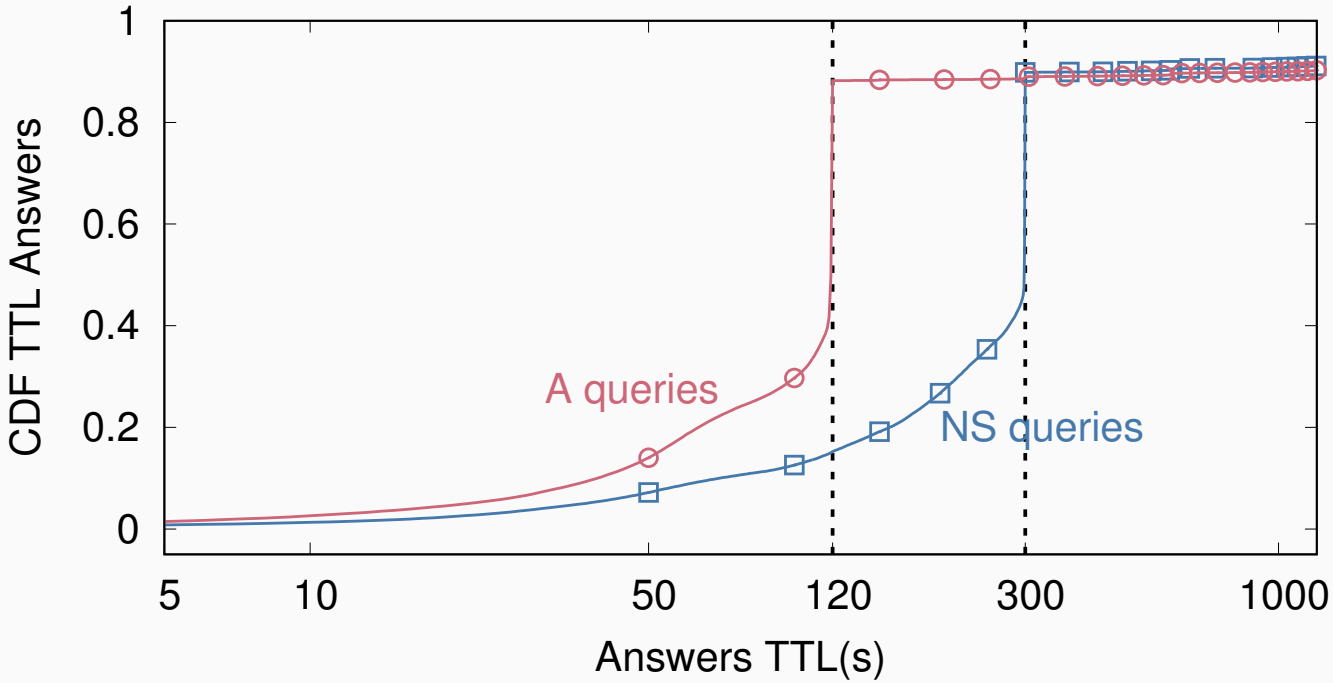
Are resolvers parent- or child-centric?

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) mutliple 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.

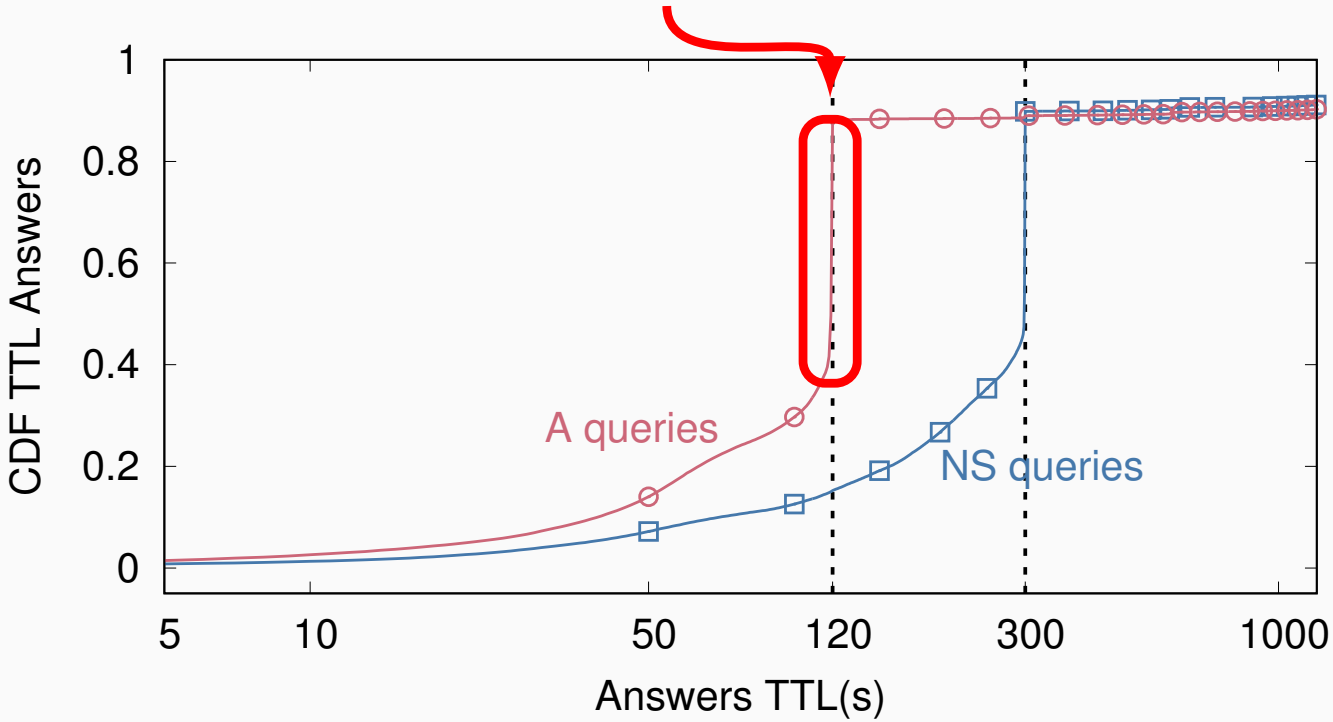


- Remember: TTL parents: 2 days

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

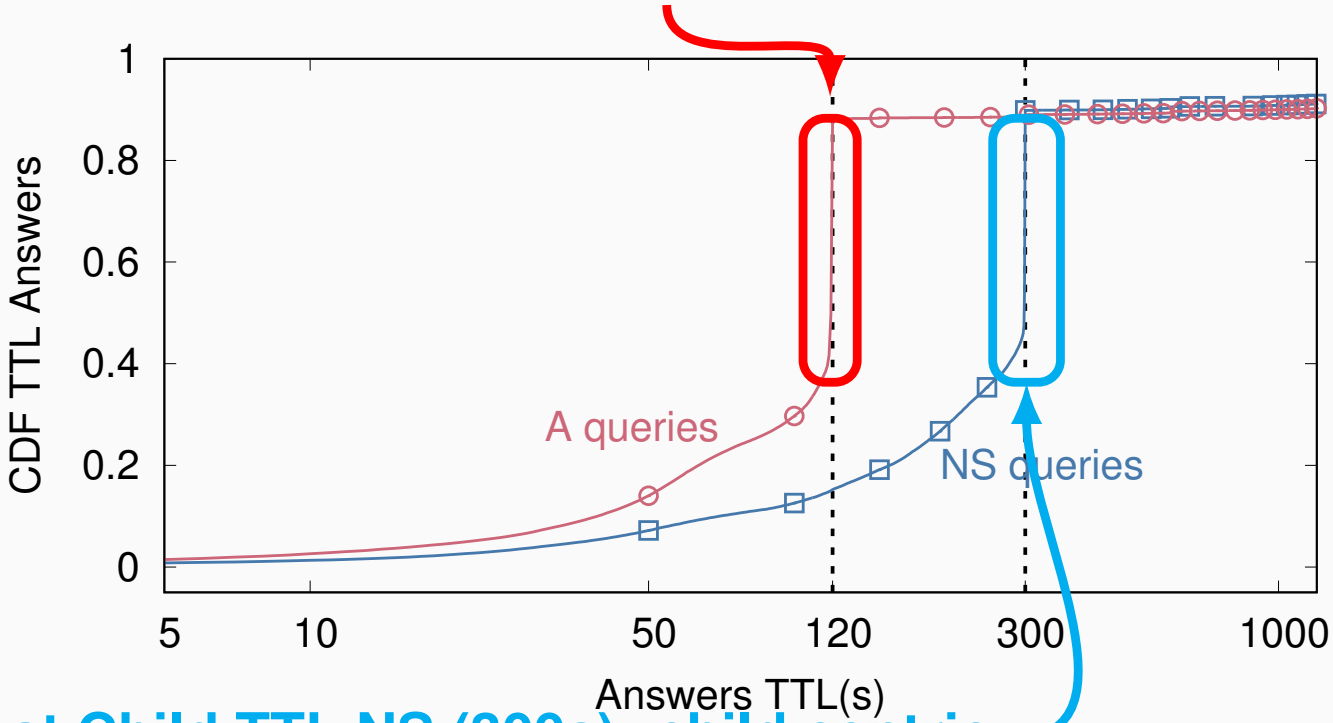


- Remember: TTL parents: 2 days

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



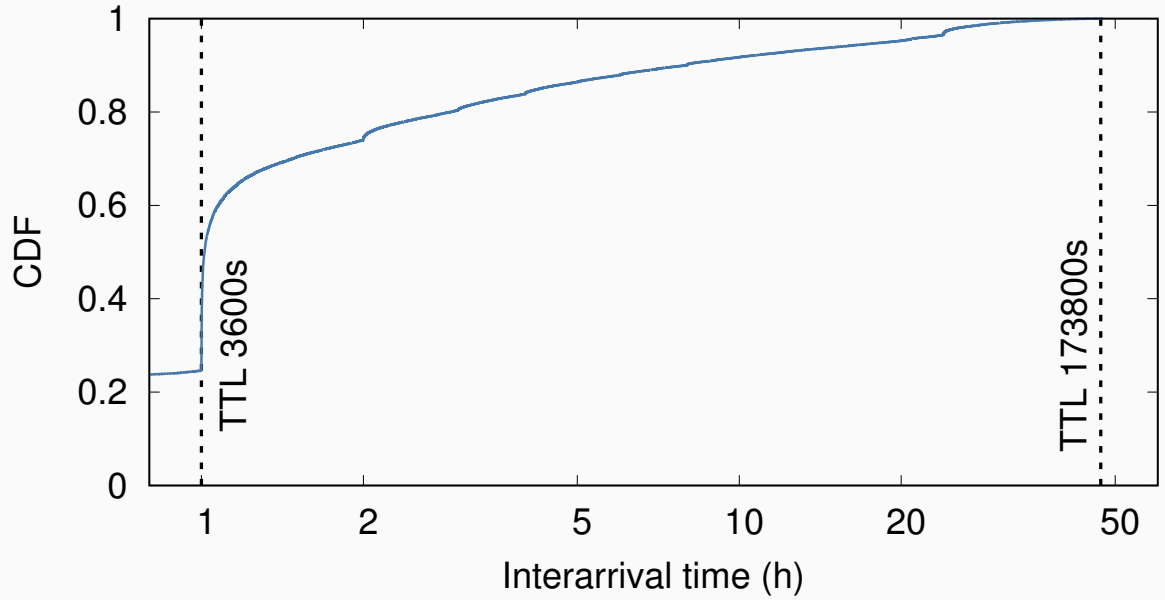
Spike at Child TTL NS (300s): child centric

- 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**

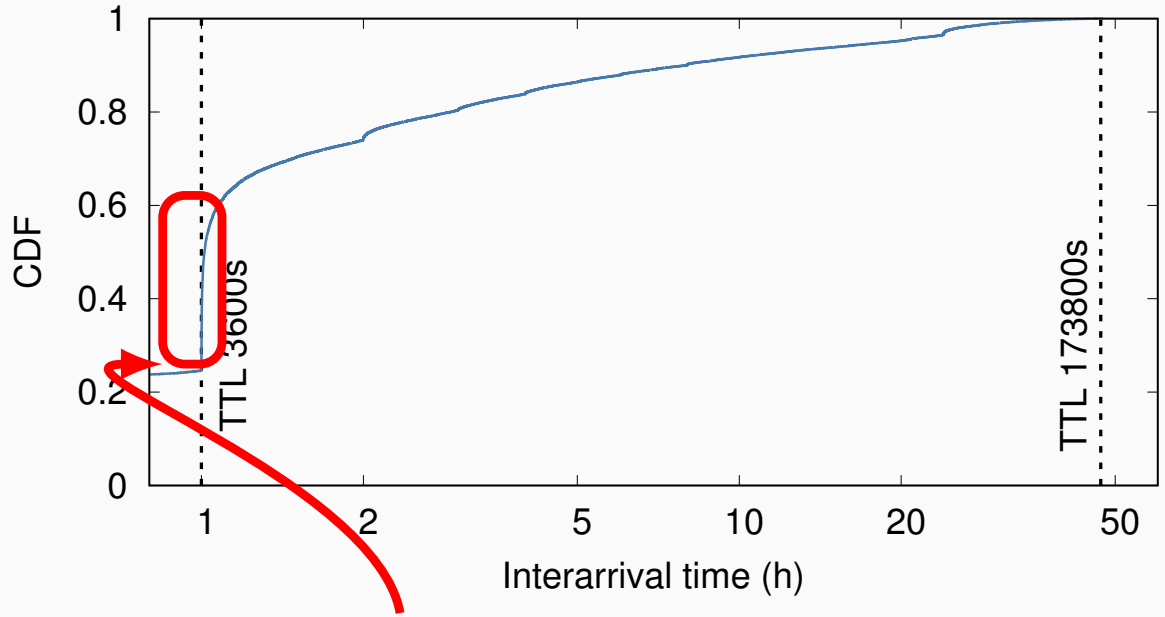


We confirmed this with a second-level domain (paper)

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**



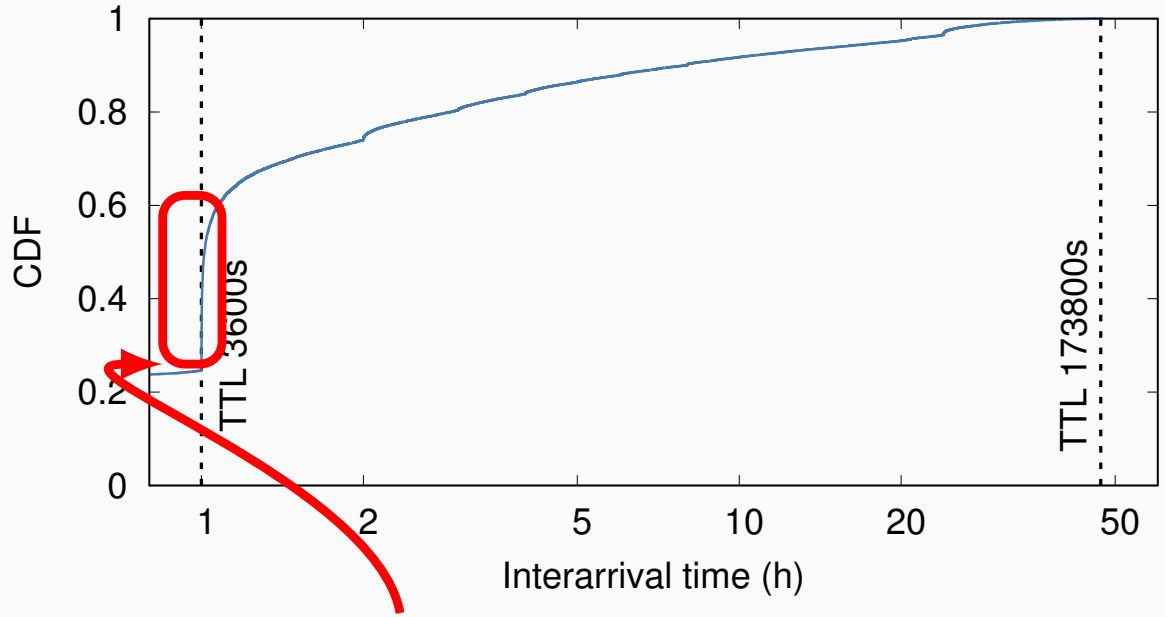
Spike at Child TTL A (3600s): confirm child centric for TLD

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**

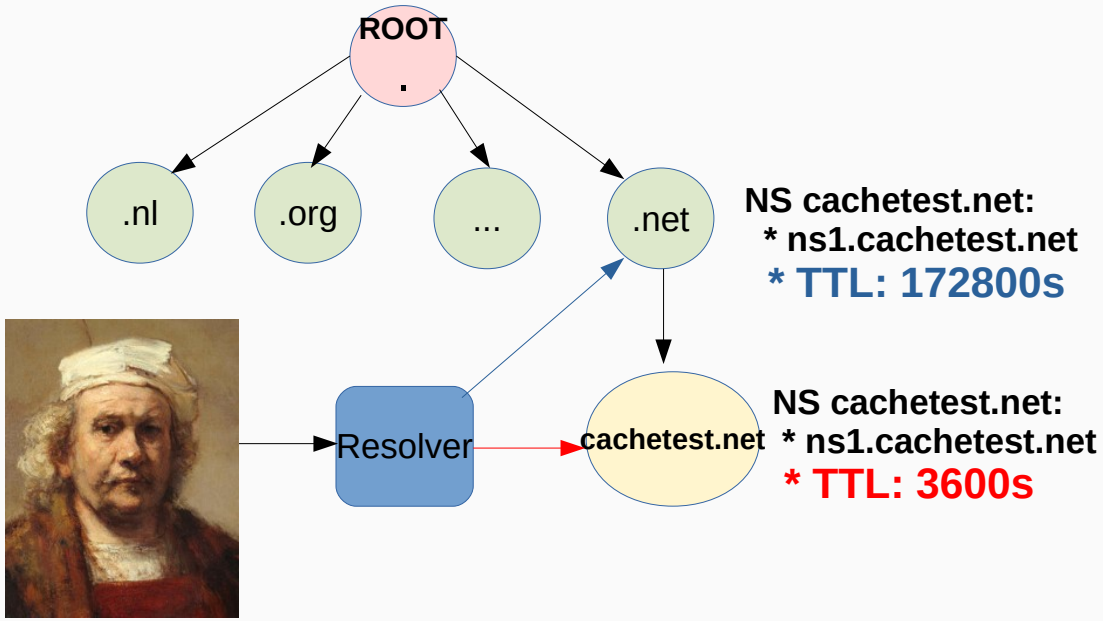


Spike at Child TTL A (3600s): confirm child centric for TLD

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

Most resolvers wil use child TTLs

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

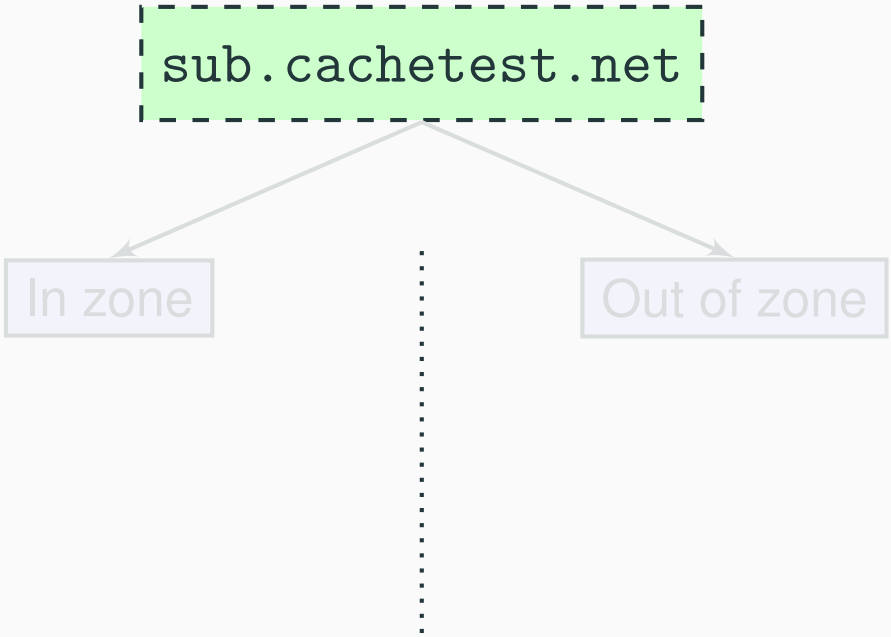


Which TTL will Rembrandt use?
Parent (172800s) or child (TTL: 3600s)

