Presented at IETF 109 - MAPRG on 16th November 2020

CLOUDFLARE

Oblivious DNS over HTTPS (ODoH)

A Practical Privacy Enhancement to DNS

IETF Draft: draft-pauly-dprive-oblivious-doh-02
Initial Detailed Report: https://sudheesh.info/papers/odoh-ietf-109.pdf

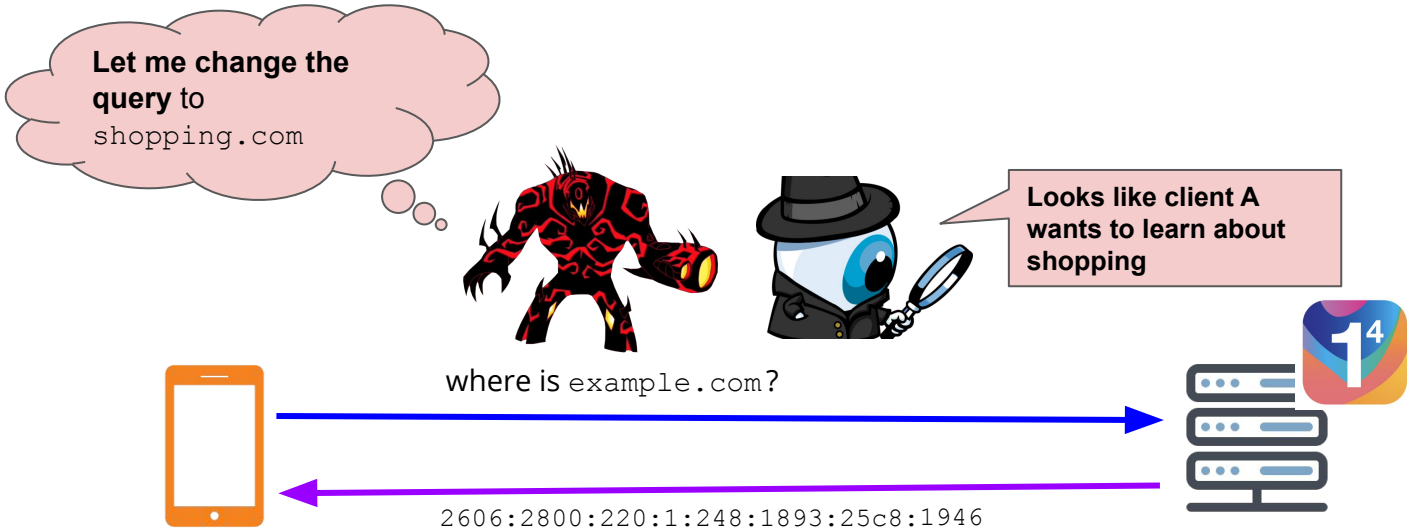
Measurements and Feasibility

Sudheesh Singanamalla[*†], Suphanat Chunhapanya[*], Marek Vavruša[*], Tanya Verma[*], Peter Wu[*], Marwan Fayed[*], Kurtis Heimerl[†], Nick Sullivan[*], Christopher Wood[*]

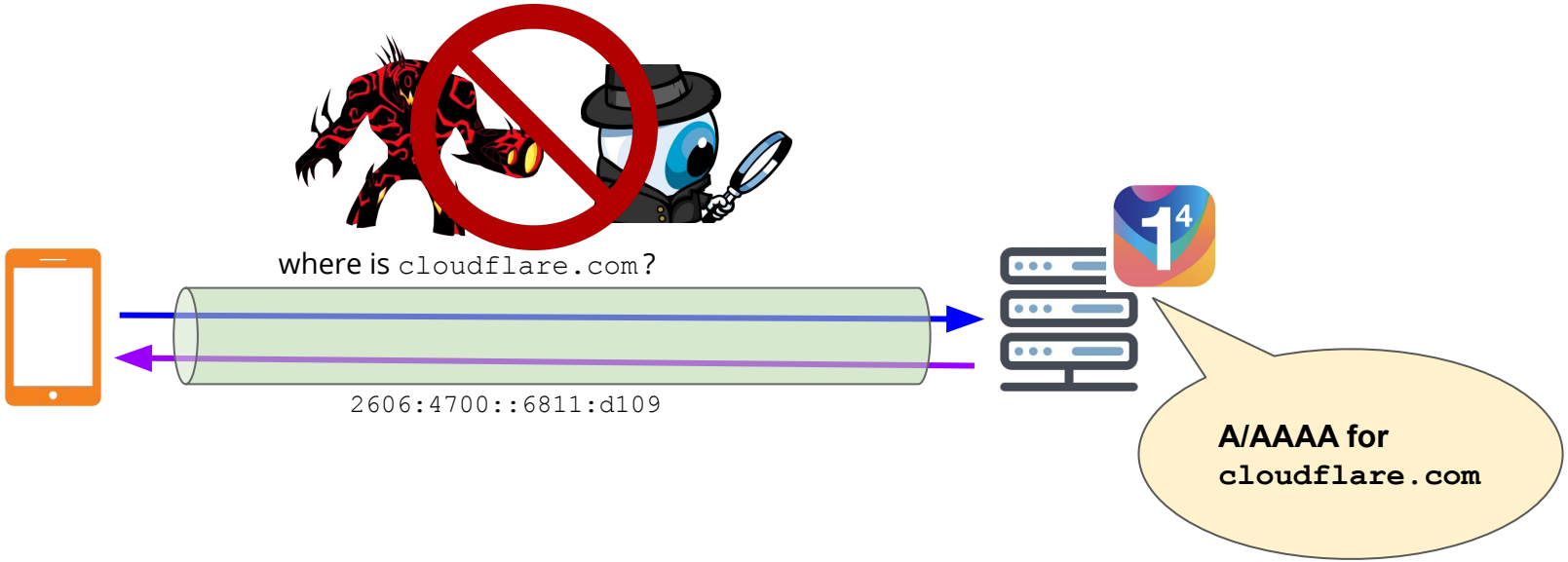
[*]Cloudflare Inc. [†]University of Washington

Do53: Plain-text UDP exposes DNS messages

Most Widely Used Variant of the protocol (92% daily traffic to *1.1.1.1*)



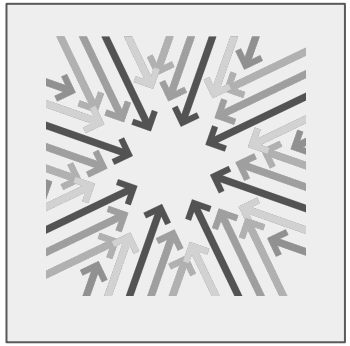
DoH: Encrypts Stub-to-Resolver link



DoH Analysis

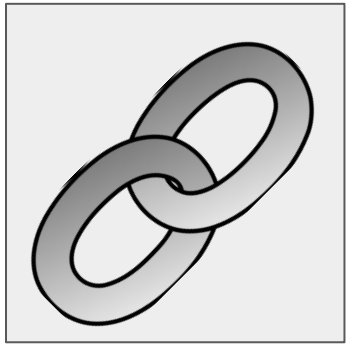
1. **[Böttger *et al.*]** Switch to DoH does not significantly impact page load times and improves user security
2. **[Housel *et al.*]** Performance of encrypted protocols vary by choice of DoH resolver
3. **[Sundaresan *et al.*]** Page load times can be improved by prefetching.

The gaps in DoH that ODoH fills



Centralization of services

Small number of deployments



Association of query to clients

Resolver operators can associate query to clients

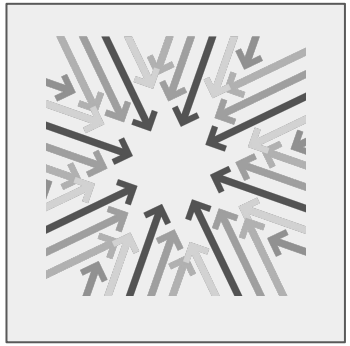
Privacy by Policy

Privacy backed by privacy policy. Needs explicit efforts like Mozilla TRR List.

