If you cite this paper, please use the CoNEXT reference: M.S. Lenders, C. Amsüss, C. Gündogan, M. Nawrocki, T.C. Schmidt, and M. Wählisch. 2023. Securing Name Resolution in the IoT: DNS over CoAP. In *Proceedings of the 19th International Conference on emerging Networking EXperiments and Technologie (CoNEXT 2023), December 5–8, 2023, Paris, France.*

Securing Name Resolution in the IoT: DNS over CoAP

Martine S. Lenders
Freie Universität Berlin
Berlin, Germany
m.lenders@fu-berlin.de

Christian Amsüss
Vienna, Austria
christian@amsuess.com

Cenk Gündogan
Huawei Technologies
Munich, Germany
cenk.gundogan@huawei.com

Marcin Nawrocki
Freie Universität Berlin
Berlin, Germany
marcin.nawrocki@fu-berlin.de

Thomas C. Schmidt
HAW Hamburg
Hamburg, Germany
t.schmidt@haw-hamburg.de

Matthias Wählisch
Freie Universität Berlin
Berlin, Germany
m.waehlisch@fu-berlin.de

ABSTRACT

In this paper, we present the design, implementation, and analysis of DNS over CoAP (DoC), a new proposal for secure and privacy-friendly name resolution of constrained IoT devices. We implement different design choices of DoC in RIOT, an open-source operating system for the IoT, evaluate performance measures in a testbed, compare with DNS over UDP and DNS over DTLS, and validate our protocol design based on empirical DNS IoT data. Our findings indicate that plain DoC is on par with common DNS solutions for the constrained IoT but significantly outperforms when additional standard features of CoAP are used such as caching. With OSCORE, we can save more than 10 kBytes of code memory compared to DTLS, when a CoAP application is already present, and retain the end-to-end trust chain with intermediate proxies, while leveraging features such as group communication or encrypted en-route caching. We also discuss a compression scheme for very restricted links that reduces data by up to 70%.

CCS CONCEPTS

• **Networks** → **Network protocol design**; Application layer protocols; *Security protocols.*

ACM Reference Format:
Martine S. Lenders, Christian Amsüss, Cenk Gündogan, Marcin Nawrocki, Thomas C. Schmidt, and Matthias Wählisch. 2023. Securing Name Resolution in the IoT: DNS over CoAP. In *Proceedings of the 19th International Conference on emerging Networking EXperiments and Technologie (CoNEXT 2023), December 5–8, 2023, Paris, France.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

CoNEXT 2023, December 5–8, 2023, Paris, France

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The Internet of Things (IoT) deployment extends from simple environmental sensors, industrial monitoring and control to various consumer-grade products, such as home cameras and Smart-TVs. Most IoT operations require frequent access to data or (cloud-)services, commonly addressed by names. Unprotected name resolution from such devices—often in long range undisclosed radio networks—raises concerns regarding security and privacy, as names may carry contextual semantics that enable fingerprinting for third-parties. The system-wide implementation of a vulnerable protocol such as DNS over UDP also opens IoT deployments to large-scale botnet creation. Hence, protecting name resolution in the IoT plays a central role.

IoT devices are often constrained. These *things* commonly interconnect wirelessly and remain independent of the power grid—they typically operate on batteries or harvest energy from the environment. Hardware platforms are kept simple to prolong operation times and reduce unit costs. Even the most powerful devices according to the common IETF classification [8] show orders of magnitude less memory than general-purpose hardware platforms (see Table 2). These devices still require protected name resolution to find, *e.g.,* cloud services [16, 60], possibly via long-range radio communication through untrusted gateways, *e.g.,* [15].

Protecting the name resolution of DNS strengthens privacy and security [71]. Common uses of DNS on top of encrypted transport, though, *e.g.,* DNS over HTTPS (DoH) [27], DNS over TLS (DoT) [29], or DNS over QUIC (DoQ) [31], conflict with low hardware resources of constrained class 1 or class 2 devices. DNS over DTLS (DoDTLS) [66] does not provide means for message segmentation dictated by the small link layer SDUs of constrained networks.

Table 1: Comparison of DNS transport features. The focus of this paper is highlighted in bold.

	DNS over								
Feature	UDP	TCP	DTLS	TLS	QUIC	HTTPS	**CoAP**	**CoAPS**	**OSCORE**
Message Segmentation	✘	✓	✘	✓	✓	✓	✓	✓	✓
Message Authentication	✓	✓	✓	✓	✓	✓	✓	✓	✓
Message Encryption	✘	✘	✓	✓	✓	✓	✘	✓	✓
Message Format Multiplexing	✘	✘	✘	✘	✘	✓	✓	✓	✓
Shares protocol with application	✘	✘	✘	✘	✘	✓	✓	✓	✓
Suitability for Constrained IoT	✓	✘	✓	✘	✘	✘	✓	✓	✓
Content Secure En-route Caching	✘	✘	✘	✘	✘	✘	✘	✘	✓

In this paper, we present DNS over CoAP (DoC), a secure and privacy-friendly DNS resolution protocol for the constrained IoT. The Constrained Application Protocol (CoAP) [65] was standardized by the IETF as a lightweight IoT alternative to HTTP and is widely available. CoAP is based on UDP but provides transactional message contexts, reliability retransmission functions, and en-route caching on dedicated forward proxies. Security extensions either use Datagram Transport Layer Security (DTLS) [52] or Content Object Security for Constrained RESTful Environments (OSCORE) [63]. DoC leverages these security extensions to query the DNS privately, securely, and yet efficiently enough to comply with the low-end IoT.

Designing and implementing DNS for the constrained IoT is more challenging than DoH when embedded into common IETF protocols:

(1) Common DNS answers are large and lead to packet fragmentation, which should be avoided on lossy IoT links.
(2) New CoAP methods offer differing features and tradeoffs.
(3) En-route caching has more importance in CoAP to decouple lossy IoT links from content delivery [22] and fragmentation avoidance benefits from the cache validation mechanism of CoAP [65]. This prevents a simple re-use of the purely client-based DoH caching approaches.

A feature comparison of the different DNS transports, which highlights the achievements of our work, is shown in Table 1. In summary, the main contributions of this paper read:

(1) We analyze the impact of IoT naming on the design of DNS resolution based on an empirical data set of characteristic IoT domain names. We compare with names requested via a large regional Internet eXchange Point (IXP). (Section 3)
(2) We design the DoC protocol, which efficiently embeds the DNS semantics into the rich feature set of the CoAP protocol suite to enable end-to-end protection, block-wise transfer, group communication, caching, as well as an option for compression. (Section 4)
(3) A system-level analysis conducted on real IoT hardware using key properties gathered in Section 3 reveals that DoC performance is at least on par with generic UDP-based DNS transport. Additional features increase the DoC performance further. (Sections 5 and 6)
(4) We discuss the utility of a potential new media type to transport DNS messages over DoC or DoH. (Section 7)

Table 2: Constraints of DoC target platforms [8].

Memory	Class 0	Class 1	Class 2
RAM [kBytes]	$\ll 10$	≈ 10	≈ 50
ROM [kBytes]	$\ll 100$	≈ 100	≈ 250

The remainder of this paper is structured as follows. After exploring the problem space and the related work in Section 2, we present our contributions in Sections 3 to 6, discuss further potential for DoC in Section 7, and conclude with an outlook in Section 8.

2 THE PROBLEM OF NAME RESOLUTION IN THE IOT AND RELATED WORK

2.1 The Need for Secure Name Resolution

DNS Resolvers on IoT Devices. The DNS serves as an indirection to reach IP endpoints, enabled by a DNS (stub) resolver at Internet nodes. Omitting the resolver on the IoT device would introduce several drawbacks. First, IP addresses to access Internet services, *e.g.,* cloud backends, need to be preconfigured on the IoT device. This reduces flexibility and adds additional burden, if preconfigured addresses need changing. Second, offloading the name resolution to more powerful border routers (or gateways) would require application-specific deep packet inspection on the gateway or an additional name resolution protocol between gateway and IoT device. We argue that a gateway should remain as transparent as possible and that DNS resolution should be interoperable throughout the global Internet. Finally, we emphasize that unprotected DNS over UDP is already available in popular IoT operating systems, ARM Mbed [44], RIOT [14], and Zephyr [51].

Threats to End User Privacy. Meta-data about communication can leak sensitive information such as sleeping habits [5]. One such meta-data are the hostnames IoT devices resolve [49, 53, 59]: Names provide application- or service-specific information. Plain DNS queries concurrent to (protected) application traffic may disclose the context of confidential communication, reveal behavioral patterns, or uncover hints for fingerprinting victims [41]. In this paper, we

close the gap in privacy-friendly DNS resolution [42] by designing and analyzing a lightweight protocol that obfuscates DNS query and response, built on emerging IoT standards.

Threats to Infrastructure Security. Leaked information, may it be personal or not, is a security risk by itself: An attacker can use it to infer knowledge about network and application and plan attacks accordingly. The IoT, however, repeatedly takes center stage in large-scale distributed Denial of Service (DDoS) attacks against the Internet infrastructure. The Mirai botnet, *e.g.,* caused 600 GBytes/s of incoming traffic, taking down commercial servers in 2016 [4, 25].

DNS rebinding attacks are ways to redirect a requester to malicious sites, which then can deploy malware on a device [16, 37, 58, 69]. Securing the name resolution with encryption helps mitigate such attacks and provides another factor of name authentication directly on the end device [26].

2.2 Challenges from the Constrained IoT

The use of DNS over TCP [17] mitigates DNS-based DDoS attacks, but faces a limited support by many resolvers [43]. TCP and by extension TLS introduces an increased transport complexity and while it can be deployed in some constrained scenarios [36], it is not suitable for very restricted link-layer technologies that exhibit small packet sizes and long duty-cycles, such as LoRaWAN [20]. DoT [29], DoH [27], and most recently DoQ [31] are mechanisms to protect the confidentiality and integrity of DNS traffic on the Internet. They employ transport layer security and maintain a session state between two endpoints to prevent IP spoofing. The first two approaches build on TCP and their performance significantly drops when network conditions degrade [28]. This property conflicts with constrained networks where links commonly saturate. DoQ uses UDP, but despite its performance advantages over DoT and DoH [35], deployment in low-power regimes is challenging due to its use of TLS [54].