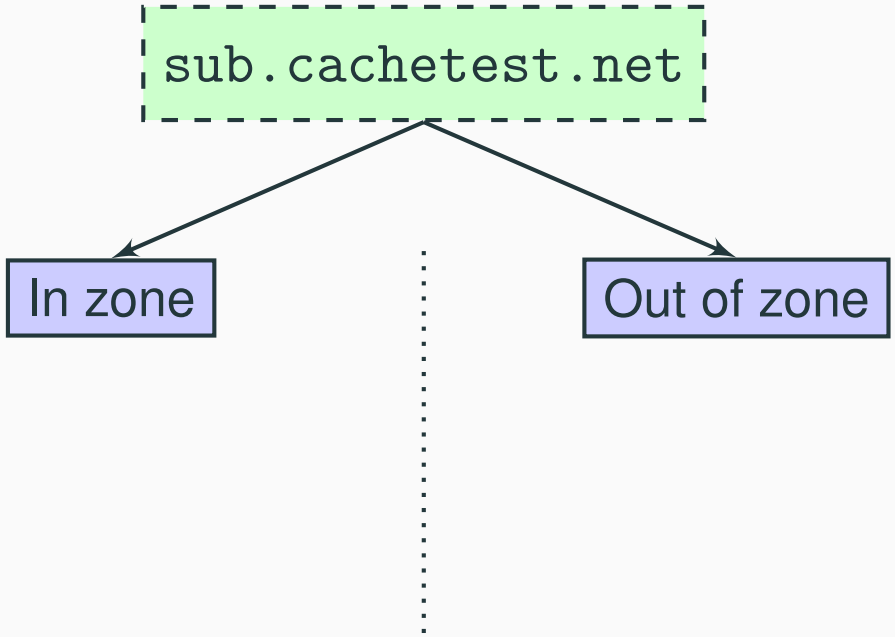
Outline

Zone configurations and Effective TTL

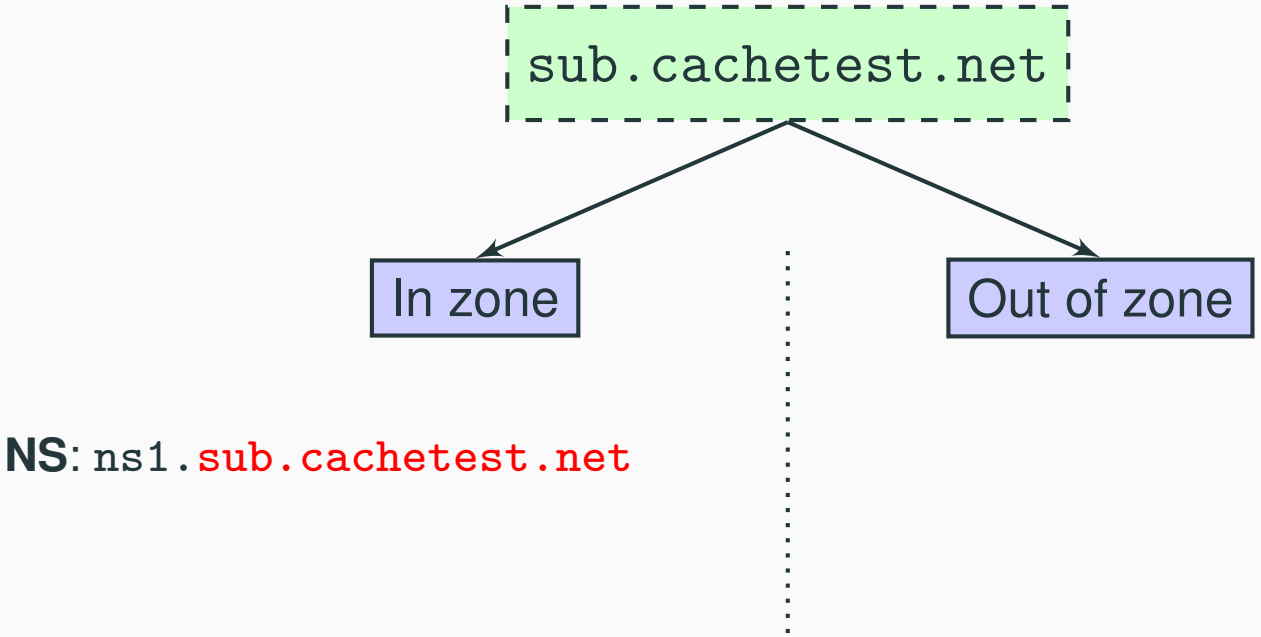
Are there dependencies between A and NS TTLs?



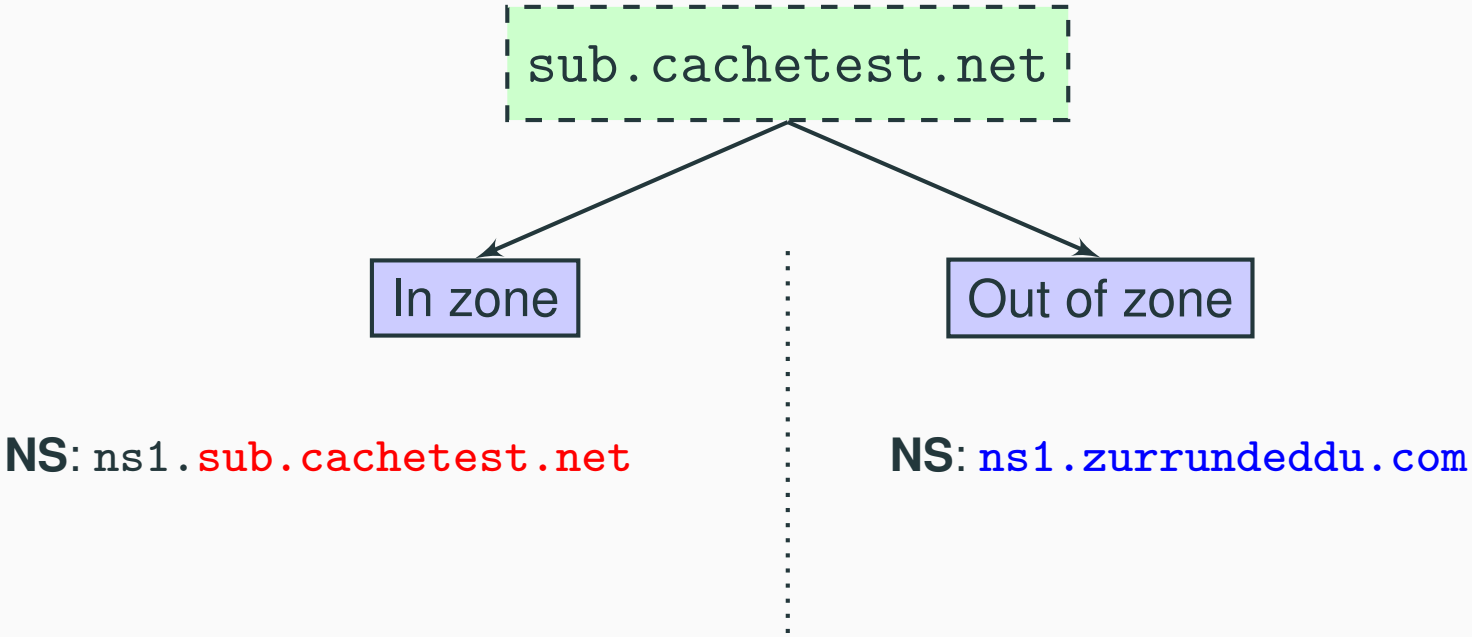
Are there dependencies between A and NS TTLs?



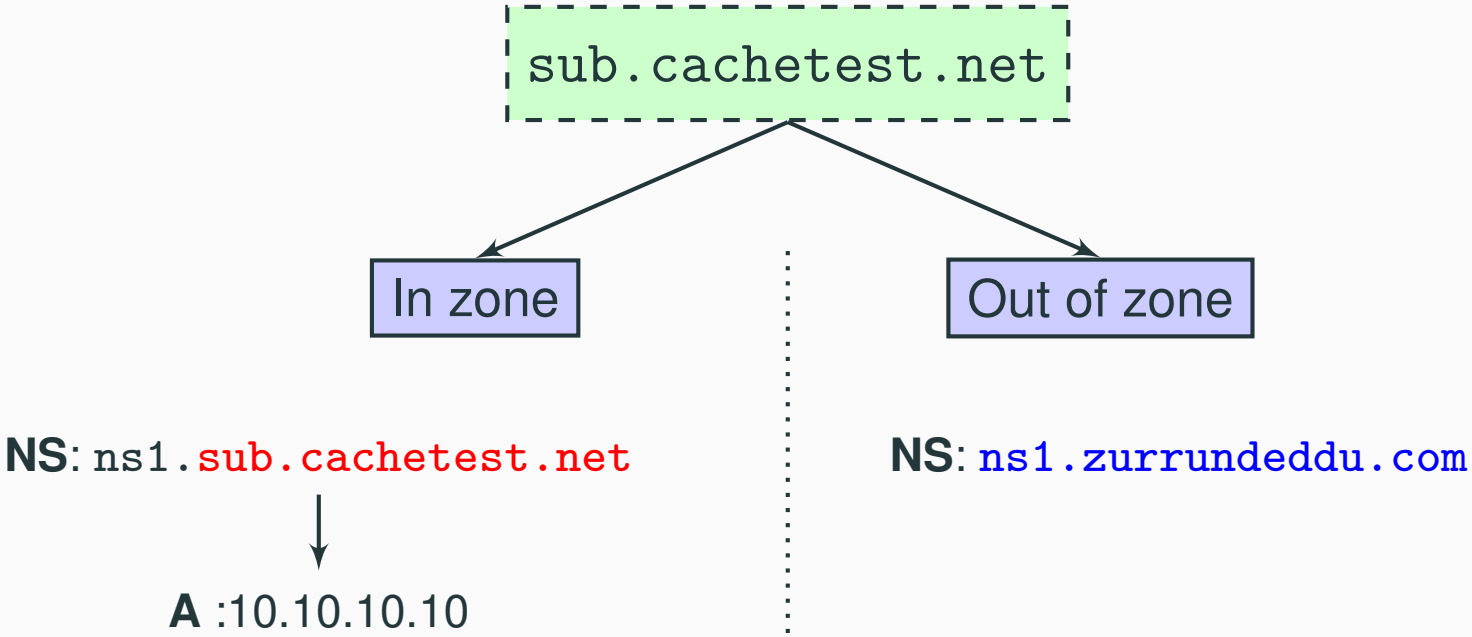
Are there dependencies between A and NS TTLs?



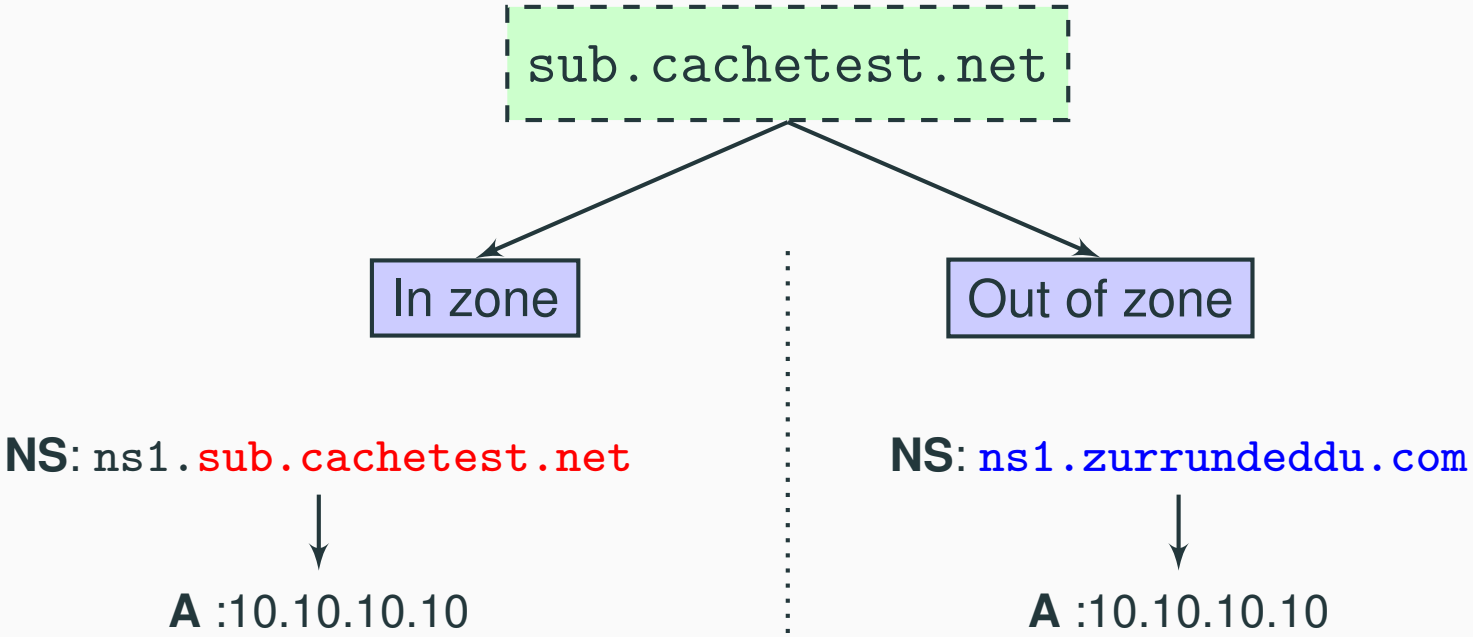
Are there dependencies between A and NS TTLs?



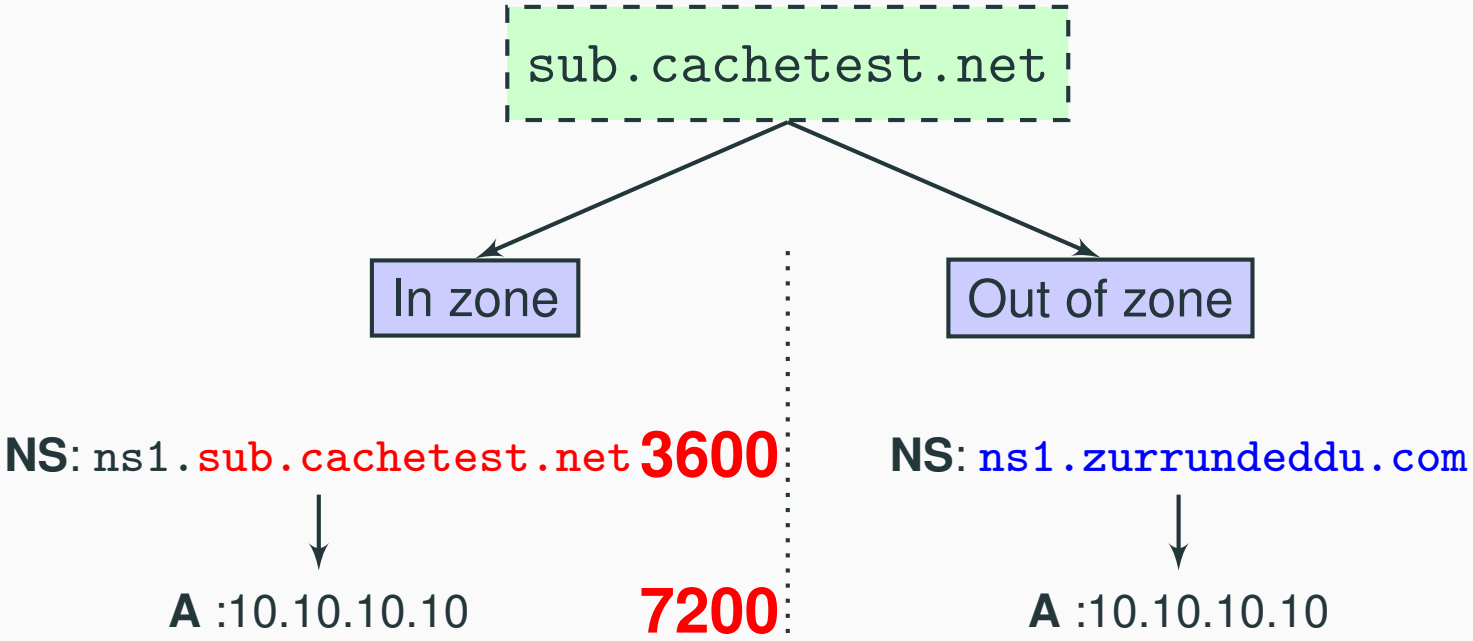
Are there dependencies between A and NS TTLs?