Regulatory Concerns

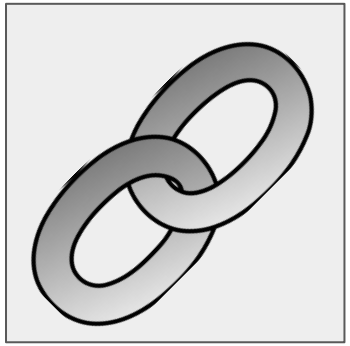
Heavy regulation to prevent monetization attempts.

The gaps in DoH that ODoH fills



Centralization of services

Small number of deployments



Association of query to clients

Resolver operators can associate query to clients

Privacy by Policy

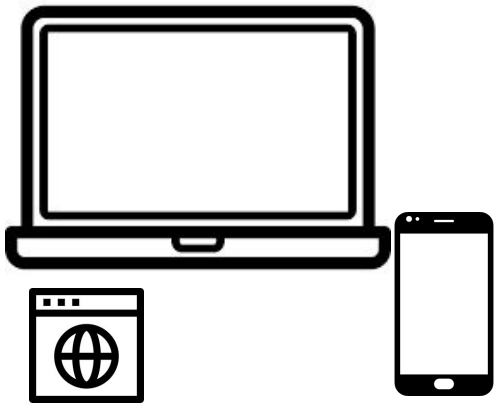
Privacy backed by privacy policy. Needs explicit efforts like Mozilla TRR List.

Regulatory Concerns

Heavy regulation to prevent monetization attempts.

Components of ODoH

Clients



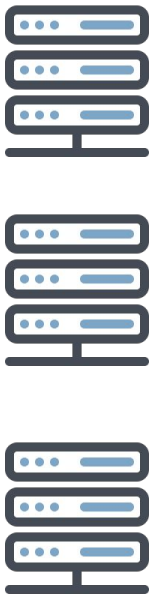
- Prepare DNS Query requests
- Receive DNS Answer responses
- Relays the request through a proxy

Goals:

1. Be able to successfully encrypt and decrypt the messages
2. Be unable to decrypt incorrectly received messages.
3. Identify maliciousness or attacks when they occur.

Components of ODoH

Proxy



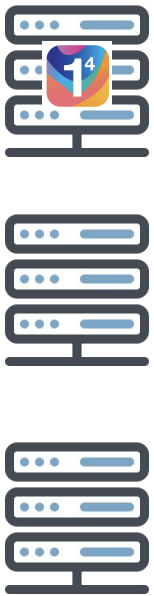
- Relay the encrypted requests to target
- Relay the encrypted responses to client
- Remove client IP addresses

Goals:

1. Remove client identifying information
2. Be unable to decrypt any messages from either the client or the target instances
3. Operated by an organization different from the target resolver

Components of ODoH

Targets

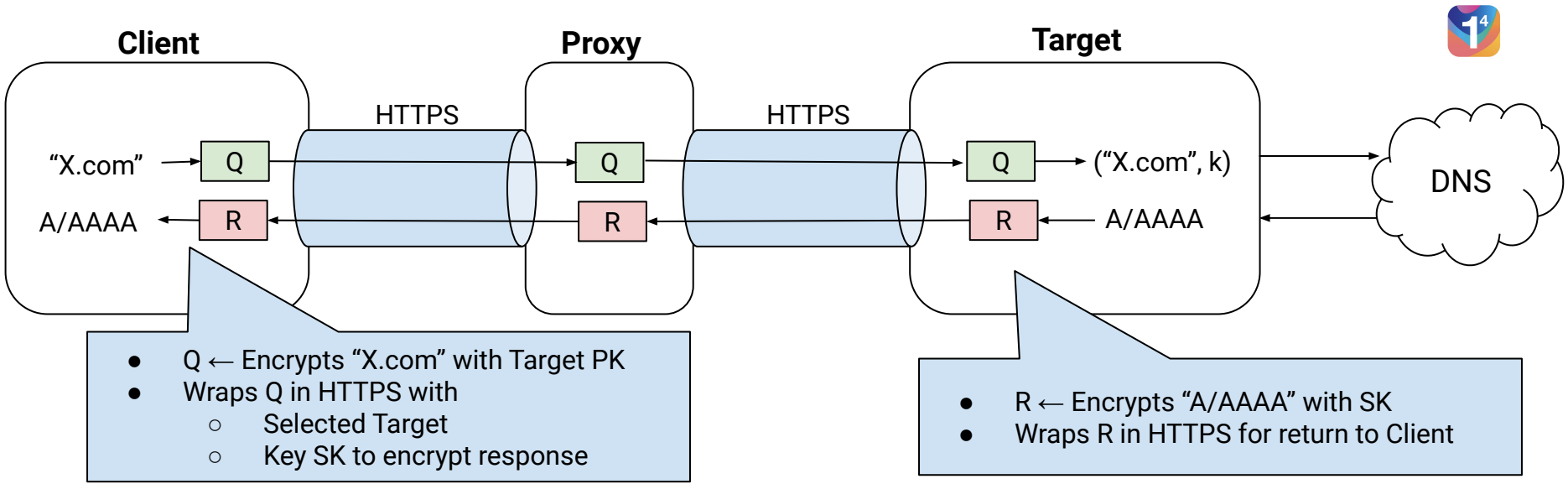


- Receive the encrypted requests from proxy
- Decrypt the query and Encrypt the answer

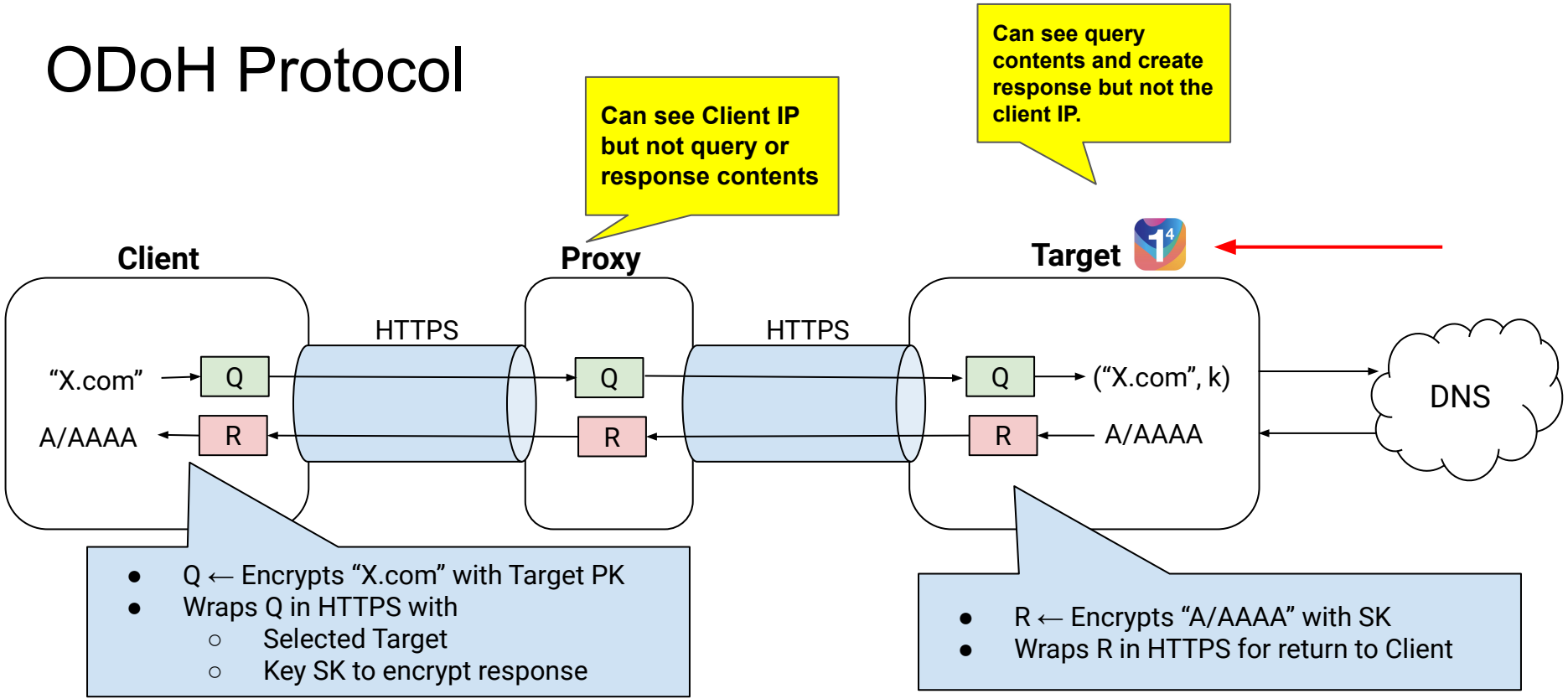
Goals:

1. Successfully decrypt the query
2. Obtain the answer from a resolver
3. Encrypt the answer and respond to proxy
4. Be unable to identify the actual client requesting the information.

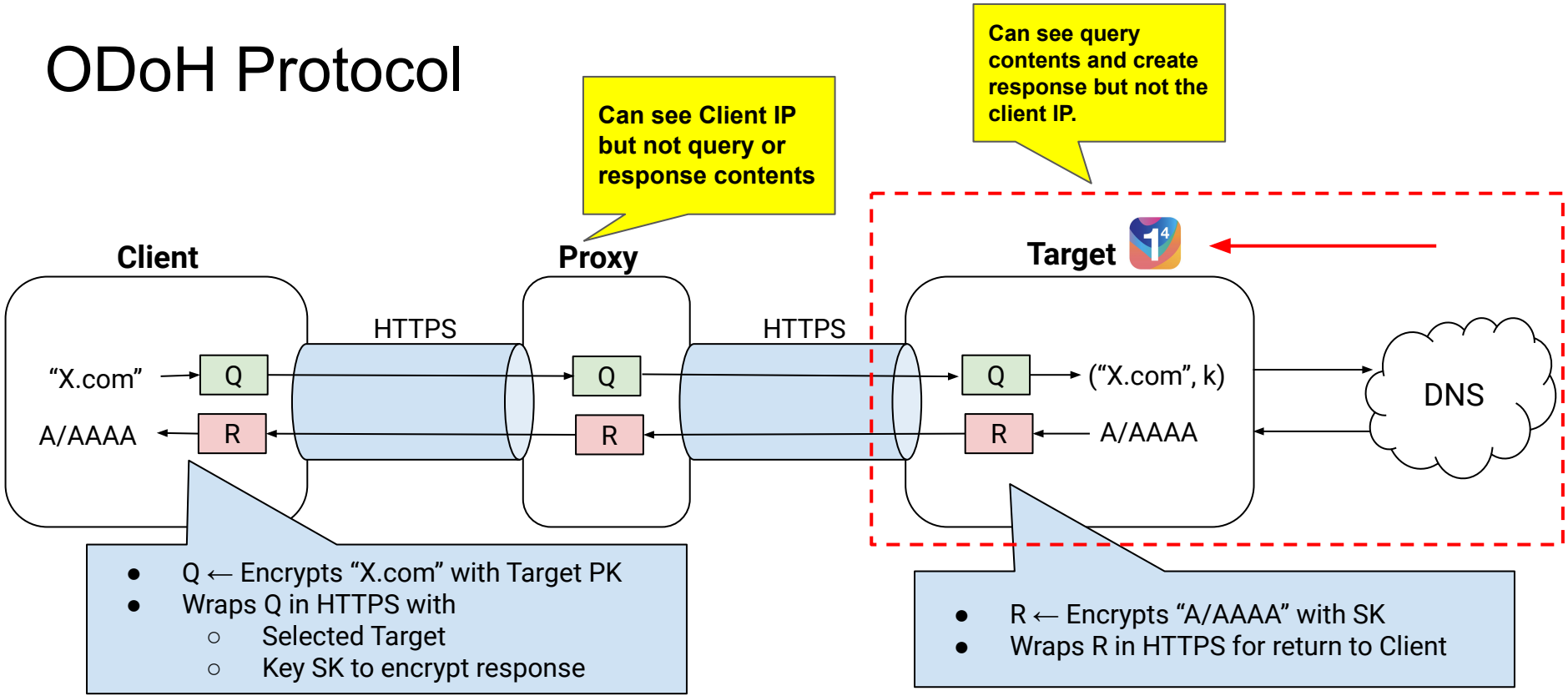
ODoH Protocol



ODoH Protocol



ODoH Protocol



ODoH Measurements Questions

Q1: What is the impact of using ODoH on **DNS response times**?

Q2: How does the usage of ODoH affect **Page Load Times** & user experience?

Q3: How does ODoH **compare to other privacy enhancing protocol** variants?

ODoH Measurements - High Level Takeaways

Q1: What is the impact of using ODoH on DNS response times?

- ODoH's higher performance relies on **choosing low latency proxy-target pairs**.
- **Service co-location** between the Target and Resolver improves response time.
- **Reusing connections** improves DNS response times when using ODoH.

Q2: How does the usage of ODoH affect Page Load Times & user experience?

- Despite a higher DNS response time, Page Load Times have minimal impact.
- Page load times do not have a perceivable impact due to usage of ODoH. (1.3 sec → 1.6 sec)

Q3: How does ODoH compare to other privacy enhancing protocol variants?

- ODoH strikes an interesting middle ground compared to conventional DNS protocols and other privacy enhancing variants.

