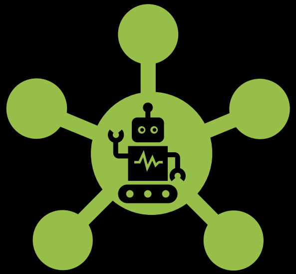
Trufflehunter: Cache Snooping Rare Domains at Large Public DNS Resolvers

Audrey Randall, Enze “Alex” Liu, Gautam Akiwate, Ramakrishna Padmanabhan, Geoffrey M. Voelker, Stefan Savage, Aaron Schulman

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Computer Science and Engineering

Harmful Internet behavior today

Spam Emails

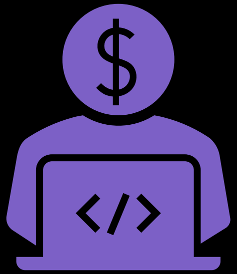


Botnets

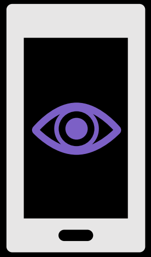
Malware

Common Internet abuse
(well studied)

Typo Squatting



Hack for Hire

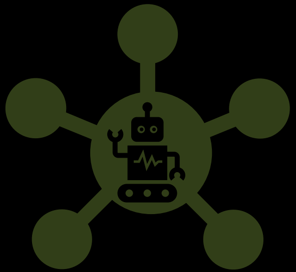


Stalkerware

Rare Internet abuse
(sparsely studied)

Harmful Internet behavior today

Spam Emails

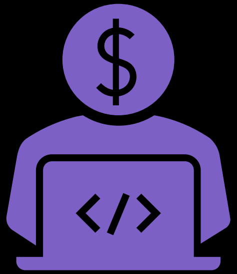


Botnets

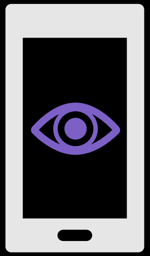
Malware

Common Internet abuse (well studied)

Typo Squatting



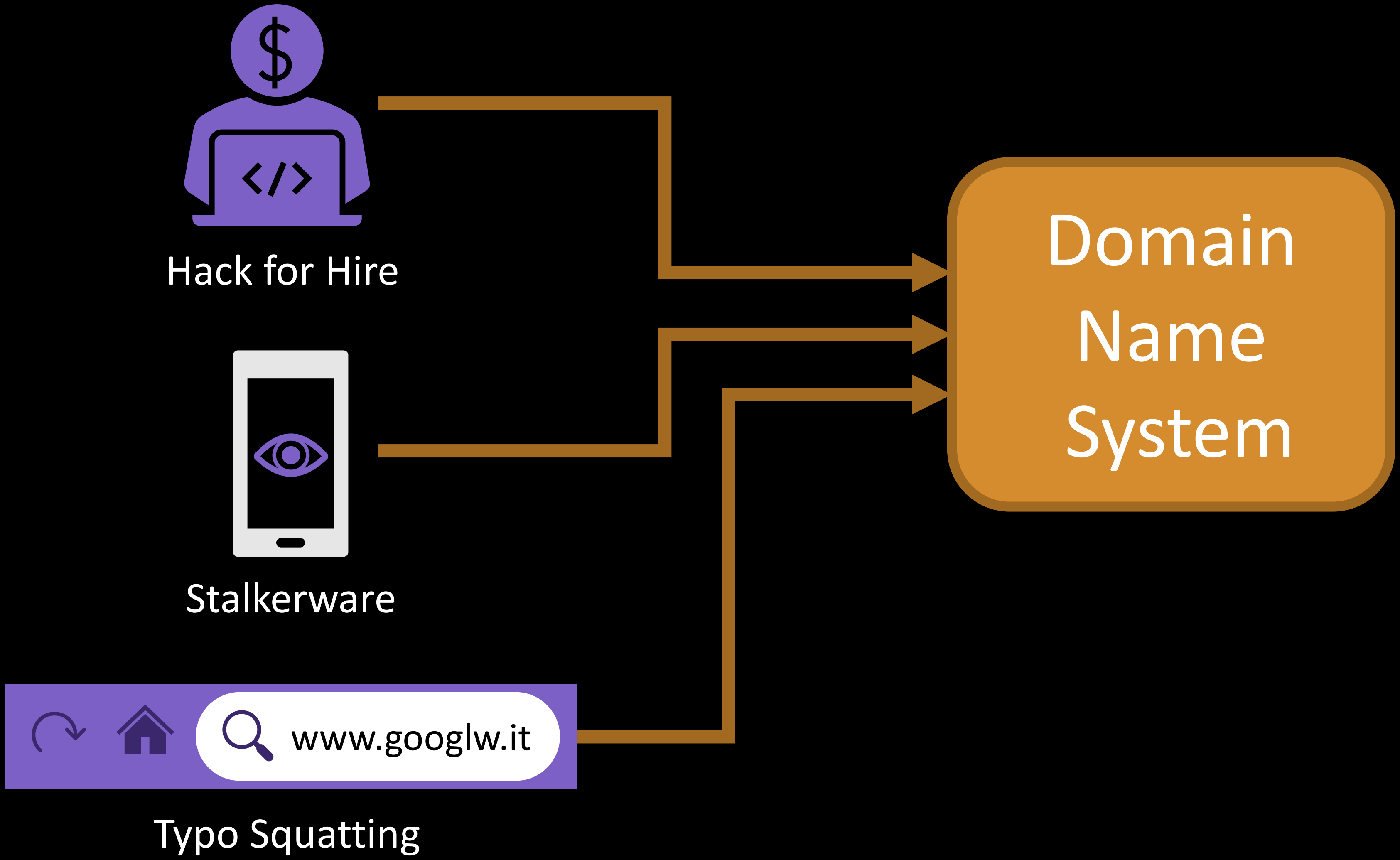
Hack for Hire



Stalkerware

Rare Internet abuse (sparsely studied)

Categories of harmful Internet behavior



If you can observe DNS requests, you can study these types of harm.

New Era in DNS: Public Resolvers

Public resolvers are gaining popularity.

They're now often used by default!

- Google home routers
- Firefox
- NYC Public WiFi

Can a third-party observer use these services to observe rare behavior?

Observing requests on public resolvers

Well-known technique: DNS cache snooping.

In the past, considered a privacy threat.

- Often used misconfigured home routers

Public DNS resolvers allow preserving privacy!

- Too many users to de-anonymize

But, public resolvers are more challenging…

- Complicated caching strategies -> some protocol noncompliance

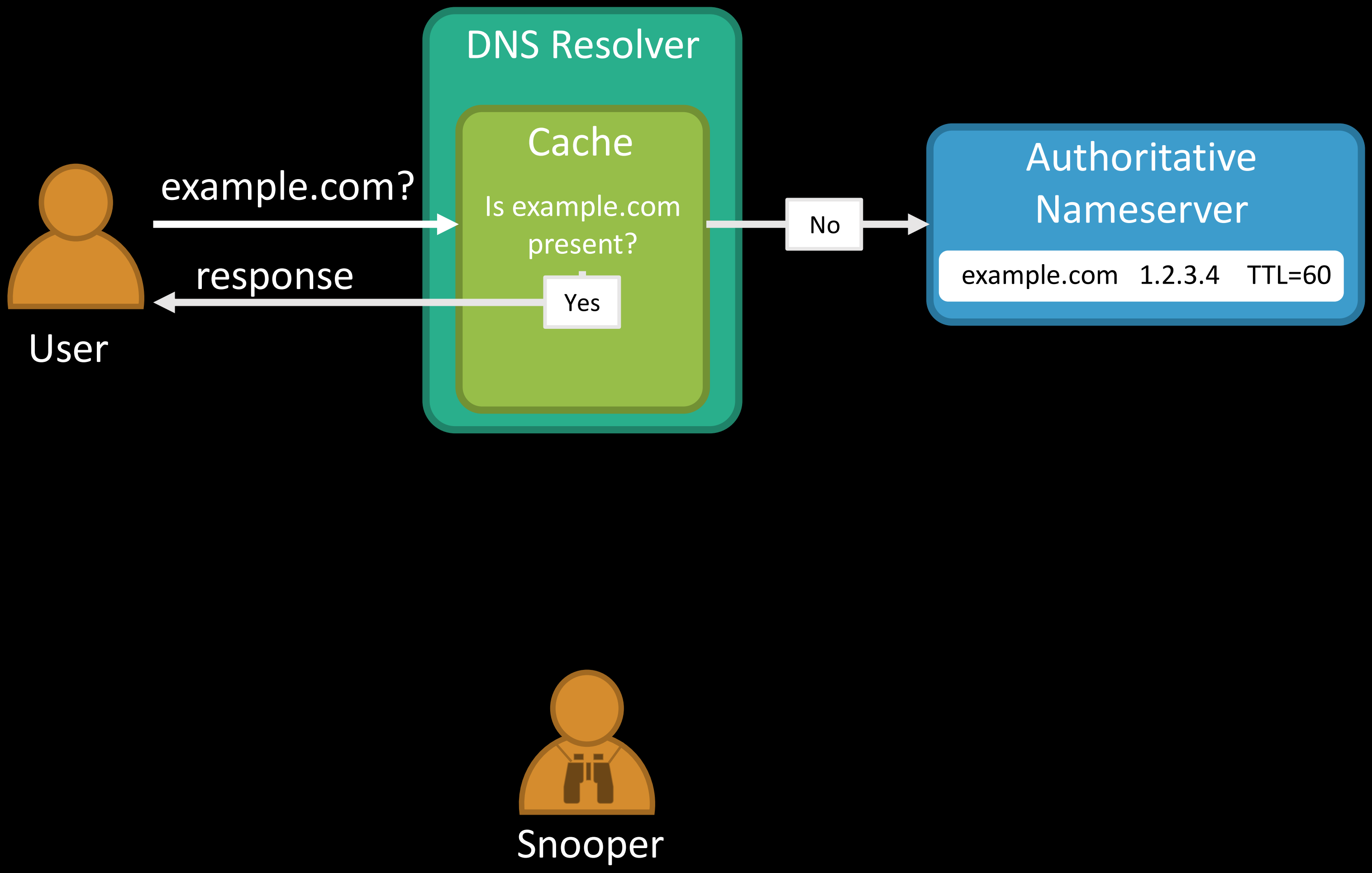
Organization of this talk

1. Background on cache snooping
2. Reverse engineering public resolver caching strategies
3. Our tool: Trufflehunter
4. Case studies

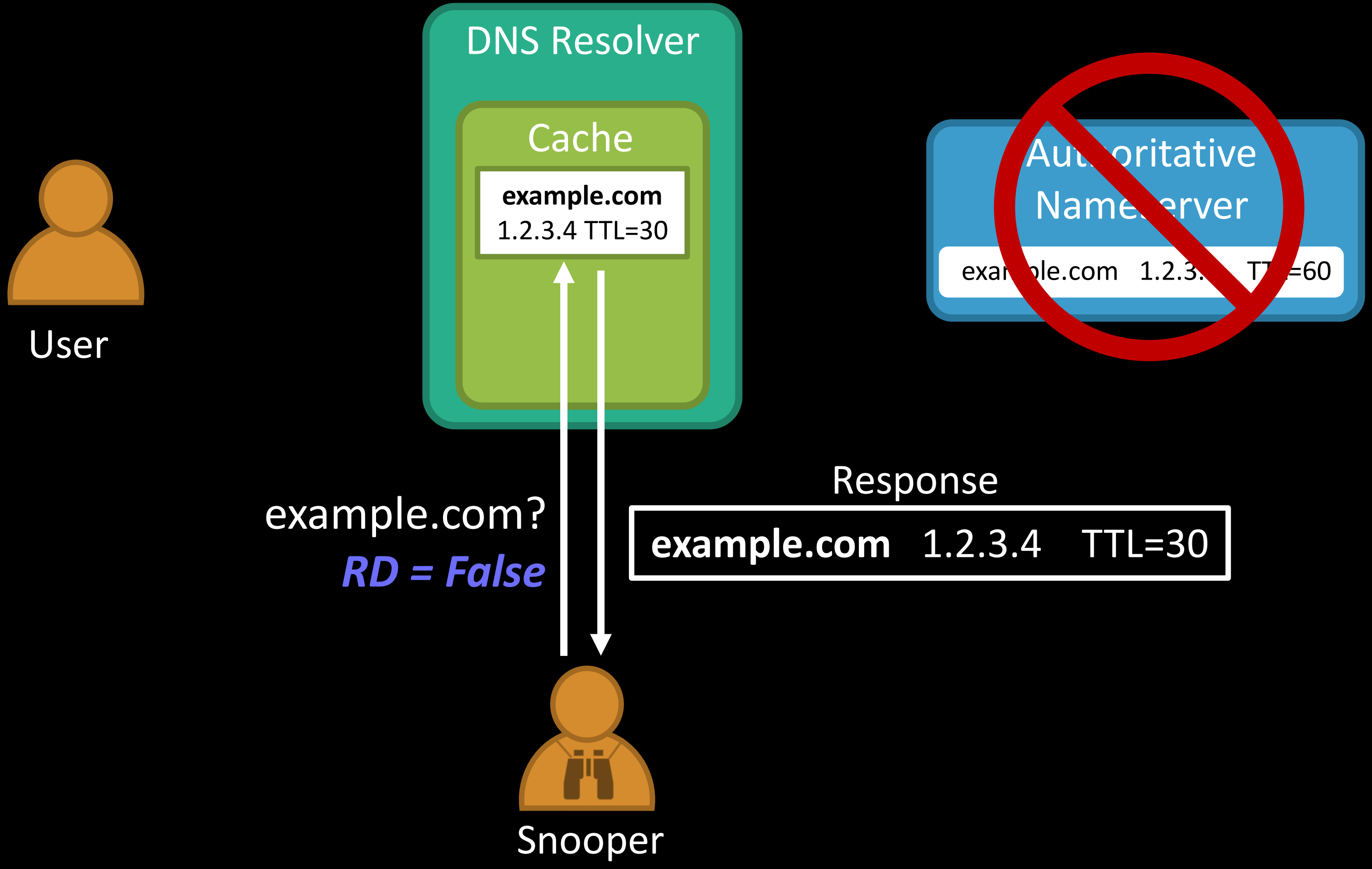
Organization of this talk

1. Background on cache snooping
2. Reverse engineering public resolver caching strategies
3. Our tool: Trufflehunter
4. Case studies

