DNS Queries over CoAP (DoC)

draft-lenders-dns-over-coap
(https://datatracker.ietf.org/doc/draft-lenders-dns-over-coap/)

Martine S. Lenders, Christian Amsüss, Cenk Gündoğan,
Thomas C. Schmidt, Matthias Wählisch

IETF 113 CoRE Meeting, 2022-03-25

Outline

Motivation

Attack Scenario

Countermeasure: Encrypt name resolution triggered by IoT devices

Possible solutions

Possible solutions:

DNS over HTTPS (RFC 8484)

DNS over TLS (RFC 7858)

Possible solutions

Possible solutions:

DNS over HTTPS (RFC 8484)

DNS over TLS (RFC 7858)

DNS over QUIC (`dprive` draft)

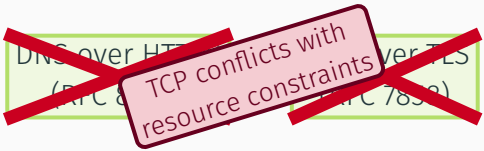
Possible solutions

Possible solutions:

- DNS over HTTPS (RFC 8484)
- DNS over TLS (RFC 7858)
- DNS over QUIC (`dprive` draft)
- DNS over DTLS (RFC 8094)

Possible solutions

Possible solutions:



DNS over QUIC
(**dprive** draft)

DNS over DTLS
(RFC 8094)

Possible solutions

Possible solutions:

Possible solutions

Possible solutions:

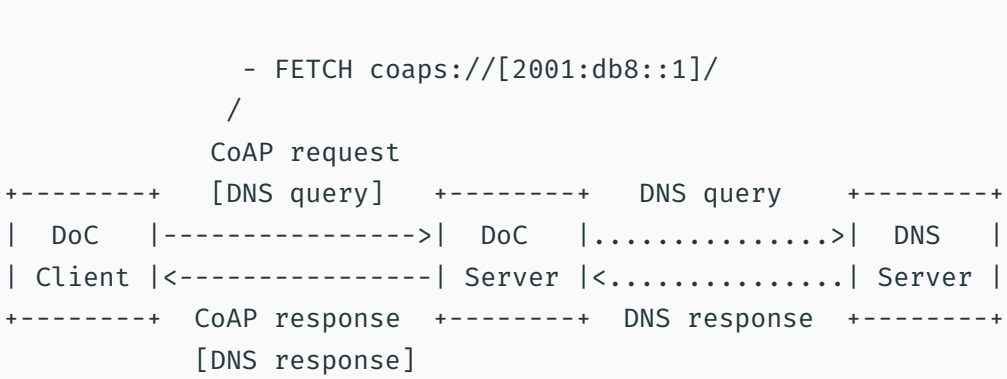
Possible solutions

Possible solutions:

Our proposal: DNS over CoAP

- **Encrypted communication** based on DTLS or OSCORE
- **Block-wise message transfer** to overcome Path MTU problem
- **Share system resources** with CoAP applications
  - Same socket and buffers can be used
  - Re-use of the CoAP retransmission mechanism

Overview


- FETCH coaps://[2001:db8::1]/
CoAP request
[DNS query]
DNS query
DoC Client
DoC Server
DNS Server
CoAP response
[DNS response]
DNS response

What happened since `interim-2021-core-12`

`draft-lenders-dns-over-coap-02`

- Remove GET and POST method specification
- Add note on ETag and response codes
- Clarify why DoQ conflicts with constrained IoT scenarios
- Clarify Content-Format / Accept handling

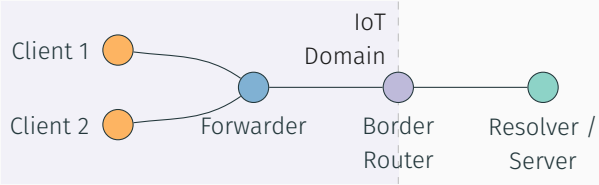
`draft-lenders-dns-over-coap-03`

- Clarify server selection to be out-of-band
- Define ”core.dns” resource type
- Add considerations on message manipulation for DoC servers
- Update considerations on unencrypted use

Evaluation: Setup

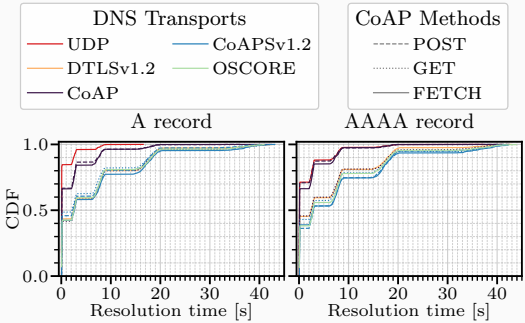
Name properties: Based on empirically measured data from IoT devices

Testbed experiments:

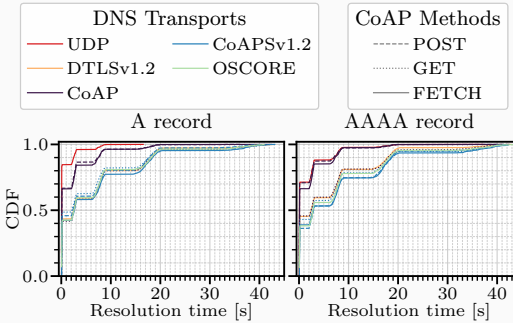


- Clients query 50 A or AAAA records for names of length 24 chars
  via DNS over UDP / DTLSv1.2 / CoAP (unencrypted) / CoAPSv1.2 / OSCORE
- Poisson distribution: $\lambda = 5$ queries / sec (ignoring `NSTART=1` requirements)
- 10 runs on IoT-nodes (incl. BR): Cortex-M3 with IEEE 802.15.4 radio