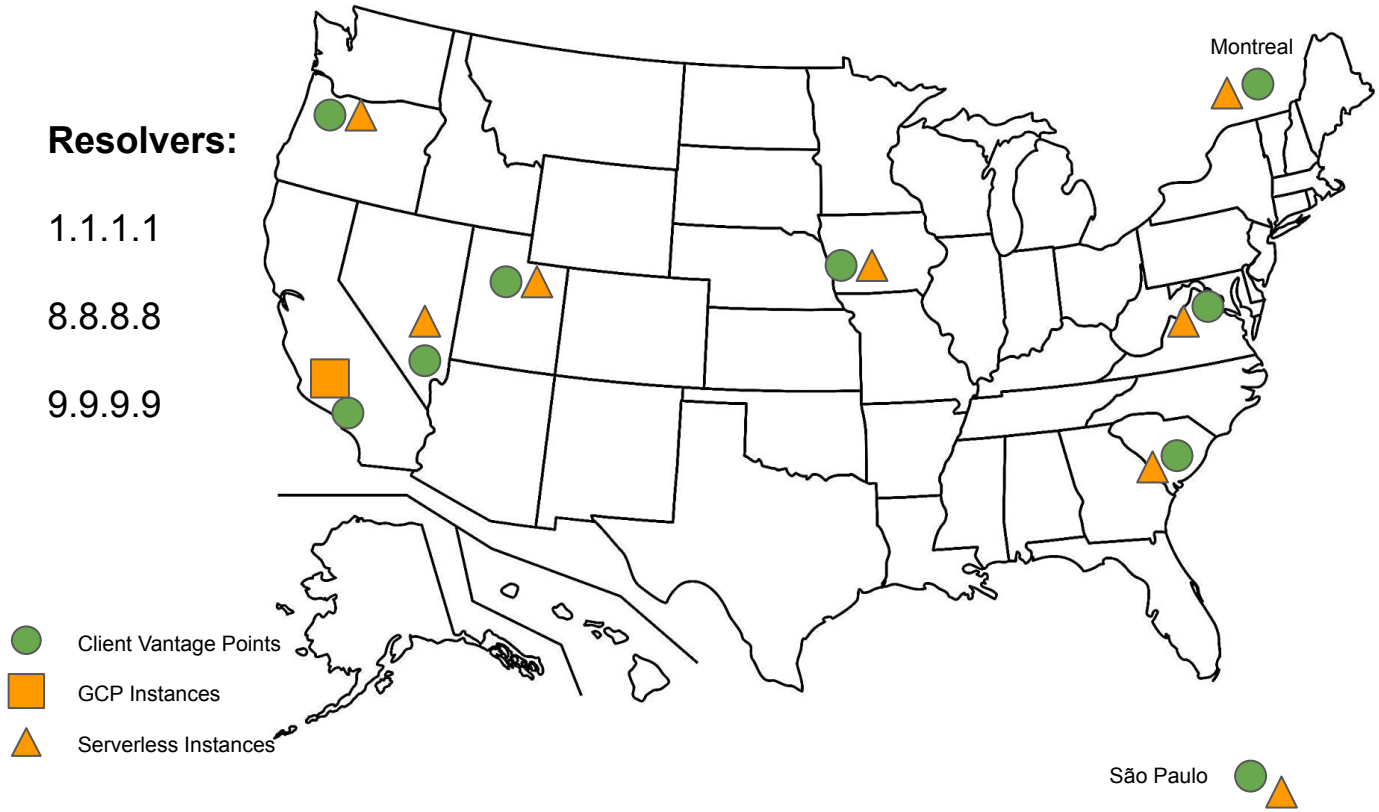
Measurement Setup and Deployments

Resolvers:

1.1.1.1

8.8.8.8

9.9.9.9



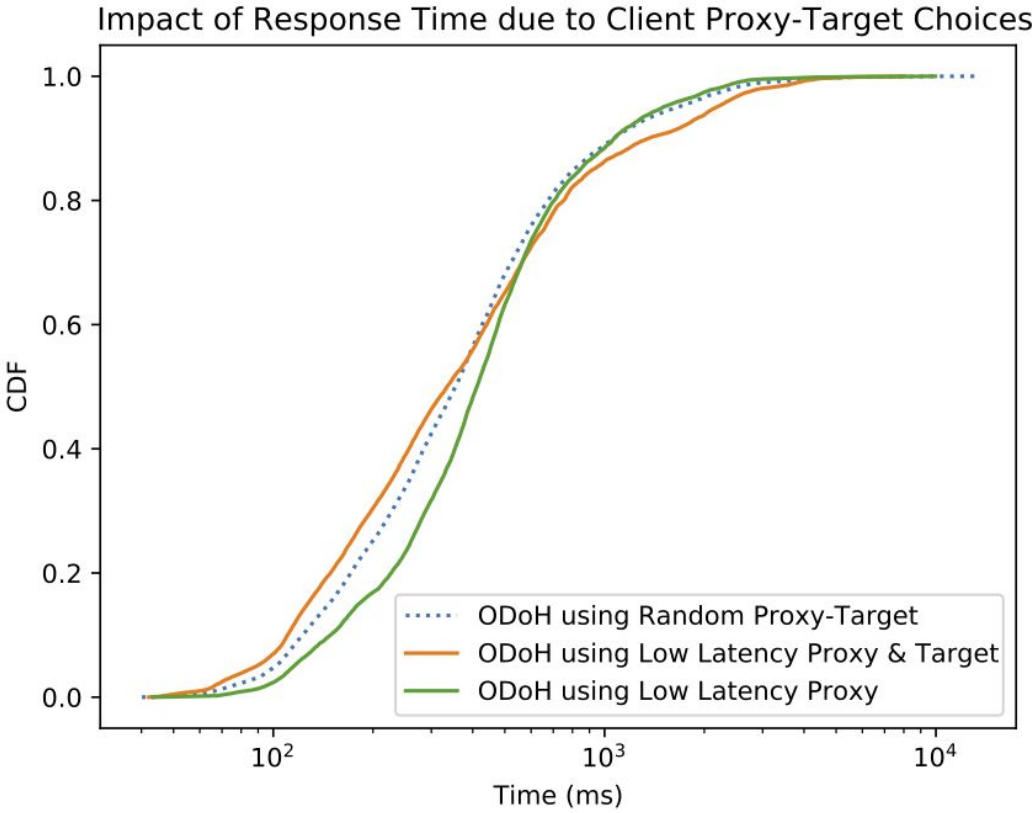
90 Client stubs
- 10 per vantage point

Experiment:
21,000 DNS req/day
or 15 requests/minute

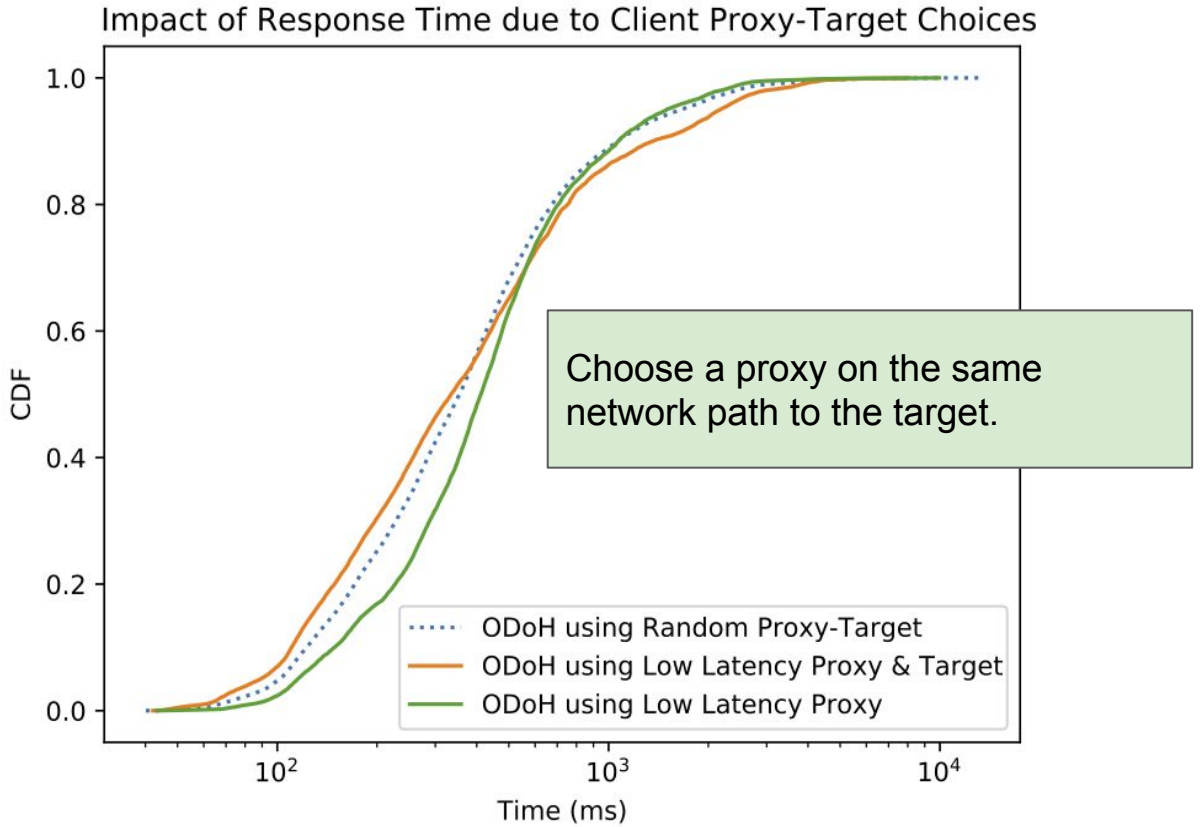
Average bandwidth:
480 Mbit/s

Clients:
1 core Intel Xeon 2 GHz CPU 3.75GB RAM x86_64

Takeaway 1: Choose Low Latency Proxy-Target Pair

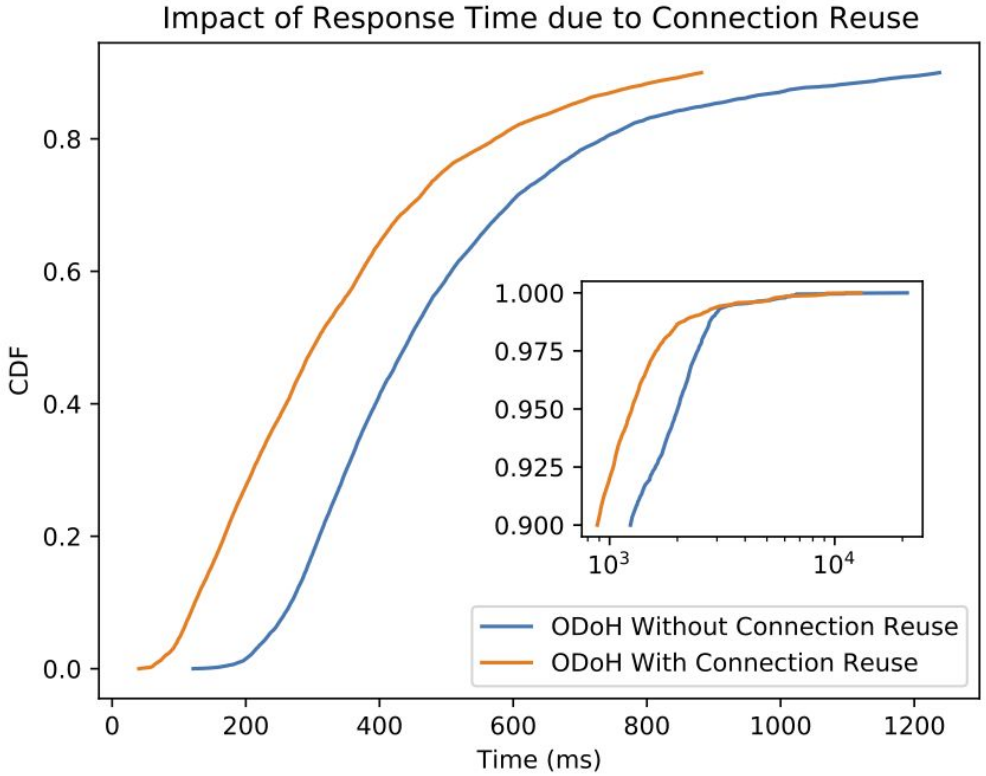


Takeaway 1: Choose Low Latency Proxy-Target Pair

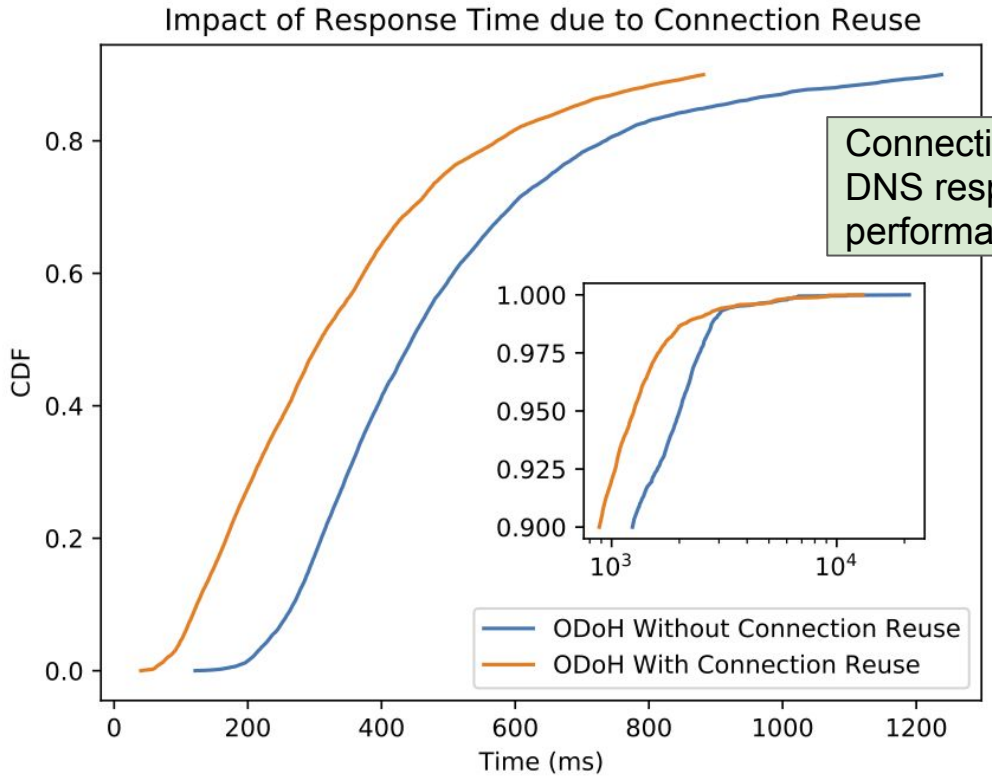


Choose a proxy on the same network path to the target.

Takeaway 2: Reuse connections!

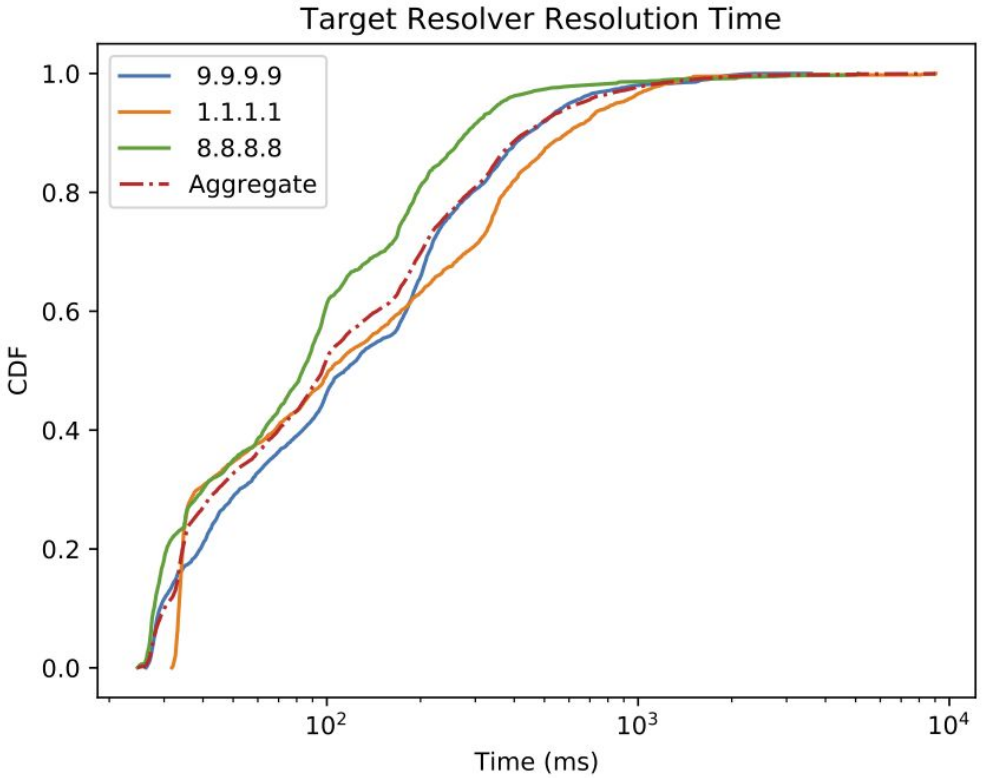


Takeaway 2: Reuse connections!