DoDTLS [52] is an alternative that also runs on datagram transport. In addition to the lower protocol complexity compared to the former approaches, this transport does not suffer from head-of-line blocking, which frequently occurs in constrained networks. Nevertheless, DoDTLS faces issues with larger messages exceeding the Path Maximum Transmission Unit (PMTU) [52] and forces applications into fragmentation. Moreover, IoT link layers, such as IEEE 802.15.4 or LoRaWAN, often support only a few hundred bytes [32, 40]. This limit is easily reached by rather small DNS queries or responses as we will show later. Even though adaption layers for IPv6, *e.g.,* 6LoWPAN [48] or Static Context Header Compression and Fragmentation (SCHC) [19], offer fragmentation between the link and network layer, they introduce higher packet loss and latency [38].

Despite the advantages of DoDTLS, there are certain drawbacks with protecting the transport in the IoT use case [23]. (*i*) IoT networks may connect to gateways that bridge between transports, *e.g.,* between UDP and TCP. This burdens the end-to-end protection, since gateways need to be included in trust relationships to re-encrypt the data between endpoints. (*ii*) A loose coupling and caching are favored techniques in the IoT to deal with mobility and network partitioning, but the established security sessions are deeply rooted on the transport and harden the endpoint paradigm. (*iii*) Long duty-cycles in lossy, constrained networks as in LoRaWAN conflict with the handshake requirements of DTLS.

While current standardization efforts for DTLS, *e.g.,* Connection Identifiers [56], help to address the drawbacks, there is another undertaking in the IETF CoRE working group to provide secure communication. OSCORE [63] protects messages with encryption on the object-level instead of the transport-level. It fully integrates with the CoAP [65] ecosystem, ensures end-to-end protection across gateways, allows for a protected multiparty communication, and makes encrypted and authenticated CoAP messages cacheable on untrusted proxies. In addition, OSCORE provides better performance compared to CoAP over DTLS [24].

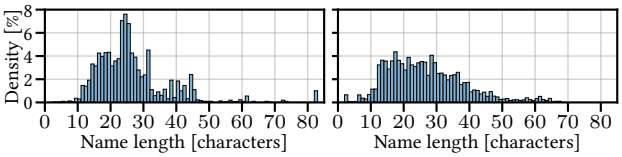
CoAP was designed as the HTTP for the IoT, and is thus the candidate for protective measures analog to DoH. The CoAP ecosystem facilitates power-efficient, privacy-friendly DNS queries in the IoT and mitigates the size of DDoS attacks through bandwidth reduction and peer authentication.

3 EMPIRICAL VIEW ON IOT DNS TRAFFIC

In this section, we motivate our design decisions. To this end, we empirically analyze DNS traffic produced by end-consumer IoT devices and compare to flow samples from a large regional European IXP.

3.1 Data Corpus

Identifying IoT-specific traffic is challenging. We rely on data from three common projects, YourThings [2], IoTFinder [49], and MonIoTr [53], all collected throughout 2019, which captured and annotated IoT traffic based on ground truth. YourThings and IoTFinder also provide traffic from desktop computers, phones, tablets, gaming consoles, and Wi-Fi access points. We exclude such traffic. All three IoT data sets include both unicast DNS and multicast DNS (mDNS) [13] traffic. As mDNS is integral to DNS Service Discovery (DNS-SD), which is used in many use cases for IoT device discovery, we keep mDNS traffic. Overall, the aggregation of all three data sources include data from over 90 consumer-grade IoT devices by more than 50 manufacturers and contains 0.2 million



(a) IoT devices **(b) Internet devices**

Figure 1: Distribution of name lengths; names queried by different devices connected via the Internet.

DNS and mDNS queries and 1.3 million corresponding responses in total. The IoTFinder data only contains responses but we can infer the queries from their question section.

To compare IoT-specific DNS traffic with DNS traffic from common Internet devices, we leverage sFlow [50] samples collected in January 2022 at a European IXP. Our IXP data contains 1.6 million unicast DNS queries and 2.4 million responses and is based on a sampling rate of 1/16000 packets. Packets are truncated to 128 bytes. For privacy regulation compliance, we strip all names and replace them with the target analysis data (*e.g.,* name lengths) before exporting them for our analysis. Before stripping, we confirmed that no side effects on the name lengths were introduced due to query name minimization [11].

3.2 Results

How long are names requested by IoT devices? Figure 1 shows a normalized histogram over all queried names seen, for both the IoT data sets (Figure 1a) and the IXP data set (Figure 1b). The statistical key properties for the name lengths of each data set and the aggregated IoT data set are shown in Table 3.

The median of the name lengths is 23 or 24, depending on the IoT data set, which is confirmed by the median of 24 of the IXP data set. Many cloud and CDN names, such as `e123.abcd.akamaiedge.net` (name modified for sake of privacy), gather around this name length. Significantly longer names are used for certain mDNS applications, *e.g.,* for reverse DNS or to identify local devices via a UUID. As such, we do not see these longer name lengths at the IXP. With a mean of 24 in name length 18.8% of 128 bytes of the IEEE 802.15.4 link layer PDU is already taken up by the name on average. LoRaWAN can have a link layer PDU of only 59 bytes, *i.e.,* the name would take 40.7% of the available space.
What kind of records are requested? Table 4 presents the relative ratio of the queried record types available in the `IN` class based on our data sets. `A` records are in all data sets the most requested records, with `AAAA` records being close second. With growing deployment of IPv6, these numbers will change in favor of more `AAAA` queries. When not accounting for mDNS, these are >99% of all records in the IoT. With mDNS we also see more records associated with service discovery, namely `ANY`, `PTR`, `SRV`, and `TXT` records [12, 13].

Table 3: Statistical key properties of domain names queried by IoT devices compared to domain names visible at an IXP. μ denotes the mean, σ the standard deviation, Q_1 the first quartile, Q_2 the second quartile (or median), and Q_3 the third quartile.

Data source	Lengths of Domain Names [chars]							
	min	max	mode	μ	σ	Q_1	Q_2	Q_3
YourThings [2]	2	83	31	24.5	9.7	18	24	30
IoTFinder [49]	7	82	24	26.8	10.5	20	24	30
MonIoTr [53]	9	83	18	27.1	14.7	18	23	30
IoT total	2	83	24	25.9	11.3	19	24	30
IXP	0	68	17	26.1	11.7	17	25	33

Table 4: Queried record types in `IN` class.

	Record Type	`A`	`AAAA`	`ANY`	`HTTPS`	`NS`	`PTR`	`SRV`	`TXT`	`Other`
IoT	w/ mDNS	53.6%	16.4%	8.2%	—	—	19.6%	1.0%	1.2%	<0.1%
	w/o mDNS	75.8%	23.5%	—	—	—	0.3%	—	0.1%	0.3%
	IXP	64.5%	17.6%	1.7%	9.1%	0.7%	1.8%	0.4%	0.7%	3.5%

Besides a great number of other less requested records (3.5% in total), we mostly see `A` and `AAAA` records at the IXP. The largest remaining part are `PTR` and `HTTPS` [62] records.
Other DNS data of interest. We also analyzed the number of entries in each DNS section and the overall response lengths. In response sections, we find events that include more than 255 entries—the overflow point of the count fields into 2-byte numbers—but the percentage is low and mostly relate to mDNS. These large numbers originate from unrequested `NS` records that advertise name servers in the authority section and the associated `A` or `AAAA` records for these advertised name servers in the additional section. Providing these optional data seems to be common practice by cloud and CDN providers. The question section, on the other hand, always contains only 1 entry. This complies with common resolver behavior to ignore queries or cause an error when they include a question section with more than 1 entry.

Responses can become very long, containing 400 to 600 bytes, in certain cases even more than 1 kByte, even for the IoT devices. This is caused by the long authority and additional sections as described above.
Lessons learned. Our analysis revealed that long names with a median of 24 characters are a given, even in IoT scenarios. Without a dedicated DNS message compression this can lead to fragmentation in constrained IoT scenarios. While not part of the main design of DoC, it offers the opportunity to use highly compressed message formats compared to the DNS wire format via the Content-Format option of CoAP.

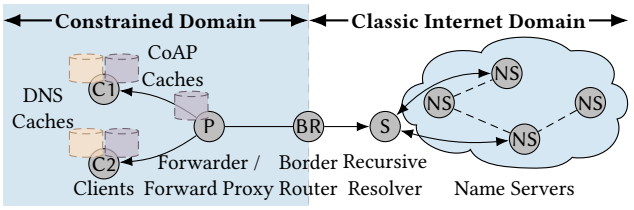


Figure 2: Typical deployment of the DNS over CoAP (DoC) architecture. DoC protects name resolution between the IoT devices and the recursive resolver.

IoT devices mainly query addresses for names with only a significant number when using DNS-SD. Group OSCORE [68] may offer an encrypted solution for multicast DNS-SD.

In constrained IoT use cases, the authority and additional sections must only be provided if necessary. Unsolicited NS records serve little purpose in a constrained environment.

4 DESIGN OF DNS OVER COAP (DOC)

In this section, we define DNS over CoAP (DoC), a protocol to query the domain name system and retrieve responses over the Constrained Application Protocol [65] (see Figure 2). The goal is to ensure message integrity and confidentiality by mapping each DNS query-response pair to a CoAP message exchange, secured on the transport via DTLS [55] or on the object level via OSCORE [63]. DoC forms thus a thin layer between CoAP and DNS, comparable to DoH [27] which maps DNS message pairs into HTTP message pairs.

Using CoAP for DNS resolution provides the following advantages in the constrained IoT: (*i*) CoAP runs on UDP. UDP does not introduce additional connection setup nor state. CoAP provides optional reliability using a simple mechanism of acknowledgments and retransmissions. (*ii*) Proxy operations and en-route response caches decouple packet losses from content delivery in wireless networks [22]. (*iii*) The Content-Format option of CoAP provides potential for future, compressed DNS messages to reduce fragmentation on the link layer. (*iv*) CoAP provides block-wise transfer to fragment and reassemble large messages on the application layer, which are likely in DNS as seen in Section 3.

4.1 Protocol Overview

For a basic message exchange, mapping DNS queries to CoAP requests and DNS responses to CoAP replies is necessary.

Request mapping. A DoC client can send a DNS query over CoAP by embedding the DNS wire format of a DNS query into a CoAP message using either GET, POST, or FETCH [70] requests, each of them provide different features.

GET and POST are also supported by DoH and thus their behavior could be mapped to DoC: Using GET, the DNS query needs to be encoded within the URI. As such, a DoC resource needs to be configured as a *URI template* [21], providing the position of the DNS query in the URI as a variable. GET allows for caching of subsequent responses but prevents block-wise transfer and requires a *URI template processor* to resolve the DNS query variable at the constrained client side. POST, on the other hand, carries the DNS query in the CoAP body, reducing complexity from an additional URI template processor. As a drawback, though, it does not allow for caching since the payload of the request is not taken into account for a cache key. To allow for both caching and block-wise transfer, a DoC client can use FETCH [70]. FETCH is currently not supported by all CoAP implementations, but extending them is trivial. As such, compared to DoH, where GET and POST are used primarily, FETCH is the preferred method for DoC. Table 5 displays the different benefits and drawbacks of the three CoAP methods.