Experiment: Resolution time

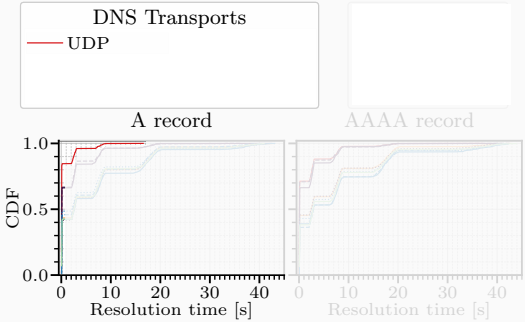


Experiment: Resolution time



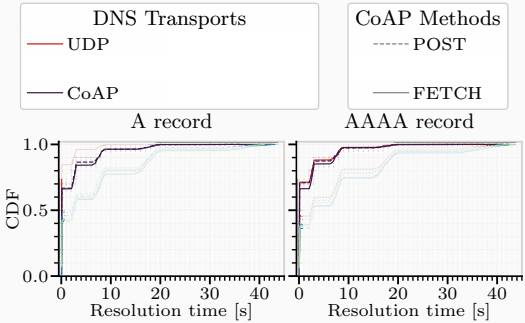
Clear performance groupings visible

Experiment: Resolution time



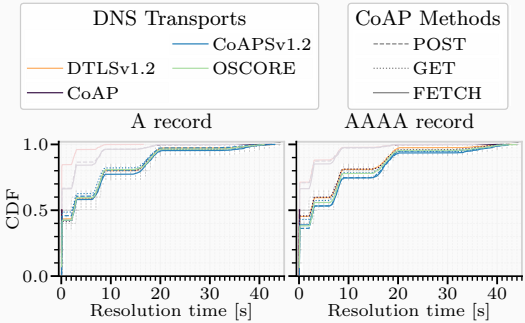
Group 1

Experiment: Resolution time & packet sizes



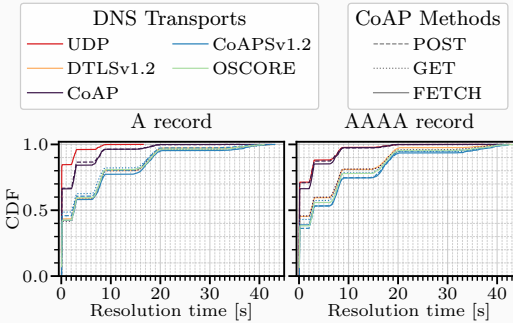
Group 2

Experiment: Resolution time & packet sizes



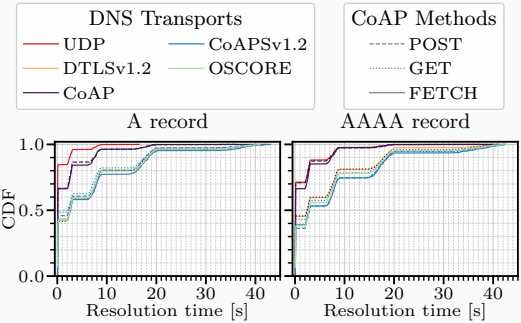
Group 3

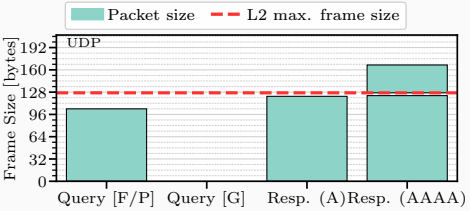
Experiment: Resolution time & packet sizes



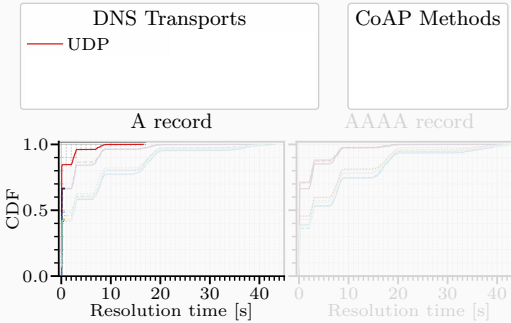
Where do performance groups come from?

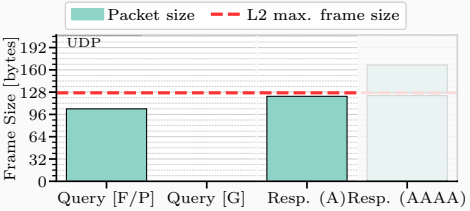
Experiment: Resolution time & packet sizes





Experiment: Resolution time & packet sizes

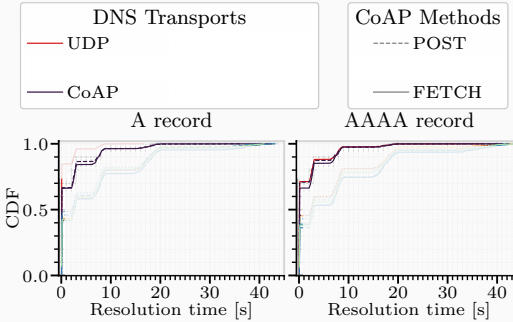


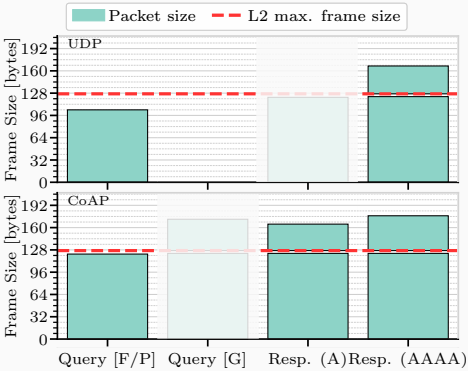


Group 1

No message fragmentation

Experiment: Resolution time & packet sizes



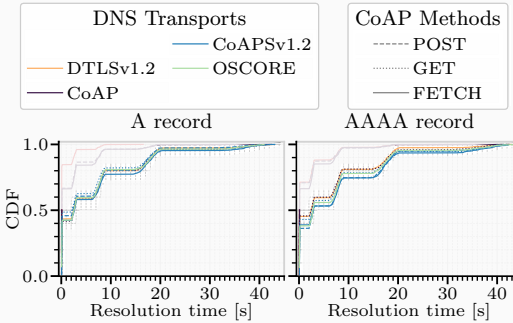


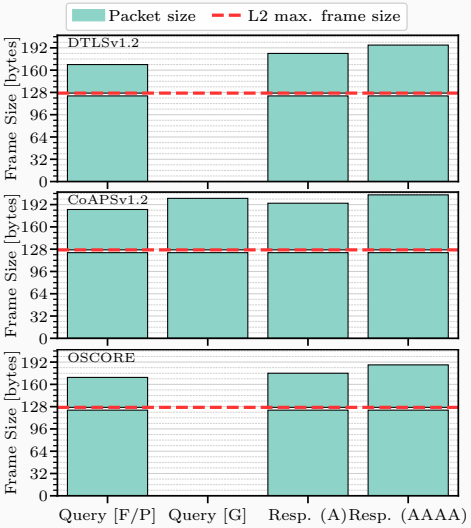
Group 2

Query unfragmented

Response fragmented

Experiment: Resolution time & packet sizes

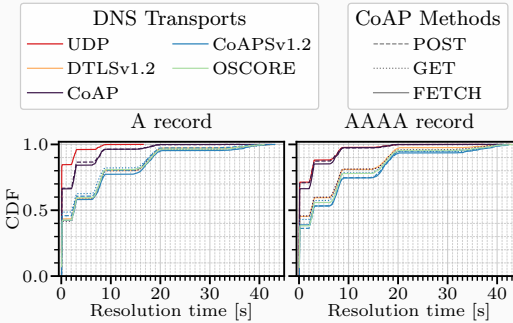




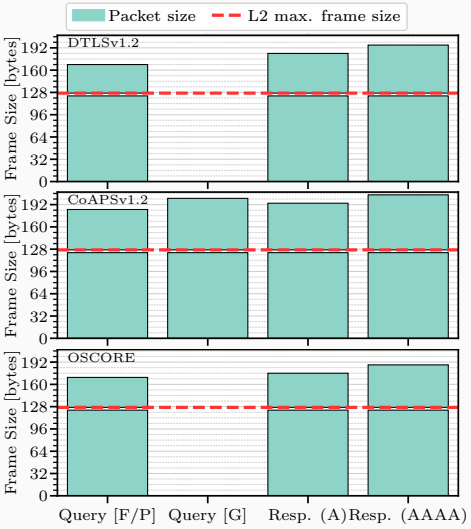
Group 3

Both messages fragmented

Experiment: Resolution time & packet sizes



⇒ Fragmentation has larger impact on performance compared to transport or CoAP method