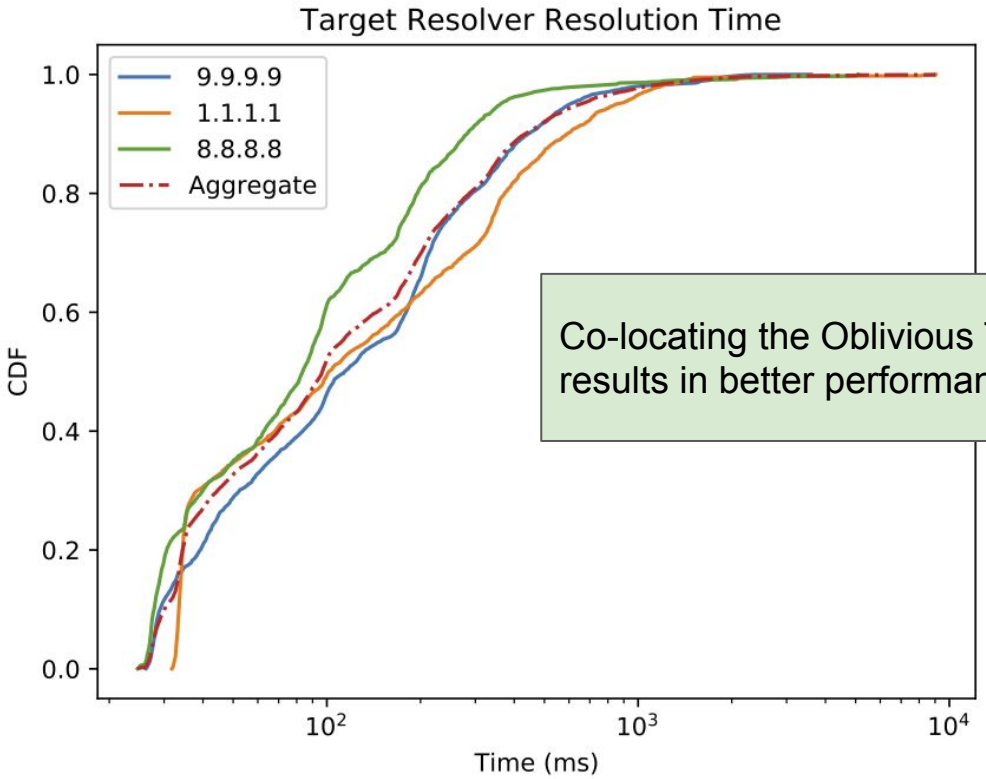


Connection reuse improves DNS response time performance by 46%.

Co-location is important: GCP Target favours 8.8.8.8

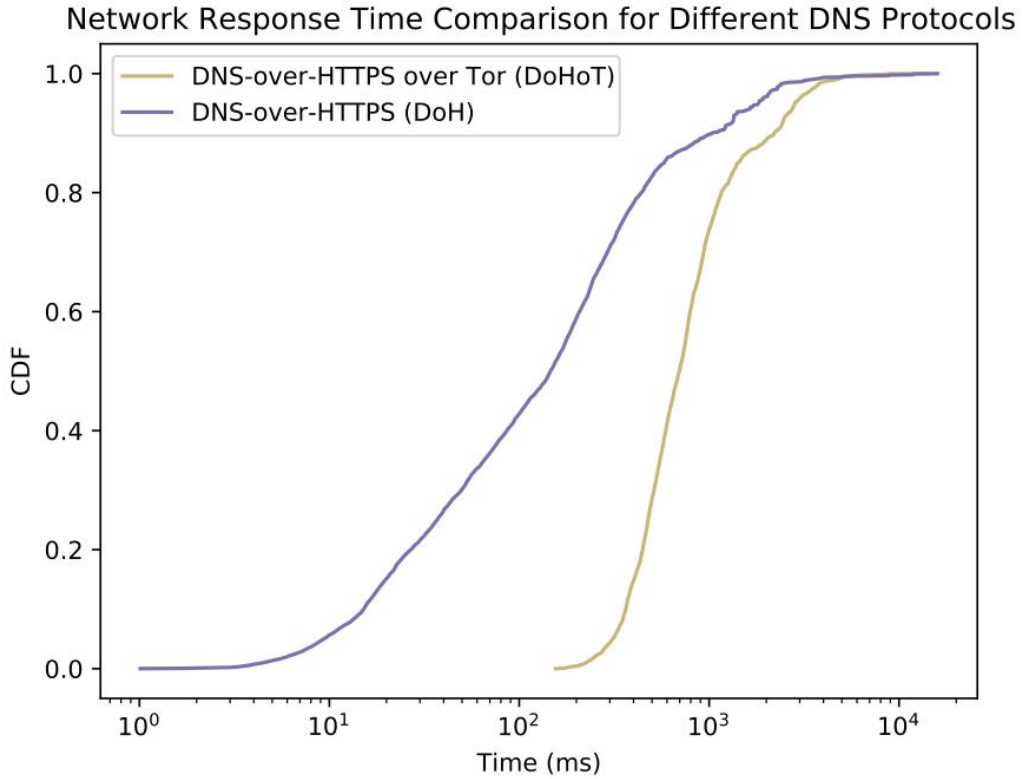


Co-location is important: GCP Target favours 8.8.8.8

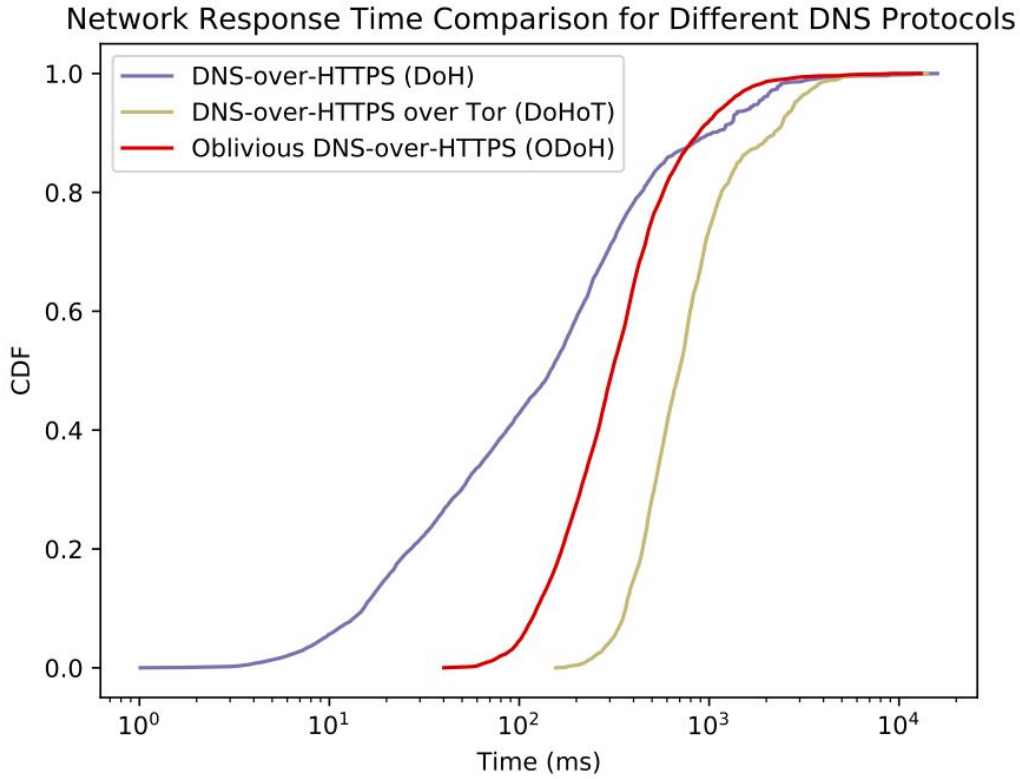


Co-locating the Oblivious Target and the Resolver results in better performance.