Table 5: Comparison of request methods considered for DoC.

	GET	POST	FETCH
Cacheable	✓	✘	✓
Application data carried in body	✘	✓	✓
Block-wise transferable query	✘	✓	✓

Response mapping. A DoC server sends a DNS response over CoAP by encoding the wire format of the DNS response in the payload of a CoAP response.

4.2 Response Caching

To decouple packet losses from content delivery in lossy, constrained network, three challenges for efficient utilization of a CoAP cache need to be tackled: (*i*) Consistency of the CoAP *cache key* equivalent DNS queries. This key is used to determine the existence of cached response copies. (*ii*) Alignment of the CoAP response caching time with DNS record times to live (TTLs) such that a DoC client does not get outdated data or triggers unnecessary data delivery. (*iii*) Leveraging of CoAP cache validation to reduce the number of large DNS response transmissions on cache timeout.

Consistent cache keys. When using CoAP FETCH or GET, the original DNS message becomes part of the cache key, either because the key includes the payload (FETCH) or the URI (GET). Since any DNS message carries an ID in the DNS header, which might be different for multiple queries of the same resource record and hostname, we propose to set this ID to 0 for either encrypted CoAP mode. This yields a deterministic wire format without introducing additional states at the client side or coordinating this ID between multiple DoC clients. DTLS and OSCORE provide sufficient defense against spoofing attacks for predictable DNS IDs.

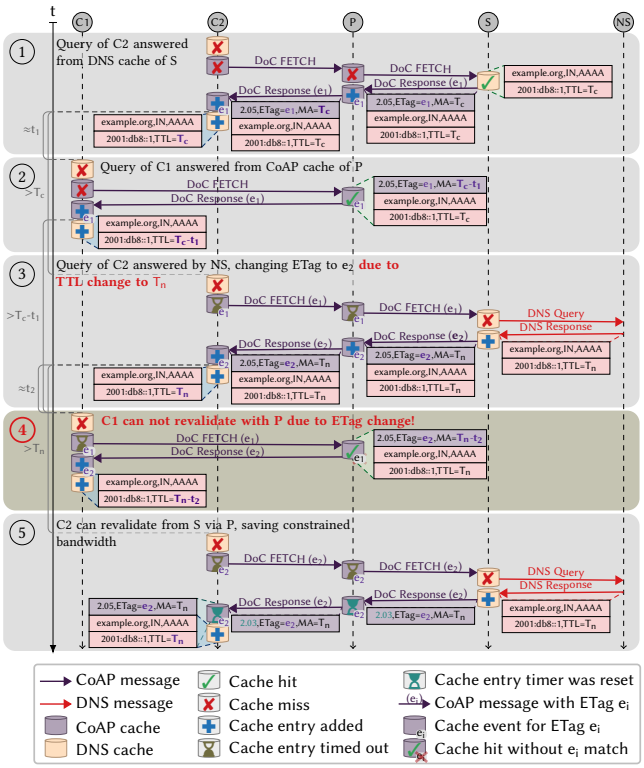


Figure 3: A name resolution using the *DoH-like* caching scheme in DoC. DoC clients C1 and C2 query `AAAA` records for `example.org` from a DoC server S via a DoC-agnostic CoAP proxy P ① ②. Only when a DNS cache miss at S occurs, the DNS name server NS is queried ③. A cache hit at P for a query from C2 adapts the TTL, causing cache re-validation to fail and to resend all information ④. Only C1 can leverage cache validation immediately just after the Max-Age (MA) update ⑤.

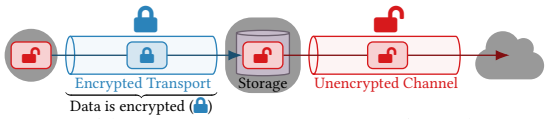
Aligning expiration timers. TTLs in DNS responses describe how long a resource record should stay in a DNS cache. They are actively maintained by all DNS caches and reflect the remaining time in a DNS response from a cache. When embedded in CoAP messages, TTLs, however, are opaque since DNS responses are treated as any other CoAP payload. DoC clients or proxies thus face two cases that impact the protocol efficiency when cached responses are retrieved. (*i*) DNS TTLs are expired, although CoAP caches consider them as valid. This leads to outdated name resolutions at DoC clients. (*ii*) Cached CoAP responses are expired, while the DNS TTLs are still valid. This leads to a reduced cache utilization and unnecessary network overhead.

Leveraging cache validation. A CoAP server may include an entity-tag (ETag) in a response to differentiate between response representations. Then, client or proxy caches use ETags to query for the validity of timed out cache entries. If a cached object is still valid, the server transmits a *small* confirmation message using the 2.03 response code to reduce the network load. After the payload of the entry changed, *i.e.,* the ETag changed, the server transmits the full response. For our evaluation, we consider two approaches to align the DNS record lifetimes with the CoAP response caching and validation model: *DoH-like*, which is based on the caching recommendations for DoH [27] as baseline and *EOL TTLs*, our improvement. *EOL TTLs* is able to leverage CoAP cache validation by setting all TTLs to 0, *i.e.,* their End of Live (EOL).

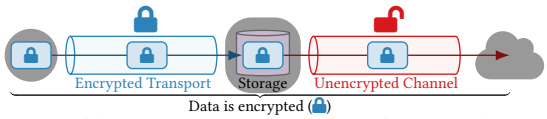
Option 1: DoH-like. This approach strictly follows RFC 8484 [27]: A DoC server sets the Max-Age option of CoAP to the minimum TTL of encapsulated DNS resource records. The Max-Age value decreases on intermediate CoAP caches. DoC clients that receive a cached response use the altered Max-Age to reduce TTLs of included resource records. A drawback of this approach is that changing DNS TTLs, either due to DNS caches or DNS operators, changes the CoAP payload and thereby affects the ETag generation. This results in failing cache re-validations and requires a full DNS response, even though only a TTL changed (see Figure 3). In common DoH deployments such overhead is of not much concern but reduces performance in the IoT (see Section 6 for details).

Option 2: EOL TTLs. Steps ③ and ④ in Figure 3 illustrate the problem of the *DoH-like* approach: Due to a change to the TTL values from the DNS cache infrastructure, the cache validation model of CoAP can be undermined. *EOL TTLs* improves the previous option by increasing success rates of cache re-validations. In detail, we propose that a DoC server sets the Max-Age CoAP option to the minimum TTL of the resource records in the DNS response and rewrites all DNS TTLs to 0.

These modifications always ensure identical ETag values for the same resource record set. During name resolution, such responses may be stored at intermediary caches, *e.g.,* on a proxy or on a client. For later requests that result in cache hits, Max-Age values are adjusted according to the CoAP freshness model. When a response arrives at the DoC client, it copies the Max-Age into the resource records to restore the correctly decremented TTL values when making them available to the local DNS system cache. Requests that hit stale cache entries trigger a cache re-validation. If only TTLs changed, then the DoC server validates the cache entries, and encodes the new TTL in the Max-Age option, leveraging the advantage for step ④ in Figure 3 that *DoH-like* was only able to take in step ⑤.



(a) Transport Layer Security (DTLS)



(b) Content Object Security (OSCORE)

Figure 4: The security modes available with CoAP.

4.3 Security Modes

The use of CoAP as transport for DNS enables two security modes to encrypt name resolutions: (*i*) Transport layer security (see Figure 4a) or (*ii*) content object security (see Figure 4b). Both modes can also be used in combination.

Transport Layer Security. CoAP over DTLS (CoAPS) [55] exhibits similar protocol behavior and security guarantees (*i.e.,* confidentiality and integrity) as TLS [54], and further contributes a modified record layer that tolerates message reordering and packet loss. One key advantage of this mode compared to DoDTLS without CoAP is the block-wise transfer to fragment large DNS queries and responses into smaller CoAP messages. This solves the problem that DoDTLS is facing: The datagram-based DTLS cannot deal with packets that exceed the PMTU but relies on negotiating a maximum response size during name resolution [52].

DTLS has two drawbacks impairing DoC, though: First, without any cross-layering, each CoAP retransmission needs to be re-encrypted and re-authenticated before delivery—retransmissions of the encrypted datagram may be rejected by the duplicate detection of the peer. Second, CoAP proxies and intermediary caching nodes must be included in the trust relationship to process CoAP messages.

Content Object Security. OSCORE [63] is a CoAP protocol extension addresses the drawbacks of DTLS. Instead of securing transport sessions between endpoint pairs, it provides integrity, authenticity, and confidentiality on an object level by protecting *entire* CoAP messages. OSCORE transforms the original CoAP message into an authenticated and encrypted CBOR Object Signing and Encryption (COSE) [61] object, and encapsulates it as a CoAP option in an outer, newly created CoAP header which only exposes the request-response mapping token, message-layer ID and the unprotected parts of the OSCORE option.

To mitigate mismatch and replay attacks, OSCORE constructs a strong message binding between requests and corresponding responses with the use of identical identifiers in their authenticated components, which persist over retransmissions. This also leads to less computational effort for encryption and authentication, since, compared to CoAP over DTLS, the OSCORE protected messages are stored in the CoAP retransmission buffer.

To enable caching on untrusted nodes, a protocol add-on for OSCORE is currently discussed [3], thus ensuring end-to-end security via third-party gateways. Likewise, there is a proposal to allow for protected group requests and responses for one-to-many communication [68].

OSCORE initially relies on pre-shared keys or preconfigured certificates. DTLS provides a built-in key exchange protocol to establish temporary session keys between two endpoints. This enables perfect forward secrecy: Leaked keys cannot be used to decrypt past correspondences. A lightweight authenticated key exchange for OSCORE is under development, though: Ephemeral Diffie-Hellman over COSE (EDHOC) [64].

5 COMPARISON OF LOW-POWER DNS TRANSPORTS

In this section, we evaluate memory usage, packet sizes, and resolution times of DoC and compare with DNS over UDP and DNS over DTLS in different communication setups. Our DoC configurations include the unencrypted use, CoAPS, and OSCORE, using the FETCH, GET, and POST methods.