A New Content-Format: Numerical analysis

Problem:

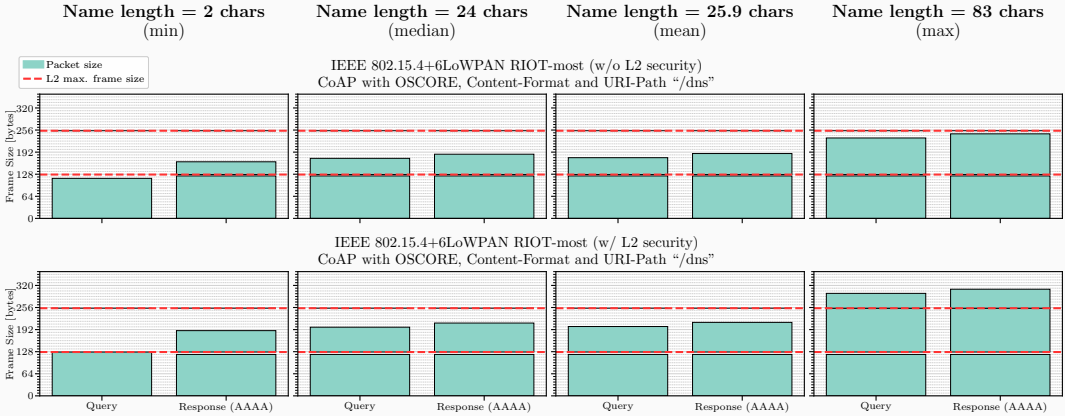
Realistic query and response sizes lead to fragmentation, using OSCORE & 802.15.4

Name length = 2 chars (min)	**Name length = 24 chars** (median)	**Name length = 25.9 chars** (mean)	**Name length = 83 chars** (max)

A New Content-Format: Numerical analysis

Problem:

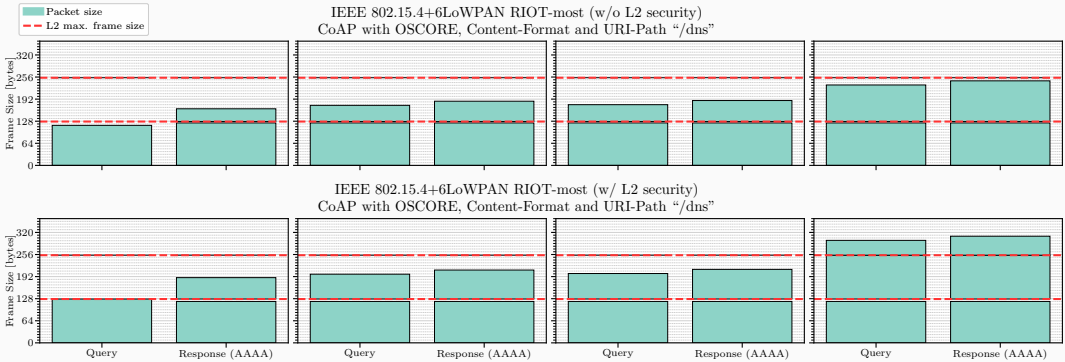
Realistic query and response sizes lead to fragmentation, using OSCORE & 802.15.4



A New Content-Format: Numerical analysis

Problem:

Realistic query and response sizes lead to fragmentation, using OSCORE & 802.15.4



⇒ Reduce packet size via compression

A New Content-Format: Some ideas

Goal: Reduce packet size

Idea:

- Omit authority and additional sections in DNS responses
- Question section always size 1: omit QDCount field
- Make class and type optional (imply IN/AAAA)
- Self-delimiting numeric values for classes, types, counts, TTLs, etc?
- Question section optional in responses?

Two Options:

- Question section CBOR-array, Answer section: CBOR-array of arrays?
- “remote getaddrbyname()” (i.e. query name (maybe type?), expect address as response)?

Discuss in separate draft?

Discussion: Caching and Max-Age vs. DNS TTL

Problem: CoAP Max-Age and DNS TTL may get out of sync at caching proxy

Option 1 (PR#17): Do it like DoH but

Server:

Max-Age = min(TTLs)

Client:

$TTL_{new} = TTL_{old}$ - (min(TTLs) - Max-Age)

Option 2 (PR#19): Do it like DoH but

Server:

Max-Age = min(TTLs)

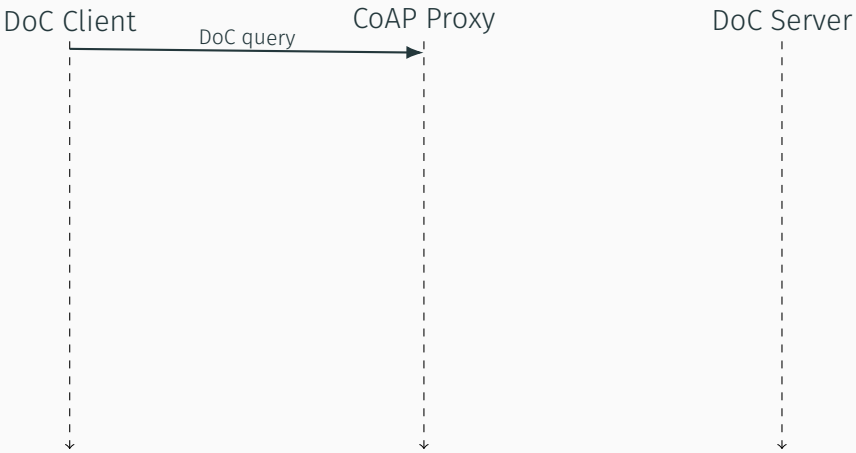
$TTL_{new} = TTL_{old}$ - min(TTLs)

Client:

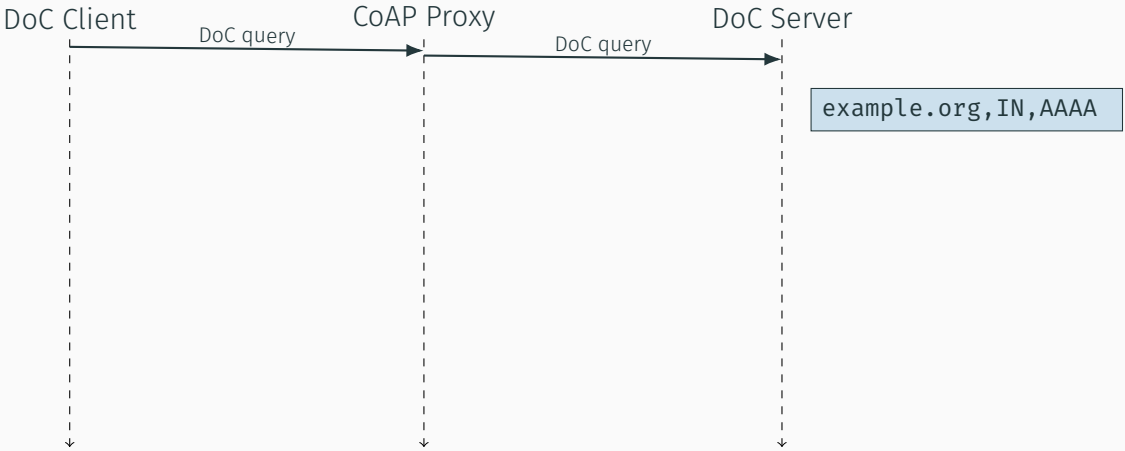
$TTL_{new} = TTL_{old}$ + Max-Age

(see GitHub-Issue #5)