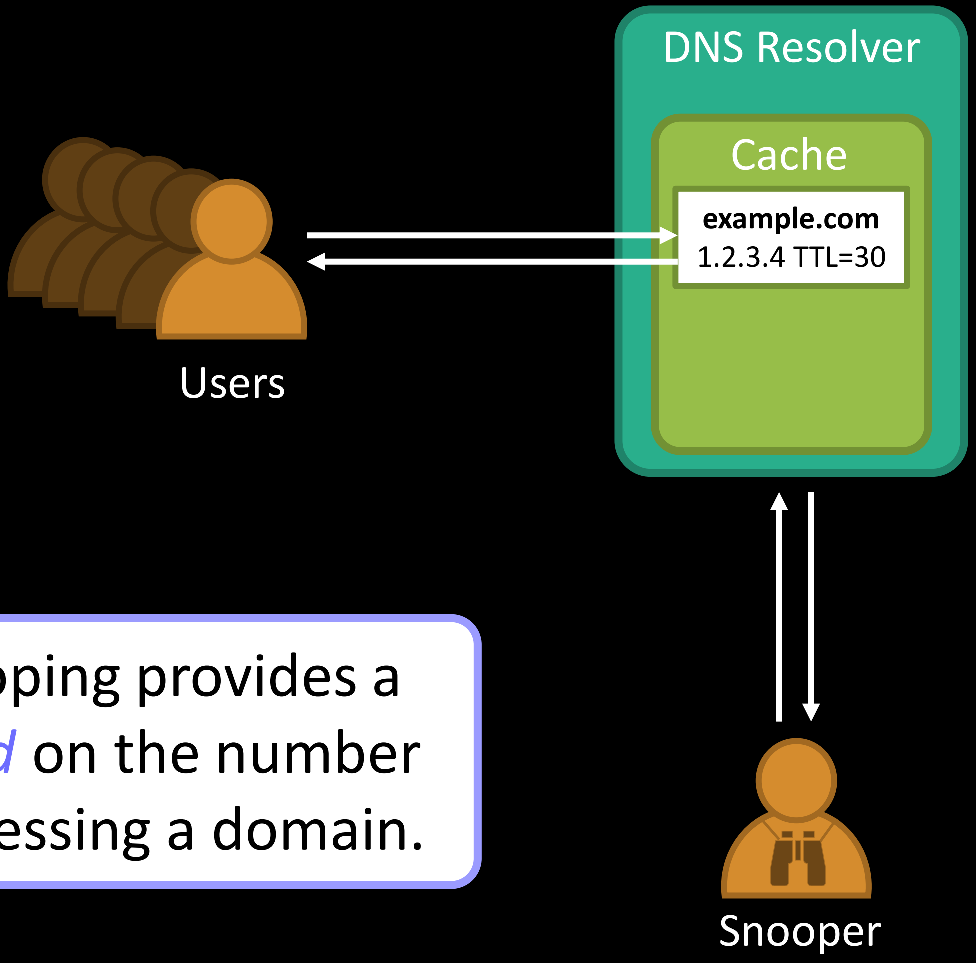
Background: How Cache Snooping Works



Background: How Cache Snooping Works



Background: How Cache Snooping Works

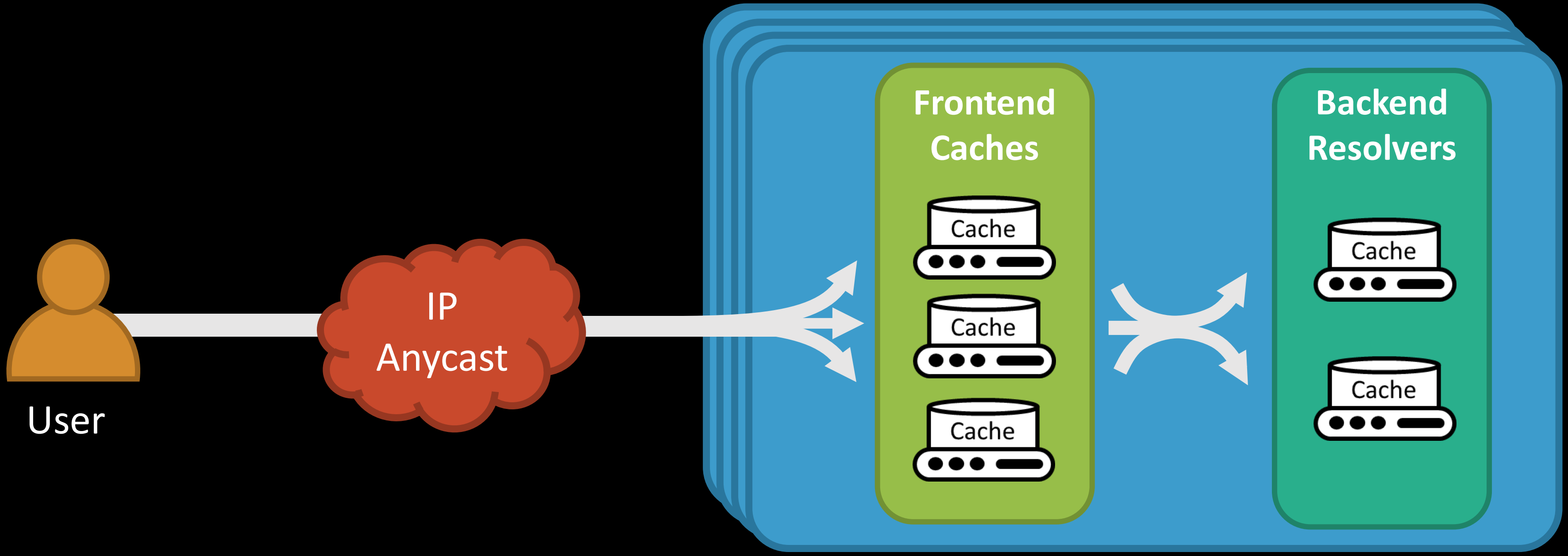


Cache snooping provides a *lower bound* on the number of users accessing a domain.

Organization of this talk

1. Background on cache snooping
2. Reverse engineering public resolver caching strategies
3. Our tool: Trufflehunter
4. Case studies

Simplified Public Resolver Cache Architecture



Public DNS Point of Presence (PoP)

Public resolvers use novel caching algorithms

Each resolver implements caching differently

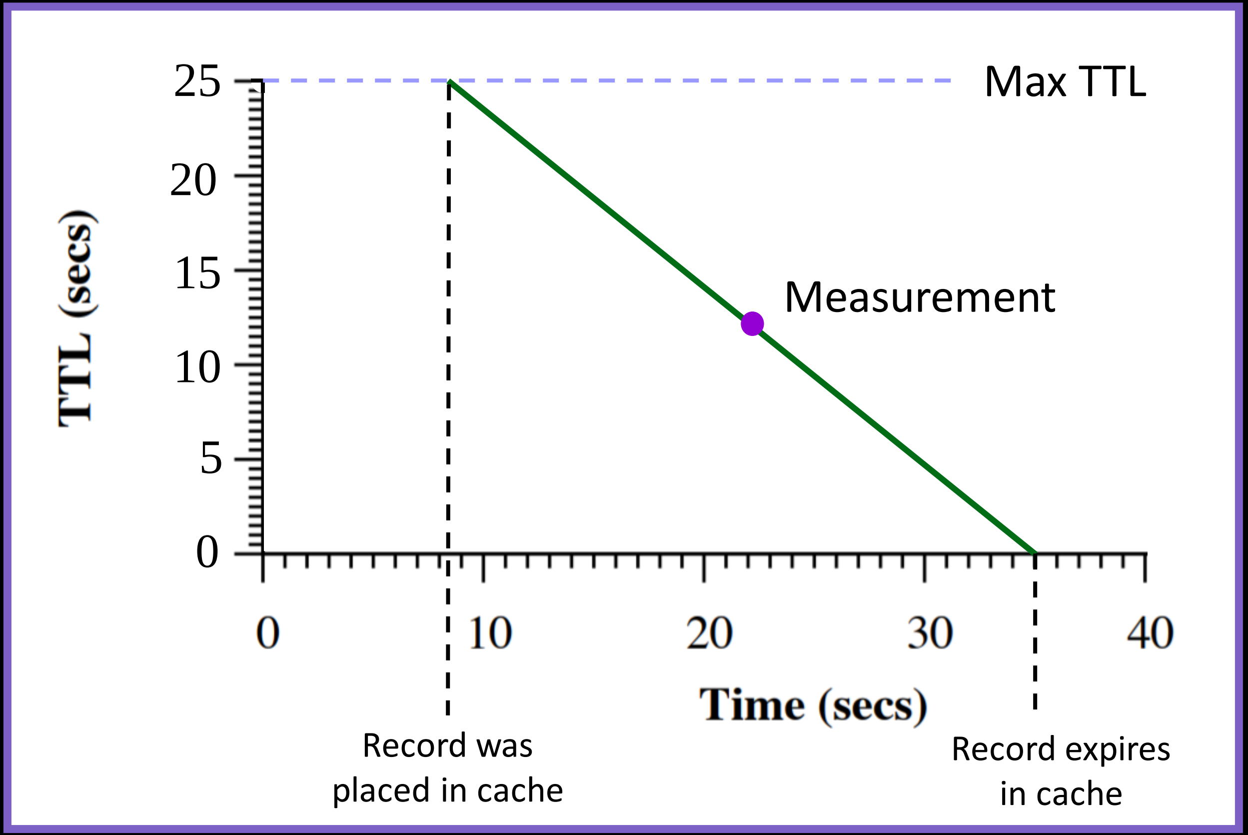
- Inconsistency causes potential problems
- Some algorithms cause TTL violations

To count filled caches, must identify which caches queries hit!

We reverse-engineered each caching strategy.

- Used only TTL, timestamp

How We Modeled Cache Architectures



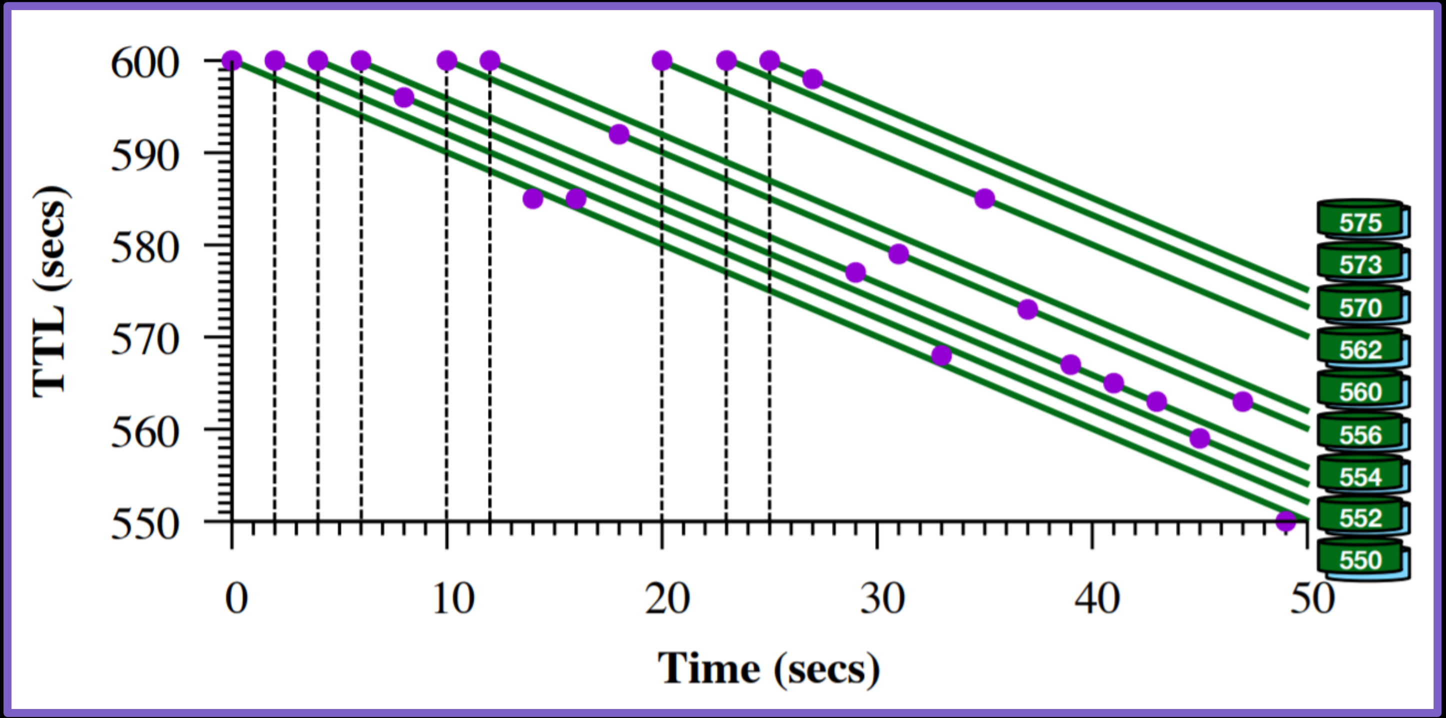
Experiment:

1. Repeatedly query resolver, fill caches
2. Observe how queries were cached: examine TTLs.

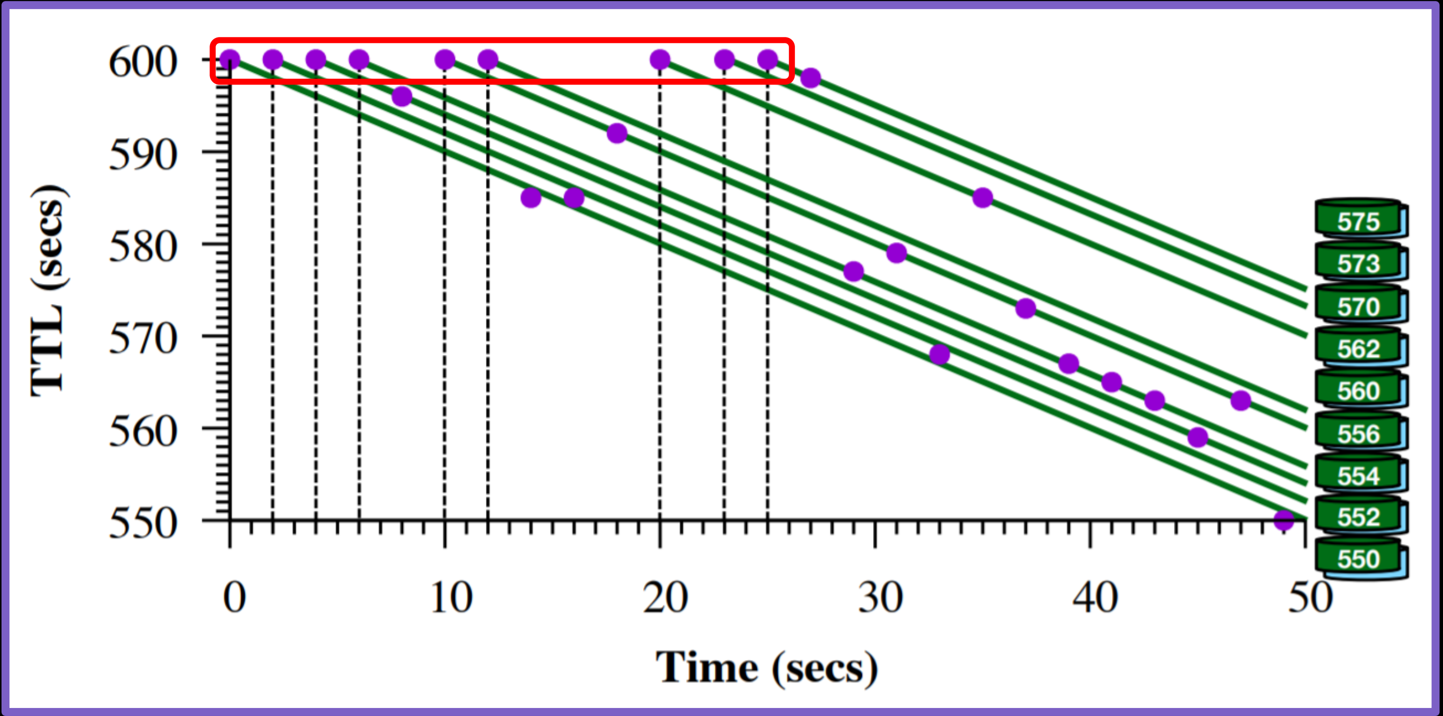
"TTL Line:" Model of how a TTL decreases in a cache.

- Rate: one second per second.