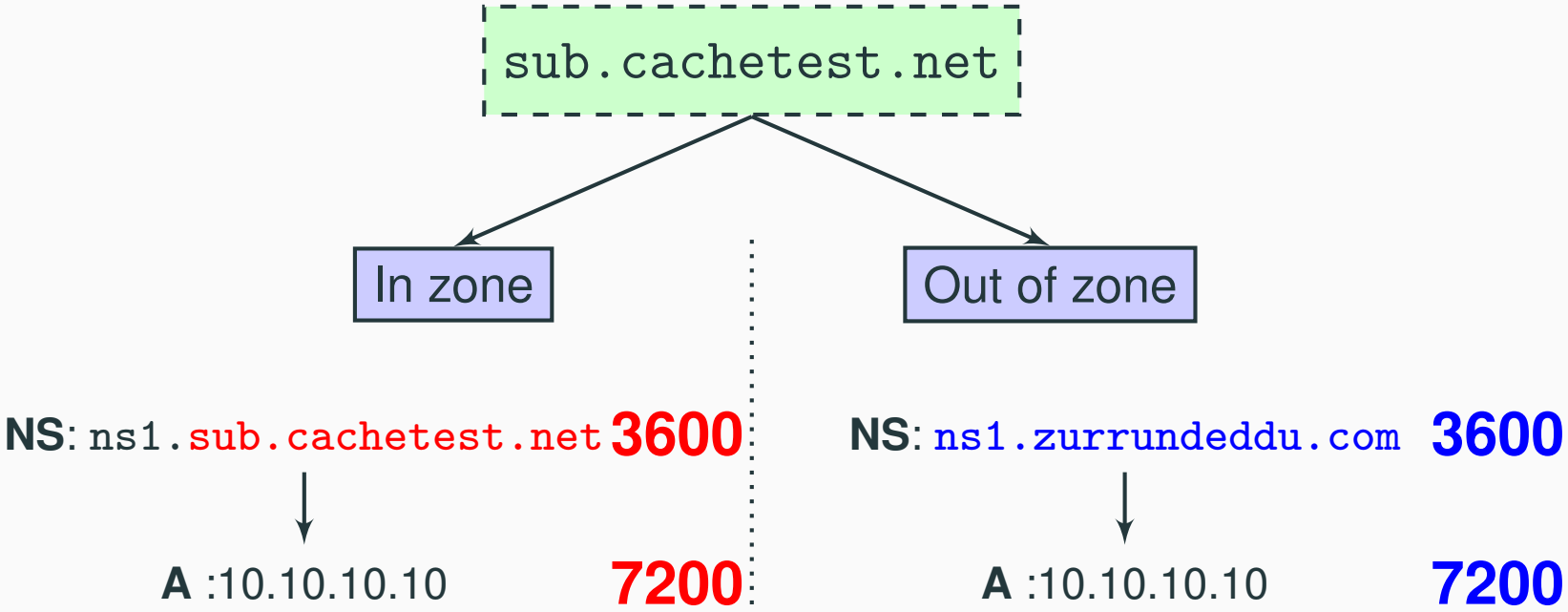
Are there dependencies between A and NS TTLs?



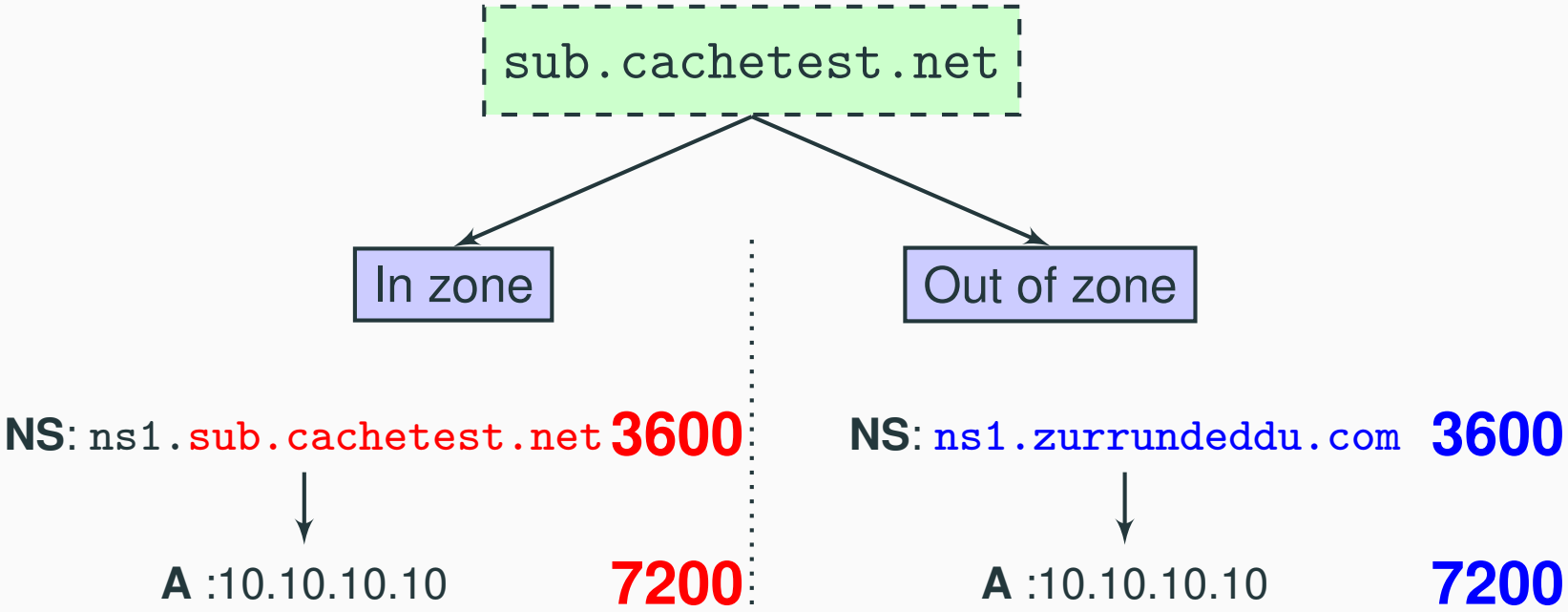
Are there dependencies between A and NS TTLs?



Are there dependencies between A and NS TTLs?

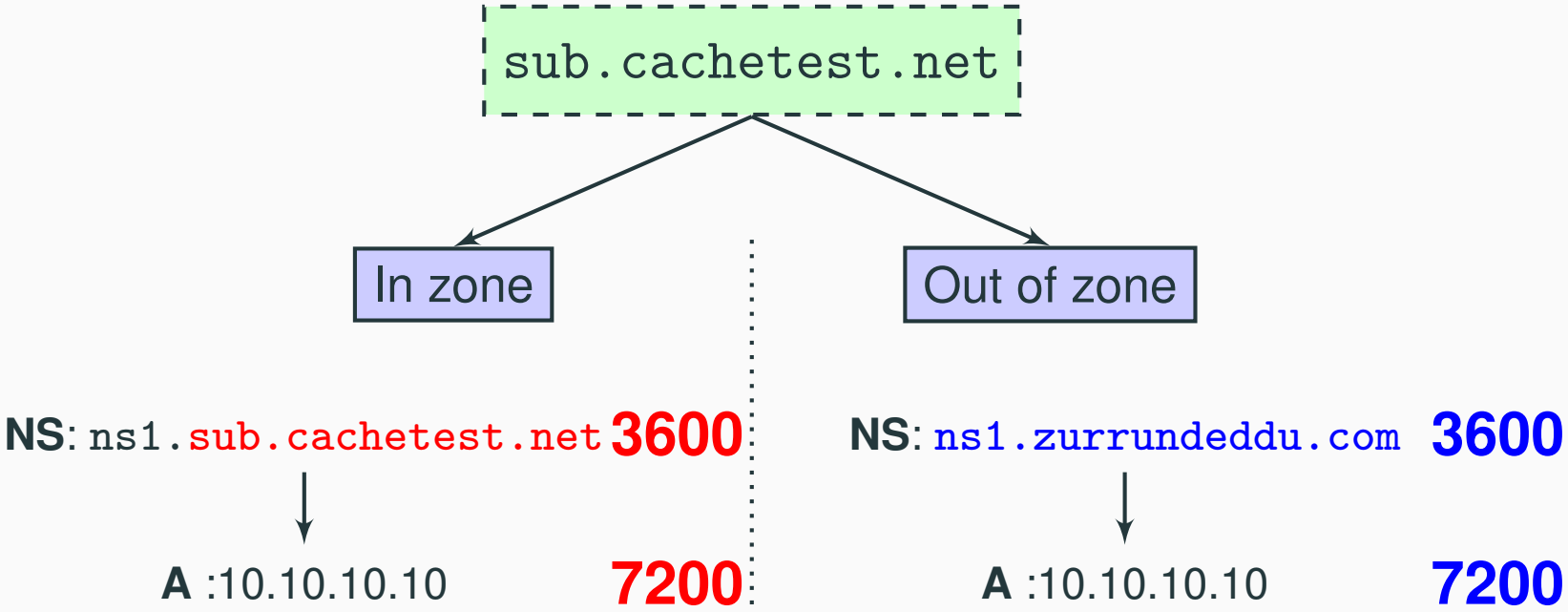


Are there dependencies between A and NS TTLs?



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

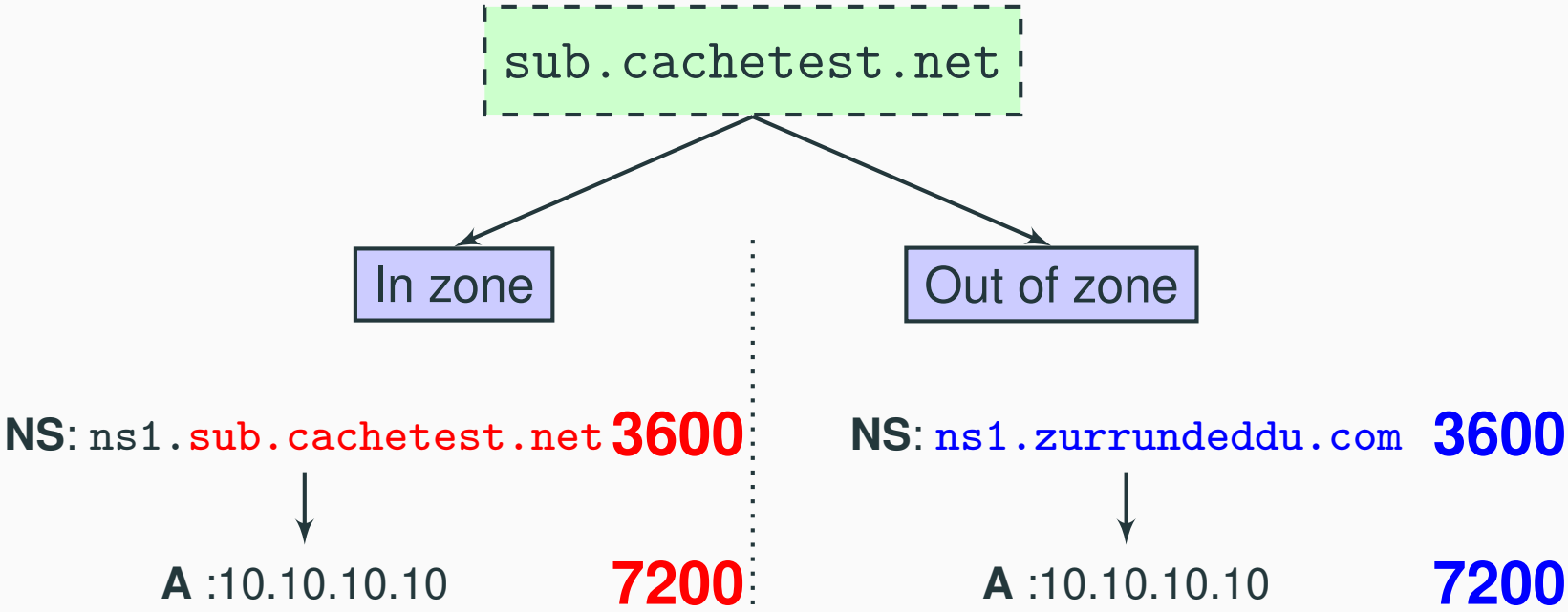
Are there dependencies between A and NS TTLs?



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

Are NS and A cached independently?

Are there dependencies between A and NS TTLs?

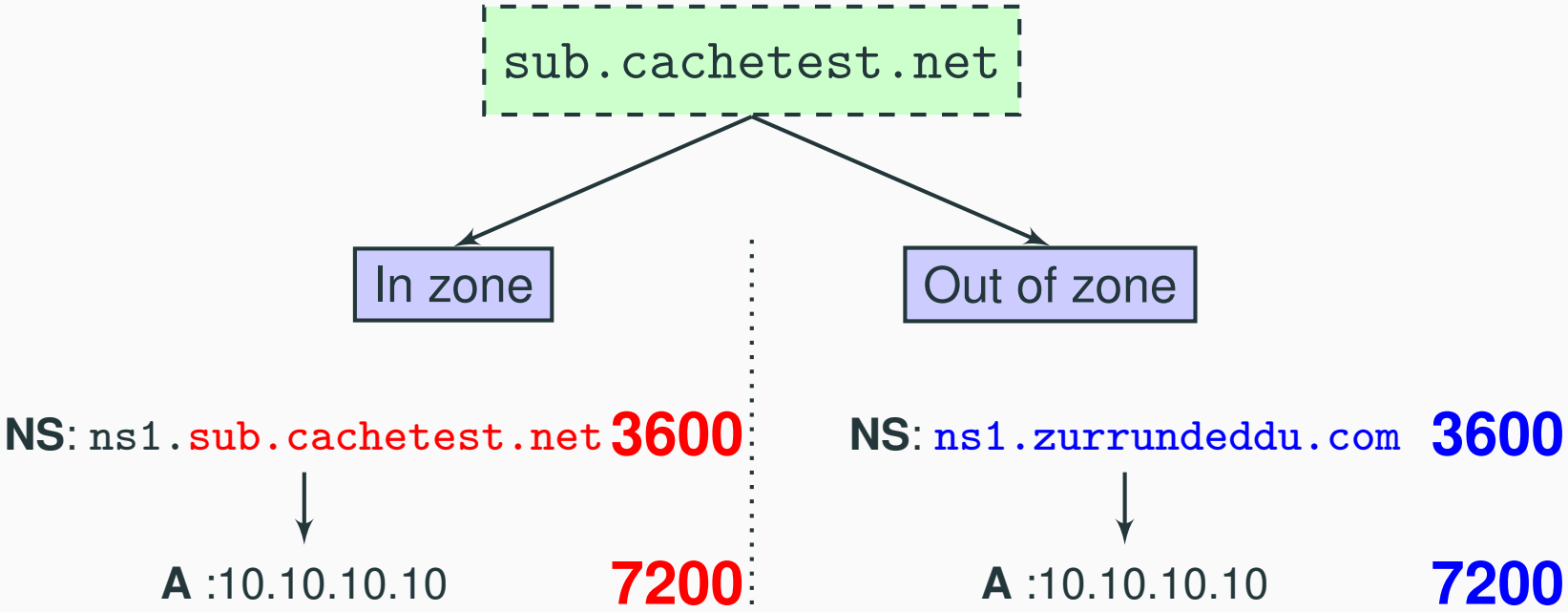


To resolve *.sub.cachetest.net, you need both **NS** and **A**

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?



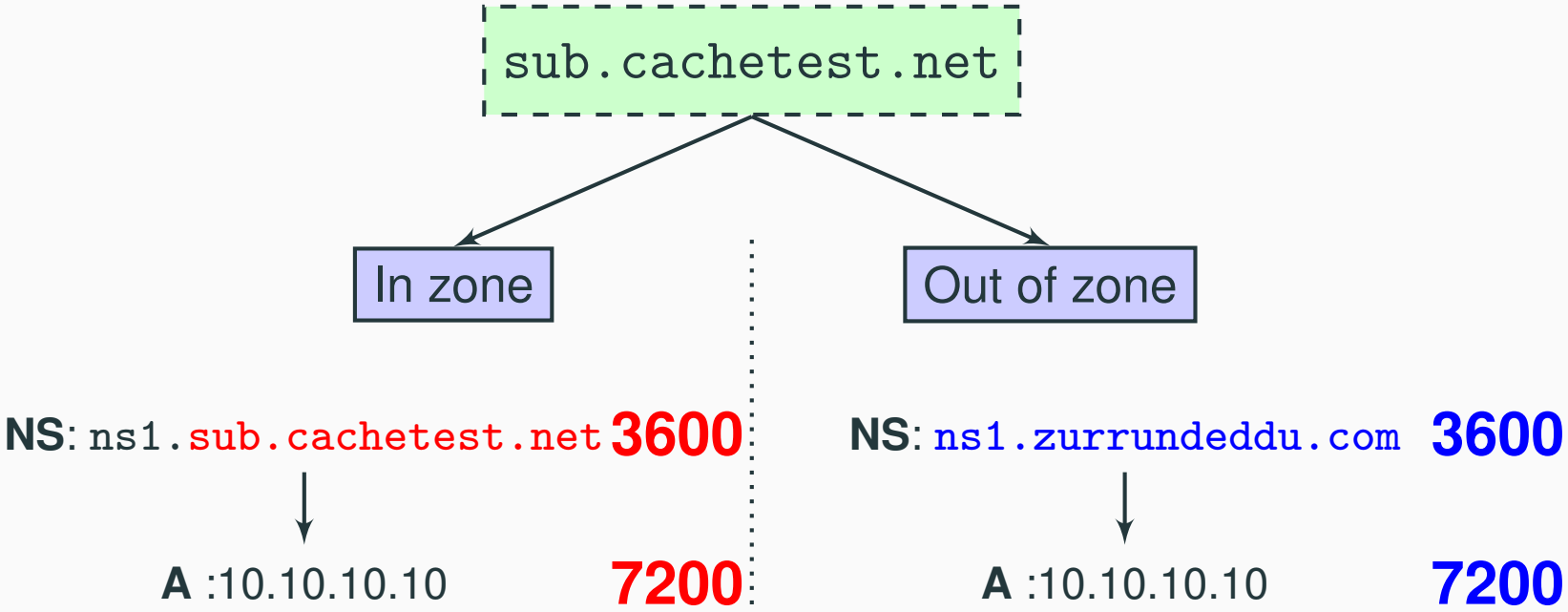
To resolve *.sub.cachetest.net, you need both **NS** and **A**

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?