5.1 Setup

Hardware and software platform. We conduct our experiments in the FIT IoT-LAB testbed [1], which supports a variety of IoT hardware environments. We choose nodes from the *Grenoble* site, since the physical stretch makes it a good candidate for multi-hop measurements. Our platform features a Cortex-M3 MCU with 64 kBytes of RAM, 512 kBytes of ROM [67], and an IEEE 802.15.4 radio [46]. The radio is configured to automatically handle link layer retransmissions and acknowledgments.

As the base of our experiments, we use RIOT (`2022.07`), which includes DNS over UDP, We provide extensions to support DNS over DTLS and DNS over CoAP. For details about the implementations, we refer to Appendix A). We stress-test each of the deployments by using asynchronous protocol features that allow for concurrently pending queries on a device. We modify a few RIOT configuration parameters to accommodate the number of queries in the air, specifically internal queue sizes to hold multiple packets.

Since DTLSv1.2 [55] still enjoys a wider deployment at the time of evaluation, we choose that version in our evaluations over 1.3 [57]. For consistent measurements, we pre-initialize DTLS sessions and OSCORE replay windows on all endpoints before starting experiments. To prevent side

effects such as lost requests or prolonged timeouts due to session re-initialization, we increase both the DTLS session timeout and the OSCORE replay window size. This proved useful when measuring the protocol effects during long experiment runs. To make the 6LoWPAN implementations of RIOT (clients) and Linux (resolver) more comparable, we deactivate the stateful address compression and set the traffic class and flow label IPv6 header fields to `0`, so they are elided.

In all experiment runs, we measure the actual name resolutions within the IoT network, and exclude the resolution times to external DNS servers.

Topology description. We construct a topology with two wireless hops as in Figure 2 for two DNS clients communicating with a DNS recursive resolver via a forwarder and a border router. The forwarder is either configured as an opaque IPv6 router, or as a CoAP forward proxy with caching capabilities. The border router node is of the same hardware as the DNS clients and the forwarder. It further connects to the host machine of the DNS resolver via Ethernet that is encapsulated in a TCP-tunneled UART connection. The DNS resolver is a simple Python implementation that uses standard libraries, such as *dnspython* and *aiocoap*, and runs on the SSH frontend. The routes in the wireless domain are constructed using the RPL routing protocol [30].

Protocol settings. We evaluate the following DNS transports (short names in parentheses): (*i*) DNS over UDP (**UDP**), (*ii*) DNS over DTLS (**DTLSv1.2**), (*iii*) DNS over unencrypted CoAP (**CoAP**), (*iv*) DNS over CoAP over DTLS (**CoAPSv1.2**), and (*v*) DNS over OSCORE (**OSCORE**). We assess CoAP and CoAPSv1.2 with the FETCH, GET, and POST methods, for OSCORE we use only FETCH since our DNS over OSCORE implementation does not support GET due to its complexity. With DTLSv1.2 we use the `AES-128-CCM-8` cipher suite [45] and with OSCORE the `AES-CCM-16-64-128` cipher mode [61], as these are the most comparable options. Both pre-shared key lengths are 9 bytes.

Communication setup. We query `A` and `AAAA` records in separate runs for 50 names of length 24 characters each, emulating real world data by choosing the median name length of the empirically analyzed names in Section 3. The name encodes an identifier to track query-response pairs over the different DNS transports, even if the transaction ID is set to 0. The query rate is Poisson-distributed with $\lambda = 5$ queries/s to generate noticeable load on the medium. The recursive resolver is mocked up to generate the desired responses. For the CoAP-based transport, the requested DNS resource is `/dns`. All runs are repeated 10 times.

5.2 Memory Consumption

We first inspect the memory consumption on our target platform for the DNS requester application of each transport.

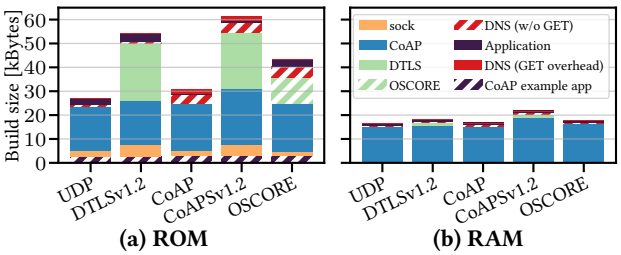


Figure 5: Memory consumption of each DNS transport with existing CoAP example application.

For better comparability, we also add the standard *gCoAP* example of RIOT, providing both server and client functionality, to account for a CoAP application already present on the device. Since the asynchronous request contexts consume a disproportionate amount of RAM compared to the core functions of each protocol, we limit the maximum number of these contexts to one. As our software platform does not use any dynamic memory allocation, we do not consider heap allocation. For measuring ROM requirements, we sum up the respective object sizes in the `.text` and `.data` sections of the RIOT image and for the RAM requirements the `.data` and `.bss` sections. See Appendix D for details.

Figure 5 displays the RAM and ROM consumption for the selected protocols including applications. The encrypted transports add a considerable amount of ROM—about 24 kBytes in the case of DTLS and about 11 kBytes in the case of OSCORE—and in the case of DTLS also about 1.5 kBytes of RAM. Notably, the DTLS part of the firmware expects more than double the memory space of the OSCORE part. This is due to DTLS requiring its own message layer, as well as asymmetric cryptography, to establish a handshake, which is not present in OSCORE.

GET support adds about 2 kBytes of ROM and 173 bytes of RAM to the overall size. About 1 kByte of this ROM contributes the URI template processor. The remainder relates to the different message handling required for the GET request, while the Content-Format option is elided.

These numbers show that for unencrypted transport, UDP remains the clear choice when it comes to memory efficiency. For encrypted DNS communication, DTLS is the most efficient transport solution, with OSCORE being a close second. If CoAP is already present for the application, OSCORE is the most efficient encrypted transport.

The comparably young DNS part for DoC has definite potential for optimization. Currently, it includes parts of the parsing and handling of certain CoAP options and is with around 4 kBytes significantly larger than the other DNS transport implementations. This header handling should be moved to the CoAP part of the firmware in the future to remove possible code duplication with application code.

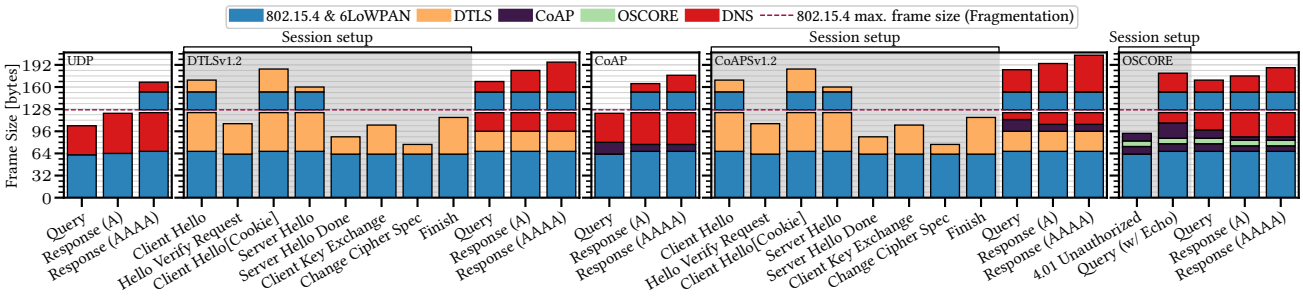


Figure 6: Maximum link layer packet sizes for each transport when resolving a name with a length of 24 characters for a single record (A or AAAA respectively).

5.3 Packet Sizes

We now measure the packet sizes of the DNS messages on all transports by capturing the IEEE 802.15.4 frames using the `sniffer_aggregator` tool of the FIT IoT-LAB testbed. Figure 6 displays the packet dissection for each packet type, segmented per communication layer. Both IPv6 and UDP headers are compressed within the 6LoWPAN header, which we group with the IEEE 802.15.4 MAC header for simplicity. The maximum PDU of IEEE 802.15.4 is marked in each plot by a red dashed line. 6LoWPAN fragments larger IEEE 802.15.4 frames, producing additional MAC and 6LoWPAN headers for each generated fragment. We represent each additional fragment with its headers above the red marker line.

We see three distinct sizes of DNS messages in our experiments. DNS queries requesting either an A or AAAA record from the DNS resolver. These queries are identical in size and only differ in their query types (A vs. AAAA). Respective responses contain either an A or AAAA record, which vary in size due to the IP address lengths.

Figure 6 further includes packet sizes of the DTLSv1.2 handshake and the OSCORE replay window initialization. We observe that DTLS—both with DTLSv1.2 and CoAPSv1.2—is at a disadvantage as the handshake messages alone already cause multiple fragmented datagrams and multiply the likelihood for packet loss during the session establishment.

DNS queries are `base64`-encoded within the GET method. This inflates requests to a size that is approximately 1.5 times larger than binary FETCH or POST queries. As such, with either CoAP-based transport a DNS query using GET will be fragmented for the median name length. Likewise, when AAAA records are requested the response will be fragmented. For CoAPSv1.2, little room is left for the DNS message itself in either message format before reaching the fragmentation limit. The same, however, is also true for OSCORE, if the Echo option required for the replay window initialization is carried in the request. Appendix B adds supplementary dissections for queries using GET.

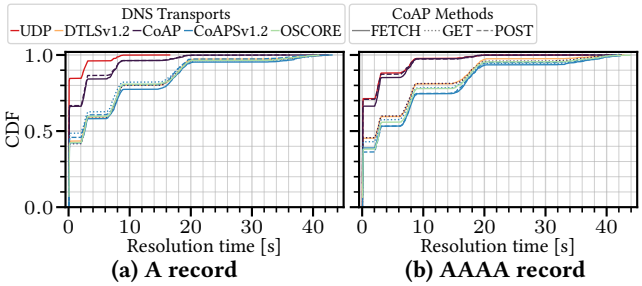


Figure 7: Resolution times for 50 queries (Poisson distributed with $\lambda = 5$ queries/s).

Overall, for unencrypted transmission, UDP is the preferred transport for mitigating fragmentation. For encrypted usage, OSCORE is the preferred method. CoAP packets can multiplex message formats with the Content-Format option. A new compressed DNS messages format could thus help to mitigate fragmentation for CoAP-based transports.

5.4 Name Resolution Times

Next, we evaluate the name resolution times for each protocol. For this, we measure the time from issuing the query by the DNS client until the IP address is parsed in the response.