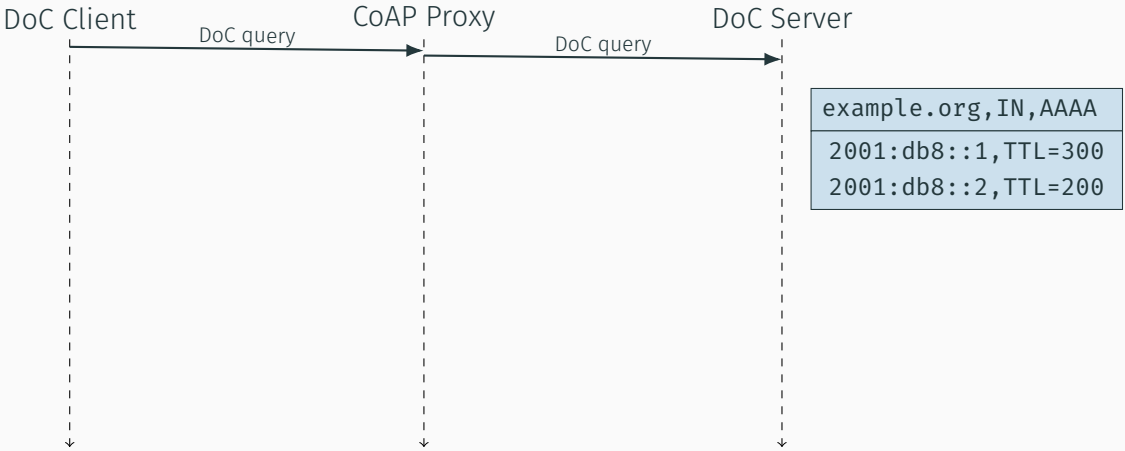
Caching and Max-Age vs DNS TTL (Option 1, DoH-like)



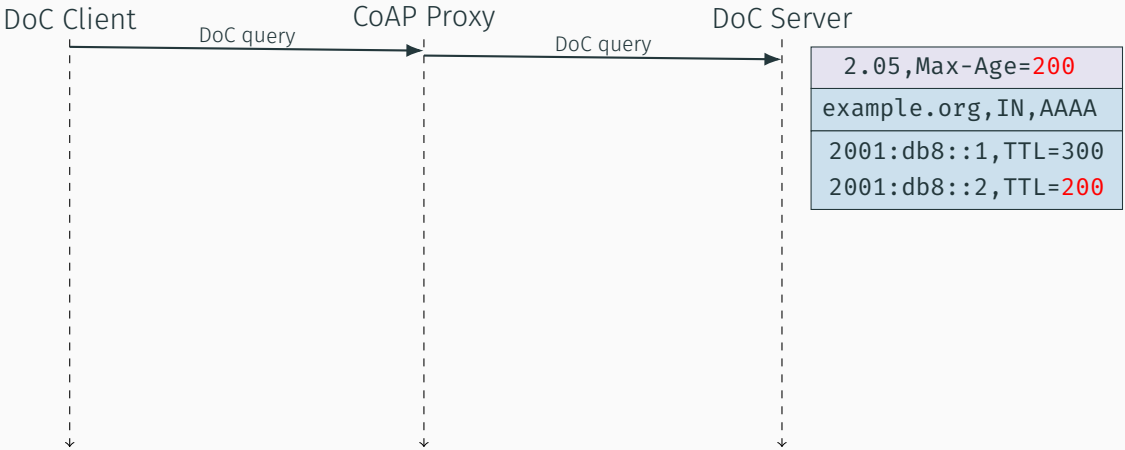
Caching and Max-Age vs DNS TTL (Option 1, DoH-like)



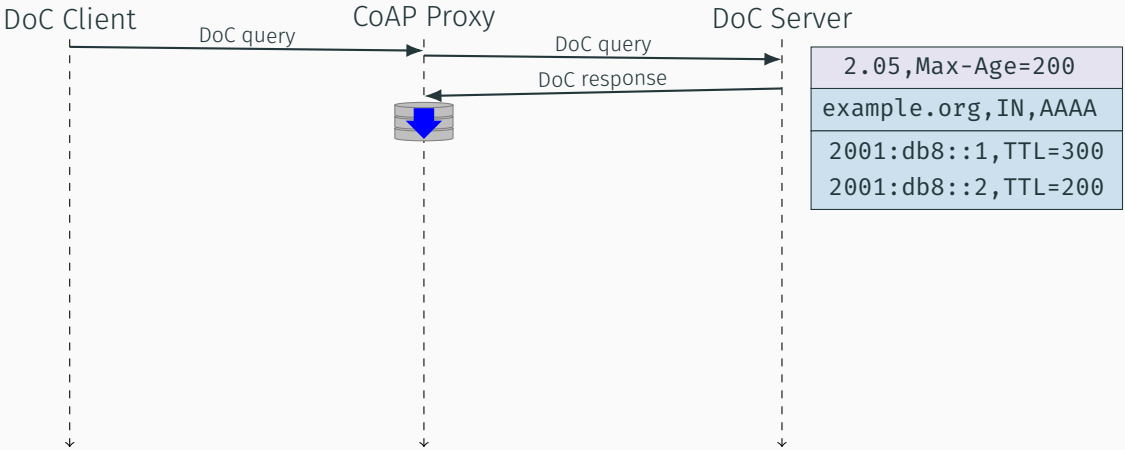
Caching and Max-Age vs DNS TTL (Option 1, DoH-like)



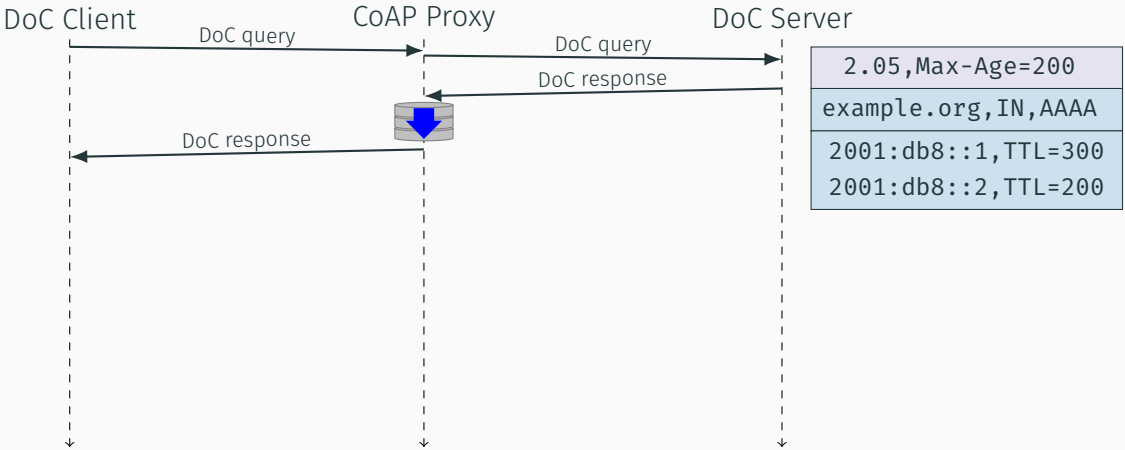
Caching and Max-Age vs DNS TTL (Option 1, DoH-like)