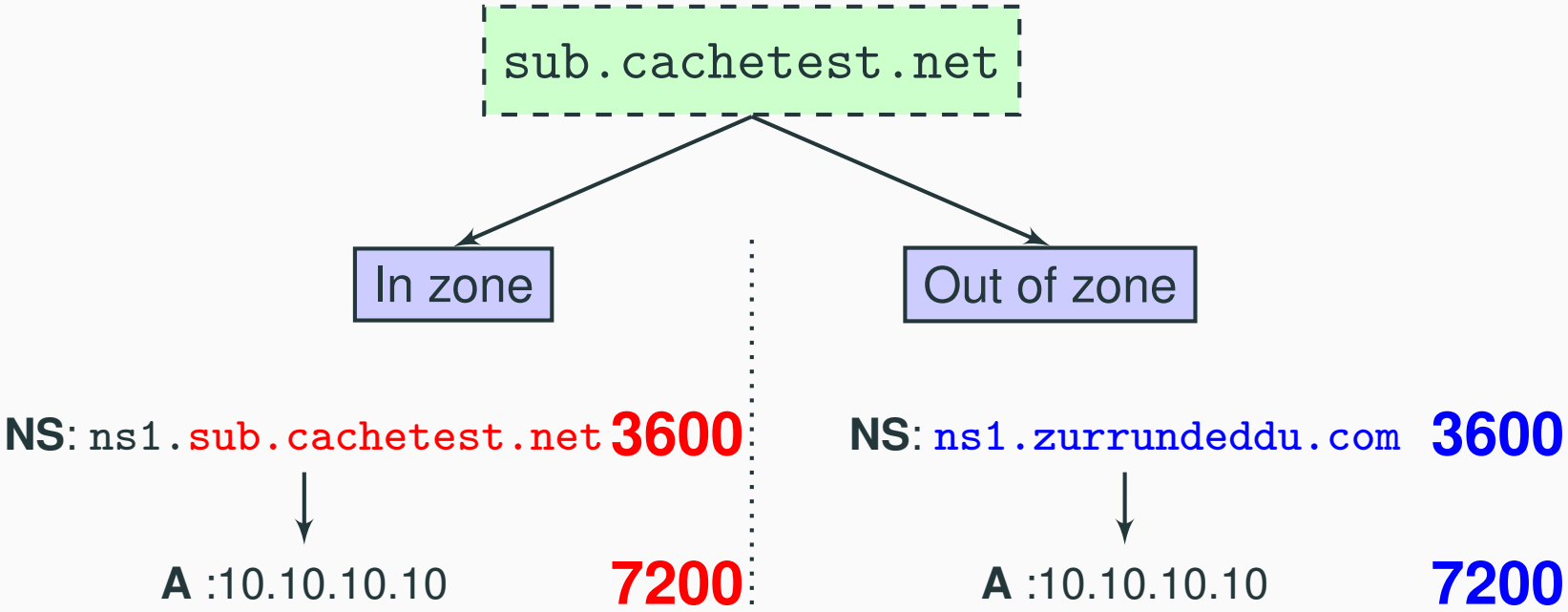
To resolve *.sub.cachetest.net, you need both **NS** and **A**

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**

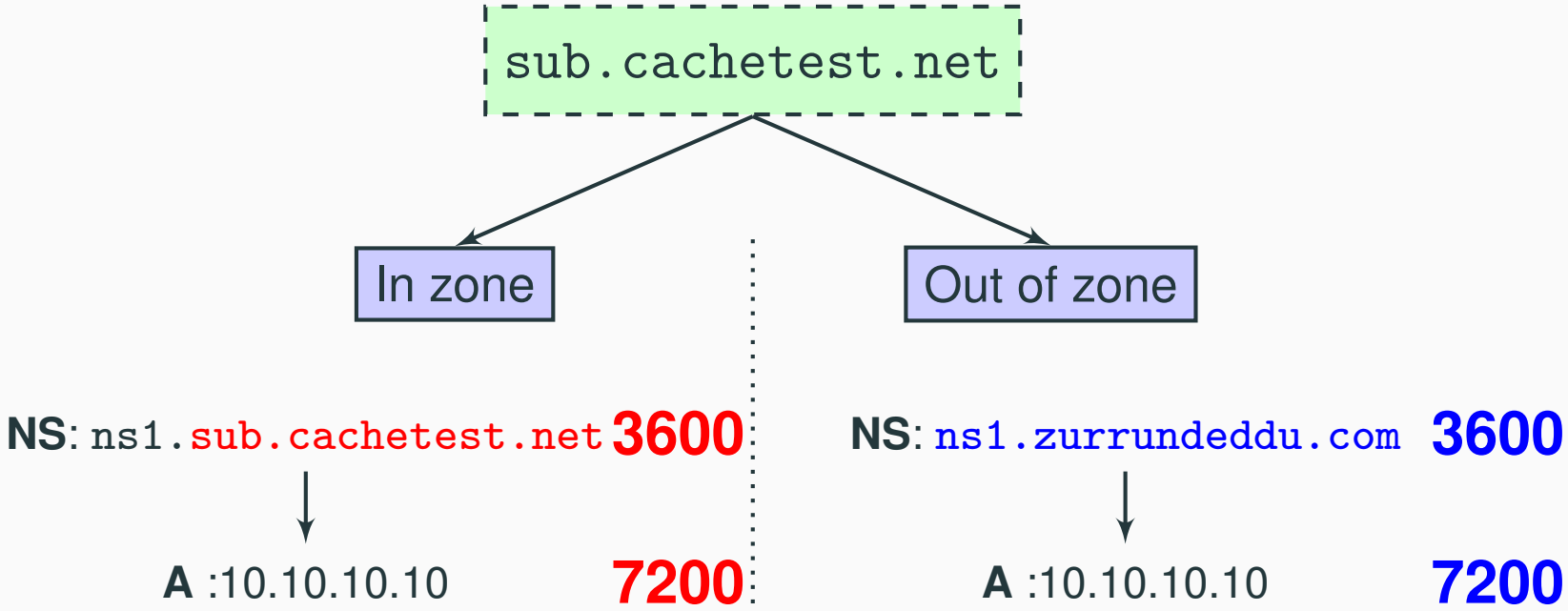
Are NS and A cached independently?

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?

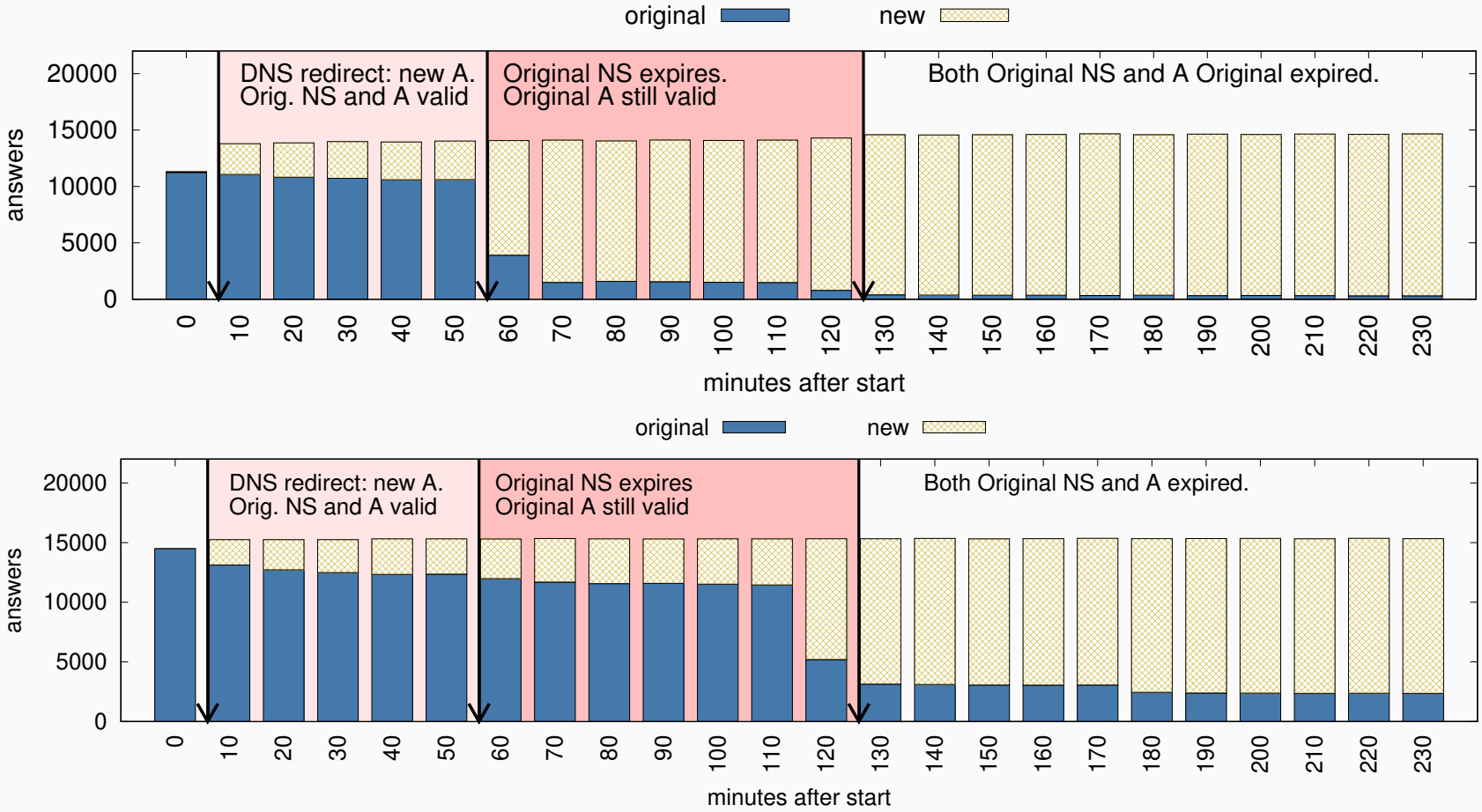
1. t=0: all Atlas VPs query (fills cache with NS and A)
2. t=4800: what happens ? NS is expired; A is still in cache:

do resolvers use the "cached A" or refresh it again?

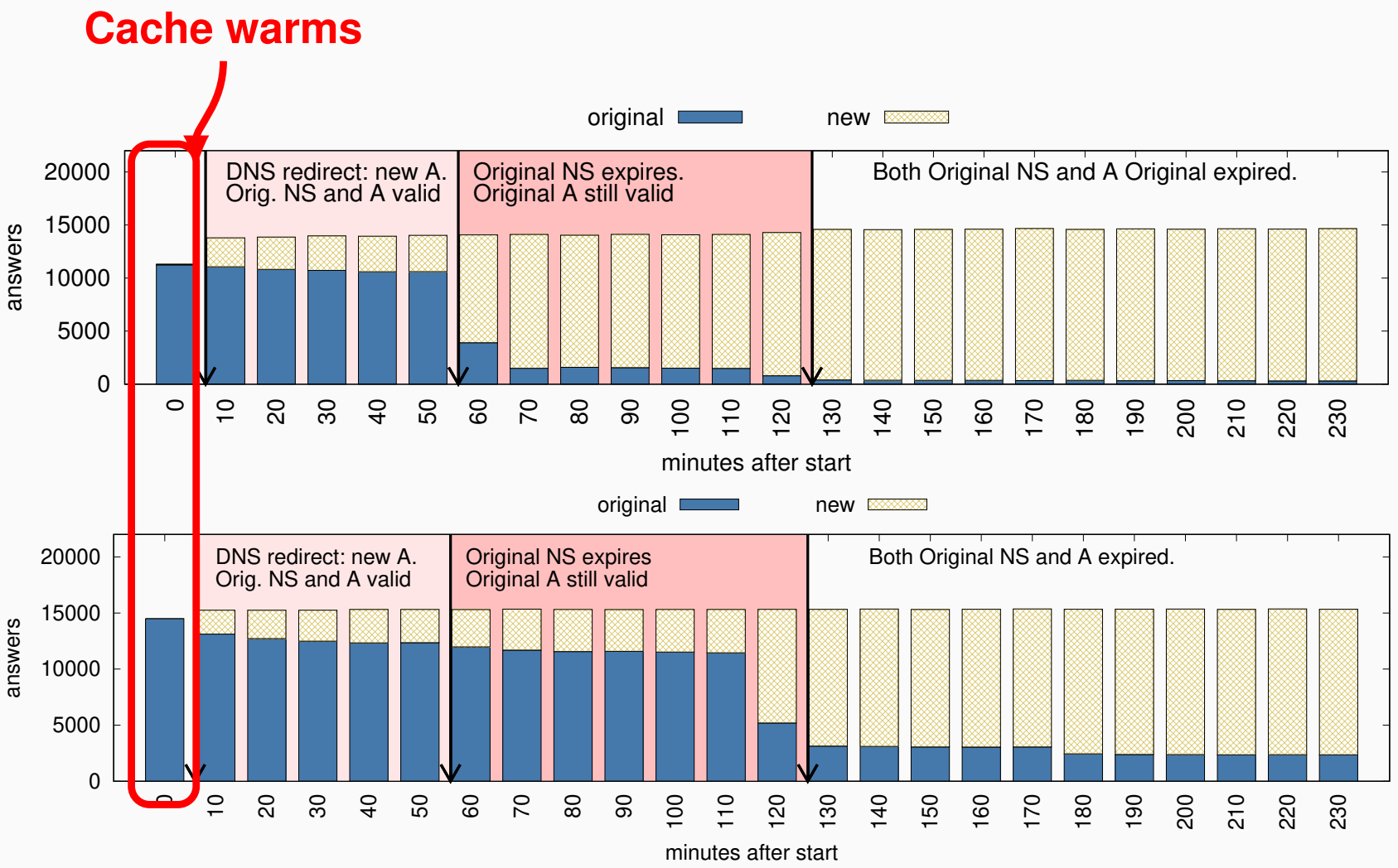
trick: at t=540, *we renumber A to 10.10.10.2* (diff answer)

Will Marcus Aurelius receive cached or new answer?