Figure 7 summarizes the distributions of resolution times for our protocols. We observe that the different transports form distinct groups in their temporal distributions due to the different packet sizes and the resulting 6LoWPAN fragmentation. For UDP requesting an A record no packet is fragmented and names resolve fastest. 85% of the queries resolve in less than 250 ms, all complete within 20 s. Requesting an AAAA record, though, plain UDP compares to unencrypted CoAP with the FETCH or POST method. The query is not fragmented, but the response is. For these transports and methods, only 65–70% of the names resolve below 250 ms, but 99% of names resolving within 20 s. The last group consists of those transports and methods, for which both queries and responses fragment. Unencrypted CoAP with GET, as well

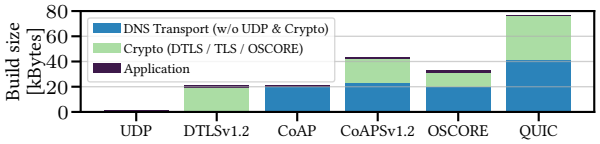


Figure 8: Code sizes of UDP-based DNS transports.

as DTLSv1.2, CoAPSv1.2, and OSCORE perform all within approximately 7% from each other, with 42–49% of A records and 37–45% of AAAA records resolve below 250 ms. All require at most 41–44 s to resolve 99% of names. These long resolution times are due to the CoAP retransmission algorithm and of no concern when considering typical IoT duty cycles, such as the ones employed by LoRaWAN or IEEE 802.15.4e.

As resolution time closely relates to packet size and the number of fragments, the same conclusions as in Section 5.3 can be drawn.

5.5 Comparison with DNS over QUIC

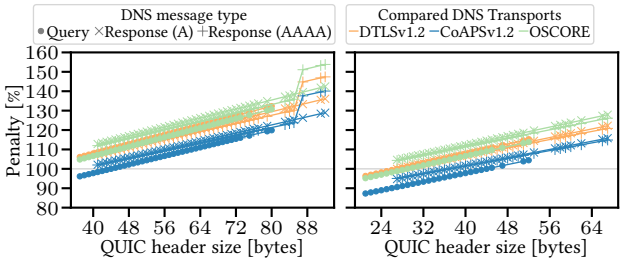
DNS over QUIC [31] could be a lightweight alternative to DNS over CoAP because DNS over QUIC is based on UDP, an IoT-friendly transport. We now compare DNS over QUIC with DNS over CoAPS, DTLS, and OSCORE. To analyze code size implementations, we base our comparison on prior work about QUIC in the IoT [18]. To analyze message sizes, we conduct a numerical evaluation.

Code size. Our point of reference is *Quant* [18], a QUIC implementation on top of RIOT. To account for differences in implementations that are not protocol-specific, we intentionally omit the UDP layer and the *sock* part of RIOT, because Quant accesses the networking modules of RIOT differently than our DNS client implementation. For further details, we refer to Appendix D.

Figure 8 shows the amount of data `.text` and `.data` sections of the generated binaries require. QUIC, including TLS, uses nearly double the ROM as any of the common IoT transports. It is worth noting that Quant includes only the QUIC client part while our implementations include both CoAP client and server code. Further optimizations proposed in [18] can only save ≈20 kBytes, which would require DNS over QUIC to use more ROM compared to DNS over CoAP.

Packet size. QUIC has variable header lengths for two reasons. First, different types of handshakes, 0-RTT, which complete without additional round trips, and 1-RTT, which require an additional round trip, lead to different header types. Second, header fields (*e.g.,* connection IDs) can be of variable sizes. To assess packet sizes realistically, we conduct a best and worst case analysis for both 0-RTT and 1-RTT handshakes.

Figure 9 shows the relative amount packet sizes DNS over QUIC would require to resolve a 24 chars name for a single



(a) 0-RTT packet **(b) 1-RTT packet**

Figure 9: Relative amount of additional data required by DNS over QUIC compared to other DNS transports when resolving a name with a length of 24 characters for a single record (A or AAAA).

A and AAAA record, compared to the other DNS transports. In the best case, *i.e.,* 1-RTT handshakes with small headers, DNS over QUIC is comparable to DNS over CoAP, but in the majority of cases DNS over CoAPS, DTLS, and OSCORE outperform DNS over QUIC (see Figure 9b). In case of 0-RTT QUIC handshakes, efficiency of DNS over QUIC decreases even more (Figure 9a). Requesting an IPv6 address in max header scenarios would lead to fragmentation into 3 fragments to carry the AAAA response over QUIC.

Summary. DNS over QUIC is less suitable in the IoT compared to DNS over CoAP. Code size is larger and packet sizes would require very use-case specific tweaking to selected header fields.

6 EVALUATION OF CACHING FOR DOC

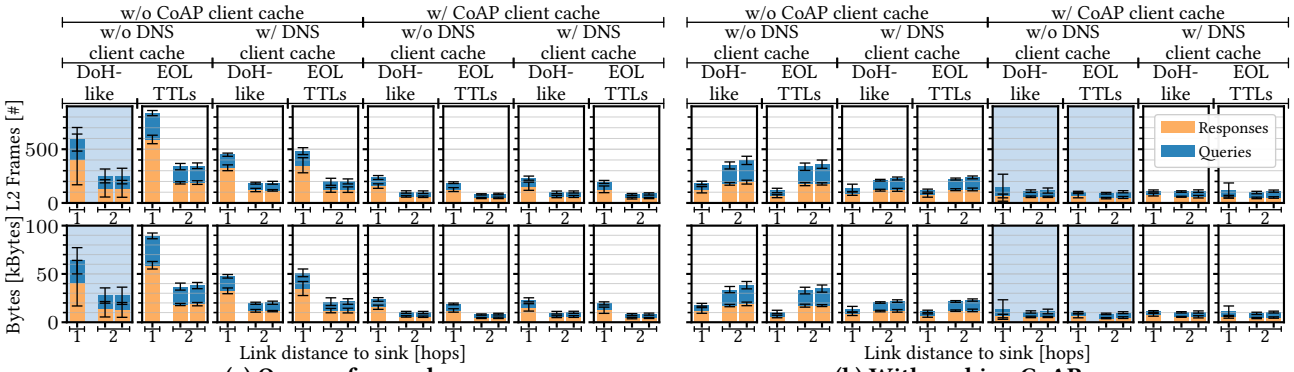
In this section, we perform a comparative assessment of caching as introduced in Section 4 using a multihop network.

6.1 Setup

We base our evaluation on the setup described in Section 5.1, except that clients now query 50 AAAA records of only eight distinct names to showcase the cache utilization.

We compare our two approaches, *DoH-like* and *EOL TTLs* (see Section 4.2) with three levels of caching: (*i*) a DNS cache at each client, (*ii*) a CoAP cache at each client, and (*iii*) a CoAP cache at the forwarder, which runs as CoAP forward proxy. When no cache at the forwarder and thus no forward proxy, we call these the *opaque forwarder* scenarios.

To focus on the impact of caching, we only evaluate unencrypted CoAP to reduce the packet size overhead of encryption and unrelated side effects stemming from that. The DNS resolver returns four AAAA records for each name query. This causes 6LoWPAN fragmentation with three fragments. All four records use the same TTL, uniformly picked between 2 and 8 s to introduce quick cache renewals.



(a) Opaque forwarder (b) With caching CoAP proxy

Figure 10: Link utilization for four AAAA record queries (Poisson-distributed with $\lambda = 5$ queries / s) for different caching solutions with CoAP FETCH, comparing DoH-like and EOL TTLs caching approaches. The scenarios highlighted in blue are evaluated in detail in Section 6.3.

6.2 Link Utilization

We show the influence of the FETCH method on the links for different caching scenarios in Figure 10. The bars with 1-hop link distance to the sink represents the link utilization between proxy and border router, the bars with distance 2 the utilization between each client and the proxy in Figure 2.

Providing a CoAP cache decreases load on all links, and using the *EOL TTLs* approach instead of *DoH-like* further decreases the load. Using DNS caching at the clients gives only little advantages. CoAP caching leads to 50% less link utilization in any configuration. Depending on which node (proxy or client) provides the CoAP cache, the upstream link directly benefits from the cache. Due to the higher amount of messages on the bottleneck link between proxy and border router, the caching can unfold its full potential, visible at that link in the *EOL TTLs* cases in Figure 10b. Here, ≈50 frames less and, depending on the cache configuration, 5-10 kBytes less need to be exchanged. Small advantages can also be observed at the client-proxy links.

6.3 Transport Retransmissions, Cache Hits

We now quantify the link stress that clients generate due to corrective actions. We track both the timestamp of each event at which a client initiates a CoAP transmission and the timestamp for cache hits including re-validations of stale entries at the clients and the proxy. For all events, we calculate the time offset to the start time of the respective DNS query. We focus on the blue scenarios of Figure 10, because they are the most interesting, and consider GET and POST in addition to FETCH.

The original requests have a negligible time offset in the range of microseconds, and since retransmissions follow a random exponential back-off mechanism [65], their time offsets scatter within specific regions (gray areas in Figure 11).

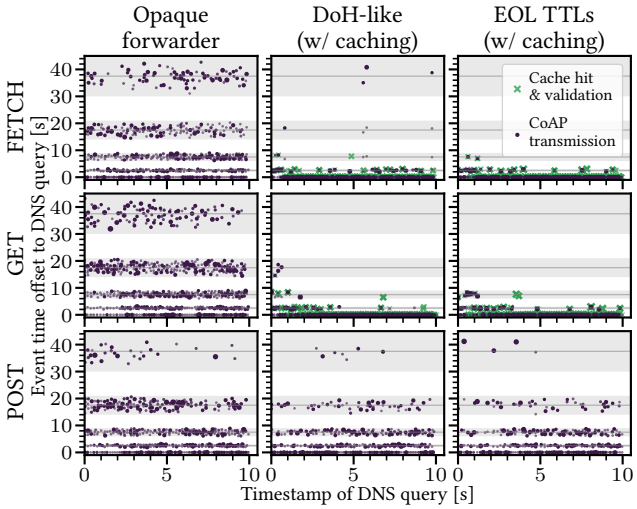


Figure 11: CoAP events of message (re-)transmissions at the client compared to the time of the initial DNS query. Retransmissions follow an exponential back-off and scatter within specific (gray) areas. Size and opacity of each marker represent the multiplicity of events in the same area.

FETCH in combination with *EOL TTLs* provides the best performance (see Figure 11). Cache hits are able to complete requests without requiring more than one retransmission for most of the DNS queries. POST requests do not utilize response caches, which degrades their performance to the level of the *opaque forwarder*.

In the *opaque forwarder* scenario, we observe about 50% more retransmissions for both GET and FETCH compared to any of the *caching approaches*. Using GET, the retransmissions in the third and fourth iteration even increase by 7%

compared to POST and FETCH. This is a result of 6LoWPAN fragmenting the query (see Section 5.3).