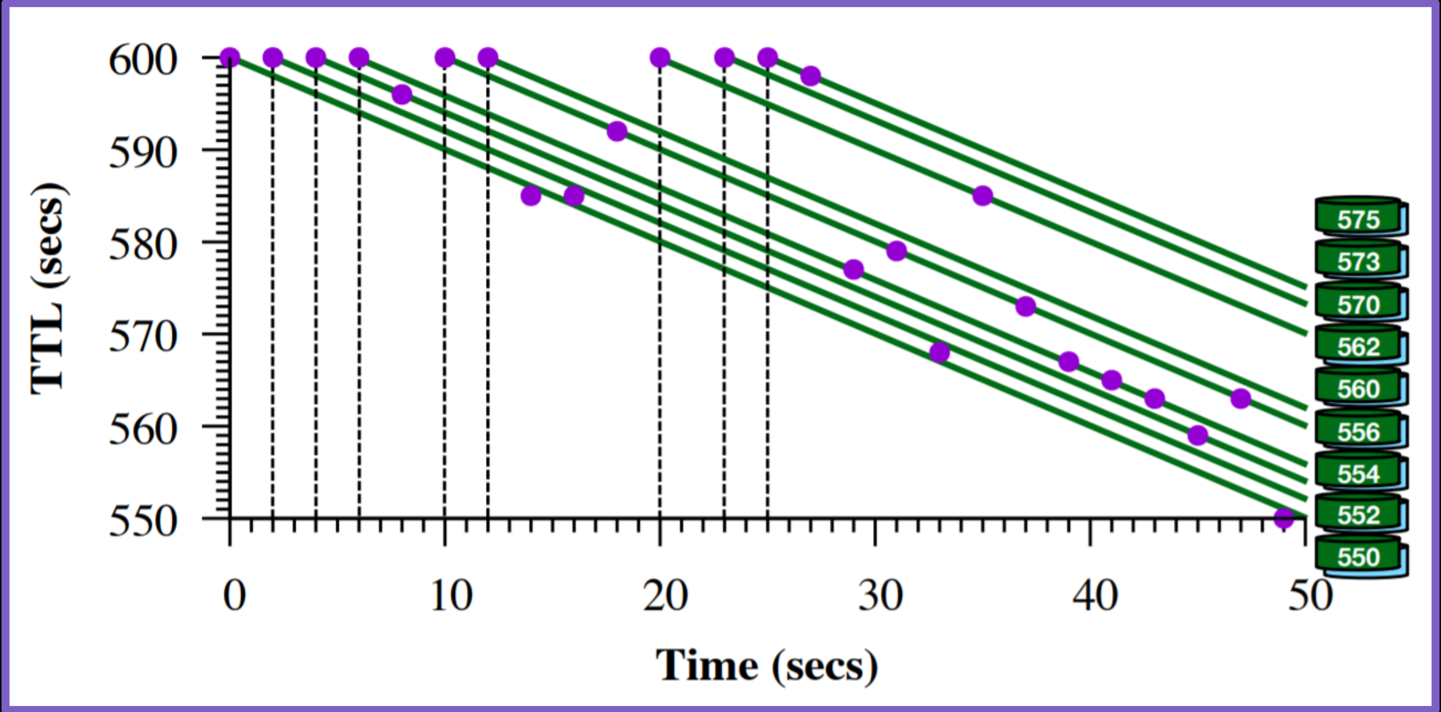
OpenDNS and Quad9

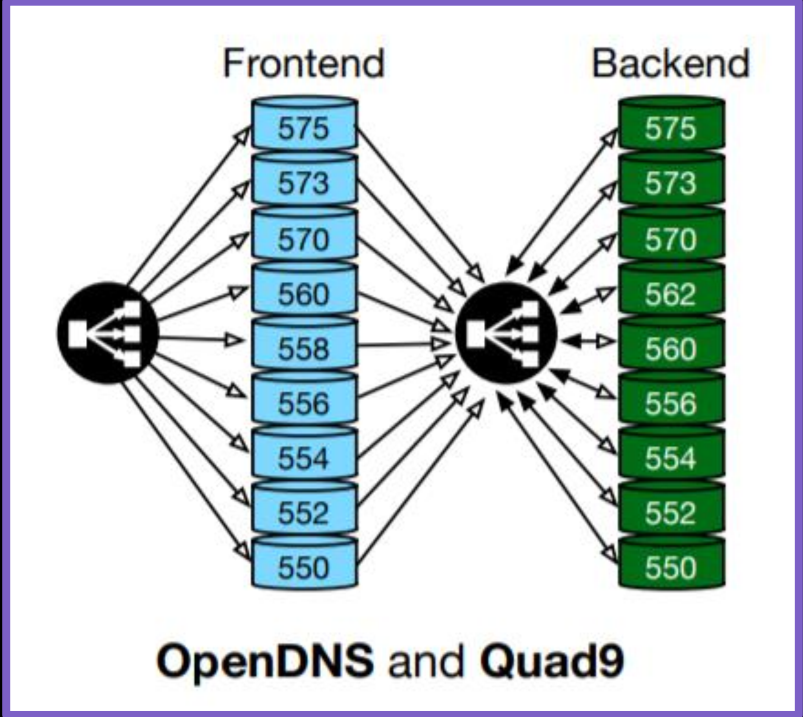


OpenDNS and Quad9

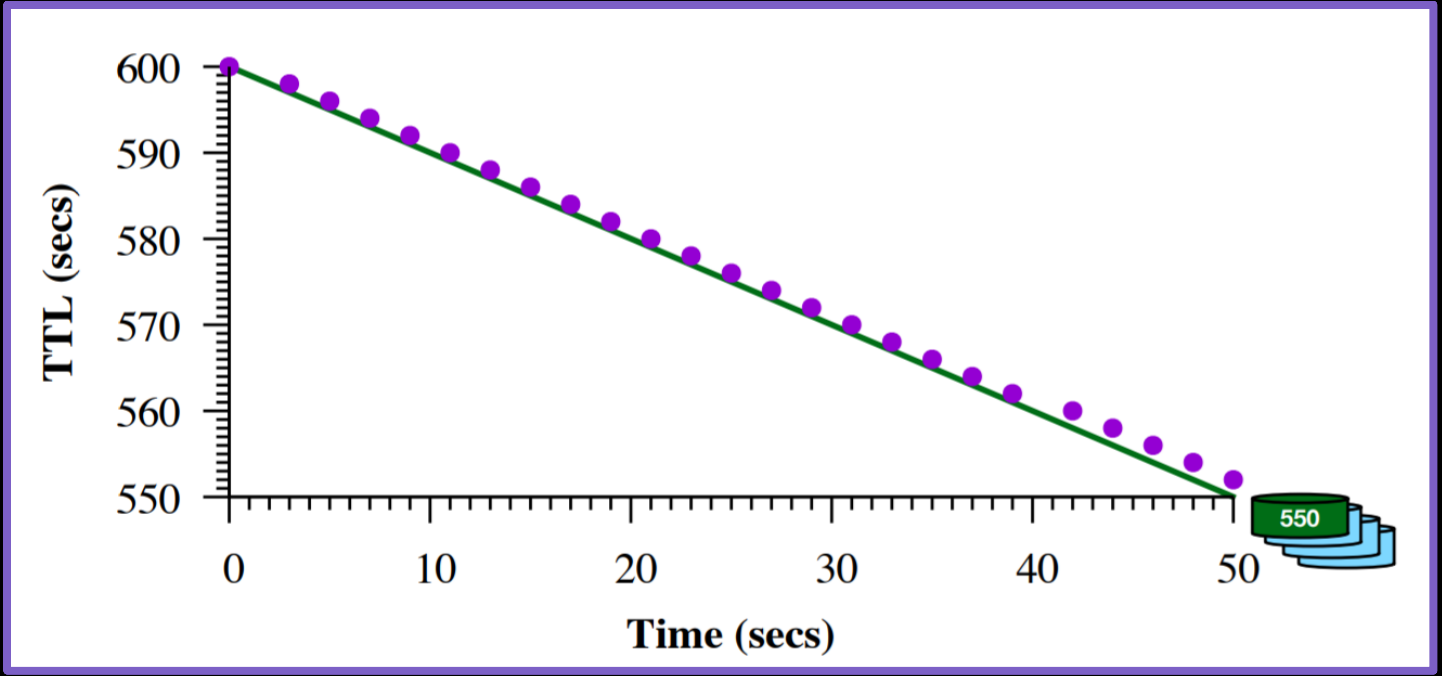


OpenDNS and Quad9

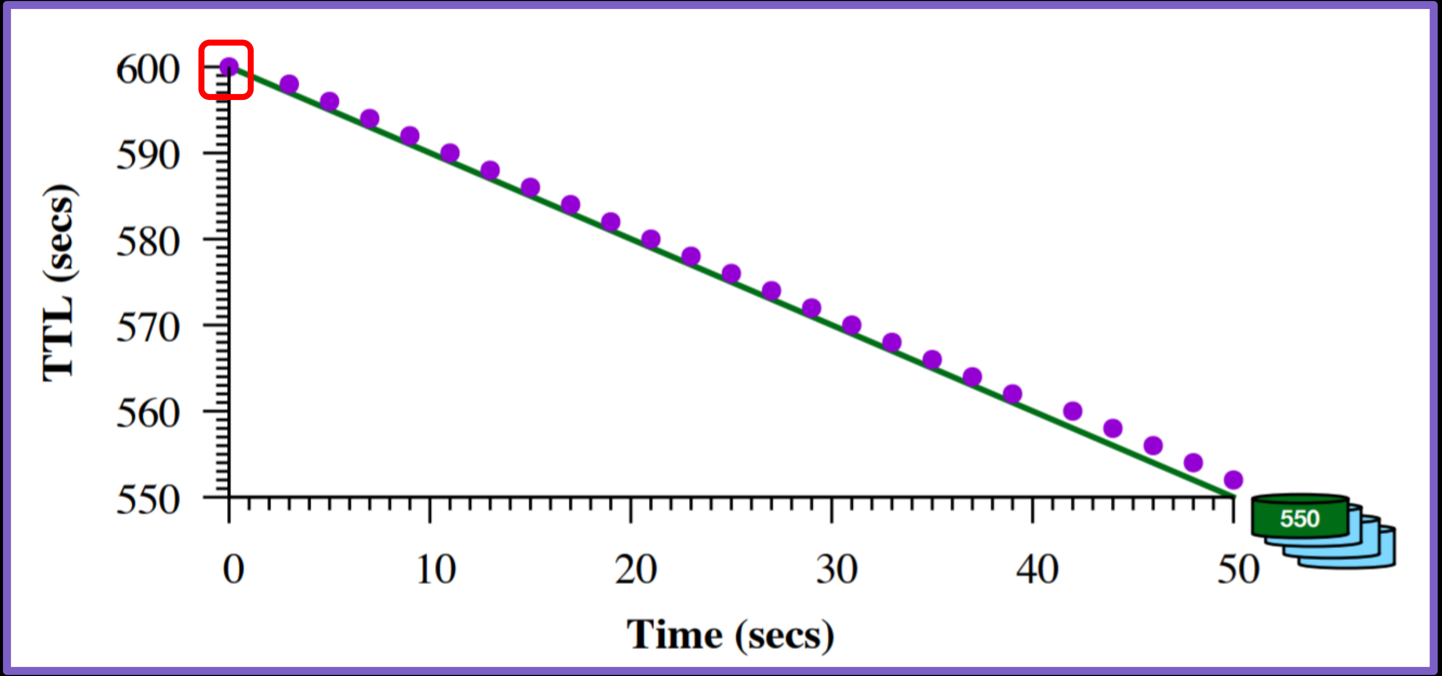




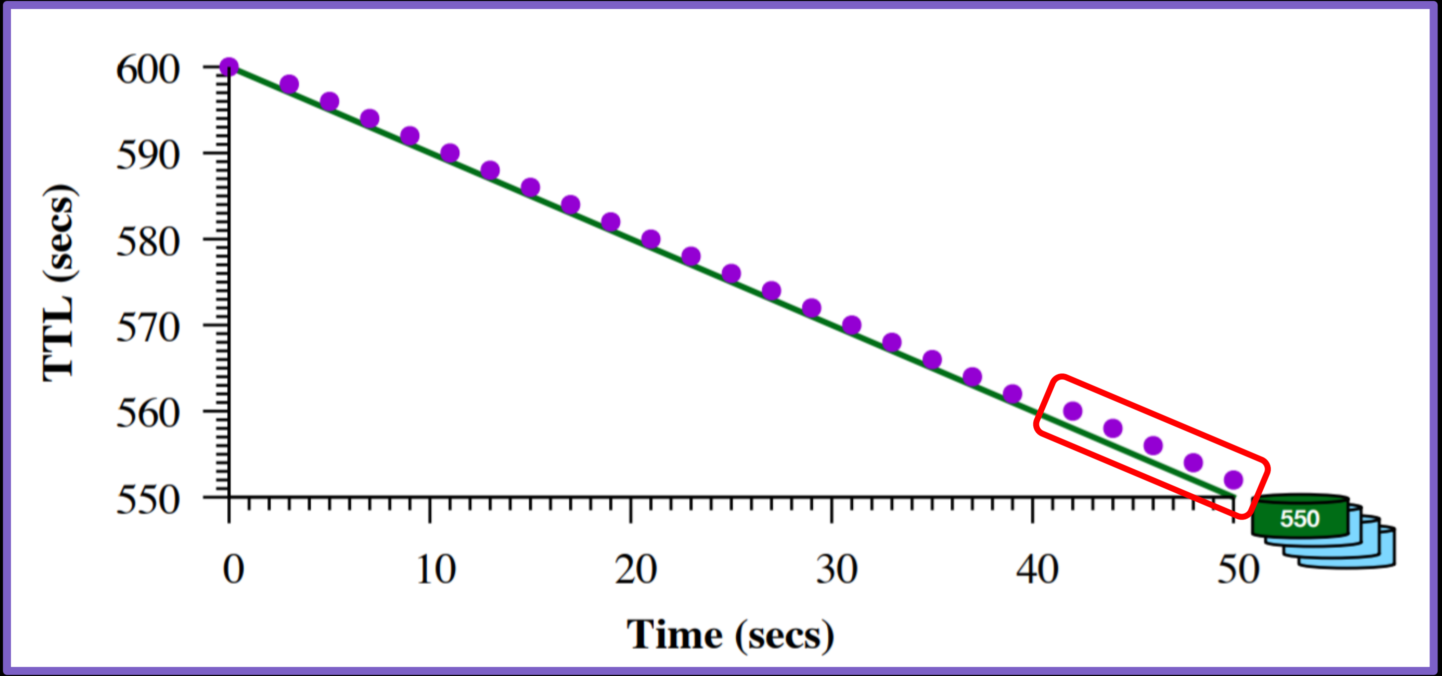
Cloudflare



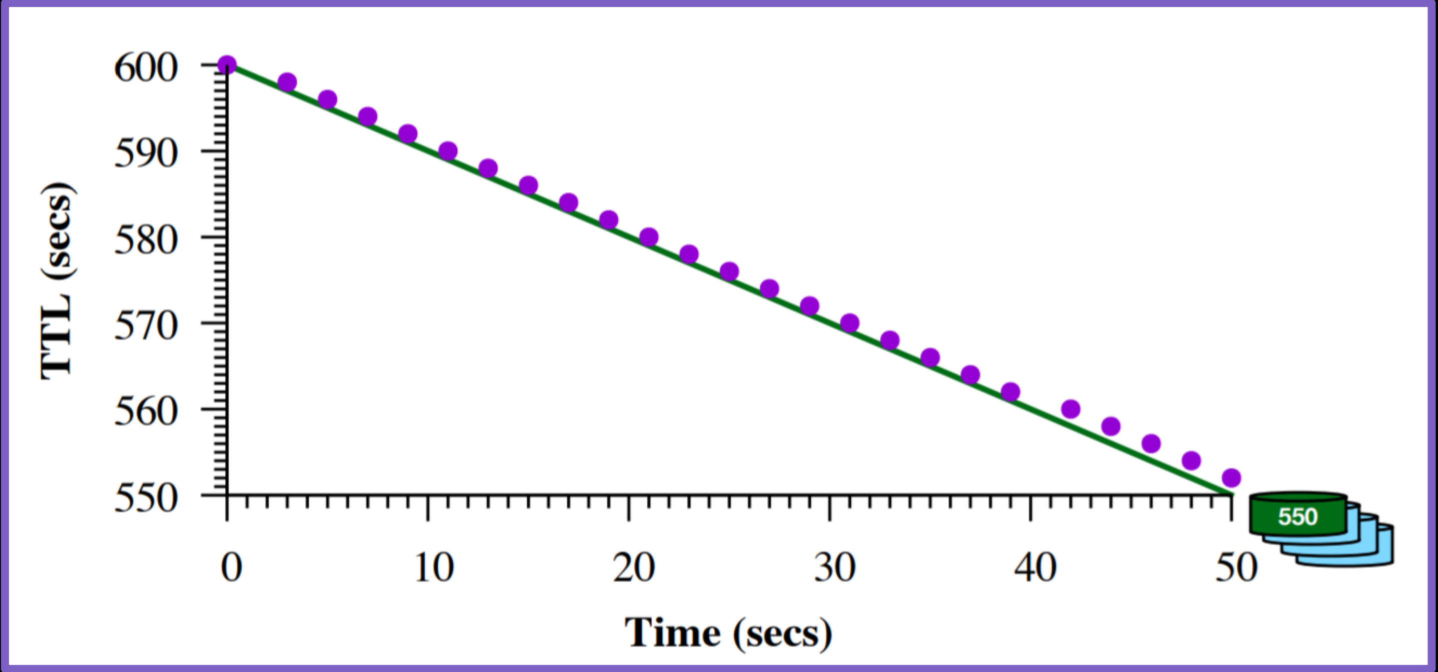
Cloudflare



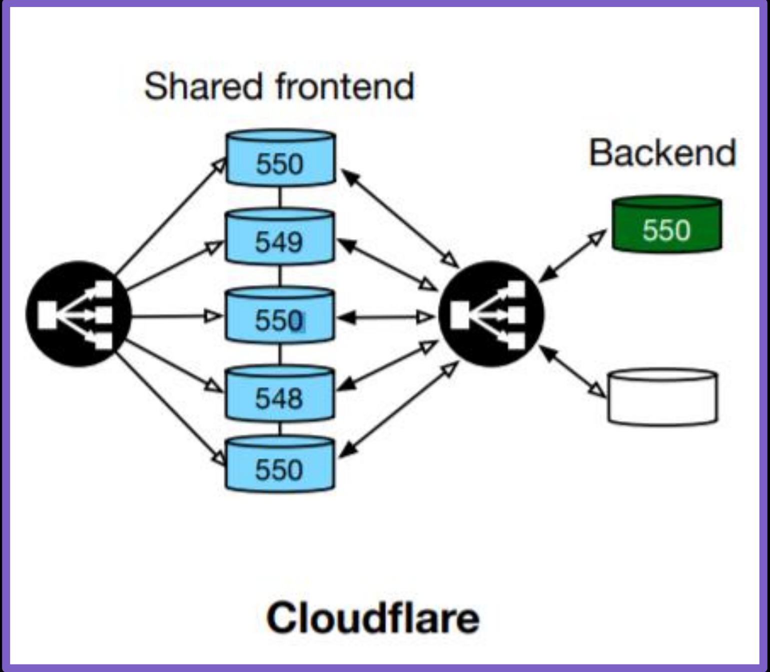
Cloudflare



Cloudflare

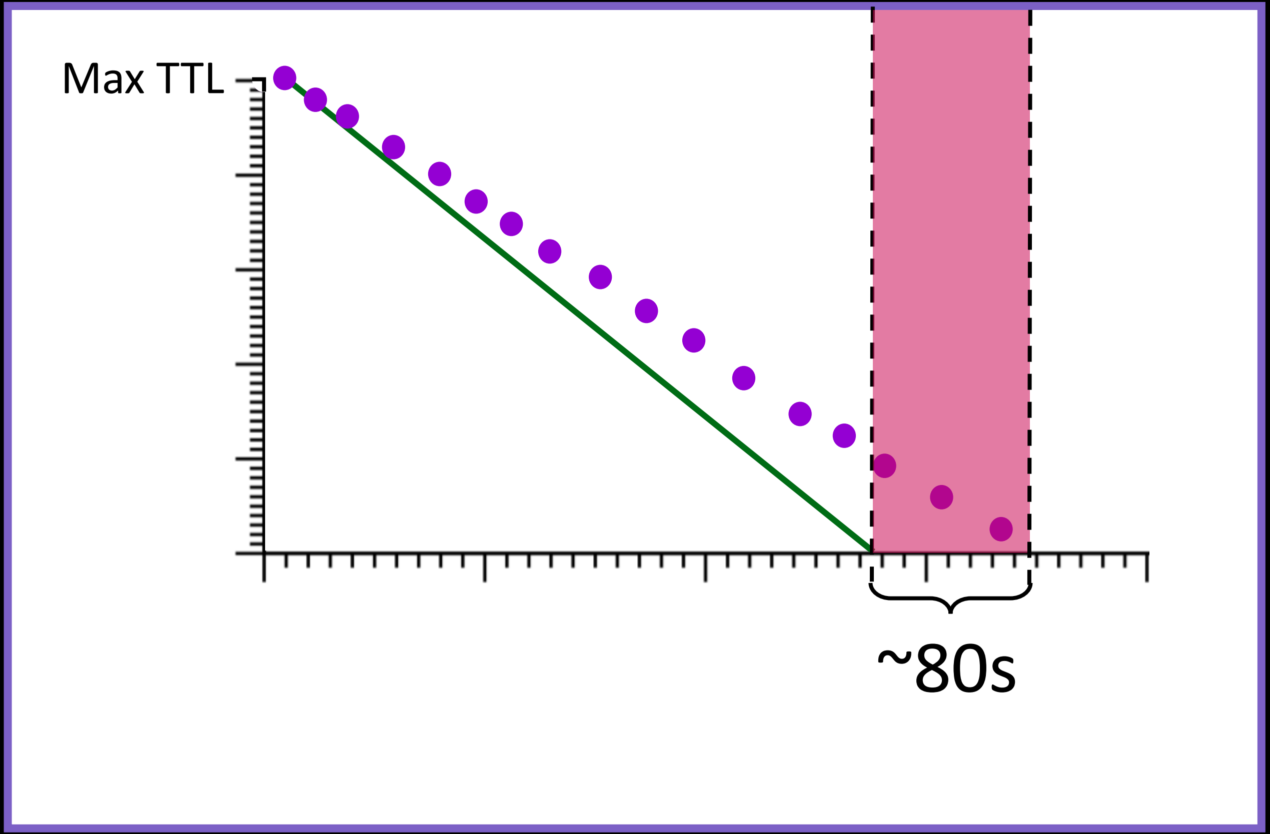


600
590
580
570
560
550
TTL (secs)
0
10
20
30
40
50
Time (secs)
550



Shared frontend
Backend
550
549
550
548
550
550
Cloudflare

Does Cloudflare's strategy lead to inaccurate TTLs?