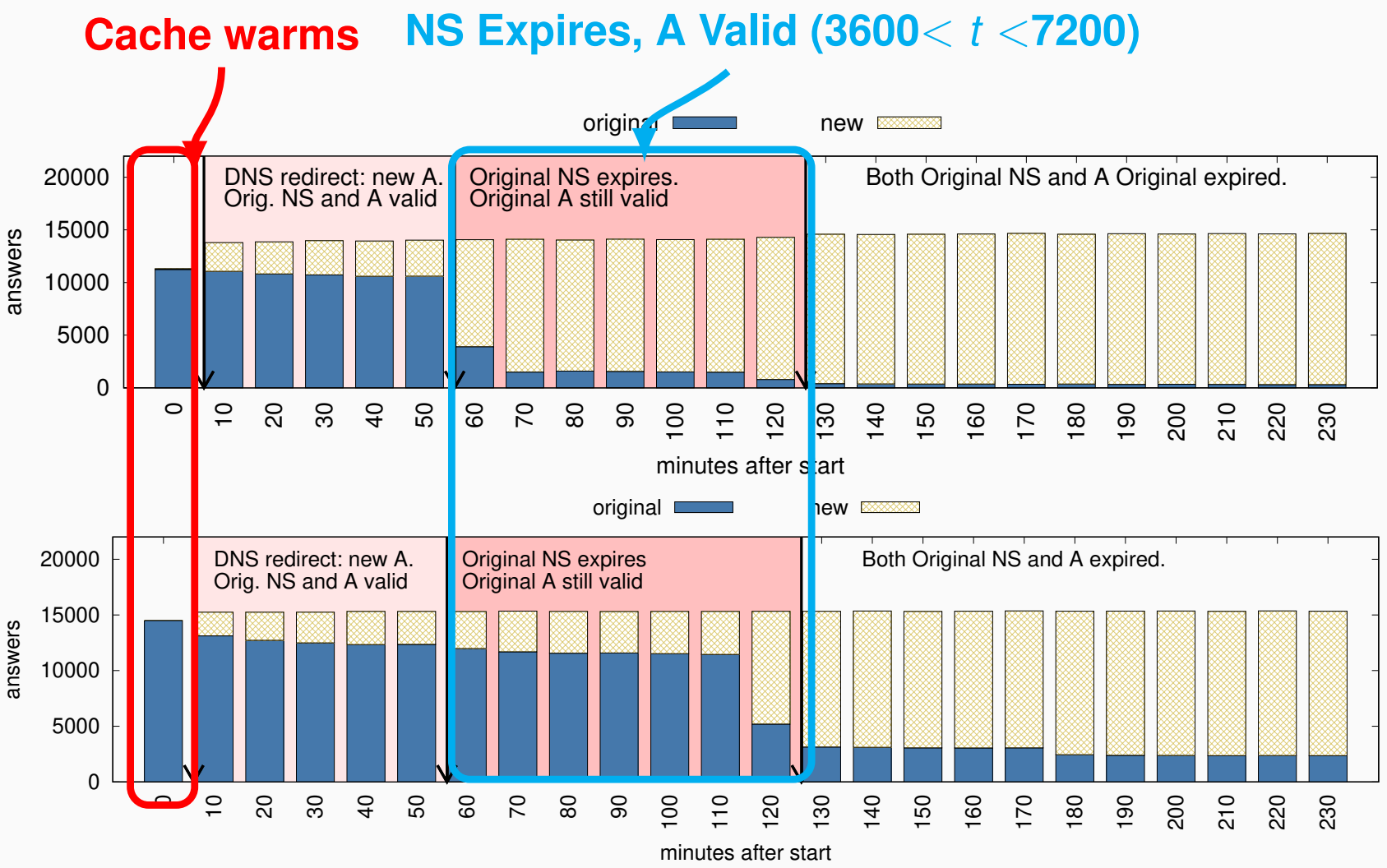
Are they dependent? Yes, for in zone



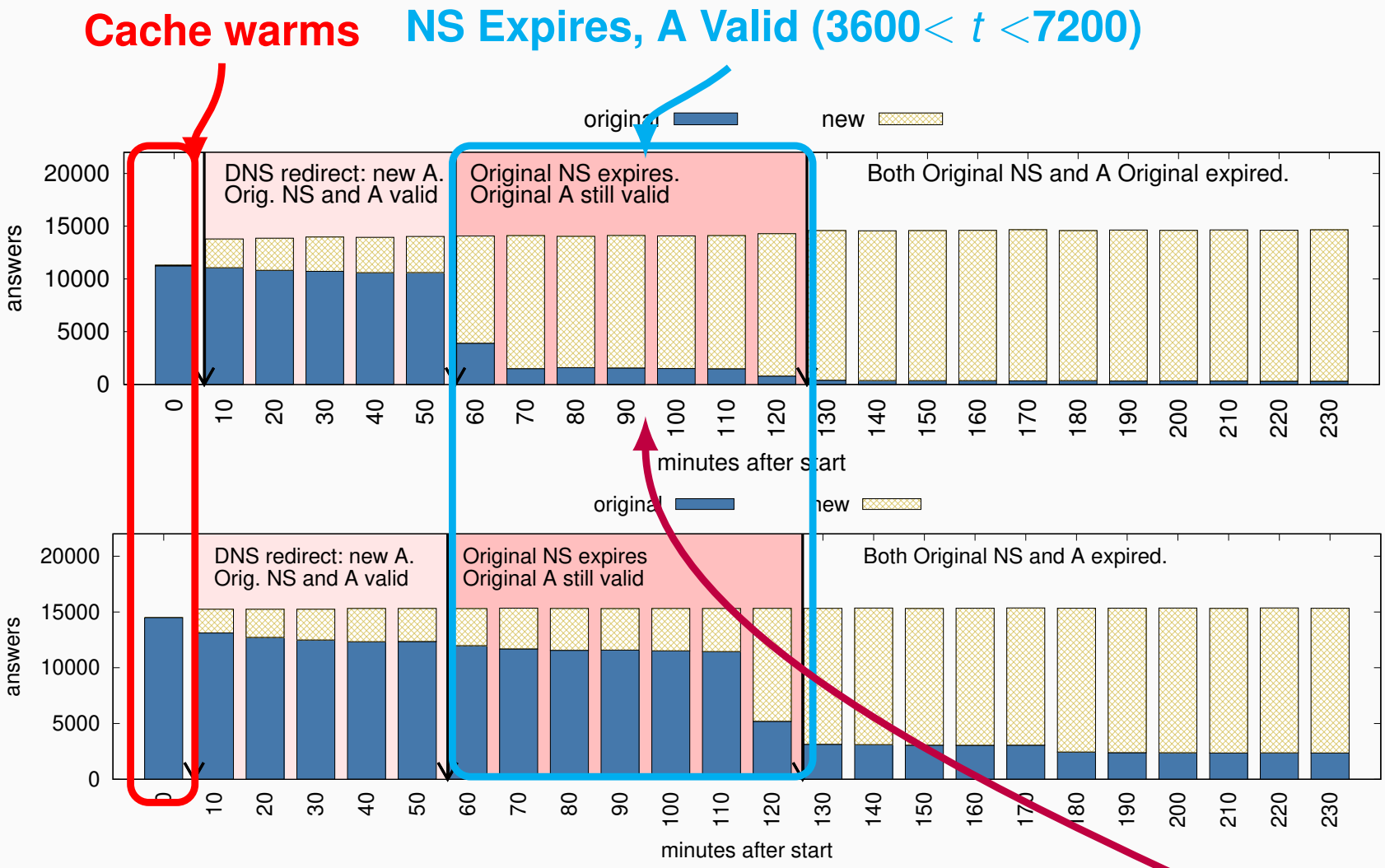
Are they dependent? Yes, for in zone



Are they dependent? Yes, for in zone

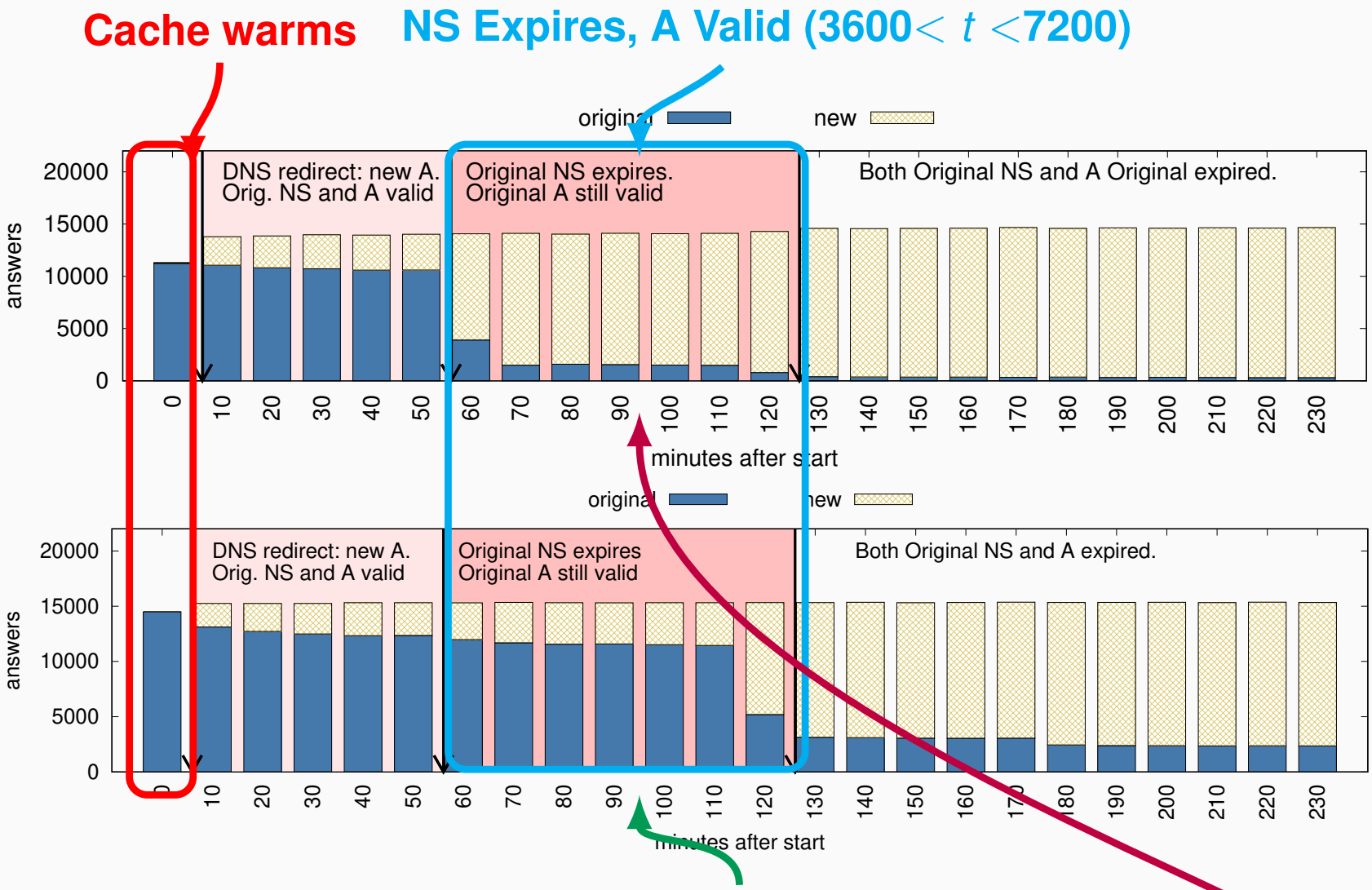


Are they dependent? Yes, for in zone



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

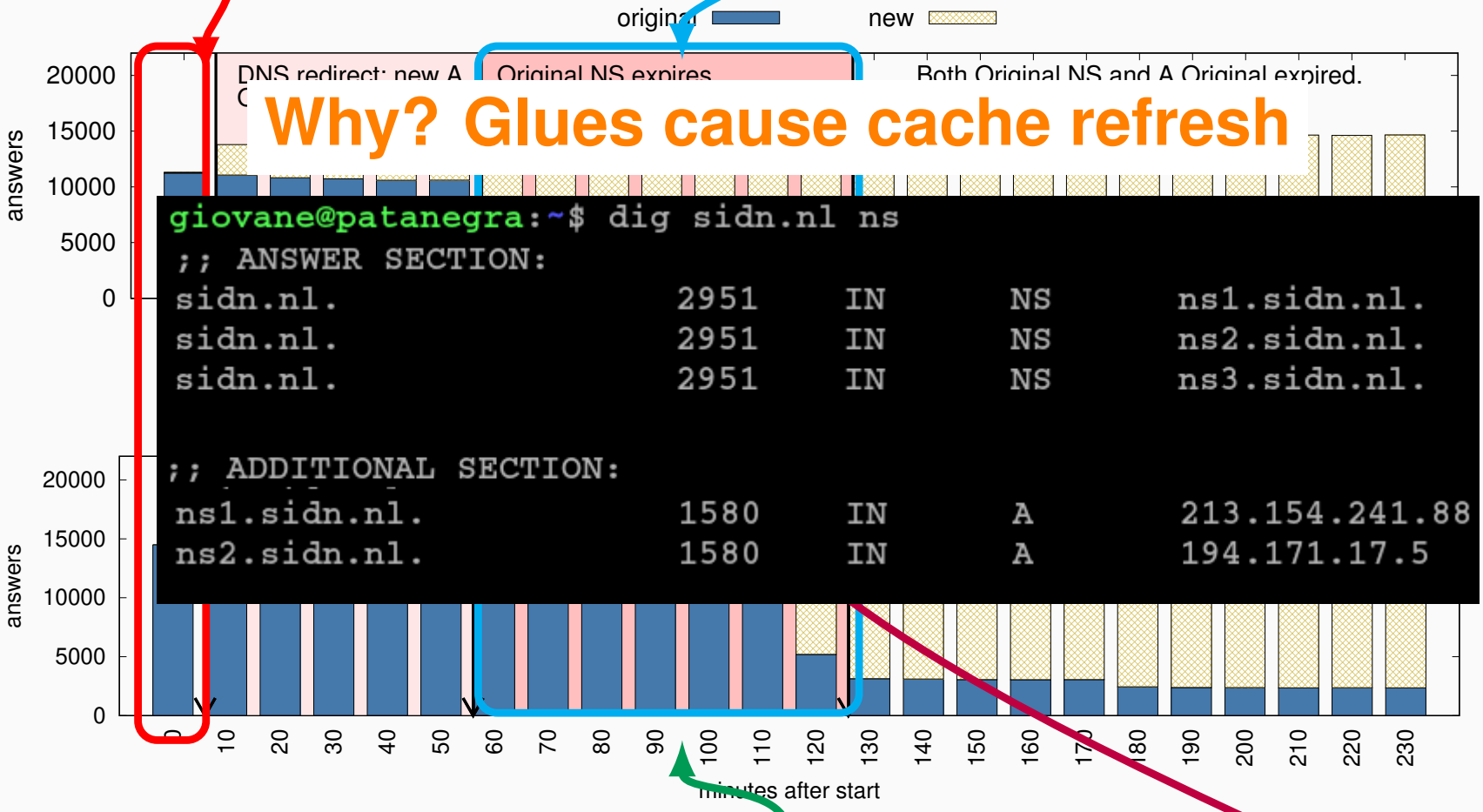
Are they dependent? Yes, for in zone



Are they dependent? Yes, for in zone

Cache warms

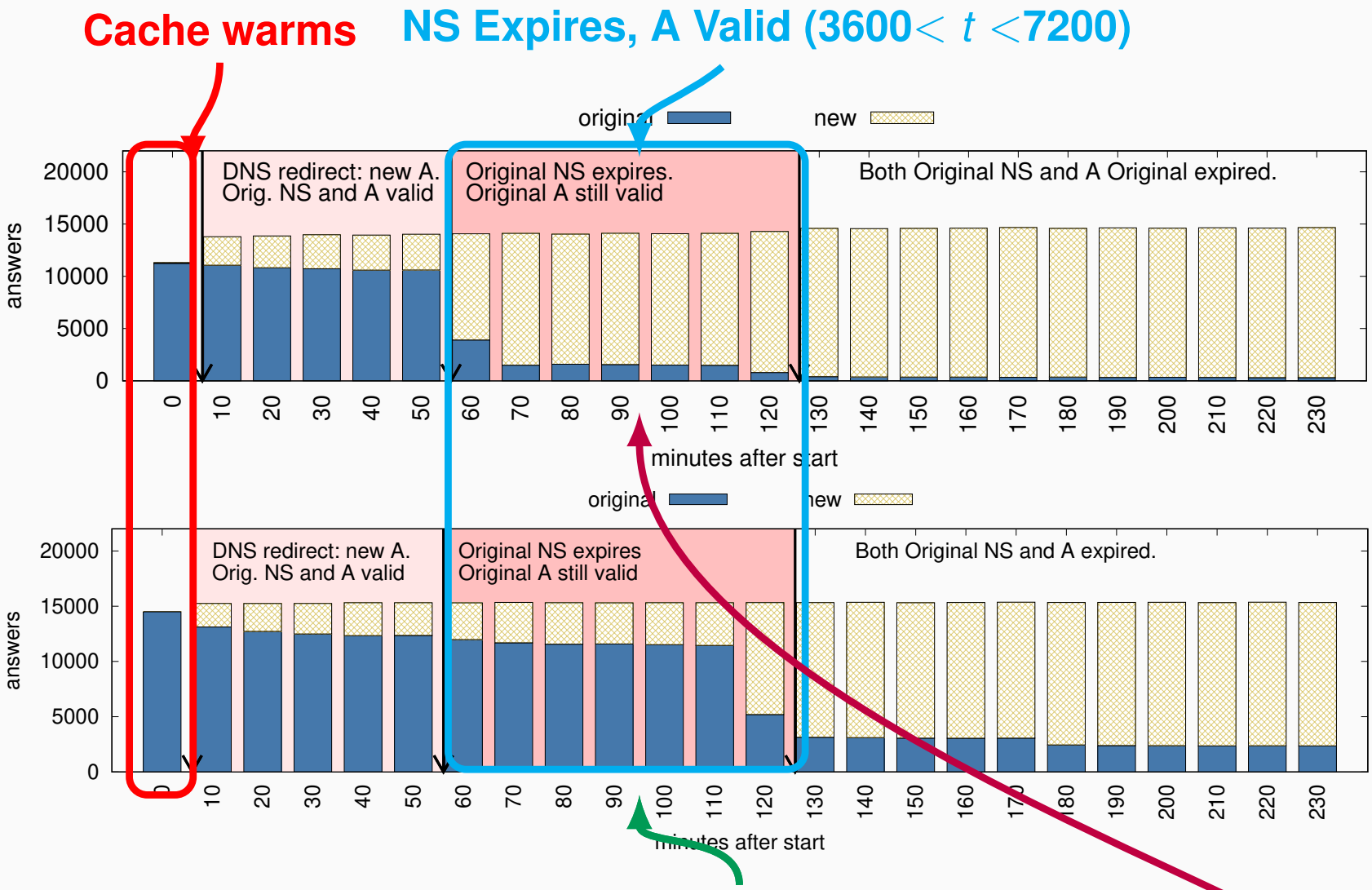
NS Expires, A Valid (3600< t <7200)



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

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

Are they dependent? Yes, for in zone



Are there dependencies between A and NS TTLs?

src:
https://en.wikipedia.org/wiki/Marcus_Aurelius

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

Outline

TTLs Use in the Wild

How are TTLs used in the wild?

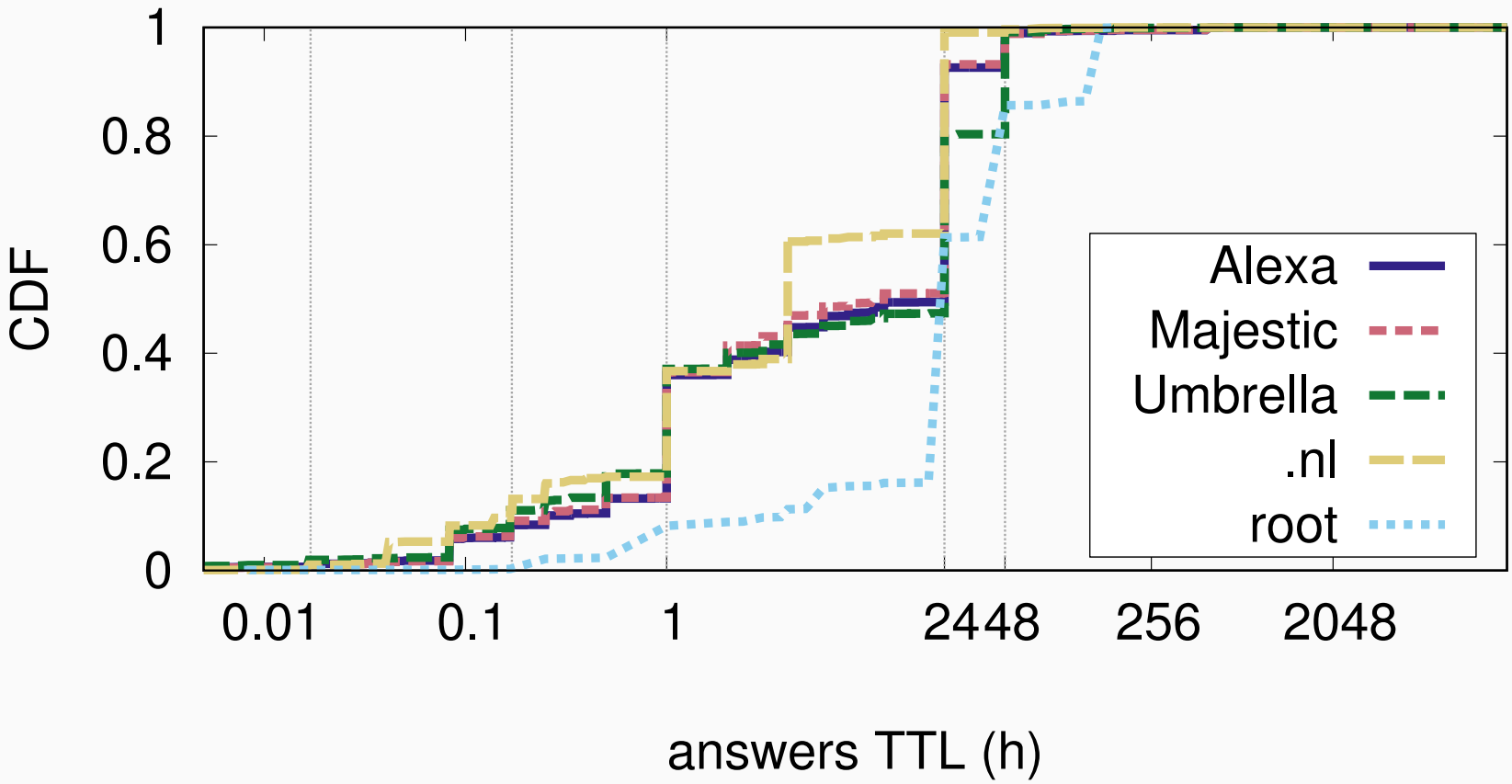
- There is no consensus how to choose TTLs
- But folks have to choose them anyway
- We use 5 lists:
  - Alexa
  - Majestic
  - Umbrella
  - `.nl`
  - Root (TLDs)
- We probe several records types
- We analyze **child TTL** values
- And discuss results with some operators

Most domains are out-of-bailiwick

	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

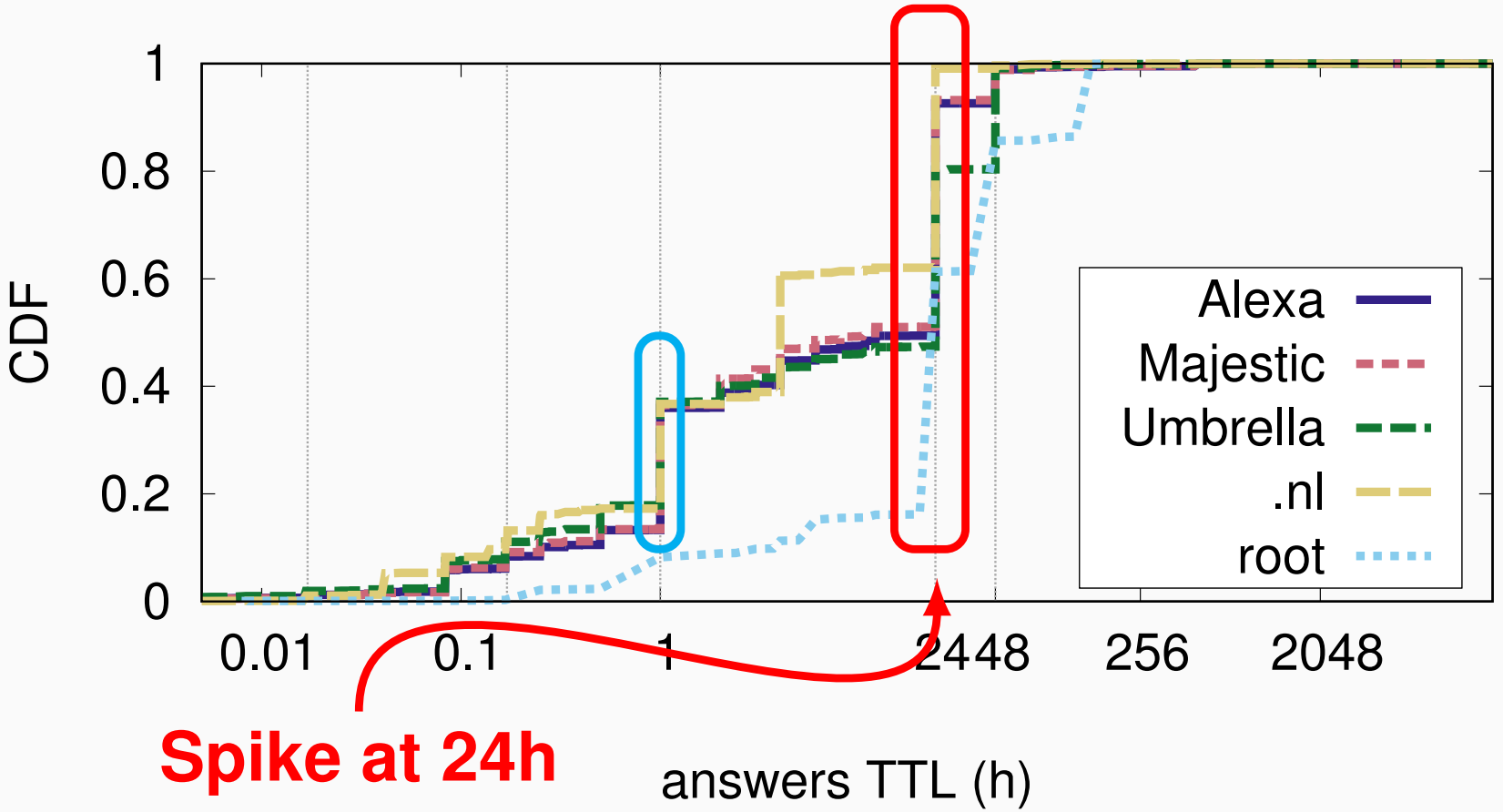
- Out of bailiwick (out-of-zone): **records are cached independently** (no glues)
- Your chosen TTLs values for different records will be respected