7 DISCUSSION AND OPTIMIZATION POTENTIALS

With the core DoC protocol at hand, impactful opportunities for security and privacy open. In addition, a series of potential optimizations and protocol enhancements become attainable, which—along with open questions and shortcomings—we discuss in the following.

Implications for security and privacy in the IoT. DoC introduces with CoAPS a new and efficient baseline for privacy and integrity of DNS queries. To this end, our protocol proposal mainly translates DNS over HTTPS to the constrained IoT. More importantly, the CoAP extension OSCORE allows for securing the DNS query messages independent of the transport. This approach to content object security enables name resolution with secure messages that are cacheable and can traverse a gateway without requiring a trust relation for transcoding. This implies that IoT devices can soften their trust relation with gateways in the future.

How to reduce the DNS packet overhead? DNS messages account for the largest parts of the packets in DoC. Hence, DNS compression schemes beyond the generic methods of 6LoWPAN [48] or SCHC [47] promise enhanced efficiency as concluded in Section 5.3.

One evident enhancement arrives from operators preferring shorter host names in communication with constrained IoT devices. Query name minimization, as applied on the backbone, is not an option, as it relies on the queried server being the authoritative name server of the remaining name part [11, 33]. However, superfluous DNS header fields may be compressed. Klauck and Kirsche [34] proposed a compression for mDNS/DNS-SD messages for 6LoWPAN, but their approach focuses on compatibility with the DNS wire format. DoC offers the opportunity to use different message formats via its use of a new Content-Format. Specifically, the Concise Binary Object Representation (CBOR) [9] offers a standardized, structured, and space-efficient encoding.

In addition, CoAP messages carry a transactional context that matches a reply to its request. Exploiting this, we argue for the following practices to reduce packet overhead. A DoC question could be encoded as a CBOR array, containing up to three entries: the name (as text string), an optional record type (as unsigned integer), and an optional record class (as unsigned integer). If record type and class are elided, DoC implies `AAAA` and `IN`, respectively. A DoC response can be matched to the request. Hence, the encoding could use only one CBOR array, which contains the DNS answer section. This could be nested for several, separate answer sections. An answer section includes a TTL, a name, and an optional type specifier using the space-efficient encoding of CBOR. For an answer with two arrays, DoC additionally identifies the question section (formatted as specified above).

In our evaluation, we could verify that the wire-format of an `AAAA` response packet compresses from 70 bytes down to 24 bytes—a reduction by 66% (see Figure 6). Further suffix and prefix compression, as well as referencing redundant values, can be provided with Packed CBOR [7]. The CBOR working group of the IETF currently discusses such a format [39].

How to protect the integrity of the DNS TTLs? DoC depends on the CoAP Max-Age option to track elapsed caching time, which a DoC client then uses to decrement DNS TTLs. The integrity of the Max-Age option, however, cannot be guaranteed, because it is altered on—potentially untrusted—intermediaries. An adversary with malicious intent, or a faulty proxy behavior may impair TTLs on the client by using incorrect Max-Age values.

For *EOL TTLs*, a potential mitigation is to include a second Max-Age value that is protected by OSCORE. A DoC client compares both Max-Age values, deduces inconsistent modifications, *e.g.,* larger values than the original TTLs, and discards the response when the consistency check fails. For the *DoH-like* caching scheme, responses include the original TTLs, which can be used to perform consistency checks instead of including an additional Max-Age value. This approach mitigates the use of outdated DNS records, but still allows for unauthorized reduction of TTLs, which affects the caching performance.

How to ensure an efficient cache re-validation? A naïve ETag generation calculates a hash over the CoAP message payload to identify a specific DNS response. However, in addition to changing TTLs, DNS resolvers often rearrange resource records within responses, *e.g.,* for load balancing reasons. This modifies the binary representation of DNS messages, and thus their resulting ETag values. One approach to achieve similar load balancing effects without altering the message is to sort incoming records at the DoC server and randomize records at the DoC client.

How to utilize OSCORE group communication with DNS? A big advantage of OSCORE over other encrypted DNS transports, including DoH and DoT, is that it supports group communication, *i.e.,* multicast [68]. This makes it a candidate for encrypted DNS-SD utilizing mDNS. On the other hand, multicast can be very energy intensive, especially in constrained networks. Future work should include an analysis of DNS over Group OSCORE that carefully weighs the benefits against the drawbacks.

Why DNS over CoAP at all? Compared to the more established DNS over DTLS, CoAP could be seen as an uncommon choice: Why introduce yet another layer between DNS and transport security? However, CoAP has several advantages compared to just using DTLS. First and foremost, it is easily

translatable into DoH via a CoAP-to-HTTP proxy. As shown in Section 5 it can benefit in memory usage from sharing the same protocol used as the application. This also helps with obfuscation of name resolution within application traffic, as the unprotected parts are equivalent. Together with OSCORE, it yields the potential to provide both secure, but also loss-decoupled communication via encrypted, en-route caching. Last, when looking at LPWANs, the object security provided by OSCORE provides the opportunity for further header compression with SCHC, providing valuable space for the encrypted DNS message. Transport security can not provide this as it obfuscates the headers in question.

Is our approach limited to CoAP? The ease of transition from DoH to DoC illustrates that our approach is not limited to CoAP, but could also be applied to other application protocols in the IoT domain, such as MQTT.

8 CONCLUSION AND OUTLOOK

The current constrained IoT lacks a protocol for privacy-friendly, secure name resolution such as DoH for the regular Internet. We presented DNS over CoAP (DoC), which leverages the rich feature set of the CoAP protocol suite to provide an energy-efficient and end-to-end protected name resolution for constrained networks. In a comprehensive analysis, we compared DNS over UDP, CoAP, DTLS, CoAP over DTLS, and CoAP with OSCORE in experiments based on full-featured implementations.

Our findings revealed that the performance of name resolution is primarily driven by packet sizes. While CoAP has a space-efficient protocol encoding, the choice of the request method largely impacts the packet overhead and the additional CoAP features at hand. FETCH was identified as the preferred method to GET and POST as it allows for block-wise transfer of queries and for caching of responses. Correspondingly, OSCORE outperformed CoAPS for protecting DoC as it seamlessly integrates with the semantics of CoAP and its header encoding, leading to smaller packets and lower memory usage. Transforming DNS responses into a deterministic form favors the cache validation model of CoAP and reduces bandwidth demands as well as loss rates further.

Future work shall optimize the coding efficiency by defining a comprehensive compression scheme for DNS messages. This will enfold impact by reducing fragmentation and increasing reliability in low-power and lossy regimes. We will also focus on a DoC integration for mDNS protected by Group OSCORE to enable service discovery.

Artifacts. We make all artifacts, including implementations, configurations, and experiment results, publicly available after the double-blind review process.

REFERENCES

[1] Cedric Adjih, Emmanuel Baccelli, Eric Fleury, Gaetan Harter, Nathalie Mitton, Thomas Noel, Roger Pissard-Gibollet, Frederic Saint-Marcel, Guillaume Schreiner, Julien Vandaele, and Thomas Watteyne. 2015. FIT IoT-LAB: A large scale open experimental IoT testbed. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE Press, Piscataway, NJ, USA, 459–464.

[2] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security Evaluation of Home-Based IoT Deployments. In *IEEE S&P 2019*. 1362–1380. https://doi.org/10.1109/SP.2019.00013

[3] Christian Amsüss and Marco Tiloca. 2023. *Cacheable OSCORE*. Internet-Draft – work in progress 06. IETF.

[4] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1093–1110.

[5] Noah Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. 2019. Keeping the Smart Home Private with Smart(er) IoT Traffic Shaping. In *Proc. on Privacy Enhancing Technologies Symposium*, Vol. 2019. 128–148. Issue 3. https://doi.org/10.2478/popets-2019-0040

[6] Emmanuel Baccelli, Cenk Gündogan, Oliver Hahm, Peter Kietzmann, Martine Lenders, Hauke Petersen, Kaspar Schleiser, Thomas C. Schmidt, and Matthias Wählisch. 2018. RIOT: an Open Source Operating System for Low-end Embedded Devices in the IoT. *IEEE Internet of Things Journal* 5, 6 (December 2018), 4428–4440. http://dx.doi.org/10.1109/JIOT.2018.2815038

[7] Carsten Bormann. 2023. *Packed CBOR*. Internet-Draft – work in progress 08. IETF.

[8] C. Bormann, M. Ersue, and A. Keranen. 2014. *Terminology for Constrained-Node Networks*. RFC 7228. IETF.

[9] C. Bormann and P. Hoffman. 2020. *Concise Binary Object Representation (CBOR)*. RFC 8949. IETF.

[10] C. Bormann and Z. Shelby. 2016. *Block-Wise Transfers in the Constrained Application Protocol (CoAP)*. RFC 7959. IETF.

[11] S. Bortzmeyer. 2016. *DNS Query Name Minimisation to Improve Privacy*. RFC 7816. IETF.

[12] S. Cheshire and M. Krochmal. 2013. *DNS-Based Service Discovery*. RFC 6763. IETF.

[13] S. Cheshire and M. Krochmal. 2013. *Multicast DNS*. RFC 6762. IETF.

[14] RIOT OS Community. 2022. RIOT Documentation – DNS defines. https://doc.riot-os.org/group__net__dns.html, last accessed 29-05-2023.

[15] TTN Community. 2022. The Things Network. https://www.thethingsnetwork.org/, last accessed 04-12-2022.

[16] Hesselman Cristian, Kaeo Merike, Chapin Lyman, Claffy Kimberly, Seiden Mark, McPherson Danny, Piscitello Dave, McConachie Andrew, April Tim, Latour Jacques, and Rasmussen Rod. 2020. The DNS in IoT: Opportunities, Risks, and Challenges. *IEEE Internet Computing* 24, 4 (2020), 23–32.

[17] J. Dickinson, S. Dickinson, R. Bellis, A. Mankin, and D. Wessels. 2016. *DNS Transport over TCP - Implementation Requirements*. RFC 7766. IETF.

[18] Lars Eggert. 2020. Towards Securing the Internet of Things with QUIC. In *Proc. of 3rd NDSS Workshop on Decentralized IoT Systems and Security (DISS)* (San Diego, CA, USA). Internet Society (ISOC).

[19] O. Gimenez and I. Petrov. 2021. *Static Context Header Compression and Fragmentation (SCHC) over LoRaWAN*. RFC 9011. IETF.

[20] C. Gomez, J. Crowcroft, and M. Scharf. 2021. *TCP Usage Guidance in the Internet of Things (IoT).* RFC 9006. IETF.