Max drift we saw: ~80s (TTL=3hrs)

Drift scales with max TTL, so problems likely to be minimal?

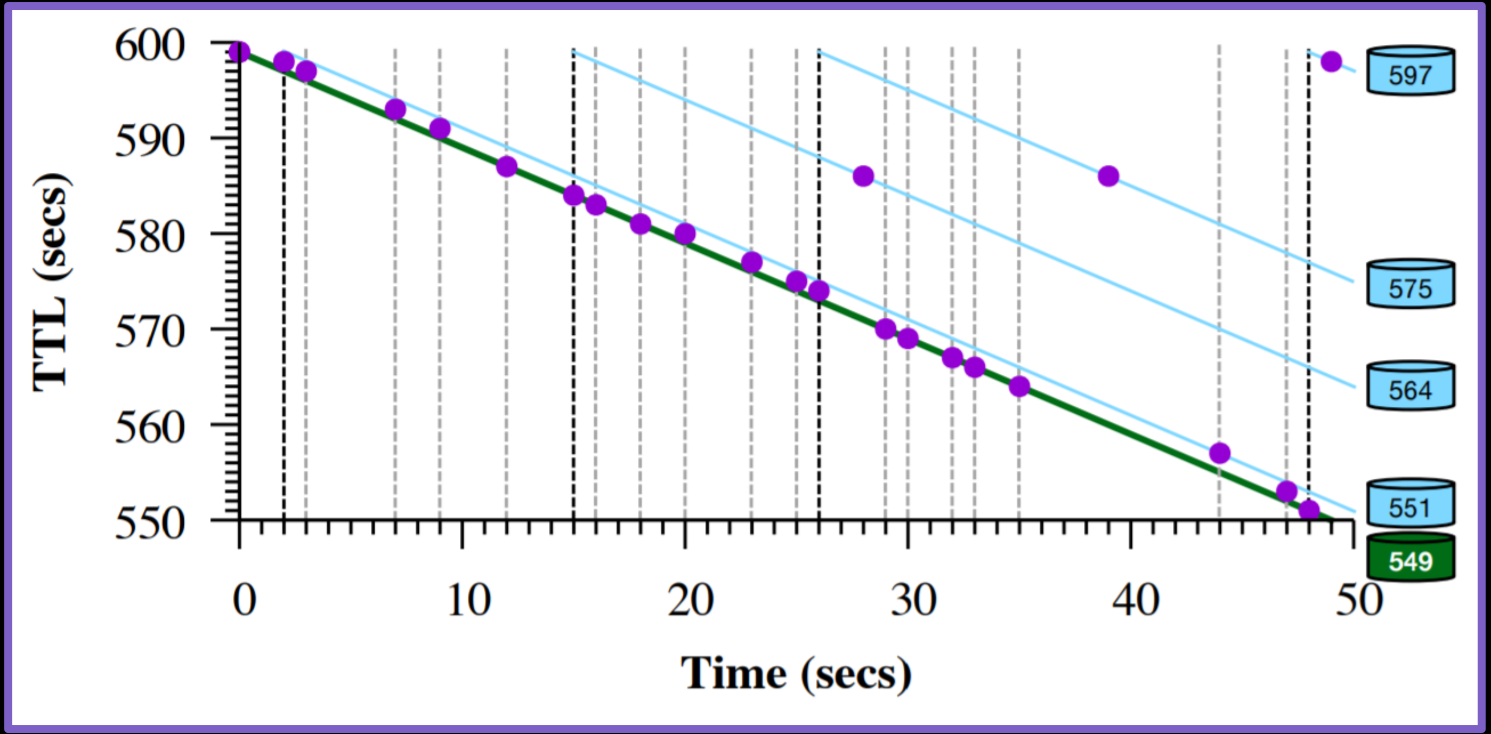
And then there's Google DNS…

Prior work observed Google "mystery caches"

- Schomp et al. found initial TTL correct, subsequent TTLs often incorrect
- Rohprimardho et al.: "Ghost caches"

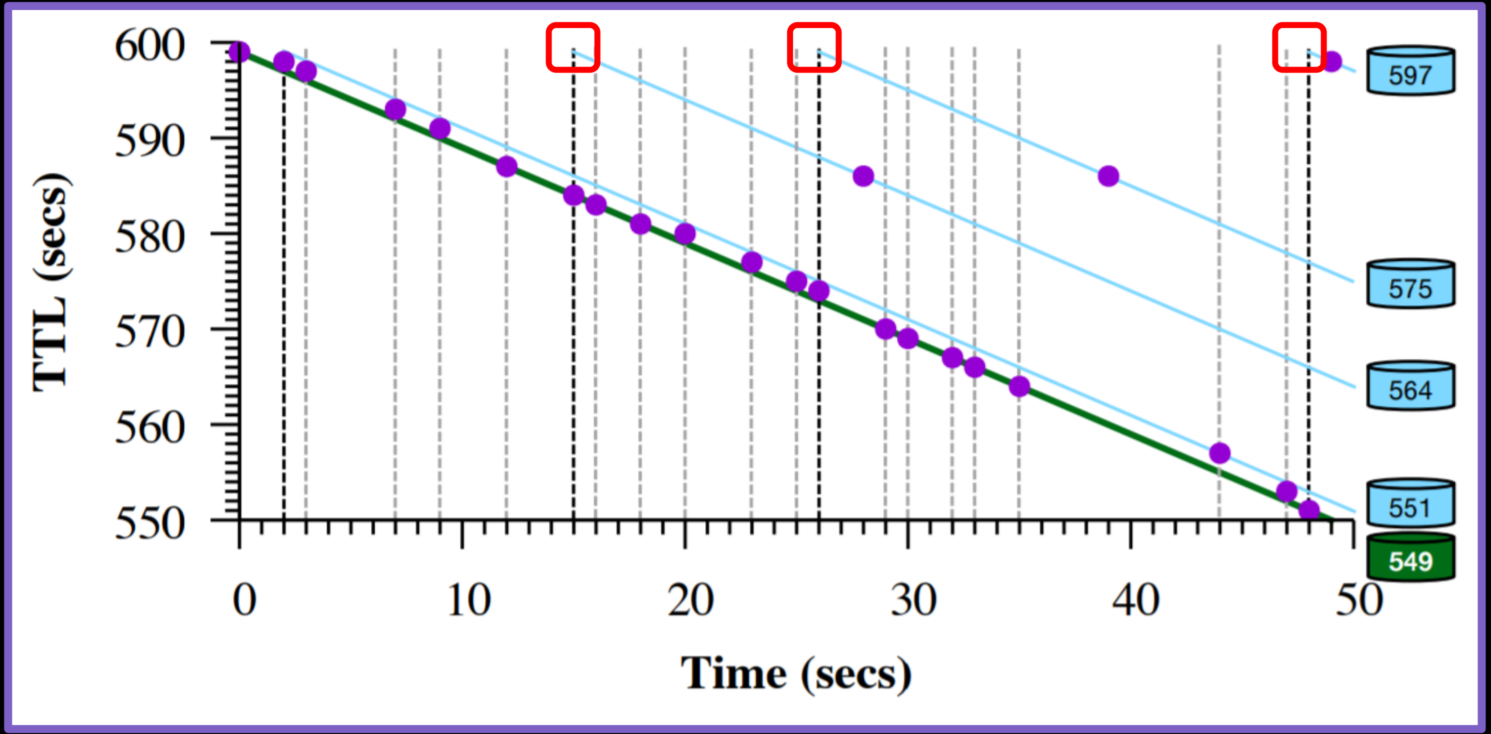
Why are caches getting filled without being queried?

Google DNS



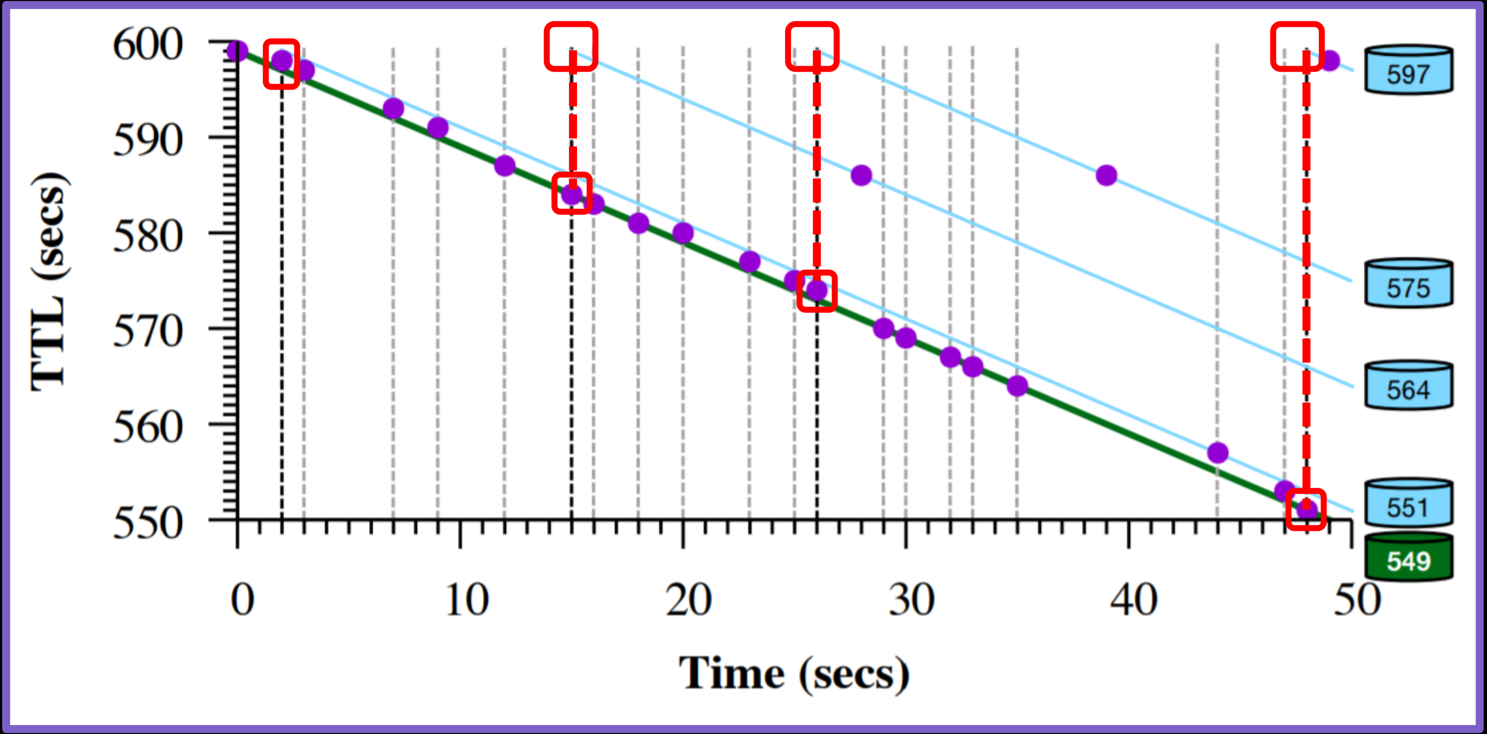
600
590
580
570
560
550
TTL (secs)
0
10
20
30
40
50
Time (secs)
597
575
564
551
549

Google DNS



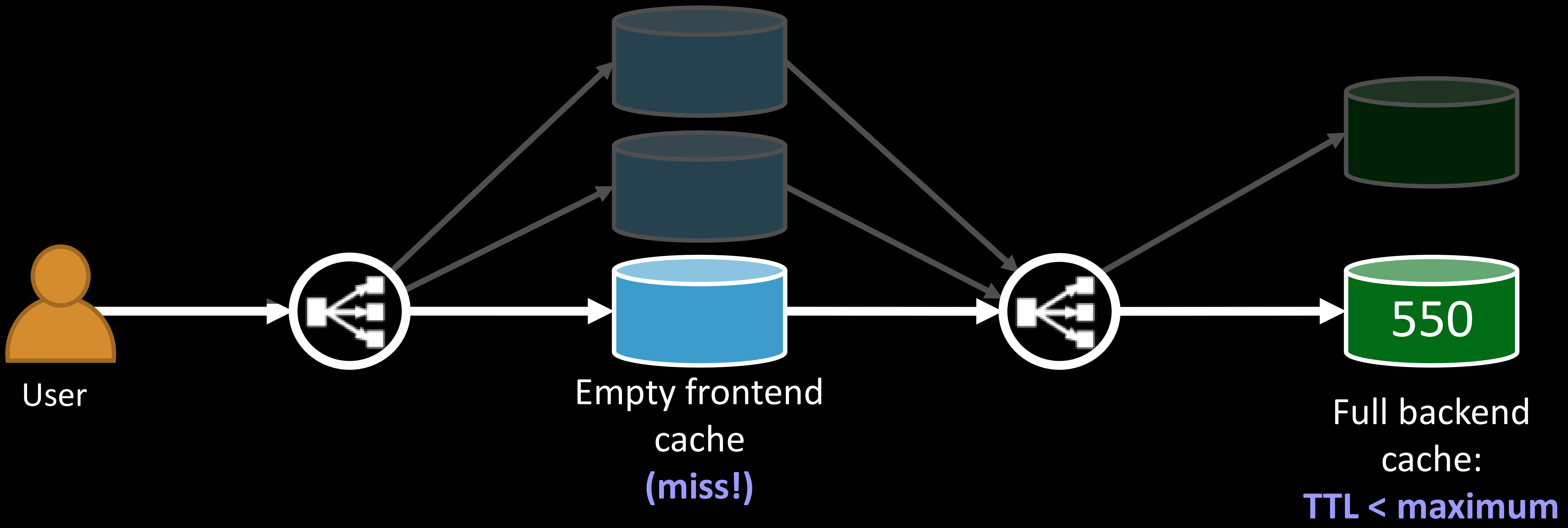
No measurements!