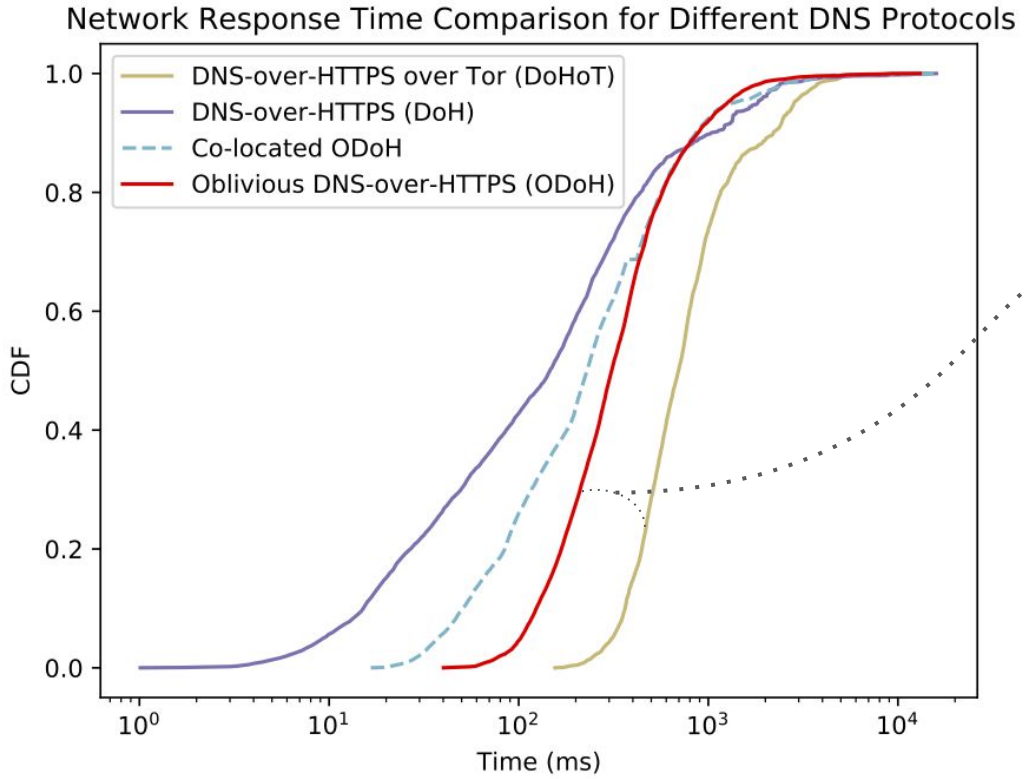
Comparing ODoH With Other DNS Protocols



Comparing ODoH With Other DNS Protocols



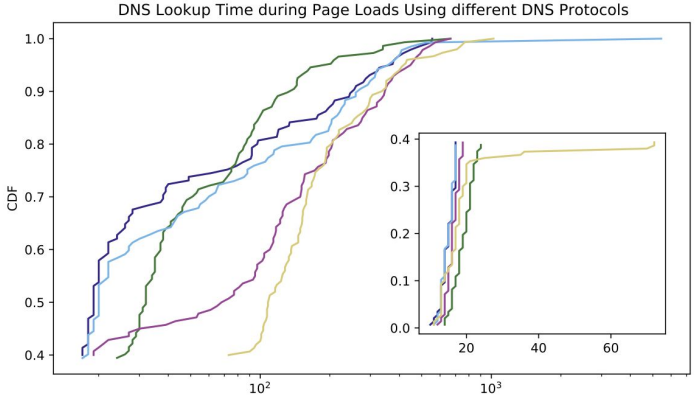
Comparing ODoH With Other DNS Protocols

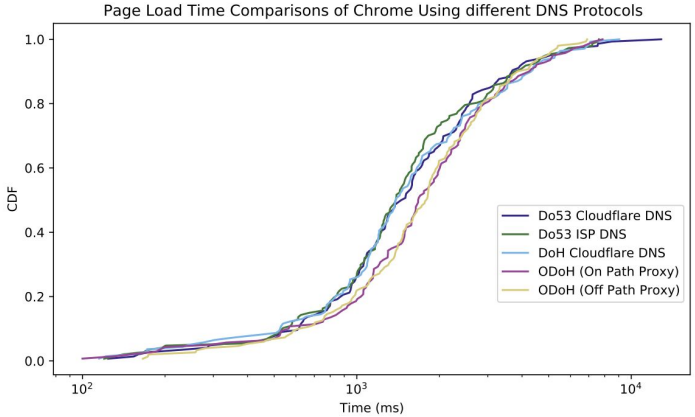


Encrypted DNS Protocols:

DNSCrypt and **Anonymous DNSCrypt**

In-Browser Measurements





Measurements taken from a single vantage point (Chrome using Local Stub resolver):

- Client node in a lab university wireless network (200 Mbps DL / 8Mbps UL)
- Experimental setup with on-path proxy
- 500 randomly chosen websites from the Top 2000 in Tranco dataset
- PLT taken after entire navigation page load event (`loadEventEnd`)
  - Worst case result

Page load times increase by ~20% (without co-location) compared to Do53 based usage and can be attributed to network topology differences.

Summary and Conclusion

1. Performance impacts in the protocol are **purely network topology effects**.
2. **Service co-location** will result in **increased response time performance**.
3. Client **choosing on-path proxy** results in higher response time performance.
4. Clients are encouraged to **reuse https connections** to avoid TLS+TCP handshake overheads.
5. ODoH has minimal total page load time impacts or perceivable user experience impacts.
6. **ODoH is a practical privacy enhancing protocol for DNS.**

Available Code and Paper

Please find the server and client implementations at:

- https://github.com/cloudflare/odoh-server-go
- https://github.com/cloudflare/odoh-proxy-worker
- More variants on https://github.com/cloudflare/

The library implementations:

- https://github.com/cloudflare/odoh-rs
- https://github.com/cloudflare/odoh-go

Clients:

- https://github.com/cloudflare/odoh-client-go
- https://github.com/cloudflare/odoh-client-rs

Report Link: **https://sudheesh.info/papers/odoh-ietf-109.pdf**

A special shoutout to:

- **Jonathan Hoyland**
- Tommy Pauly
- Eric Kinnear
- Wesley Evans

And countless others who made this work possible.

Backup Slides

Explicit Attempts at Privacy and Prior Measurements

1. Range Queries with random noise and usage of Privacy Information Retrieval (PIR) techniques in addition to DNSSEC [Zhao et al.]
2. Exponential Network bandwidth usage and Insecure to channel adversaries [Castillo-Perez et al.]
3. Using mix-cascades or cryptographic mixes to anonymize user traffic.