[21] J. Gregorio, R. Fielding, M. Hadley, M. Nottingham, and D. Orchard. 2012. *URI Template.* RFC 6570. IETF.

[22] Cenk Gündogan, Christian Amsüss, Thomas C. Schmidt, and Matthias Wählisch. 2020. Toward a RESTful Information-Centric Web of Things: A Deeper Look at Data Orientation in CoAP. In *Proc. of 7th ACM Conference on Information-Centric Networking (ICN)* (Montreal, CA). ACM, New York, 77–88. https://doi.org/10.1145/3405656.3418718

[23] Cenk Gündogan, Christian Amsüss, Thomas C. Schmidt, and Matthias Wählisch. 2022. Content Object Security in the Internet of Things: Challenges, Prospects, and Emerging Solutions. *IEEE Transactions on Network and Service Management (TNSM)* 19, 1 (March 2022), 538–553. https://doi.org/10.1109/TNSM.2021.3099902

[24] Martin Gunnarsson, Joakim Brorsson, Francesca Palombini, Ludwig Seitz, and Marco Tiloca. 2021. Evaluating the performance of the OSCORE security protocol in constrained IoT environments. *Internet of Things* 13 (2021), 100333.

[25] Hang Guo and John Heidemann. 2020. Detecting IoT Devices in the Internet. *IEEE/ACM Transactions on Networking* 28, 5 (October 2020), 2323–2336.

[26] Jessica Haworth. 2019. SANS reveals the top attacks for 2019 at RSA Conference. https://portswigger.net/daily-swig/sans-reveals-the-top-attacks-for-2019-at-rsa-conference.

[27] P. Hoffman and P. McManus. 2018. *DNS Queries over HTTPS (DoH).* RFC 8484. IETF.

[28] Austin Hounsel, Kevin Borgolte, Paul Schmitt, Jordan Holland, and Nick Feamster. 2020. Comparing the Effects of DNS, DoT, and DoH on Web Performance. In *Proceedings of The Web Conference 2020* (Taipei, Taiwan) *(WWW '20)*. ACM, New York, NY, USA, 562–572.

[29] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. 2016. *Specification for DNS over Transport Layer Security (TLS).* RFC 7858. IETF.

[30] J. Hui and JP. Vasseur. 2012. *The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams.* RFC 6553. IETF.

[31] C. Huitema, S. Dickinson, and A. Mankin. 2022. *DNS over Dedicated QUIC Connections.* RFC 9250. IETF.

[32] IEEE 802.15 Working Group. 2016. *IEEE Standard for Low-Rate Wireless Networks.* Technical Report IEEE Std 802.15.4™–2015 (Revision of IEEE Std 802.15.4-2011). IEEE, New York, NY, USA. 1–709 pages.

[33] Burton S. Kaliski Jr. 2022. Minimized DNS Resolution: Into the Penumbra. https://ipj.dreamhosters.com/wp-content/uploads/2023/01/253-ipj.pdf. *The Internet Protocol Journal* 25, 3 (Dec. 2022).

[34] Ronny Klauck and Michael Kirsche. 2013. Enhanced DNS message compression - Optimizing mDNS/DNS-SD for the use in 6LoWPANs. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. 596–601. https://doi.org/10.1109/PerComW.2013.6529565

[35] Mike Kosek, Trinh Viet Doan, Malte Granderath, and Vaibhav Bajpai. 2022. One to Rule Them All? A First Look at DNS over QUIC. In *Proc. of PAM (LNCS, Vol. 13210)*. Springer, Cham, 537–551.

[36] Sam Kumar, Michael P. Andersen, Hyung-Sin Kim, and David E. Culler. 2020. Performant TCP for Low-Power Wireless Networks. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA, 911–932.

[37] lefty. 2019. [Dumpsterfire] DNS rebinding attacks. http://www.firemountain.net/pipermail/dumpsterfire/2019-June/000090.html.

[38] Martine S. Lenders, Thomas C. Schmidt, and Matthias Wählisch. 2021. Fragment Forwarding in Lossy Networks. *IEEE Access* 9 (October 2021), 143969 – 143987. https://doi.org/10.1109/ACCESS.2021.3121557

[39] Martine S. Lenders, Thomas C. Schmidt, and Matthias Wählisch. 2022. *A Concise Binary Object Representation (CBOR) of DNS Messages.* IETF Internet Draft – work in progress 01. IETF. https://datatracker.ietf.org/doc/draft-lenders-dns-cbor/

[40] LoRa Alliance. 2019. *RP002-1.0.0 LoRaWAN™Regional Parameters.* Technical Report. LoRa Alliance. https://lora-alliance.org/wp-content/uploads/2019/11/rp_2-1.0.0_final_release.pdf

[41] Chaoyi Lu, Baojun Liu, Zhou Li, Shuang Hao, Haixin Duan, Mingming Zhang, Chunying Leng, Ying Liu, Zaifeng Zhang, and Jianping Wu. 2019. An End-to-End, Large-Scale Measurement of DNS-over-Encryption: How Far Have We Come?. In *Proc. of ACM IMC*. ACM, New York, NY, USA, 22–35.

[42] Minzhao Lyu, Hassan Habibi Gharakheili, and Vijay Sivaraman. 2022. A Survey on DNS Encryption: Current Development, Malware Misuse, and Inference Techniques. *ACM Comput. Surv.* (July 2022).

[43] Jiarun Mao, Michael Rabinovich, and Kyle Schomp. 2022. Assessing Support for DNS-over-TCP in the Wild. In *Proc. of PAM (LNCS, Vol. 13210)*. Springer, Cham, 487–517.

[44] ARM Mbed. 2021. DNS Resolver - API references and tutorials | Mbed OS 6 Documentation. https://os.mbed.com/docs/mbed-os/v6.16/apis/dns-apis.html, last accessed 29-05-2023.

[45] D. McGrew and D. Bailey. 2012. *AES-CCM Cipher Suites for Transport Layer Security (TLS).* RFC 6655. IETF.

[46] Microchip 2009. *Low Power 2.4 GHz Transceiver for ZigBee, IEEE 802.15.4, 6LoWPAN, RF4CE, SP100, WirelessHART, and ISM Applications (AT86RF231).* Microchip. https://ww1.microchip.com/downloads/en/DeviceDoc/doc8111.pdf Rev.8111C.

[47] A. Minaburo, L. Toutain, C. Gomez, D. Barthel, and JC. Zuniga. 2020. *SCHC: Generic Framework for Static Context Header Compression and Fragmentation.* RFC 8724. IETF.

[48] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. 2007. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks.* RFC 4944. IETF.

[49] Roberto Perdisci, Thomas Papastergiou, Omar Alrawi, and Manos Antonakakis. 2020. IoTFinder: Efficient Large-Scale Identification of IoT Devices via Passive DNS Traffic Analysis. In *IEEE EuroS&P 2020*. 474–489. https://doi.org/10.1109/EuroSP48549.2020.00037

[50] P. Phaal, S. Panchen, and N. McKee. 2001. *InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks.* RFC 3176. IETF.

[51] Zephyr Project. 2022. DNS Resolve – Zephyr Project Documentation. https://docs.zephyrproject.org/3.2.0/connectivity/networking/api/dns_resolve.html, last accessed 29-05-2023.

[52] T. Reddy, D. Wing, and P. Patil. 2017. *DNS over Datagram Transport Layer Security (DTLS).* RFC 8094. IETF.

[53] Jingjing Ren, Daniel J. Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. 2019. Information Exposure for Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach. In *Proc. of the Internet Measurement Conference (IMC)*. ACM.

[54] E. Rescorla. 2018. *The Transport Layer Security (TLS) Protocol Version 1.3.* RFC 8446. IETF.

[55] E. Rescorla and N. Modadugu. 2012. *Datagram Transport Layer Security Version 1.2.* RFC 6347. IETF.

[56] E. Rescorla, H. Tschofenig, T. Fossati, and A. Kraus. 2022. *Connection Identifier for DTLS 1.2.* RFC 9146. IETF.

[57] E. Rescorla, H. Tschofenig, and N. Modadugu. 2022. *The Datagram Transport Layer Security (DTLS) Protocol Version 1.3.* RFC 9147. IETF.

[58] Christian Rossow. 2014. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *Proc. of NDSS*. Internet Society, 15 pages.

[59] Said Jawad Saidi, Srdjan Matic, Oliver Gasser, Georgios Smaragdakis, and Anja Feldmann. 2022. Deep Dive into the IoT Backend Ecosystem. In *Proc. of the 22nd ACM SIGCOMM Conf. on Internet Measurement*

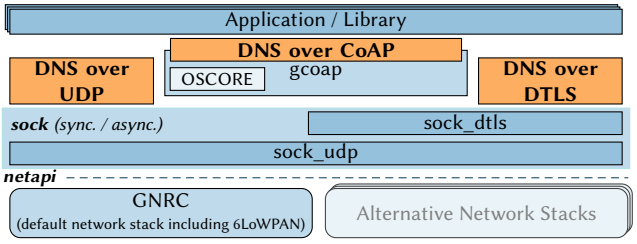


Figure 12: DNS, DoDTLS, and DoC in RIOT.

(IMC '22) (IMC '22). Association for Computing Machinery, New York, NY, USA, 488–503. https://doi.org/10.1145/3517745.3561431

[60] Said Jawad Saidi, Srdjan Matic, Georgios Smaragdakis, Oliver Gasser, and Anja Feldmann. 2022. Deep Dive into the IoT Backend Ecosystem. In *Proc. of the 22nd ACM Internet Measurement Conference (IMC)*. ACM, 488–503. https://doi.org/10.1145/3517745.3561431

[61] J. Schaad. 2017. *CBOR Object Signing and Encryption (COSE)*. RFC 8152. IETF.

[62] Benjamin M. Schwartz, Mike Bishop, and Erik Nygren. 2023. *Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)*. Internet-Draft – work in progress 12. IETF.

[63] G. Selander, J. Mattsson, F. Palombini, and L. Seitz. 2019. *Object Security for Constrained RESTful Environments (OSCORE)*. RFC 8613. IETF.

[64] Göran Selander, John Preuß Mattsson, and Francesca Palombini. 2023. *Ephemeral Diffie-Hellman Over COSE (EDHOC)*. Internet-Draft – work in progress 19. IETF.

[65] Z. Shelby, K. Hartke, and C. Bormann. 2014. *The Constrained Application Protocol (CoAP)*. RFC 7252. IETF.

[66] J. Snijders. 2017. *Deprecation of BGP Path Attribute Values 30, 31, 129, 241, 242, and 243*. RFC 8093. IETF.