Google DNS

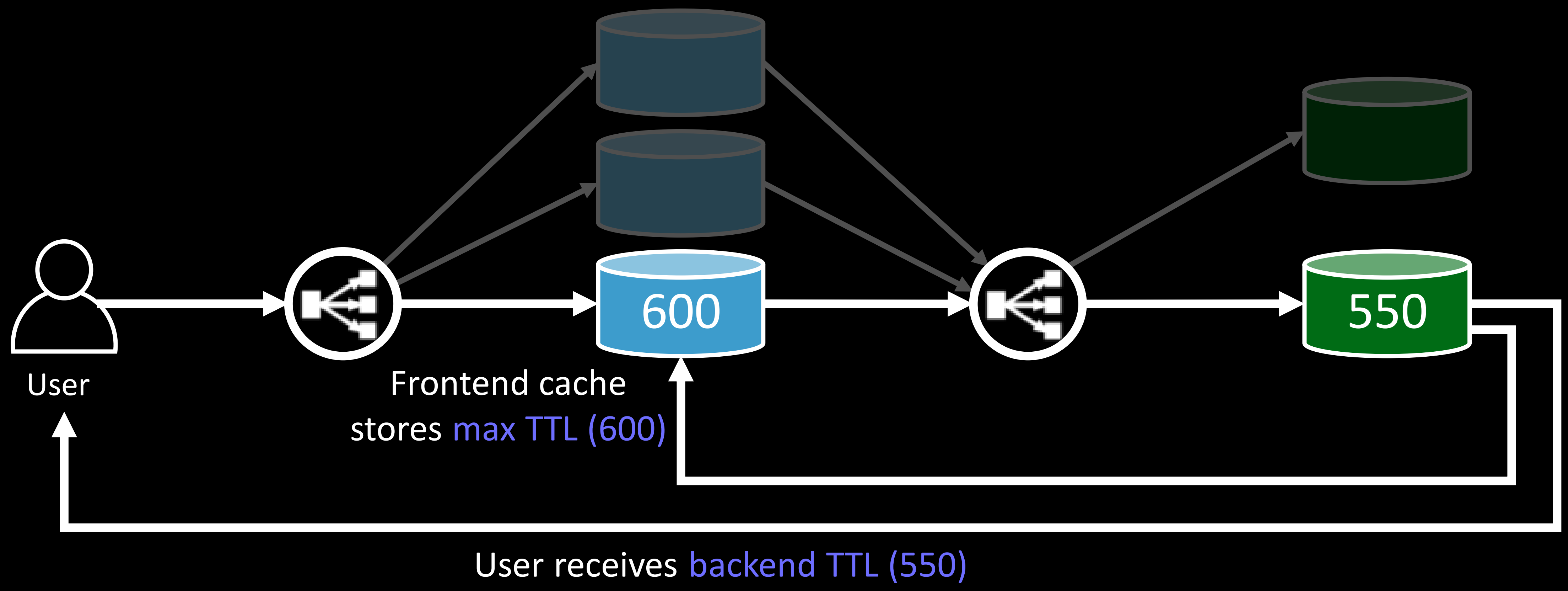


600
590
580
570
560
550
TTL (secs)
0
10
20
30
40
50
Time (secs)
597
575
564
551
549

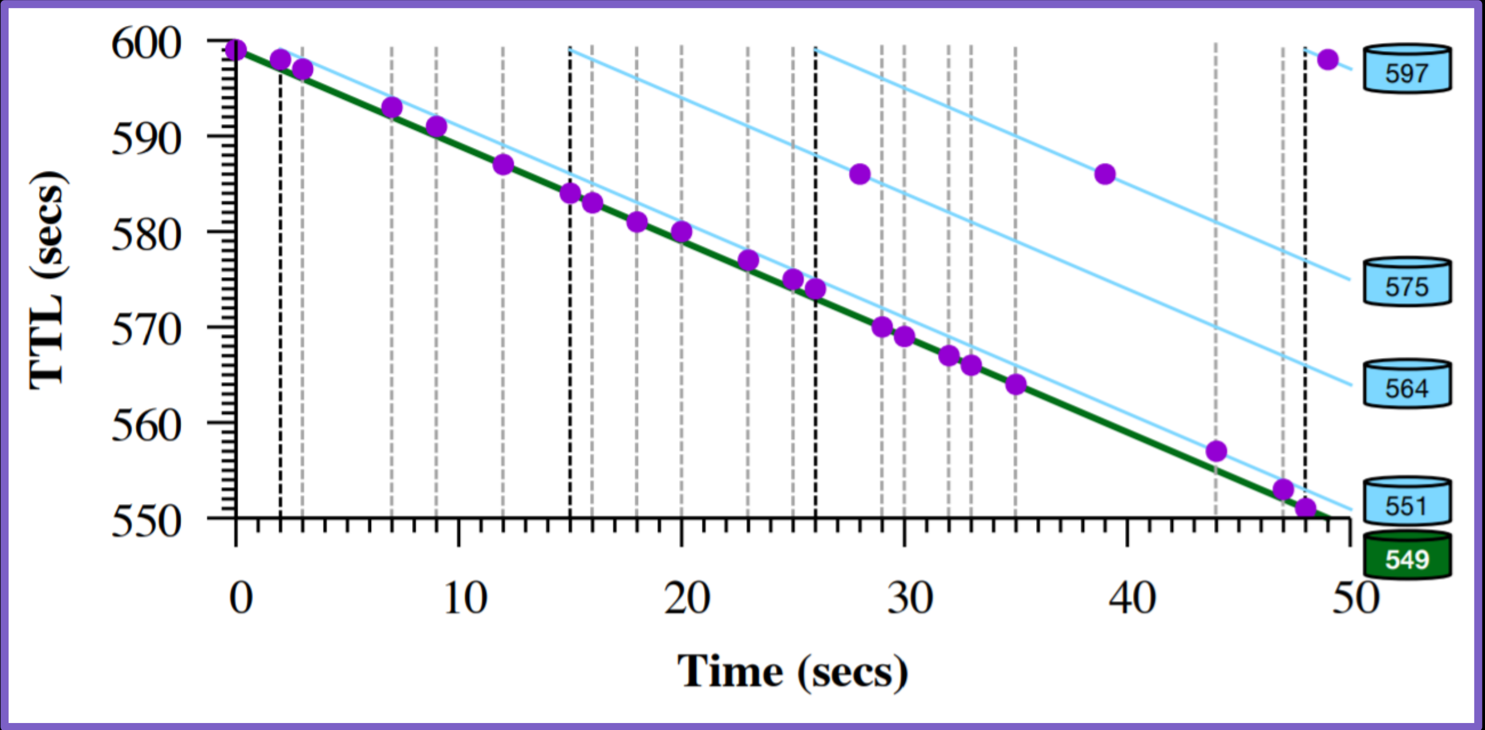
Google DNS: Dynamic Caching

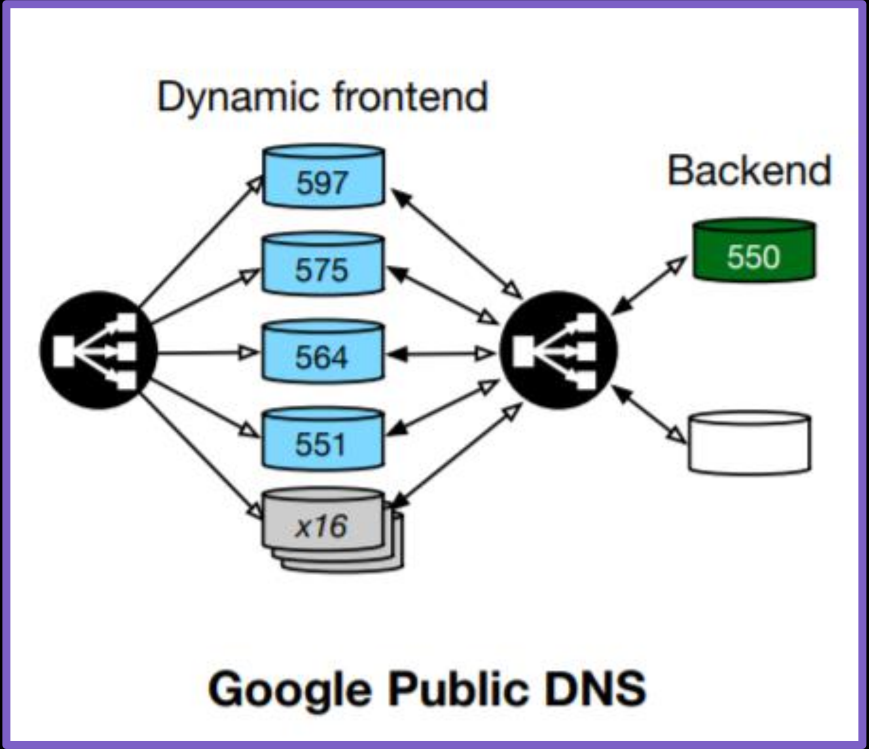


Google DNS: Dynamic Caching

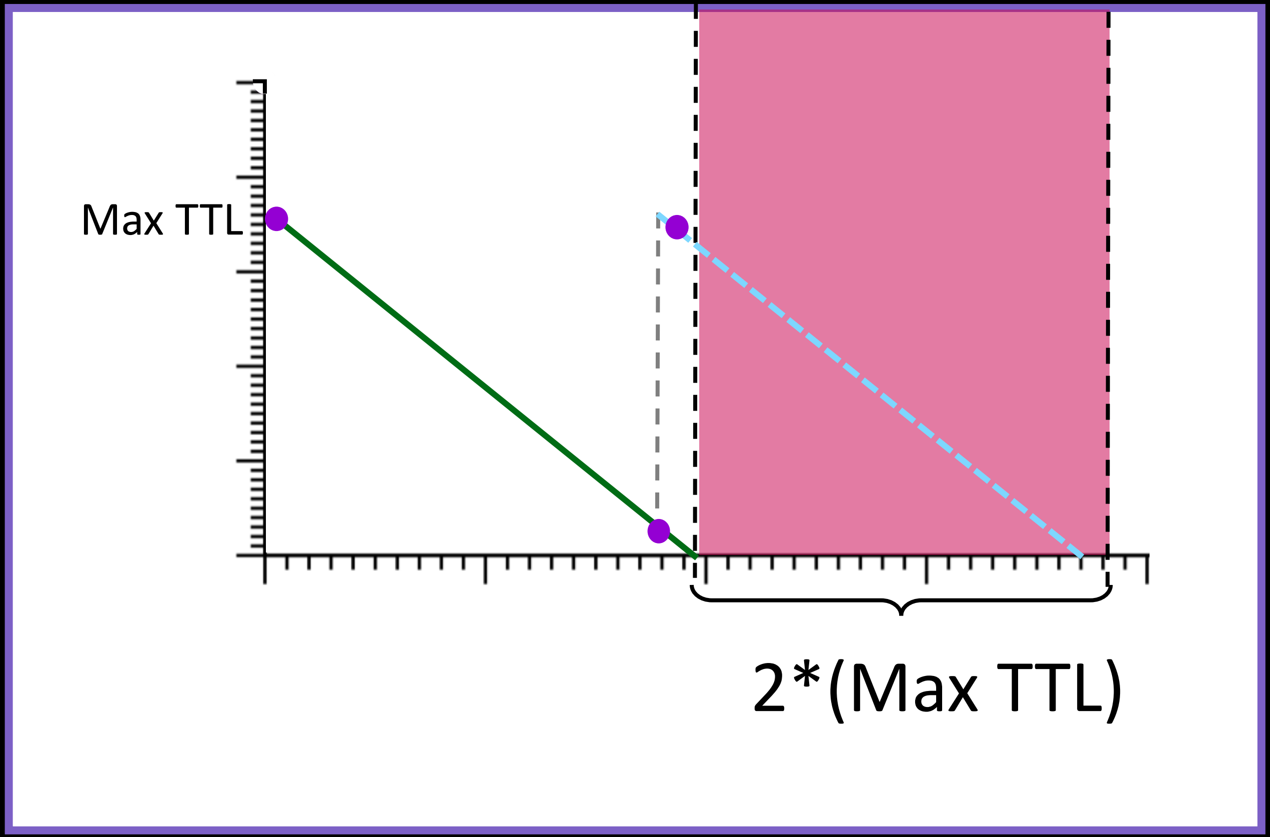


Google DNS: Dynamic Caching





Does Google's strategy lead to inaccurate TTLs?