NS records have longer TTLs (>24h)



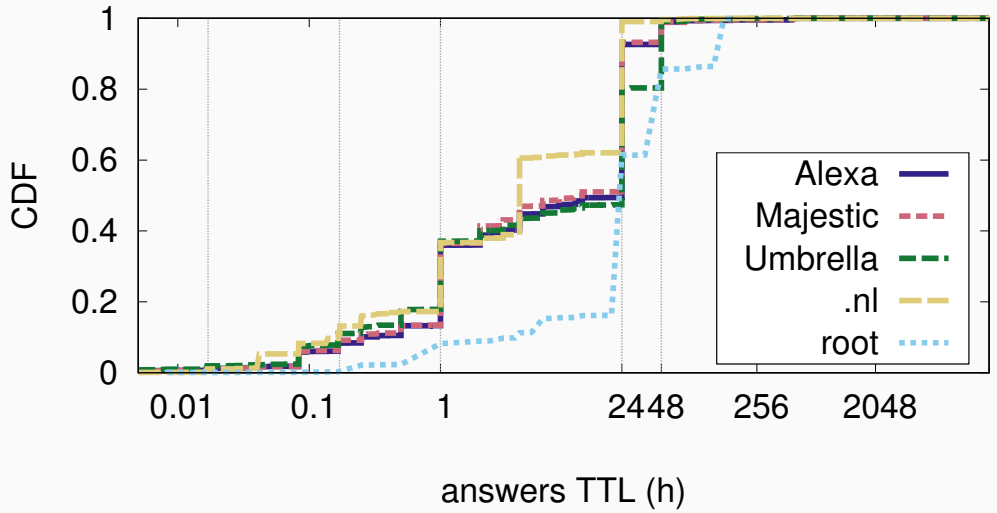
- > 60% NS records are long (**Good** for caching and performance)
- But 40% are one hour or less (not so good)

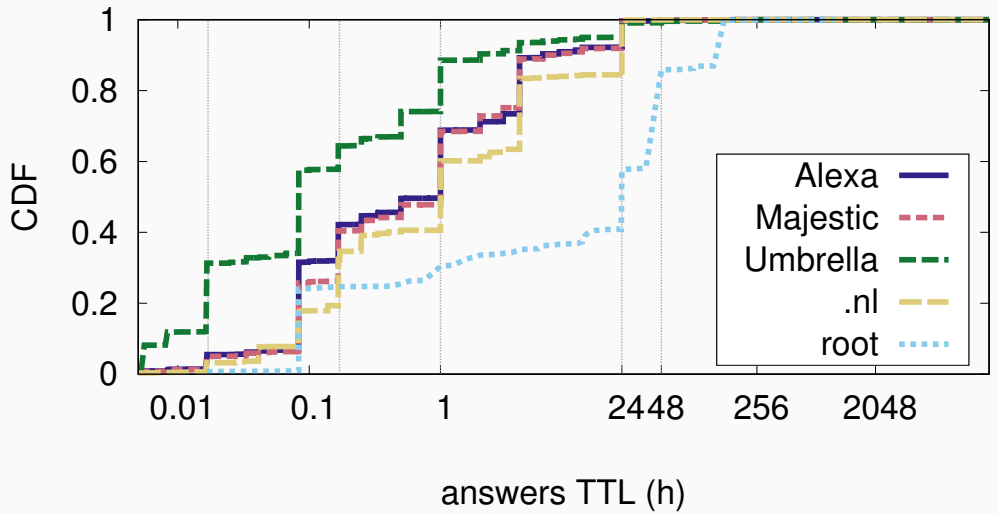
NS records have longer TTLs (>24h)



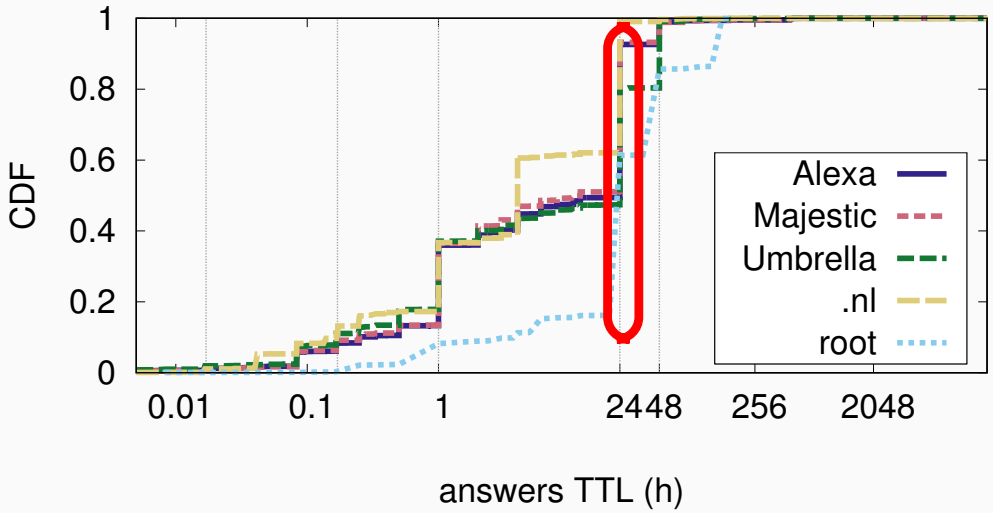
- > 60% NS records are long (**Good** for caching and performance)
- But 40% are one hour or less (not so good)

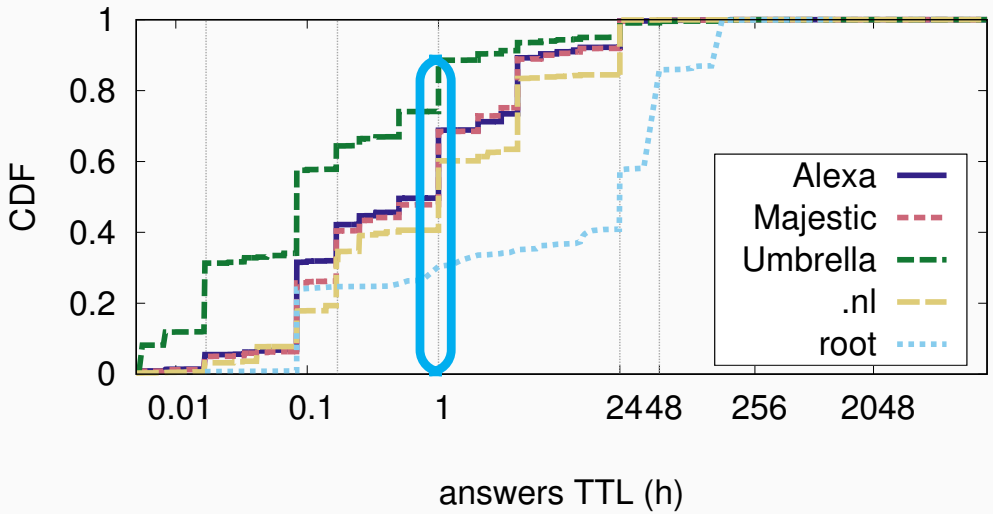
A records TTLs far shorter than NS



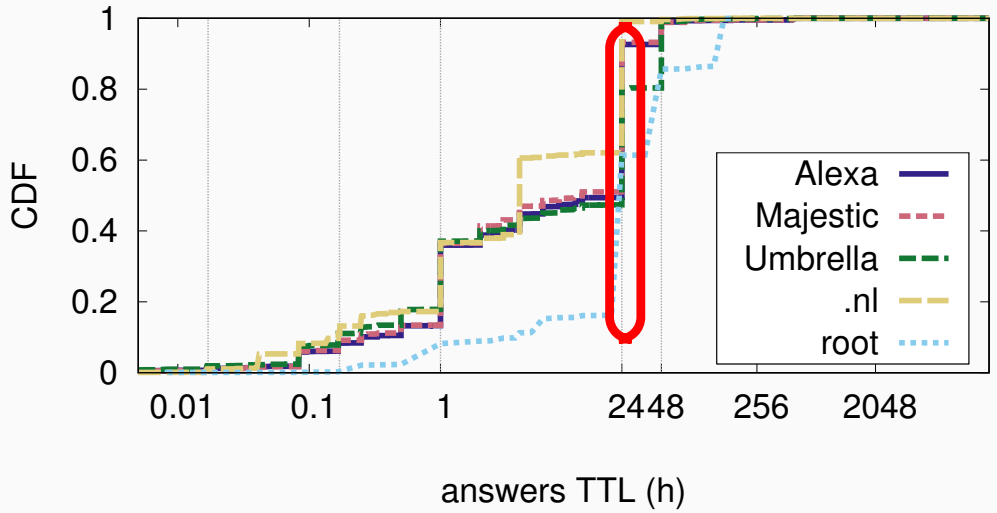


A records TTLs far shorter than NS

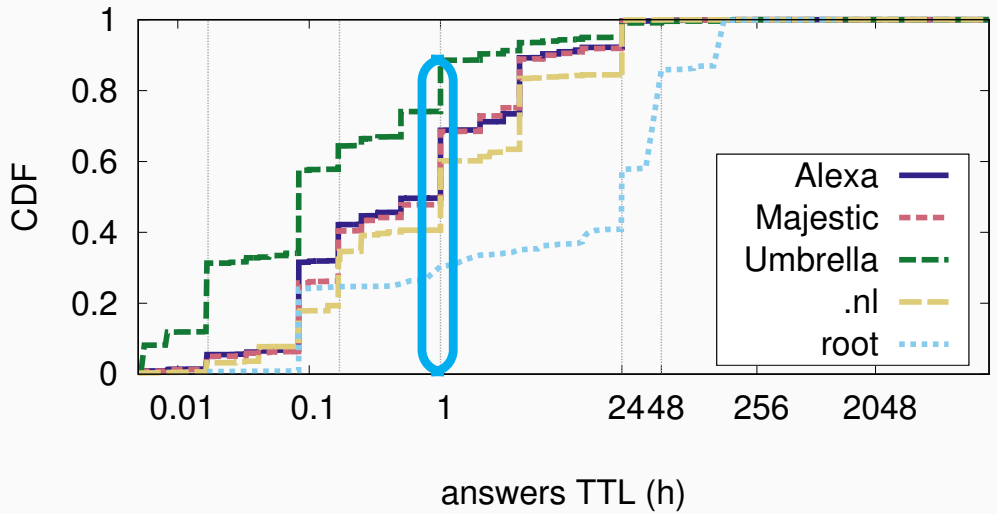




A records TTLs far shorter than NS



Shorter A records TTLs leads to poor caching



Operators Notification: 3 changed their TTLs

- We found **34 TLDs** with short TTL for NSes (<=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 from 300s to 86400s

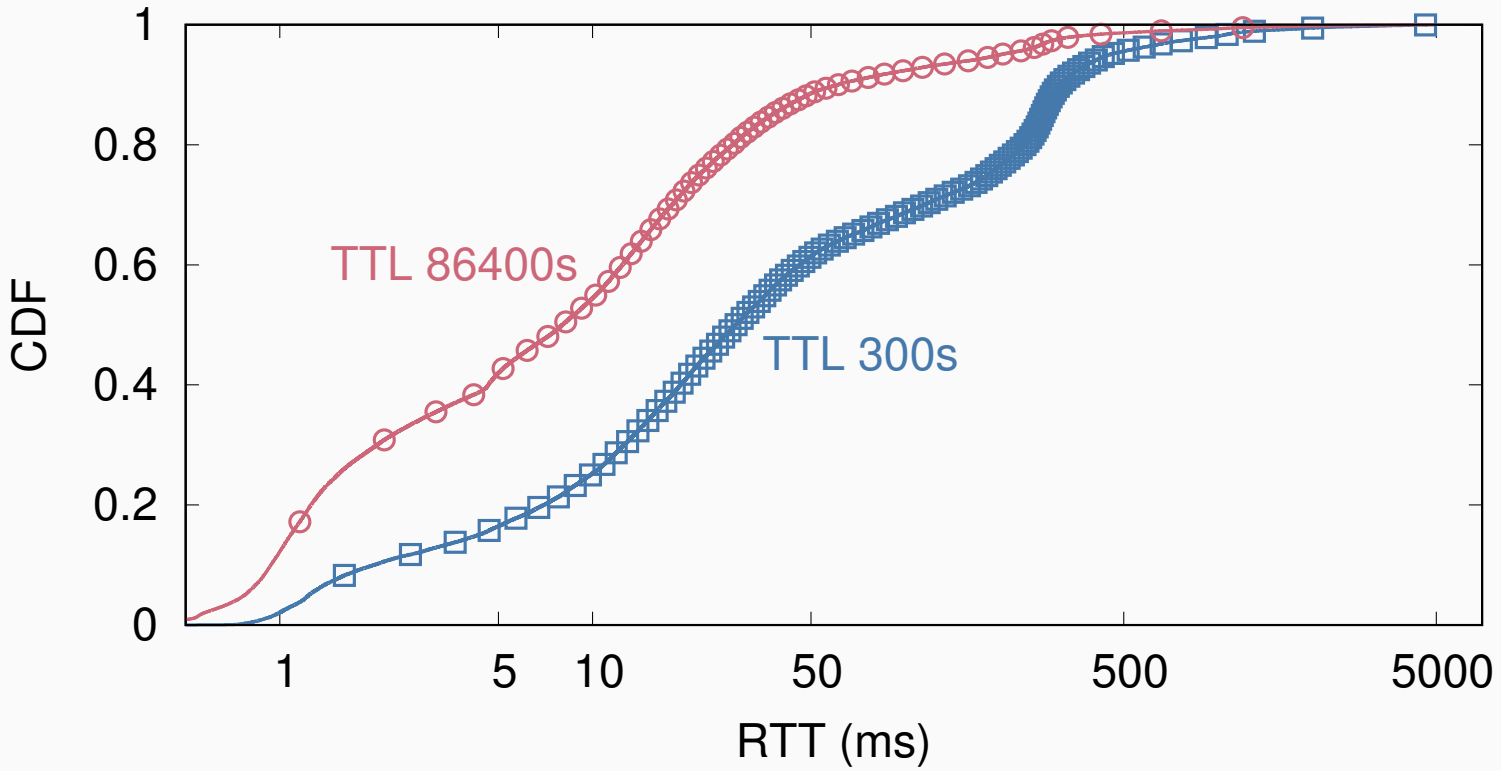


Figure 4: RTT from RIPE Atlas VPs for NS .uy queries (NS)

.uy latency reduced a lot!

- .uy NS TTL from 300s to 86400s

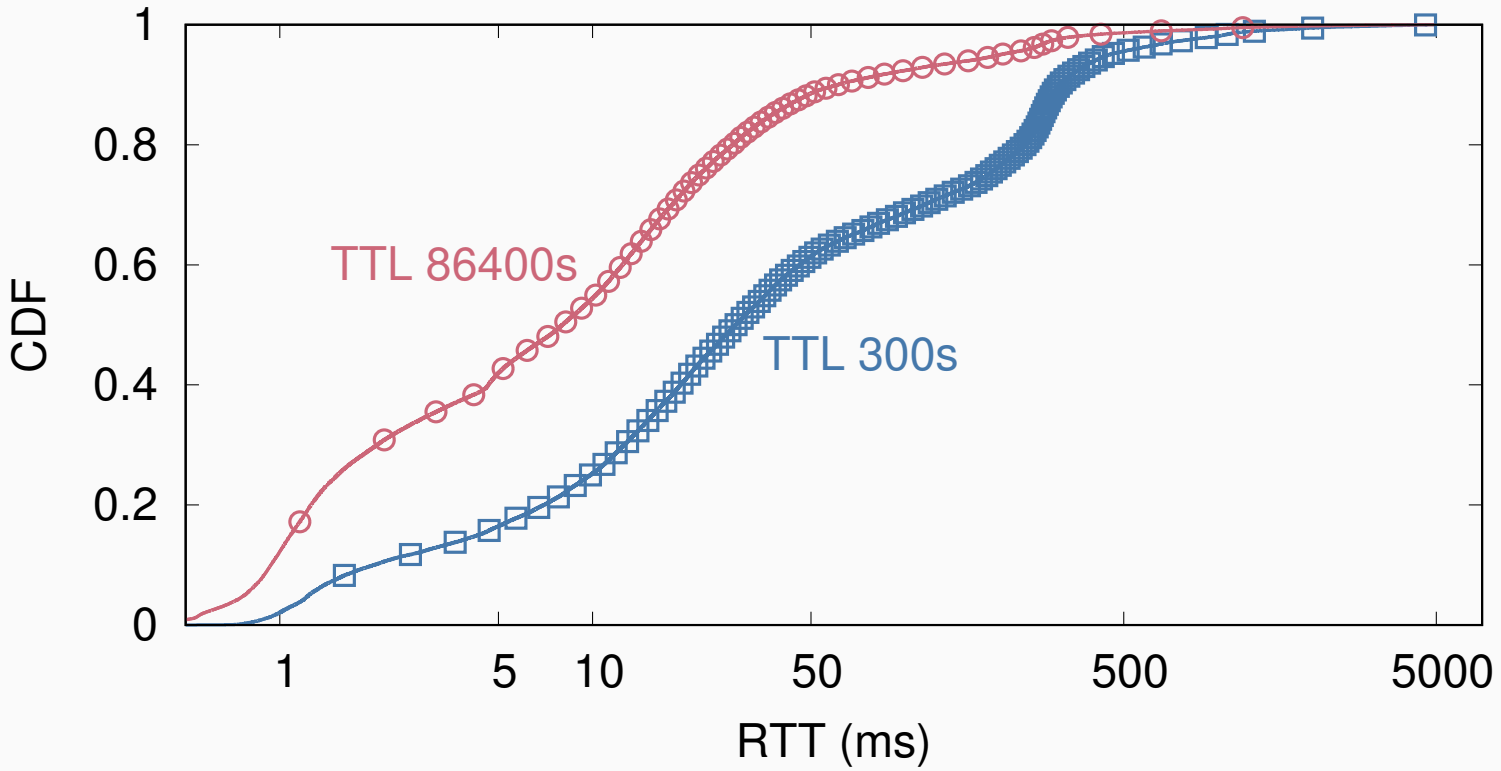


Figure 4: RTT from RIPE Atlas VPs for NS .uy queries (NS)

.uy latency reduced a lot!