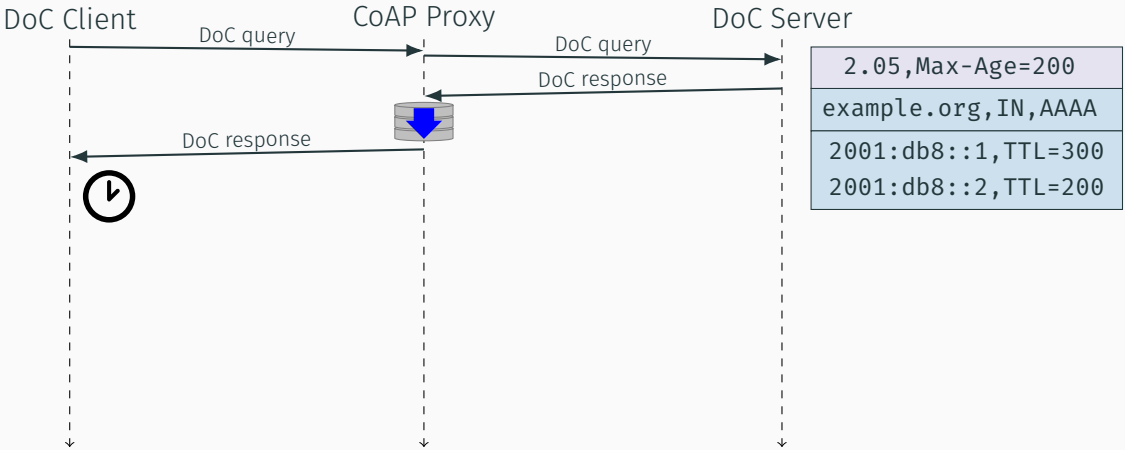
Caching and Max-Age vs DNS TTL (Option 1, DoH-like)



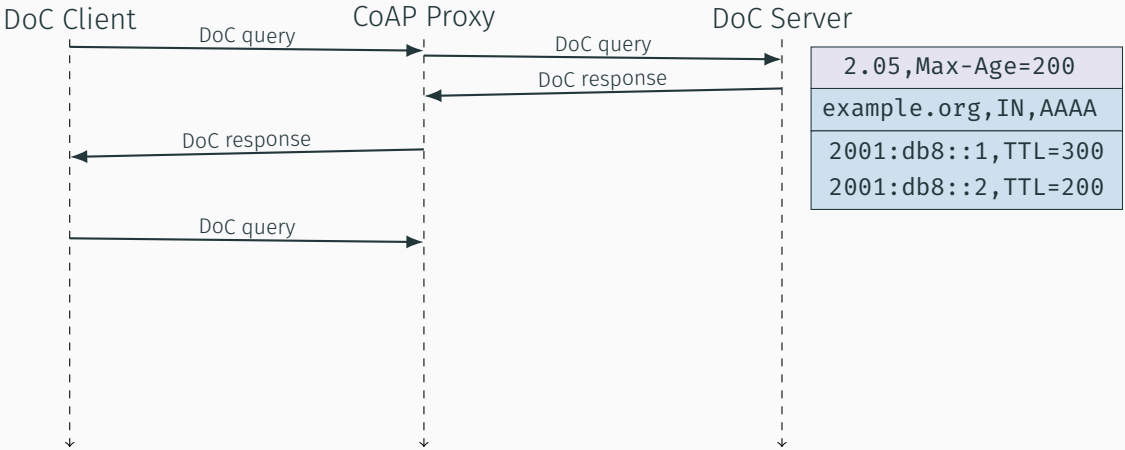
Caching and Max-Age vs DNS TTL (Option 1, DoH-like)



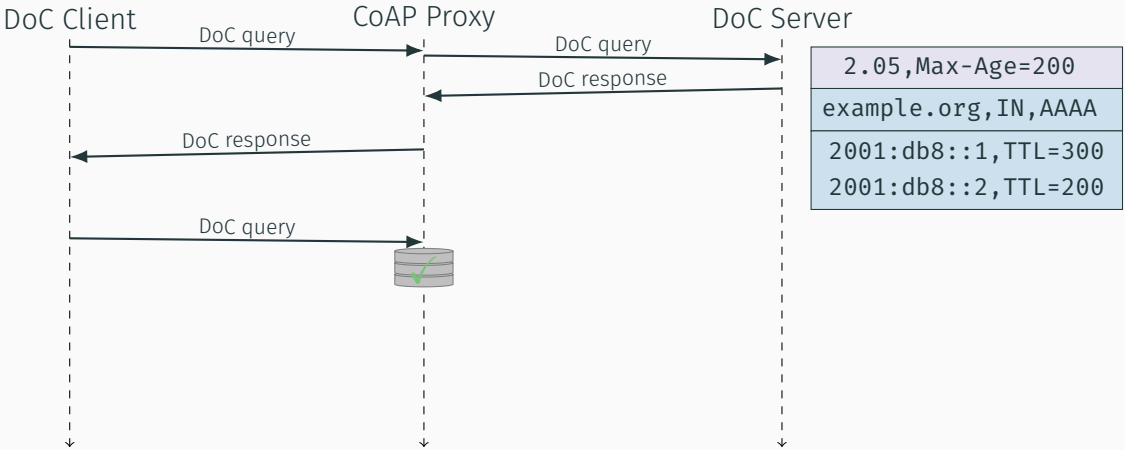
Caching and Max-Age vs DNS TTL (Option 1, DoH-like)



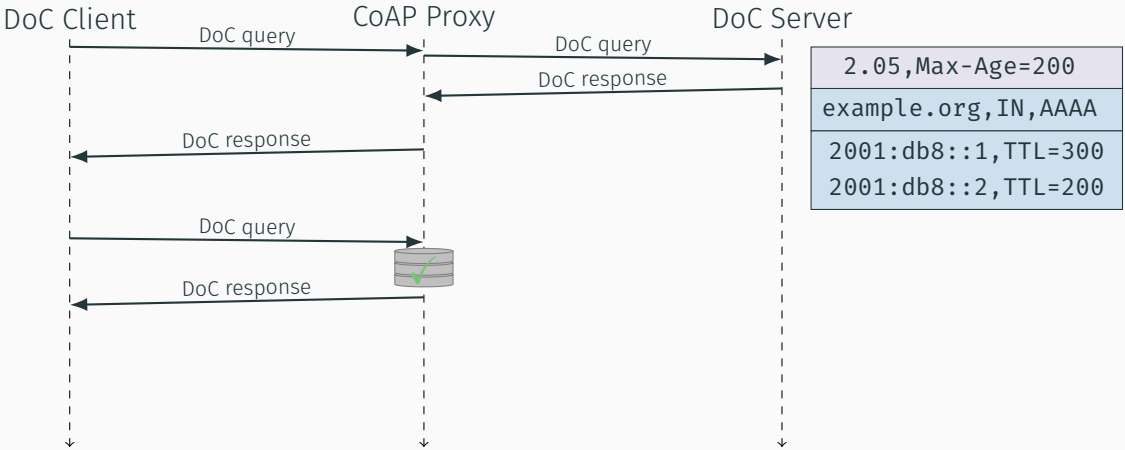
Caching and Max-Age vs DNS TTL (Option 1, DoH-like)