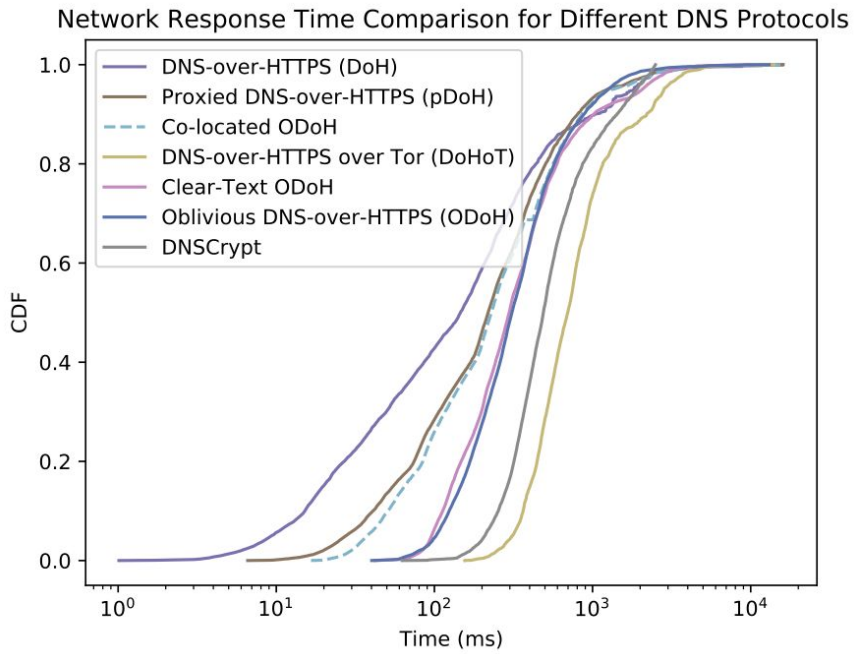
Wait … What's Different? Other Protocols?

1. DoH / DoT encrypt the network channel and sending a plain-text message over an encrypted channel.
2. DNSCrypt uses a resolver public key and exchange a short term key and send encrypted DNS queries over a non-encrypted channel.
3. Anonymous DNSCrypt introduces public non-logging proxies to route DNSCrypt requests and responses.
4. DoH over Tor (DoHoT) enables layered Tor encryption to provide privacy and security guarantees
5. ODoH performs both query encryption and uses an encrypted network channel while not impacting user perceived performance.

Protocol Implementation

- Interoperable implementation with Go lang and Rust.
- Client capable of spawning *C* client processes performing *N* queries at *R* DNS requests per minute.
- HPKE
  - KEM: X25519
  - KDF: SHA256
  - AEAD: AES128-GCM
- Additional support for
  - X448, P256, P521 KEMs
  - SHA384, SHA512 KDFs and
  - AES256-GCM, ChaCha20Poly1305 AEADs.

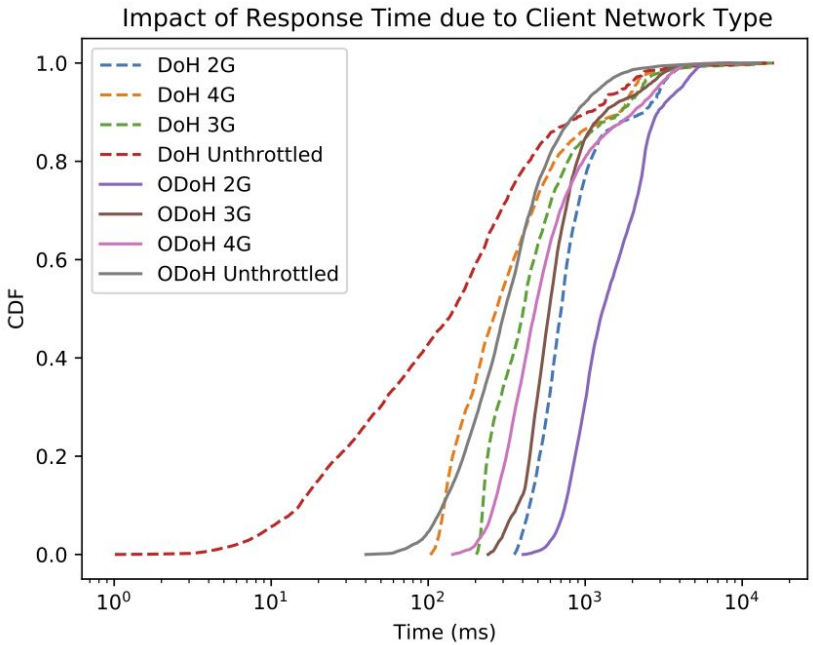
Comparing Other Architectural Variants



Protocol	Request Path	Security	Privacy
Plain DNS (Do53)	C → R	No	No
DNS over HTTPS (DoH)	C → R	Yes	No*
Proxied DoH	C → P → R	Yes	No
Oblivious DoH (ODoH)	**C → P → T → R**	**Yes**	**Yes**
Cleartext ODoH	C → P → T → R	Yes	No
Co-located ODoH	C → P → (T+R)	Yes	Yes
DNSCrypt	C → R	Yes	No*
Anonymous DNSCrypt	C → P → R	Yes	Yes
DoH over Tor (DoHoT)	C → Tor → R	Yes	Yes

C: Client, R: Resolver, T: Target, P: Proxy

Network Type Impacts



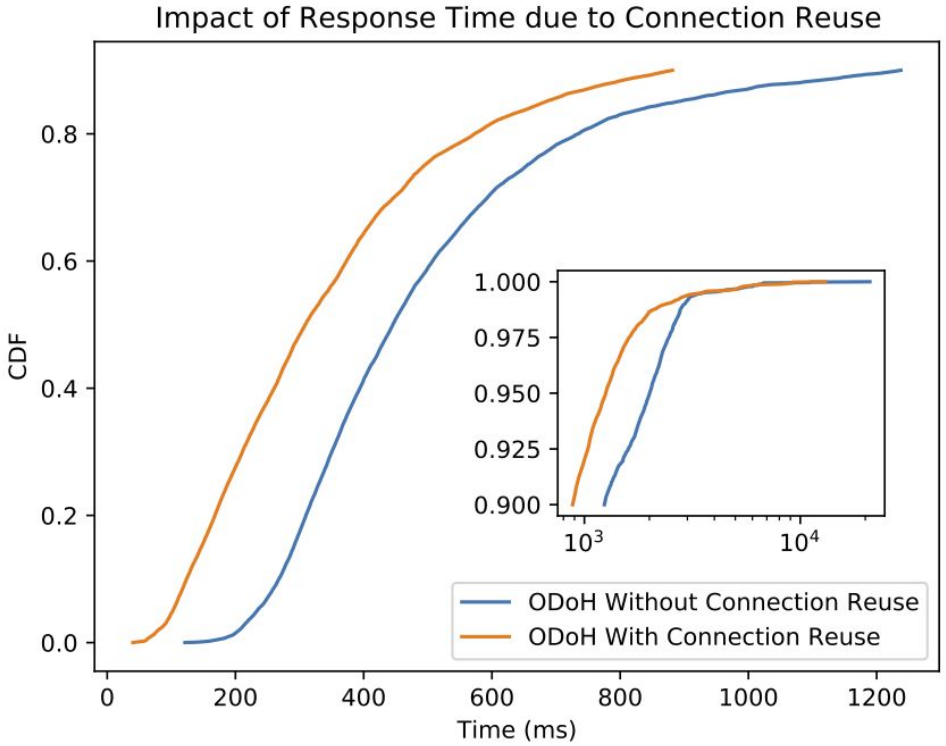
Throttled the client instances using qdisc with egress bandwidth:

2G: 0.56 Mbps with 350ms latency

3G: 1.25 Mbps with 200ms latency

4G: 12 Mbps with 100ms latency

Takeaway 2: Reuse connections!



Connection reuse improves DNS response time performance by 46%.

Some leakage of client identity due to reuse of session keys

- **No sensitive information** in either cleartext or encrypted form **is leaked**

Possible for clients to configure and force new connections if necessary.