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

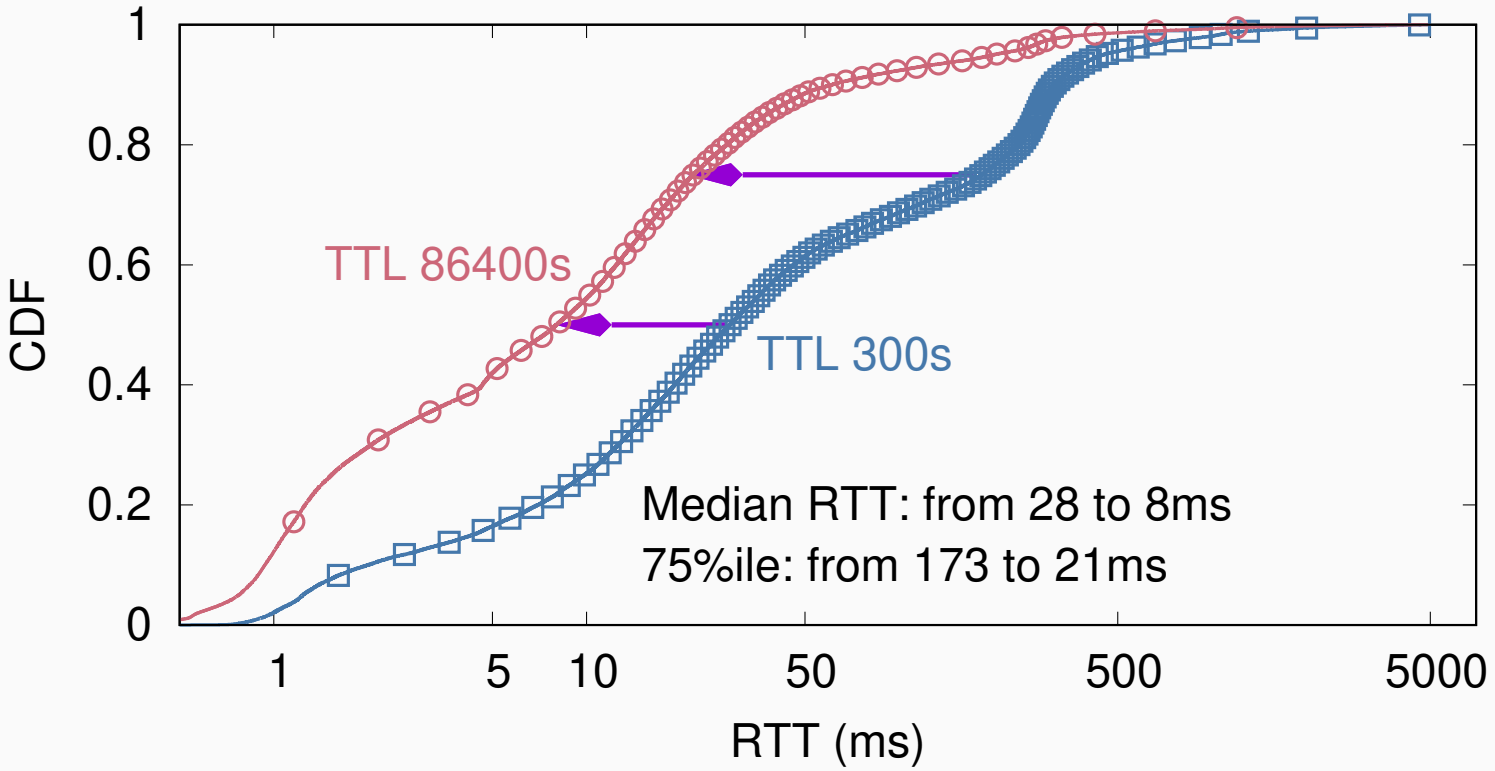


Figure 5: RTT from RIPE Atlas VPs for NS .uy queries (NS)

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

.uy latency reduced for all regions

Check for Atlas location bias

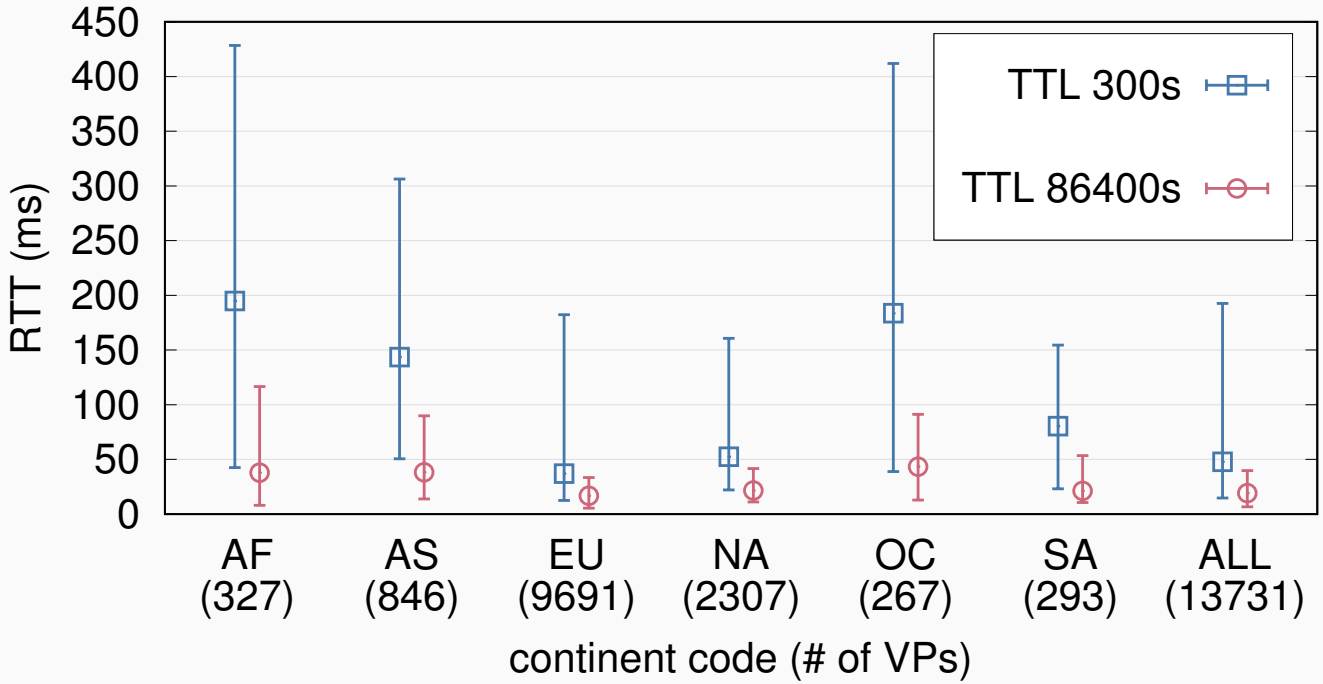


Figure 6: Median RTT as seen by RIPE Atlas VPs per region

Longer TTL → longer caching → faster answers

.uy latency reduced for all regions

Check for Atlas location bias

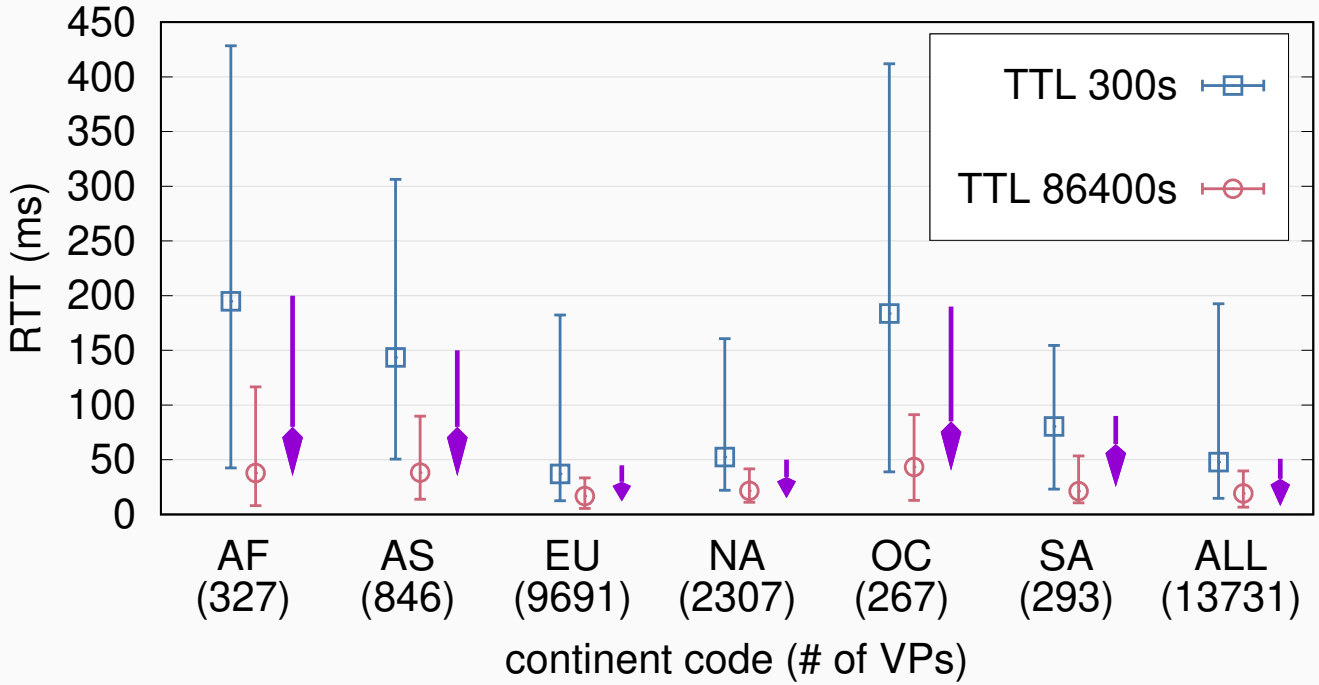


Figure 7: Median RTT as seen by RIPE Atlas VPs per region

Longer TTL → longer caching → faster answers

Up to 150ms median latency reduction (AF)

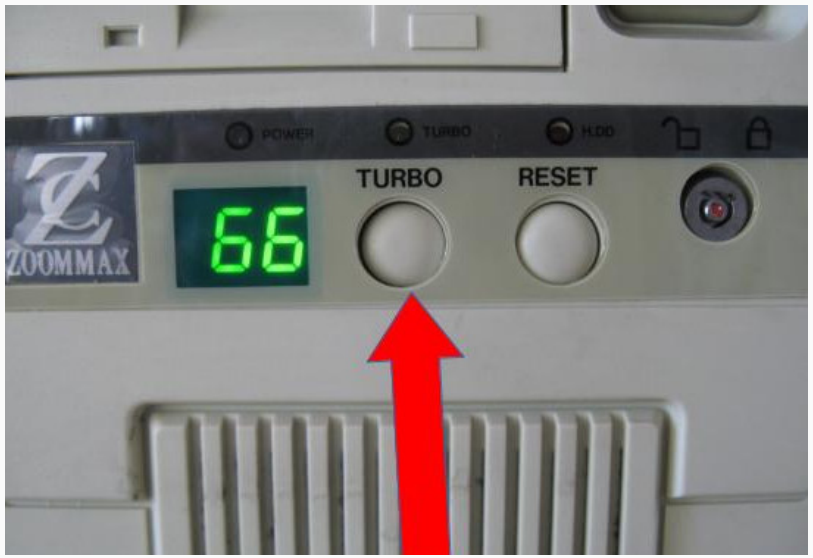
We are no Luiz Suárez... but

- We still helped Uruguayan `.uy` users
- And two other countries:
  - One in East Africa
  - Another one in the Middle East
- Experiment 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

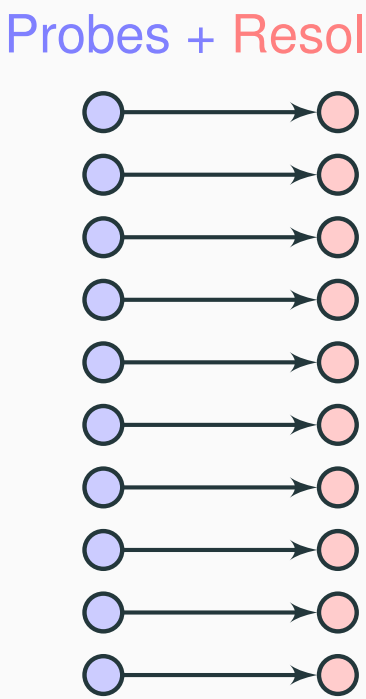
Outline

Caching (Longer TTL) vs Anycast

Caching vs Anycast

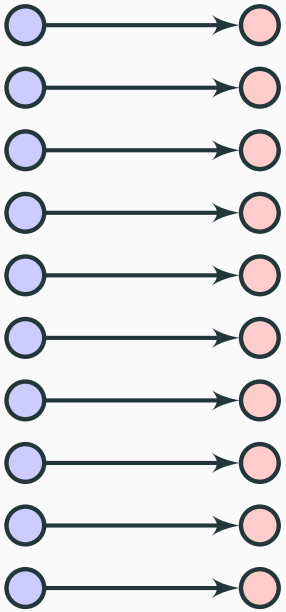
- People and CDNs spend lots on huge anycast deployments
- OPs could say: "*I'll have short TTL since I use anycast*", because anycast can make it up for it.
- **Does anycast really beats caching?**

Caching vs Anycast: experiment

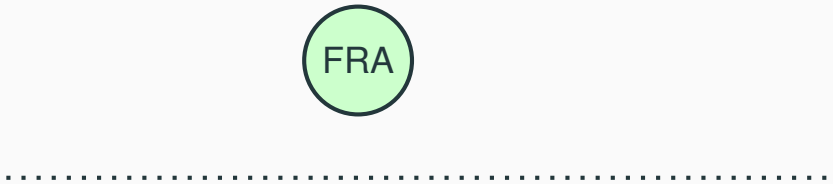


Caching vs Anycast: experiment

Probes + Resolver

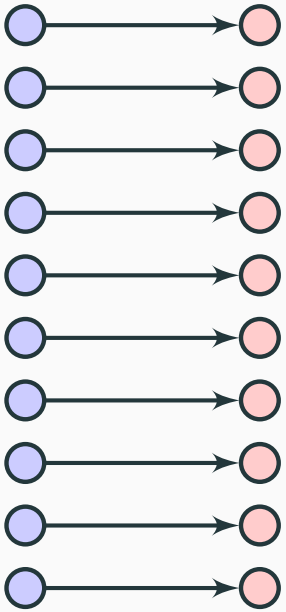


Unicast (EC2)

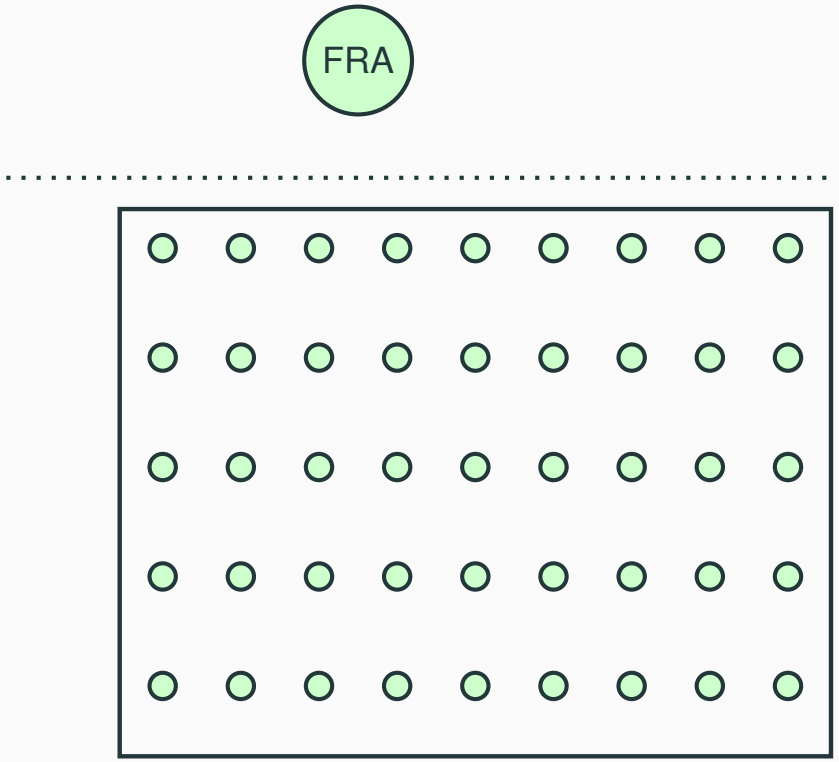


Caching vs Anycast: experiment

Probes + Resolver



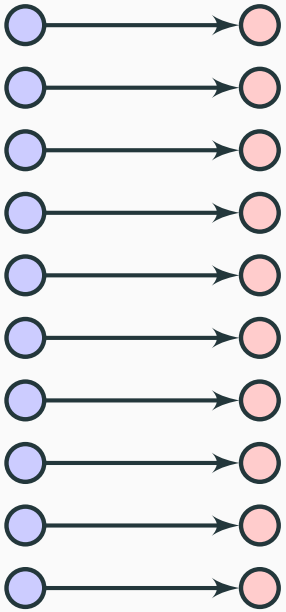
Unicast (EC2)



Anycast (Route53)

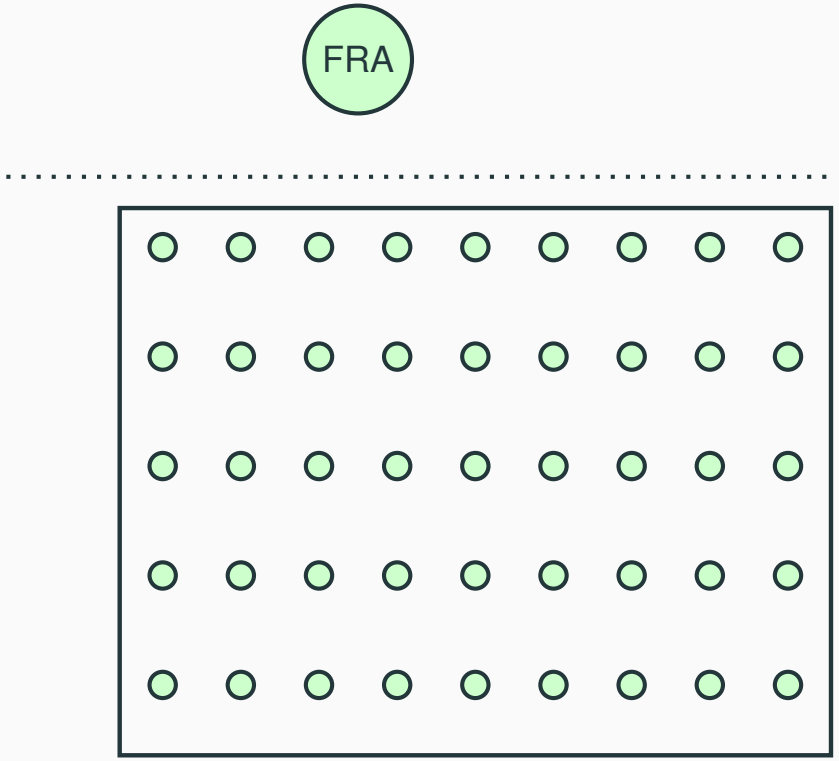
Caching vs Anycast: experiment

Probes + Resolver



Unicast (EC2)