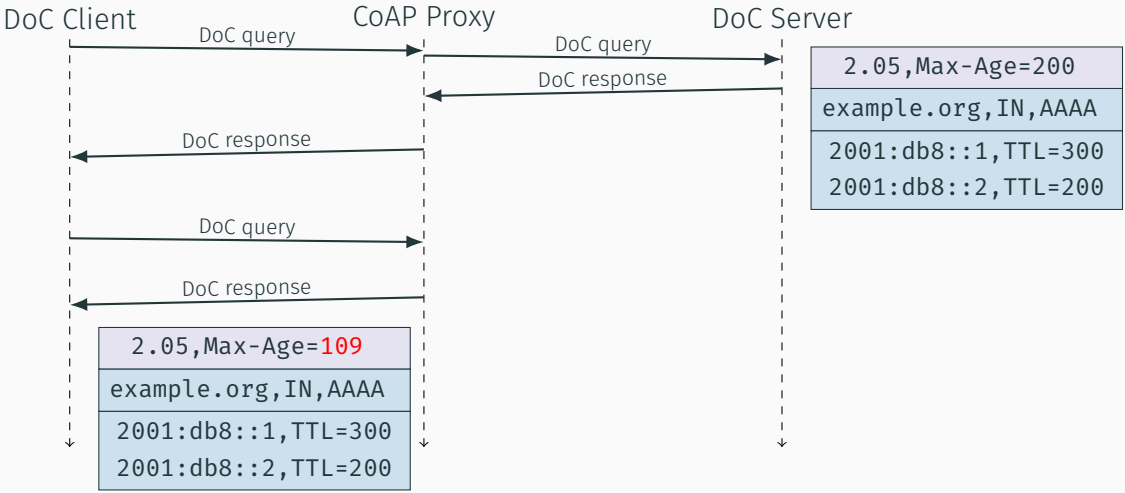
Caching and Max-Age vs DNS TTL (Option 1, DoH-like)



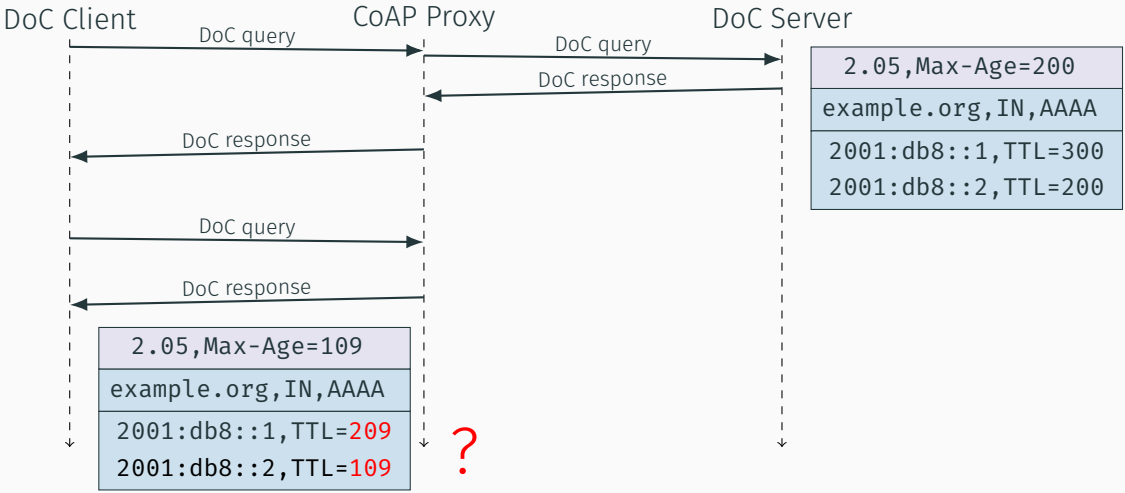
Caching and Max-Age vs DNS TTL (Option 1, DoH-like)



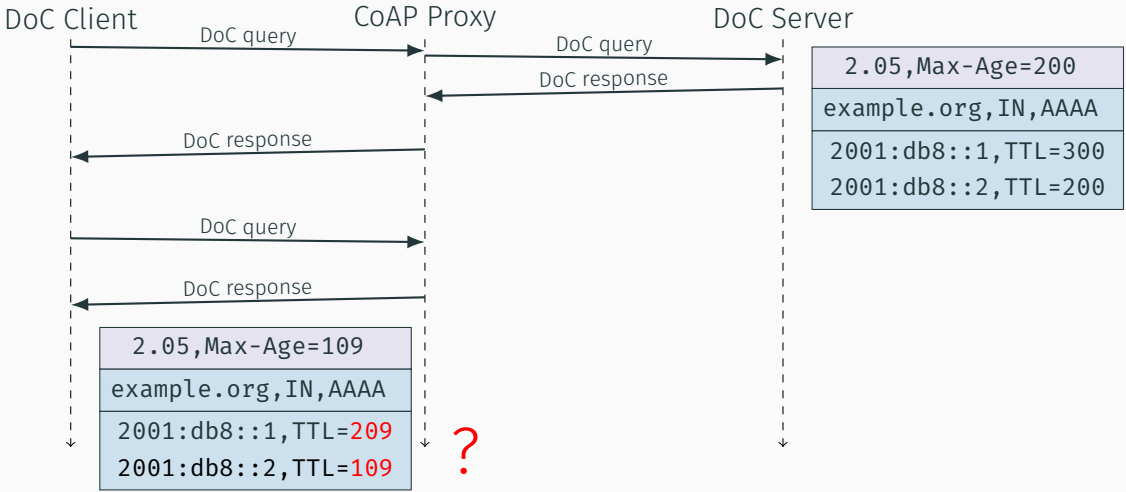
Caching and Max-Age vs DNS TTL (Option 1, DoH-like)



Caching and Max-Age vs DNS TTL (Option 1, DoH-like)

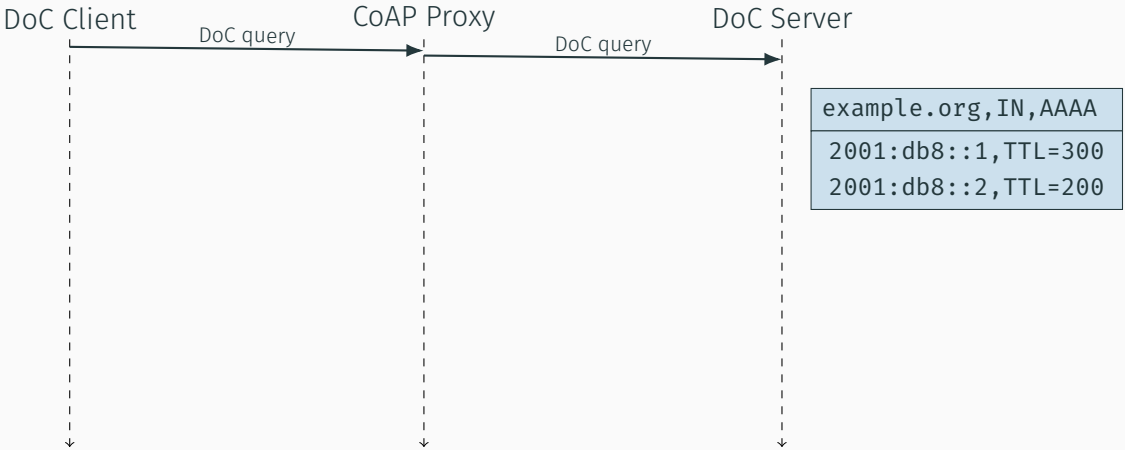


Caching and Max-Age vs DNS TTL (Option 1, DoH-like)