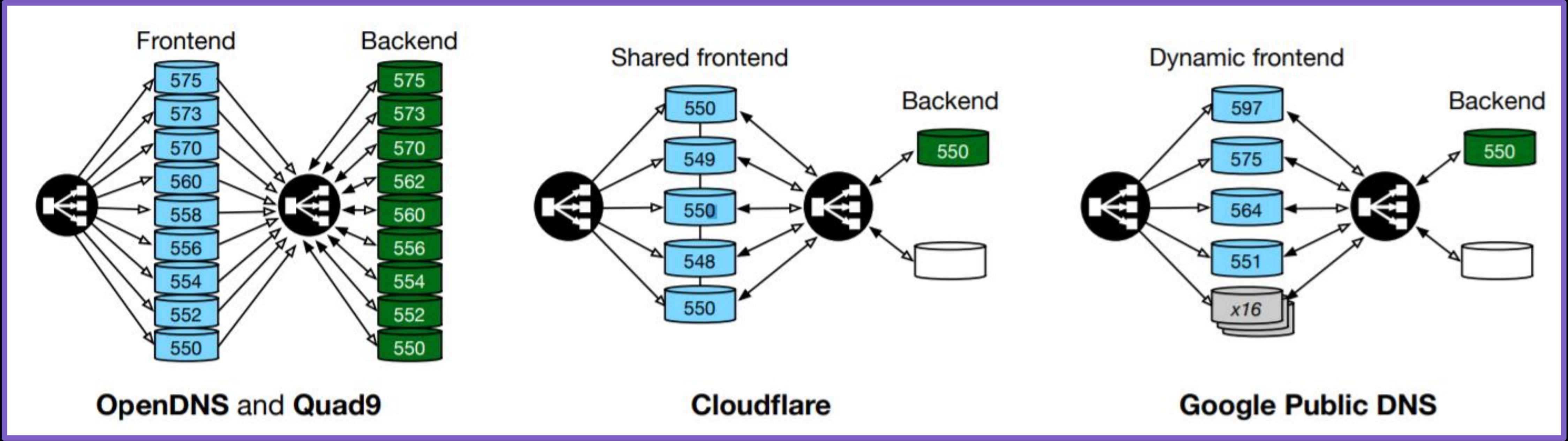


"Extra" front-end caches cleared when backend TTL expires.

Maximum drift: 2 * (max TTL).

Question: Why store max TTL in frontend caches?

Summary of caching strategies



Organization of this talk

1. Background on cache snooping
2. Reverse engineering public resolver caching strategies
3. Our tool: Trufflehunter
4. Case studies

Trufflehunter

Distributed measurement tool

- Deployed on CAIDA’s Ark project

Sends DNS queries across the U.S.

Interprets the responses, estimate counts of users

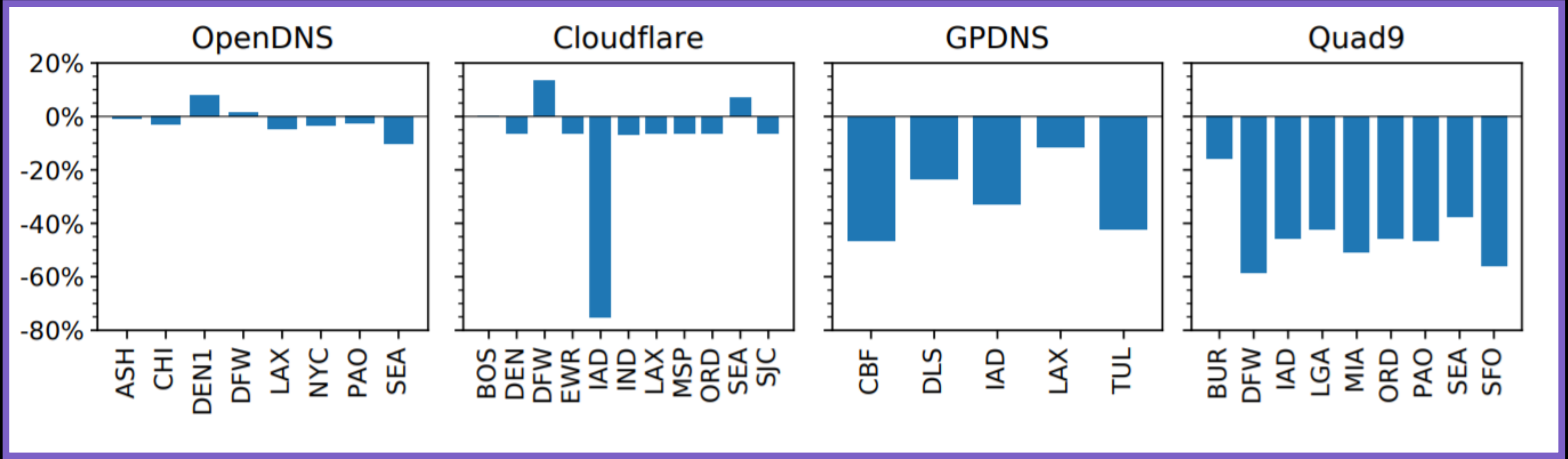
Three months of data: March 6 – May 29 2020

How accurate is Trufflehunter at estimating filled caches?

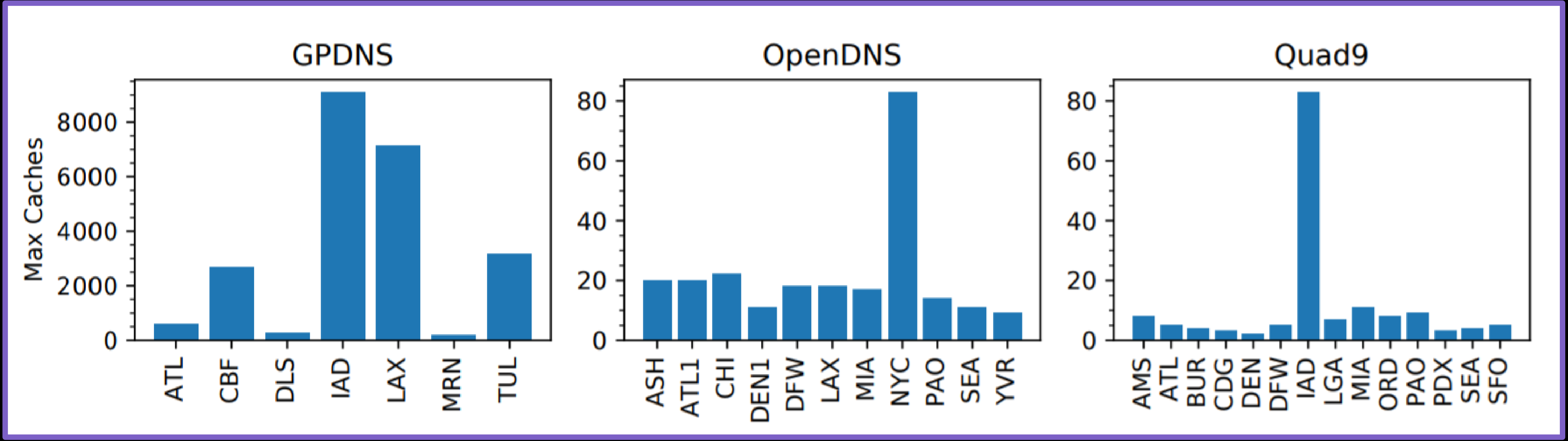
Experiment:

- Place domain we control into caches
- Observe it with Trufflehunter
- Requests to our authoritative nameserver = true number of filled caches

Error in number of filled caches:



Bounds on Observable Users



(Cloudflare has only one visible cache per PoP.)

Organization of this talk

1. Background on cache snooping
2. Reverse engineering public resolver caching strategies
3. Our tool: Trufflehunter
4. Case studies

Case Studies

Three case studies:

- Stalkerware
- Contract Cheating
- Typo Squatting

Previously, all were hard to measure – little data available about prevalence.

Case Study #1: Stalkerware

Stalkerware: emerging spyware threat.

- Often records location, keyboard, ambient sound/video
- Can hide its presence

We download and profile 24 apps

- 6 dual use: Usually marketed for parental control, employee surveillance.
- 16 overt: “Undetectable”
- Record network traffic: extract DNS requests

Why is stalkerware hard to study by other means?

Prior work: clinical settings

- Individual one-on-one sessions: low sample size
- Few to zero overt apps found in the wild

Targets have often already reset devices

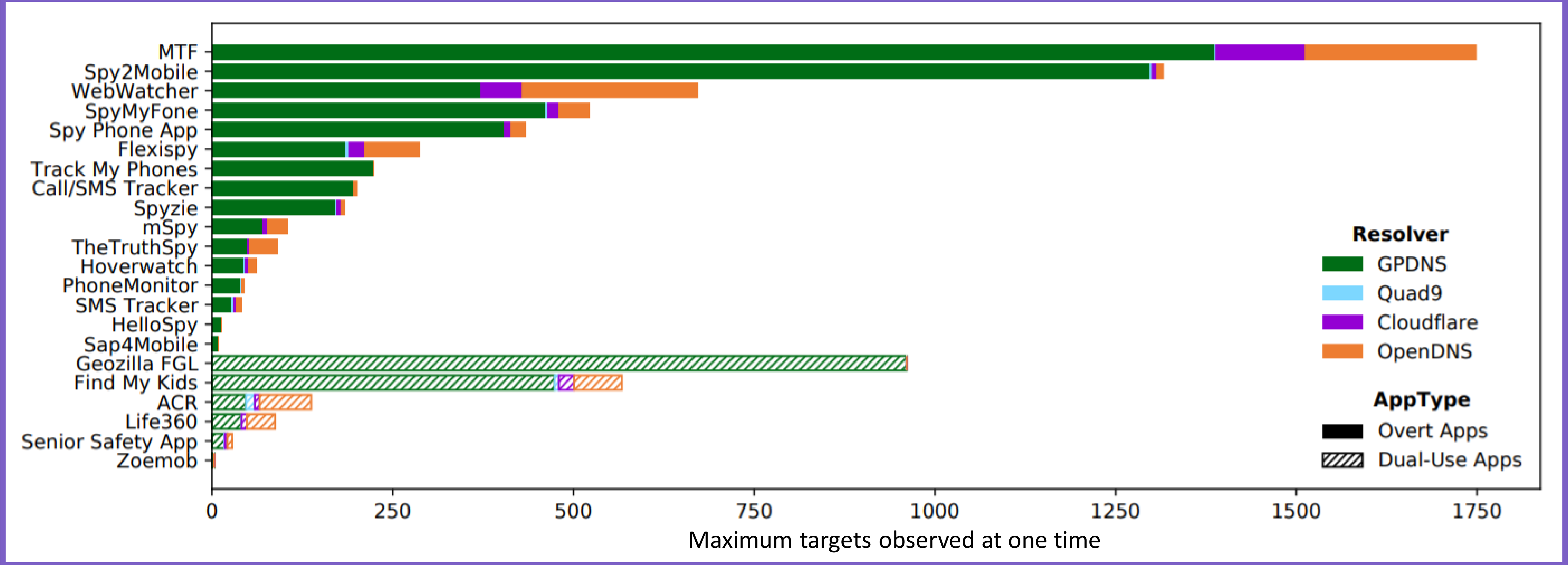
Clinics often lack technical expertise

From Counting Caches to Counting Devices

Stalkerware often makes DNS requests automatically, at regular intervals.