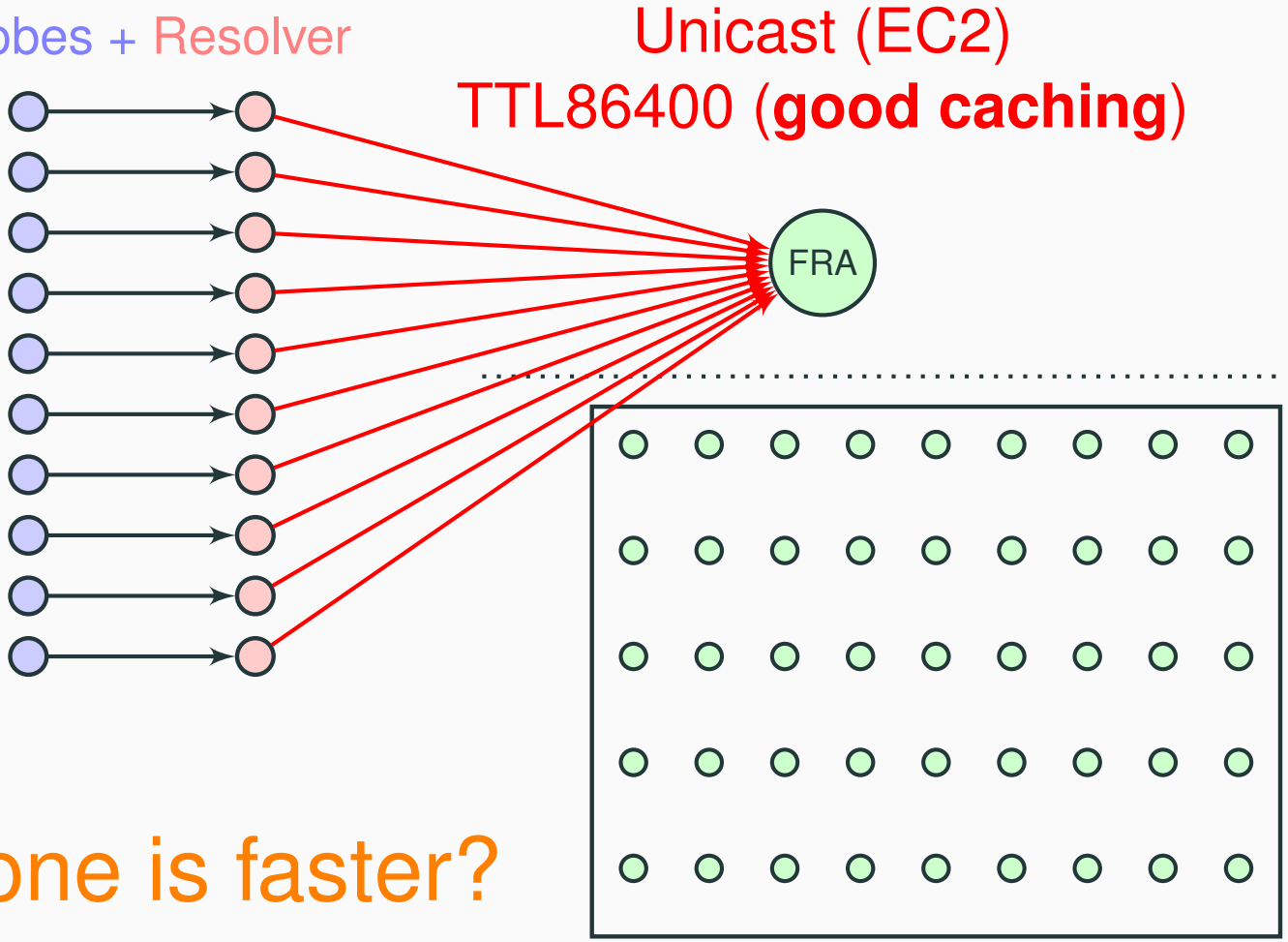
TTL86400 (**good caching**)



TTL60s (**no caching**)

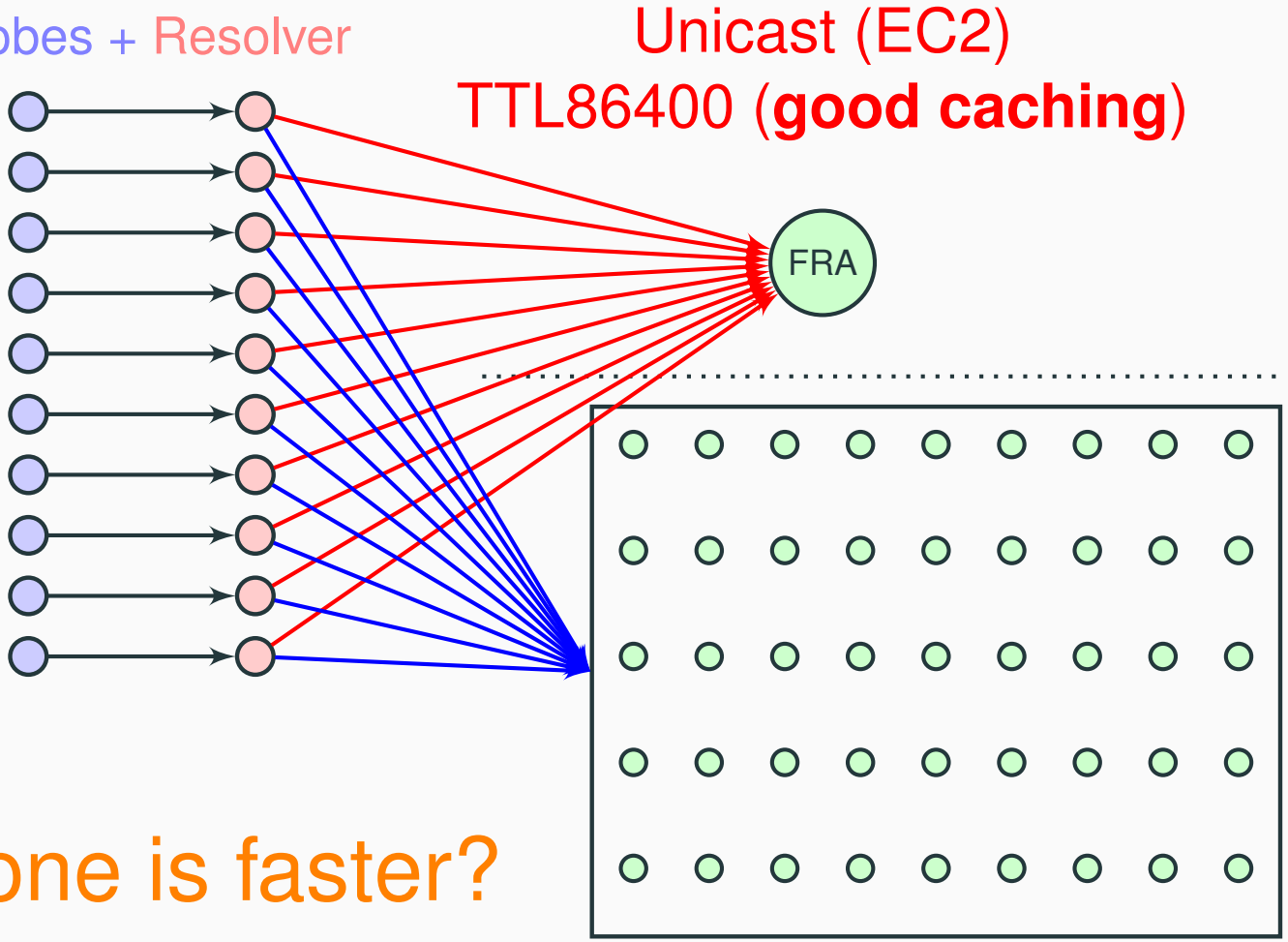
Anycast (Route53)

Caching vs Anycast: experiment



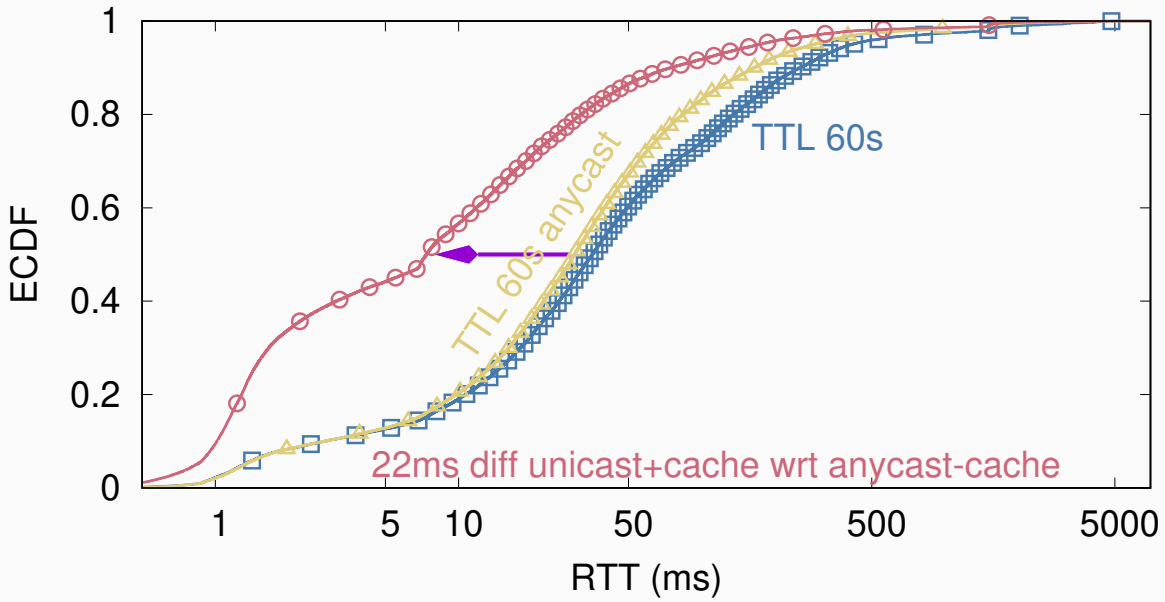
Which one is faster?

Caching vs Anycast: experiment



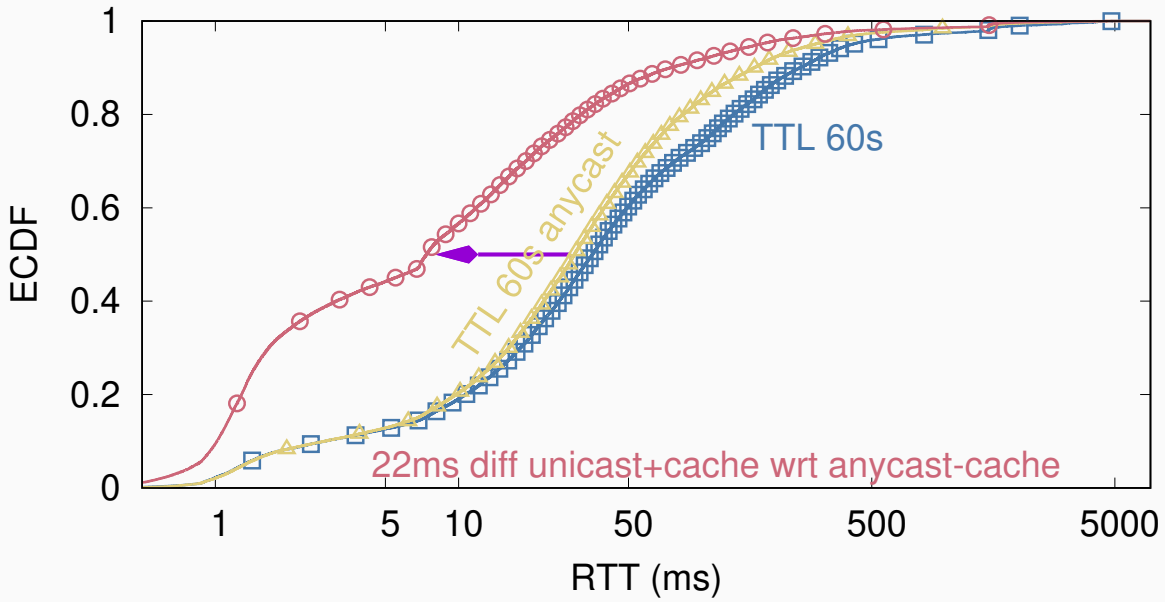
Which one is faster?

TTLs (caching) matter more than anycast



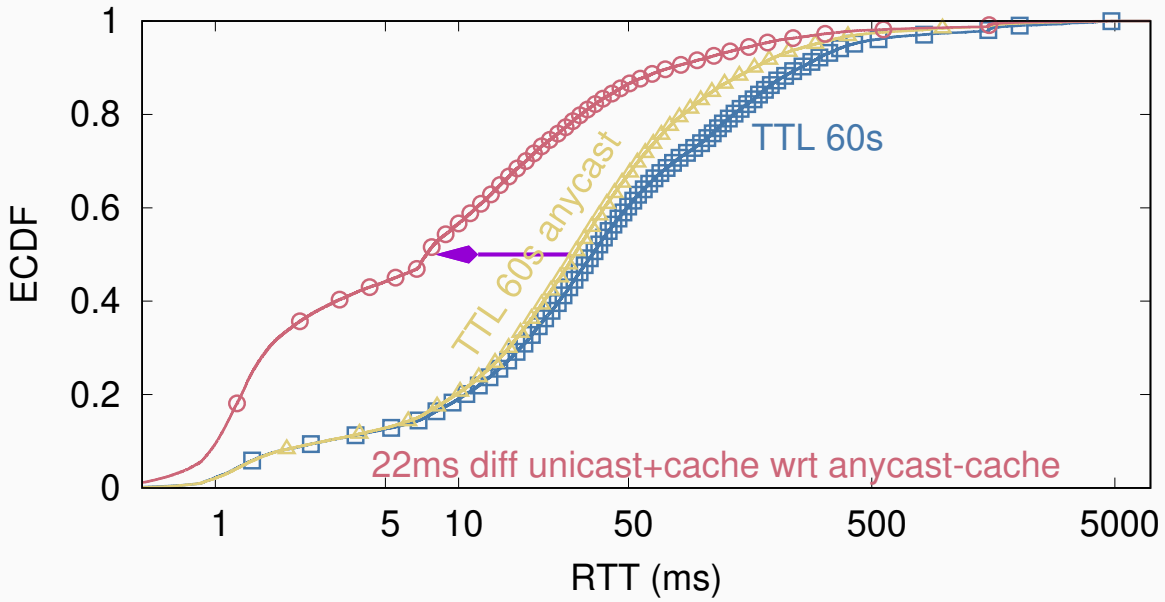
- Caching near client beats even great server infrastructure!
  - Anycast TTL60 (no cache): **29.96ms** (median)
  - Unicast TTL86400 (cache): **7.38ms** (median):
  - **22ms median latency reduction**
- Query load: **77% down** with caching
- so TTLs matter more for performance
  - (anycast is great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

TTLs (caching) matter more than anycast



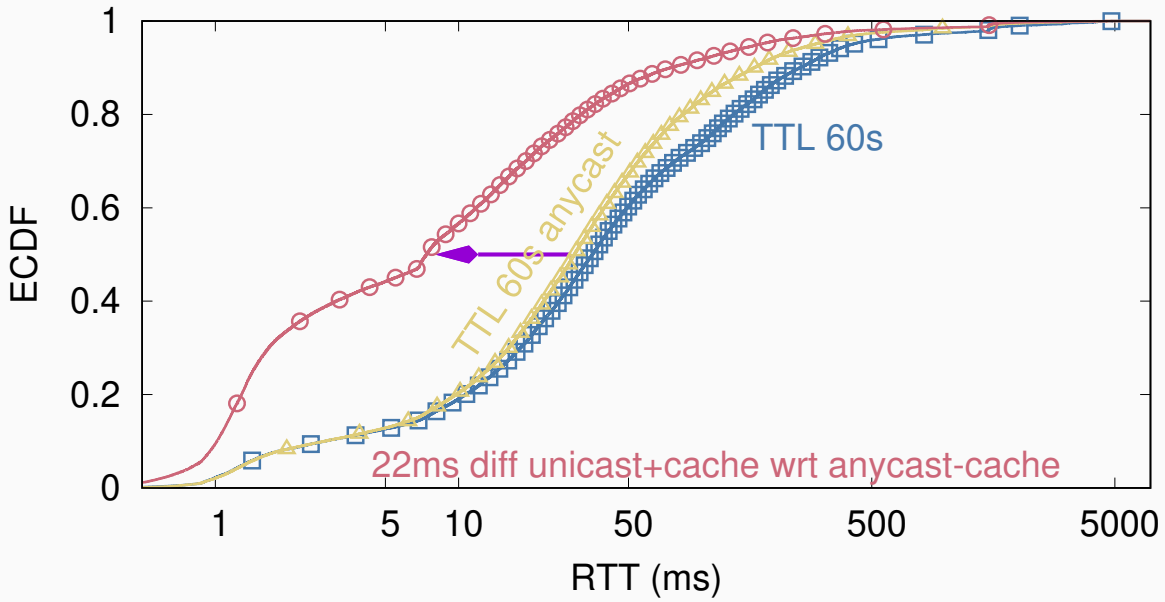
- Caching near client beats even great server infrastructure!
  - Anycast TTL60 (no cache): **29.96ms** (median)
  - Unicast TTL86400 (cache): **7.38ms** (median):
  - **22ms median latency reduction**
- Query load: **77% down** with caching
- so TTLs matter more for performance
  - (anycast is great 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]

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]

Outline

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

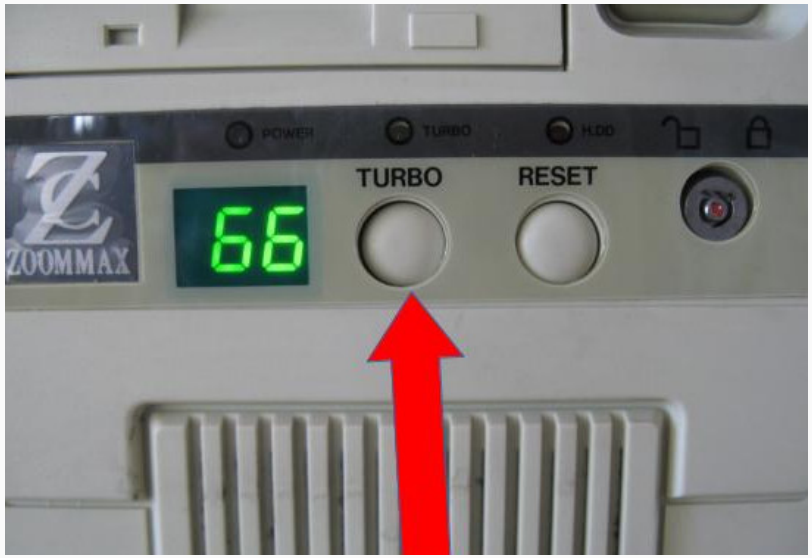
Organizations must weight these trade-offs to find a good balance

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
- 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.