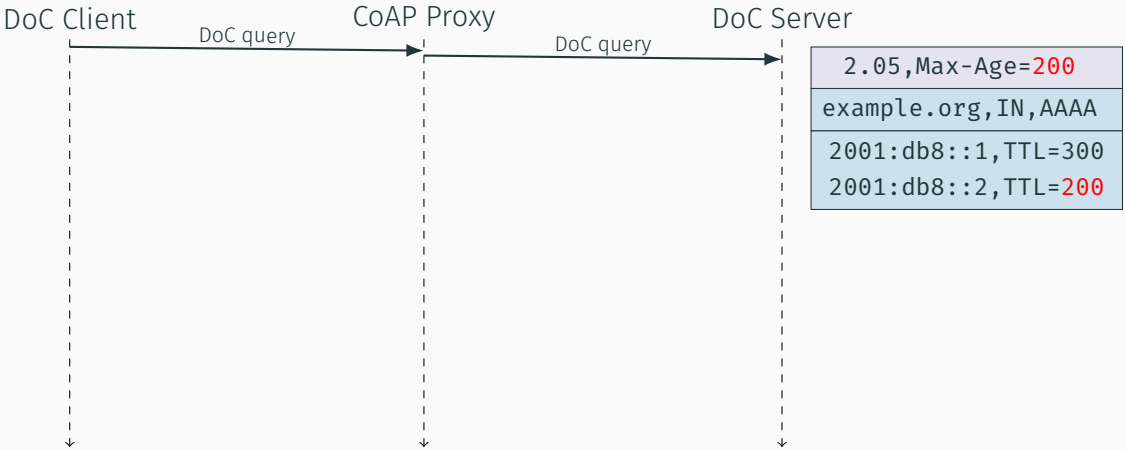


Mostly trying to stay compatible with DoH

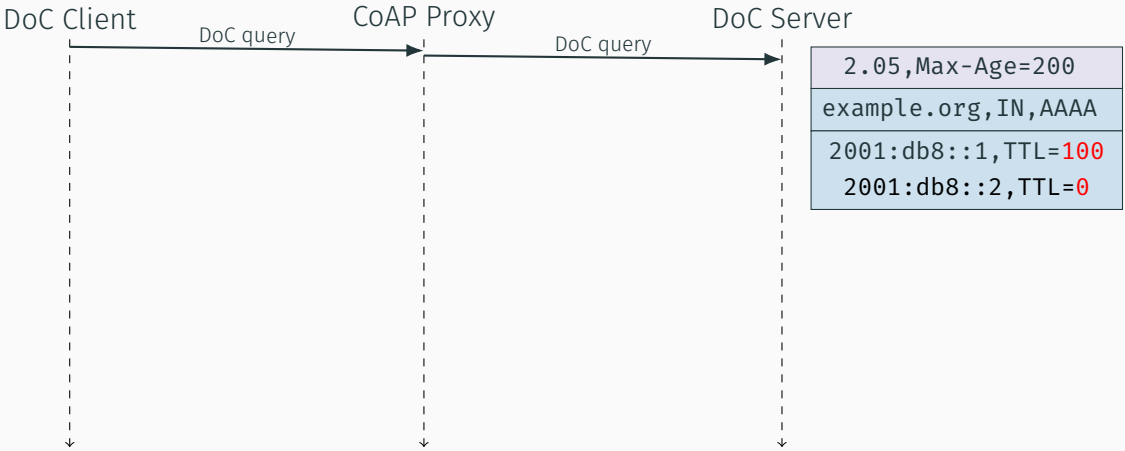
Caching and Max-Age vs DNS TTL (Option 2, adapt TTLs)



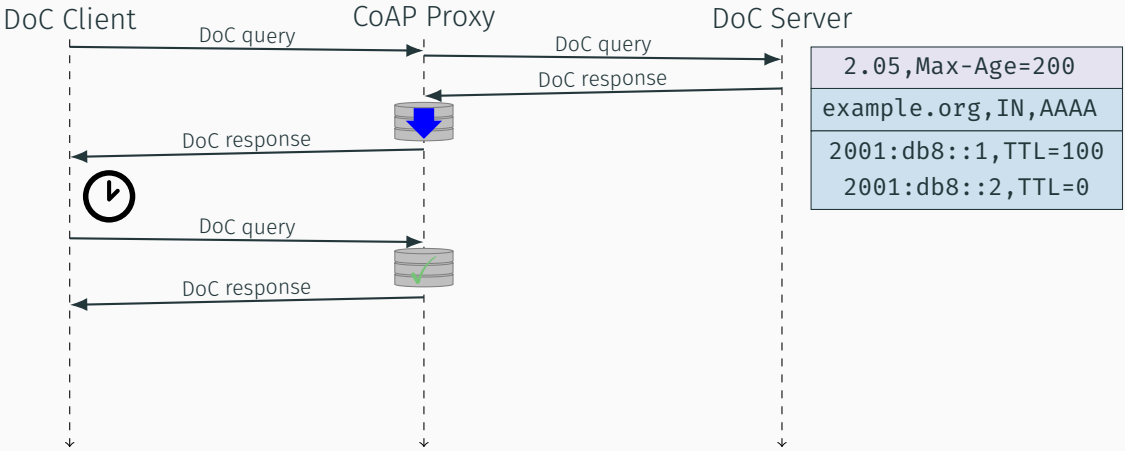
Caching and Max-Age vs DNS TTL (Option 2, adapt TTLs)



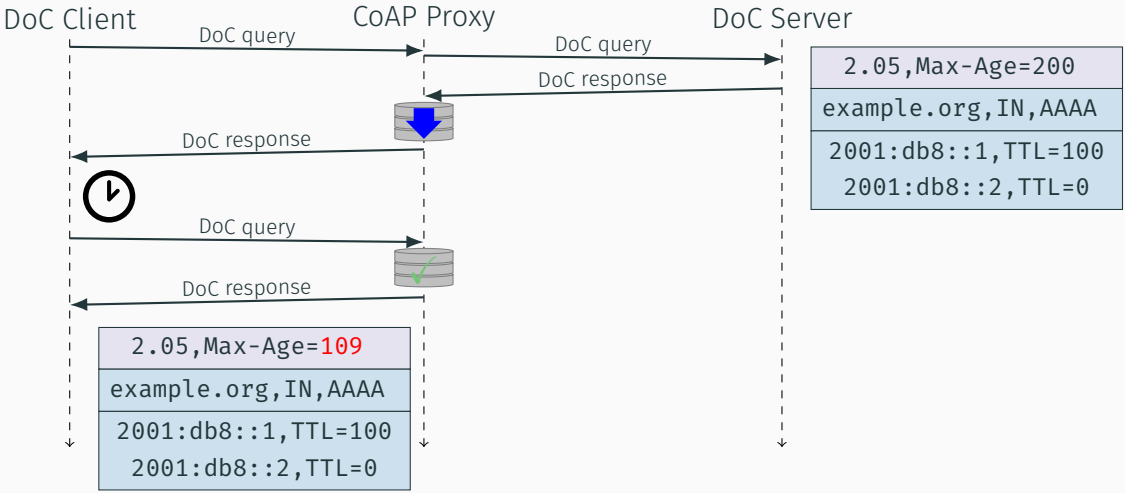
Caching and Max-Age vs DNS TTL (Option 2, adapt TTLs)