[67] STMicroelectronics 2018. *High-density performance line ARM©-based 32-bit MCU with 256 to 512KB Flash, USB, CAN, 11 timers, 3 ADCs, 13 communication interfaces (STM32F103REY)*. STMicroelectronics. https://www.st.com/resource/en/datasheet/stm32f103re.pdf DS5792 Rev 13.

[68] Marco Tiloca, Göran Selander, Francesca Palombini, John Preuß Mattsson, and Jiye Park. 2022. *Group OSCORE - Secure Group Communication for CoAP*. Internet-Draft – work in progress 17. IETF.

[69] Theodor Ts'o. 2022. [Cryptography] Cryptographic signing of software is security theater. https://www.metzdowd.com/pipermail/cryptography/2022-December/038049.html.

[70] P. van der Stok, C. Bormann, and A. Sehgal. 2017. *PATCH and FETCH Methods for the Constrained Application Protocol (CoAP)*. RFC 8132. IETF.

[71] Liang Zhu, Zi Hu, John Heidemann, Duane Wessels, Allison Mankin, and Nikita Somaiya. 2015. Connection-Oriented DNS to Improve Privacy and Security. In *Proc. of IEEE Symposium on Security and Privacy*. IEEE, Piscataway, NJ, USA, 171–186.

A IMPLEMENTATION

In this section, we introduce our framework to run DNS queries in the constrained IoT, including implementations of a DoC prototype, DNS over UDP, and DNS over DTLS. We make use of the IoT operating system RIOT [6].

The network stack of RIOT is visualized in Figure 12 (DNS building blocks in orange). The *sock* API of RIOT is agnostic to the underlying network stack, such as GNRC, and allows, among other transports, access to UDP and DTLS. Its unified access allows for a seamless composability of network building blocks, eases implementation complexity, and grants a flexible integration of third-party network stacks [6]. The existing DNS client in RIOT builds on the *sock* API to interface with the underlying network stack. Message-related operations to compose DNS queries and parse DNS responses follow a modular and reusable design. For code simplicity, this client uses the synchronous version of *sock*, so it blocks on requests until a response arrives or a timeout occurs. The *gCoAP* library provides support for CoAP [65] on top of *sock* as well. It features all request methods including GET, POST, PUT, and DELETE, but also FETCH, PATCH, and iPATCH [70]. *gCoAP* also implements proxying capabilities and supports response caches on both proxies and clients [23]. Block-wise transfer [10] as well as DTLS support are provided.

For the purposes of evaluating DNS over CoAP (DoC) we extend the stack by the following functionalities. We extend the DNS over UDP implementation to support asynchronous calls to *sock* for **non-blocking queries**. For better comparability with DoC, we support the **retransmission algorithm of CoAP** [65] for DNS over UDP, *i.e.,* 4 retransmissions using an exponential back-off. We provide a **DoDTLS client implementation** via *sock* with blocking and asynchronous capabilities and supports Pre-shared Keys (PSKs) using AES and Elliptic Curve Cryptography (ECC) via the *TinyDTLS* library. We implement a **DoC client** on top of *gCoAP*, which can be configured to use both blocking and asynchronous requests. For OSCORE support, the *libOSCORE* library is used. We also provide a lightweight **URI template processor**, which the DoC client can use to marshal the packet format of DNS queries into the URI option of CoAP GET requests. Both our DoDTLS and DoC implementation reuse the generic interface to compose and parse DNS messages of the DNS over UDP implementation of RIOT.

B ANALYSIS OF COAP PACKET SIZES

In Section 5.3 we evaluated the packet size, but did not go into detail for the different CoAP message types. In Figure 13 we show the packet sizes with block-wise transfer or GET and POST requests for DNS over CoAP.

Block-wise transfer offers a solution for the CoAP-based transports to prevent fragmentation on the link layer, allowing for confirmable segmentation into blocks within the application layer. However, only the payload can be transferred in blocks. As such, using the GET method, we are unable to send our DNS query in blocks, as it is carried, encoded in `base64` within the CoAP URI-Query option. Consequently, the GET request stays the same in all the block-wise transfer modes shown in Figure 13. With block-wise transfer, we are able to reduce the overall packet size enough to drop below

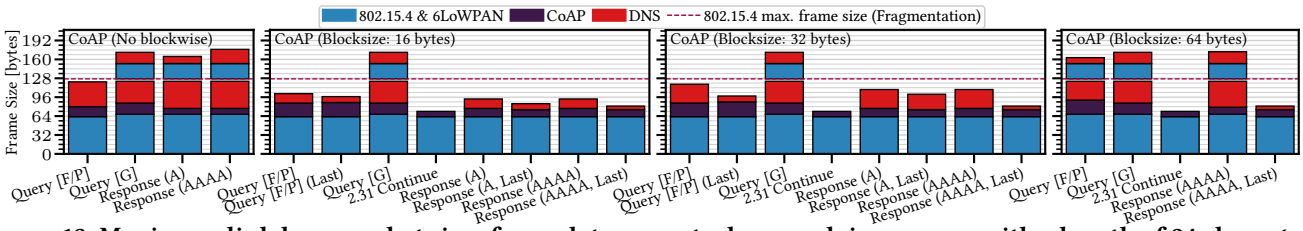
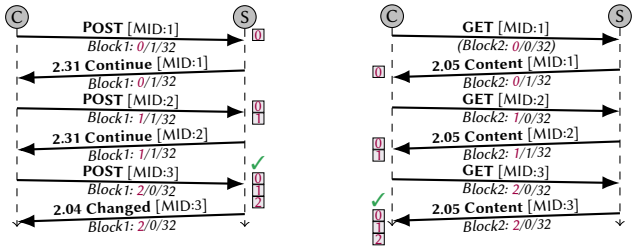
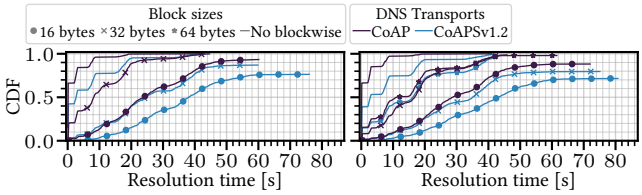


Figure 13: Maximum link layer packet sizes for each transport when resolving a name with a length of 24 characters for a single record (A and AAAA respectively) for different CoAP methods (F = FETCH, G = GET, P = POST) and block sizes. "Last" denotes the size of the last block with block-wise transfer.



(a) Block1 for requests **(b) Block2 for responses**

Figure 14: Different block-wise transfers of a 96 bytes body between a client C and a server S using 32 byte blocks in CoAP. $n/m/s$ denotes the block number, whether more blocks (or not) can be sent, and the block size.



(a) A record **(b) AAAA record**

Figure 15: Resolution times for 50 queries using FETCH with block-wise transfer. Block size 64 was only used with AAAA records, as DNS responses for A record stay below 64 bytes in size.

the fragmentation line of 6LoWPAN. Compared to fragmentation in 6LoWPAN, CoAP block-wise transfer provides us with a recovery mechanism, so even if a message is lost, we can recover from that on a block-level, so we do not have to send the whole request or response again, in case one single block or fragment gets lost.

For the setup evaluated, a block size of 32 bytes is ideal: 16 bytes makes the blocks smaller and more numerous than necessary and 64 already leads to 6LoWPAN fragmentation.

C COAP BLOCK-WISE TRANSFER

CoAP POST and FETCH, which carry a DNS query in the body of a request, provide an additional advantage in case the DNS query message size exceeds an PDU: The body can be split into multiple CoAP messages by the block-wise transfer mode. When using the Block1 [10] option, a receiver assembles the full message on successful reception of all blocks (see Figure 14a). In contrast to queries, the Block2 transfer mode [10] allows a client to request a certain block size in the response, but the server may also decide to transfer in blocks proactively, without the Block2 option being present in the initial request (see Figure 14b).

Name Resolution Times. To evaluate the name resolution times block-wise transfers, we used the same communication setup as described in Section 5 but also statically set the block size for both requests and responses to 16, 32, and 64 bytes, respectively. Block size 64 was only used with AAAA records, as only the responses for those exceed 64 bytes in the CoAP payload. We plot the temporal distributions for A and AAAA records with block-wise transfer using FETCH requests for CoAP and CoAPSv1.2 in Figure 15. For easier comparison, we include the distributions of Figure 7, which do not utilize the block-wise transfer, but we emphasize the increased range of the x-axis. We observe that the performance decreases with smaller block sizes. With a block size of 16 bytes, only ≈ 90% and 60–70% of name resolutions complete in total for CoAP and CoAPSv1.2, respectively. This is due to congestion emerging in the wireless medium, which increases the probability of packet loss for transfers with higher block counts. The lesson here, is that block-wise transfer can help mitigate the problem of fragmentation, but leads to a decrease in performance as well, if not properly congestion controlled.

D EVALUATION DETAILS

Section 5.2 We use version `9-2019-q4-major` of the *GNU ARM Embedded Toolchain*, the recommended toolchain for RIOT `2022.07`, which includes GCC v9.2.1. RIOT ships a tool to dissect the memory usage of a RIOT firmware image from module level down to function and variable level.

We group the modules of RIOT according to the following categories. *Application* contains all the machine code instructions and state information of the experiment application.

DNS contains the code of each DNS over X implementation, including the shared DNS message parser and composer. As the GET method in DoC adds a significant amount of memory for URI template processing, this is separately shown. *OSCORE* contains the code of libOSCORE including its dependencies. *CoAP* contains the code of the gCoAP library and its dependencies, as well as URI parsing. *sock* contains the code of the sock API implementation for the GNRC network stack, as well as the *sock* implementation for TinyDTLS. This was included to account for the different build sizes when using DTLS. *DTLS* contains the code of TinyDTLS including its dependencies. *CoAP example app* contains the code of the RIOT CoAP example.

Section 5.5 We compile Quant (and our DNS over CoAP/DTLS/OSCORE implementations) for RIOT version `2022.07` using the `esp-2021r2-patch3` GCC release by *espressif*. We used the ESP32 platform because Quant is available on ESP32.

E LIST OF COMMON ACRONYMS

CBOR Concise Binary Object Representation
CoAP Constrained Application Protocol
CoAPS CoAP over DTLS
COSE CBOR Object Signing and Encryption
DoC DNS over CoAP
DoDTLS DNS over DTLS
DoH DNS over HTTPS
DoT DNS over TLS
DoQ DNS over QUIC
DTLS Datagram Transport Layer Security
OSCORE Object Security for Constrained RESTful Environments