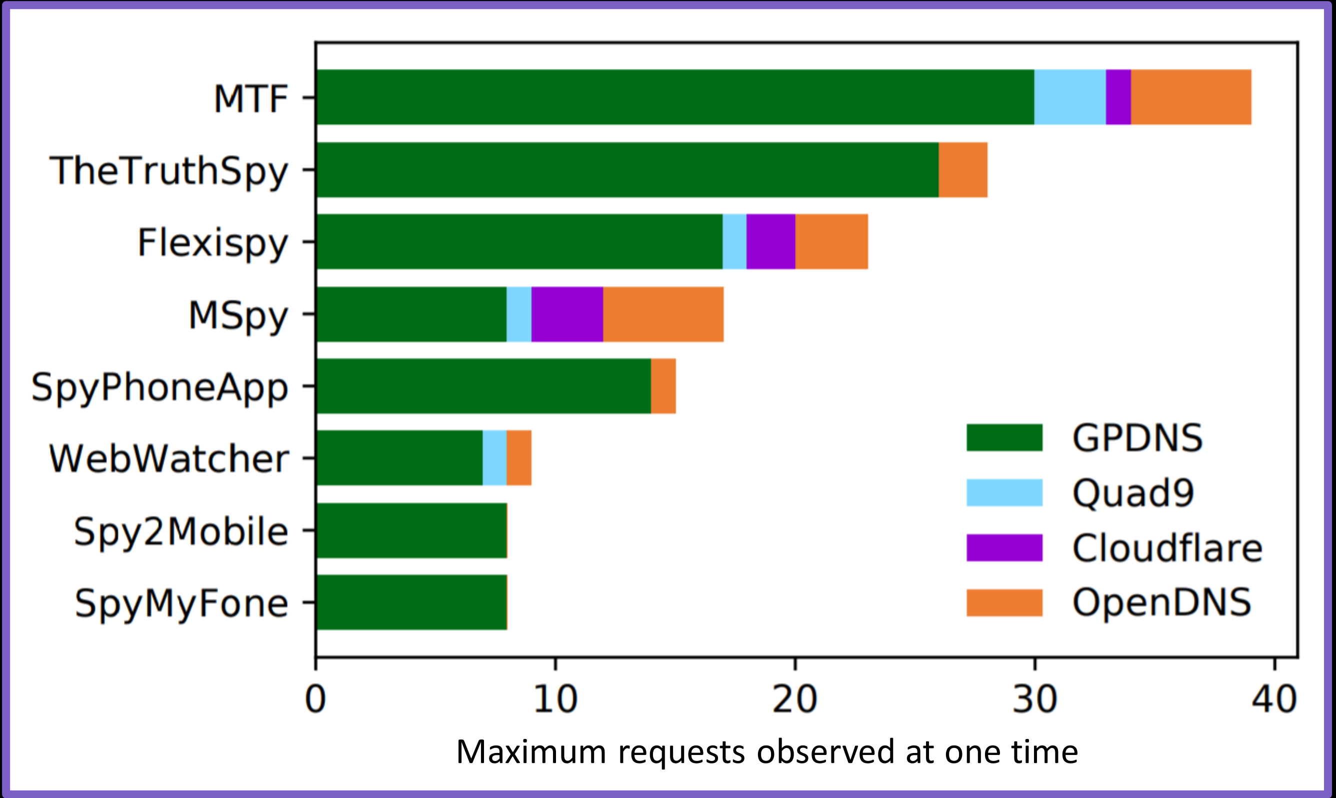
$$\text{Devices with stalkerware installed} = \frac{\text{Filled Caches}}{\text{App Request Rate}}$$

Observed Stalkerware Targets



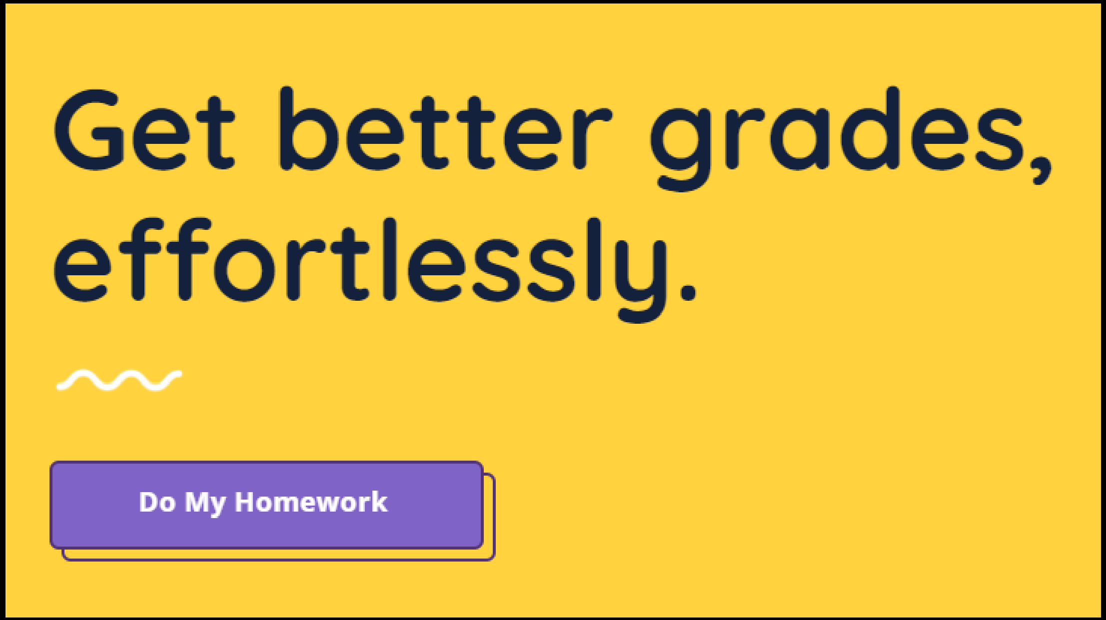
At least 5,700 people are targeted by overt stalkerware in the U.S. today.

Observed Stalkerware Dashboard Visits



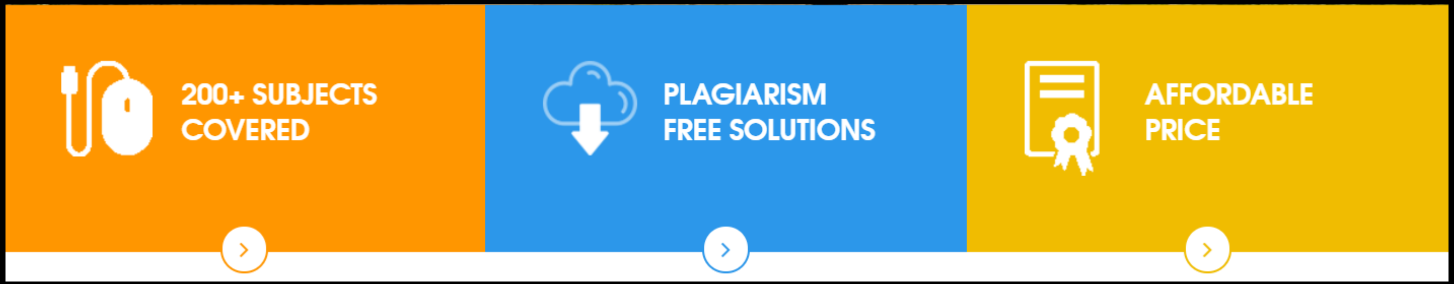
Popularity of app ≠ popularity of dashboard

Case Study #2: Contract Cheating

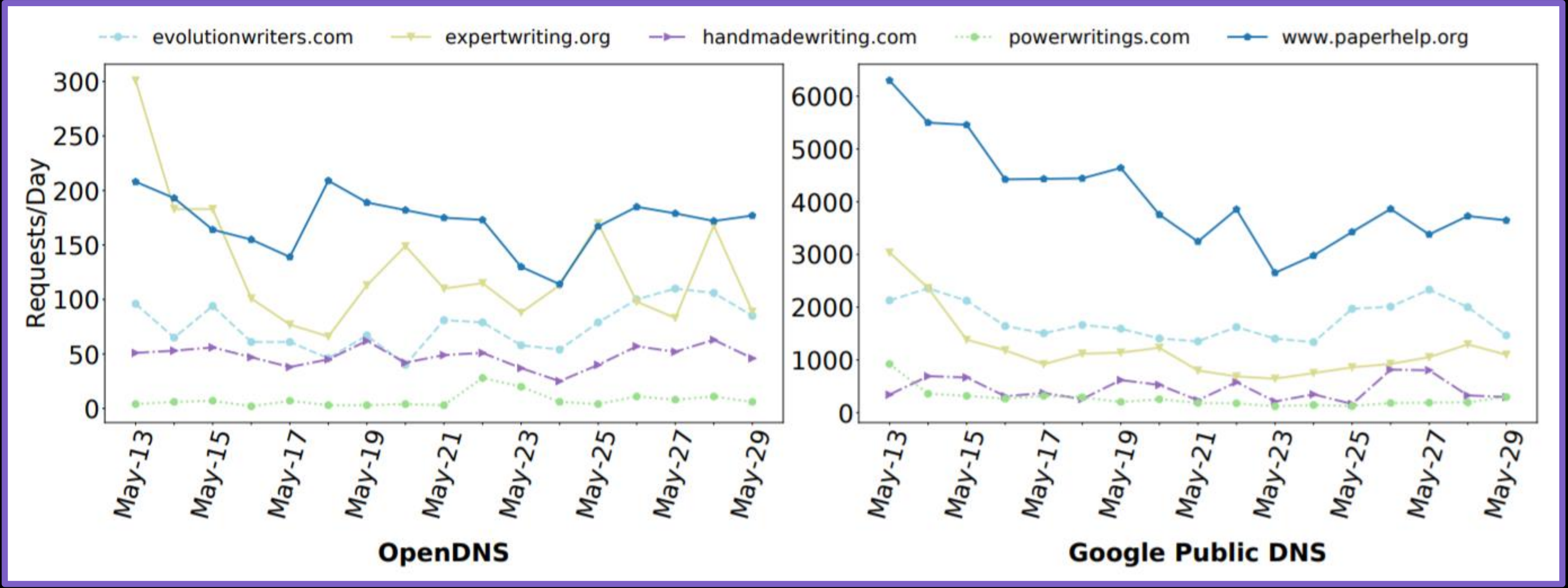


Services complete homework, projects, even entire classes

Hard to detect – original content, plagiarism checkers don't work

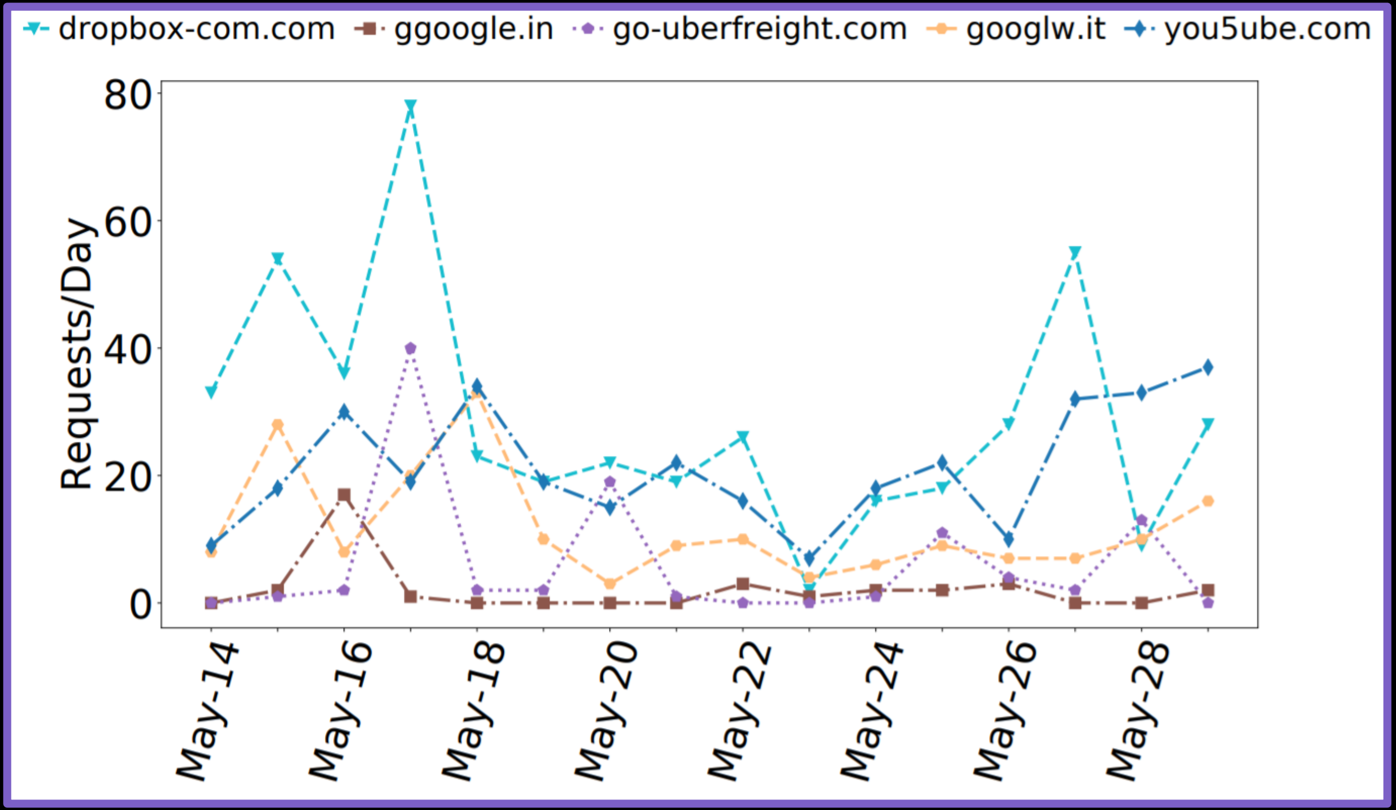


Observed Contract Cheating



Some services decrease over time:
schools letting out for summer break?

Case Study #3: Typo Squatting



Even though domains are old and probably blacklisted, we see requests.

Takeaway: Don’t get rid of cache snooping yet!

Minimal privacy concerns on public resolvers

- Too many users to de-anonymize

Can measure types of harm that are otherwise difficult to study

- Stalkerware
- Contract cheating
- New phenomena
  - Hack-for-hire services
  - Phishing

Conclusion

Public DNS resolvers enable privacy-preserving cache snooping

- Valuable measurement technique – should not be disabled

Public resolver cache architecture is complex

- We reverse engineer four resolvers' strategies
- Cloudflare, Google cause minor TTL noncompliance

We found non-trivial lower bounds of the prevalence of hard-to-study Internet phenomena.

https://github.com/ucsdsysnet/trufflehunter