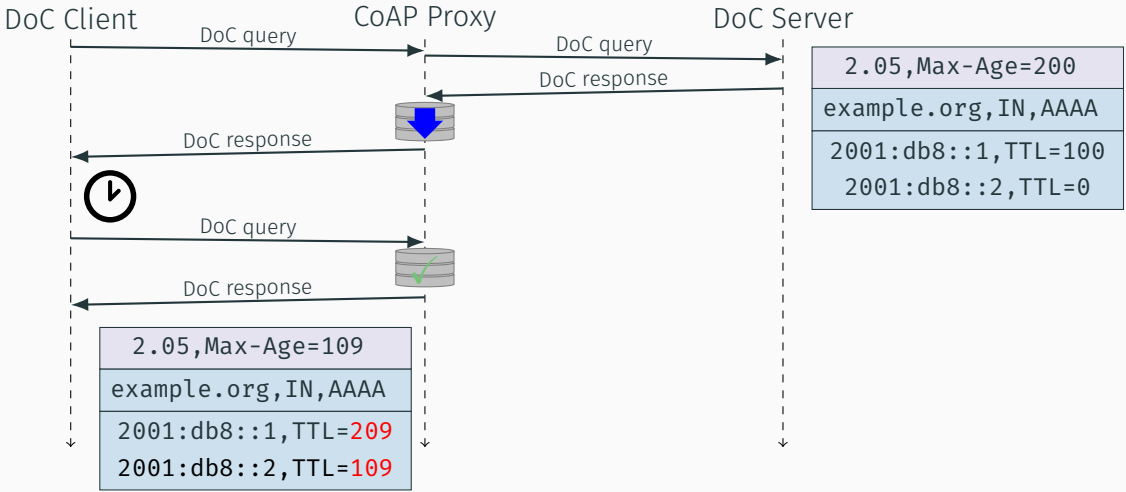
Caching and Max-Age vs DNS TTL (Option 2, adapt TTLs)



Caching and Max-Age vs DNS TTL (Option 2, adapt TTLs)



Caching and Max-Age vs DNS TTL (Option 2, adapt TTLs)



Workload mostly at server + less cache invalidation

Do we need to account for OBSERVE/Server Push?

Section 5.3:

- RFC 8484 (DoH), section 4.3: considerations on HTTP/2 Server Push
  - Deliver potential next request (*e.g.*, website for queried domain name) to client together with DNS response
  - With CoRE: *e.g.*, deliver `.well-known/core` content of CoRE-RD?
  - Requires CoAP request info in notification for proper caching
- Other use case for OBSERVE: RFC 8490, DNS Stateful Operations?

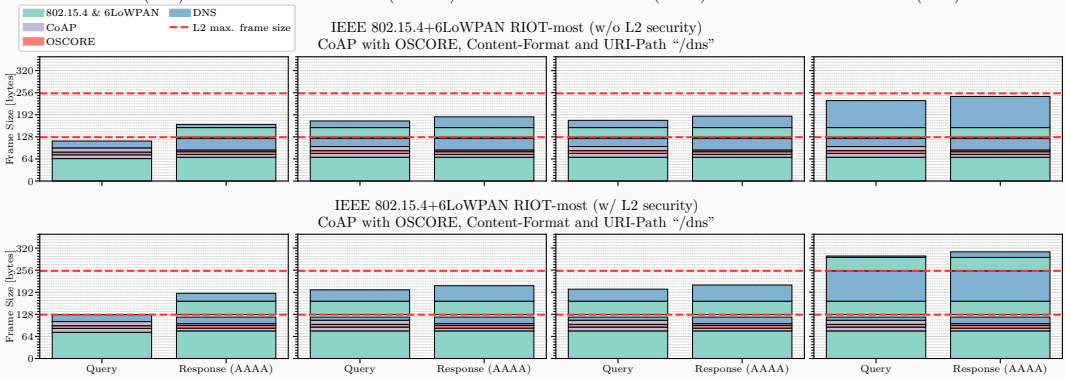
How abstract should the draft be? CoAP vs. REST

Issue #18 by Klaus Hartke proposes

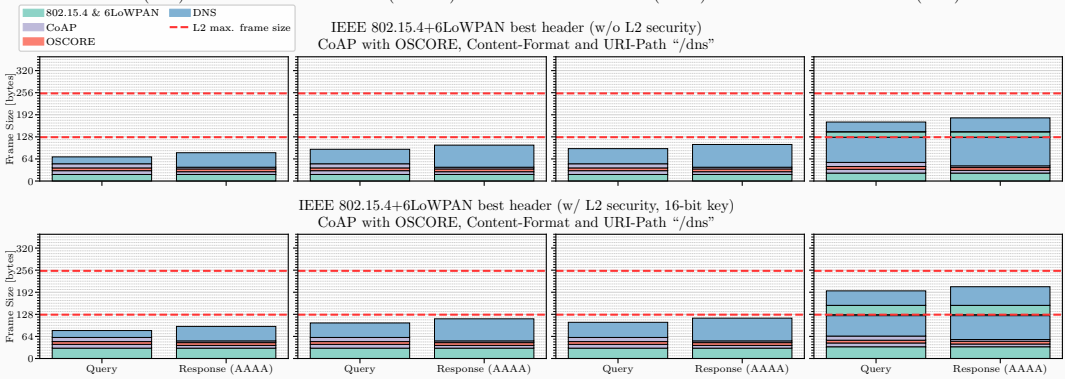
- Specify REST API to retrieve DNS information from CoAP server instead
- Leave protocol details to implementation

Backup slides

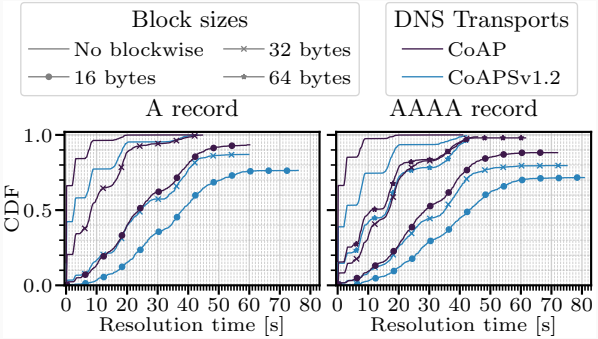
Packet sizes by layer



Packet sizes: Best case L2 headers



Block-wise transfer



- RFC 7959 only
- Not yet looked into RFC 9177