Workgroup: CoRE
Internet-Draft:
draft-amsuess-core-transport-indication-01
Published: 10 July 2021
Intended Status: Standards Track
Expires: 11 January 2022
Authors: C. Amsüss

CoAP Protocol Indication

Abstract

The Constrained Application Protocol (CoAP, [RFC7252]) is available over different transports (UDP, DTLS, TCP, TLS, WebSockets), but lacks a way to unify these addresses. This document provides terminology based on Web Linking [RFC8288] to express alternative transports available to a device, and to optimize exchanges using these.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Constrained RESTful Environments Working Group mailing list (core@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/core/.

Source for this draft and an issue tracker can be found at https://gitlab.com/chrysn/transport-indication.

Status of This Memo

Table of Contents

1. Introduction
  1.1. Terminology
  1.2. Goals
2. Indicating alternative transports
  2.1. Example
  2.2. Security context propagation
  2.3. Choice of transports
  2.4. Selection of a canonical origin
  2.5. Advertisement through a Resource Directory
3. Elision of Proxy-Scheme and Uri-Host
4. Third party proxy services
  4.1. Generic proxy advertisements
5. Client picked proxies
6. Related work and applicability to related fields
  6.1. On HTTP
  6.2. Using DNS
  6.3. Using names outside regular DNS
7. Security considerations
  7.1. Security context propagation
  7.2. Traffic misdirection
  7.3. Protecting the proxy
  7.4. Implementing proxies
8. IANA considerations
  8.1. Link Relation Types
9. References
  9.1. Normative References
  9.2. Informative References
Appendix A. Change log
Appendix B. Open Questions / further ideas
Appendix C. Acknowledgements
Author's Address

1. Introduction

The Constrained Application Protocol (CoAP) provides transports mechanisms (UDP and DTLS since [RFC7252], TCP, TLS and WebSockets since [RFC8323]), with some additional being used in LwM2M [lwm2m] and even more being explored ([I-D.bormann-t2trg-slipmux], [I-D.amsuess-core-coap-over-gatt]). These are mutually incompatible on the wire, but CoAP implementations commonly support several of them, and proxies can translate between them.

CoAP currently lacks a way to indicate which transports are available for a given resource, and to indicate that a device is prepared to serve as a proxy; this document solves both by introducing the "has-proxy" terminology to Web Linking [RFC8288] that expresses the former through the latter. The additional "has-unique-proxy" term is introduced to negate any per-request overhead that would otherwise be introduced in the course of this.

CoAP also lacks a unified scheme to label a resource in a transport-indepenent way. This document does *not* attempt to introduce any new scheme here, or raise a scheme to be the canonical one. Instead, each host can pick a canonical address for its resources, and advertise other transports in addition.

1.1. Terminology

Same-host proxy A CoAP server that accepts forward proxy requests (i.e., requests carrying the Proxy-Scheme option) exclusively for URIs that it is the authoritative server for is defined as a "same-host proxy".

The distinction between a same-host and any other proxy is only relevant on a practical, server-implementation and illustrative level; this specification does not use the distinction in normative requirements, and clients need not make the distinction at all.

hosts The verb "to host" is used here in the sense of the link relation of the same name defined in [RFC6690].

For resources discovered via CoAP's discovery interface, a hosting statement is typically provided by the defaults implied by [RFC6690] where a link like </sensor/temp> is implied to have the relation "hosts" and the anchor /, such that a statement "coap://hostname hosts coap://hostname/sensor/temp" can be inferred.

For many application it can make sense to assume that any resource has a "host" relation leading from the root path of the server without having performed that discovery explicitly.

[TBD: The last paragraph could be a contentuous point; things should still work without it, and maybe that's even better because it precludes a dynamic resource created with one transport from use with another transport unless explicitly cleared.]

When talking of proxy requests, this document only talks of the Proxy-Scheme option. Given that all URIs this is usable with can be expressed in decomposed CoAP URIs, the need for using the Proxy-URI option should never arise.

1.2. Goals

This document introduces provisions for the seamless use of different transport mechanisms for CoAP. Combined, these provide:

* Enablement: Inform clients of the availability of other transports of servers.

* No Aliasing: Any URI aliasing must be opt-in by the server. Any defined mechanisms must allow applications to keep working on the canonical URIs given by the server.

* Optimization: Do not incur per-request overhead from switching protocls. This may depend on the server's willingness to create aliased URIs.

* Proxy usability: All information provided must be usable by aware proxies to reduce the need for duplicate cache entries.

* Proxy announcement: Allow third parties to announce that they provide alternative transports to a host.

For all these functions, security policies must be described that allow the client to use them as securely as the original transport.

This document will not concern itself with changes in transport availability over time, neither in causing them ("Please take up your TCP interface, I'm going to send a firmware update") nor in advertising them (other than by the server putting suitable Max-Age values on any of its statements).

2. Indicating alternative transports

While CoAP can indicate the authority component of the requested URI in all requests (by means of Uri-Host), indicating the scheme of a requested URI (by means of Proxy-Scheme) makes the request implicitly a proxy request. However, this needs to be of only little practical concern: Any device can serve as a proxy for itself (a "same-host proxy") by accepting requests that carry the Proxy-Scheme

option. If it is to be a well-behaved as a proxy, the device should then check whether it recognizes the name indicated in Uri-Host as one of its own [TBD: Check whether 7252 makes this a stricter requirement], reject the request with 5.05 when it is not recognized, and otherwise process it as it would process a request coming in on that protocol (which, for many hosts, is the same as if the option were absent completely).

A server can indicate support for same-host proxying (or any kind of proxying, really) by serving a Web Link with the "has-proxy" relation.

The semantics of a link from C to T with relations has-proxy ("C has-proxy T", <T>;rel=has-proxy;anchor="C") are that for any resource R hosted on C ("C hosts R"), T is can be used as a proxy to obtain R.

Note that HTTP and CoAP proxies are not located at a particular resource, but at a host in general. Thus, a proxy URI T in these protocols can not carry a path or query component. This is true even for CoAP over WebSockets (which uses the concrete resource /.well-known/coap, but that can not expressed in "coap+ws" URI). Future protocols for which CoAP proxying is defined may have expressible path components.

2.1. Example

A constrained device at the address 2001:db8::1 that supports CoAP over TCP in addition to CoAP can self-describe like this:

```
Req: to [ff02::fd]:5683 on UDP
Code: GET
Uri-Path: /.well-known/core
Uri-Query: if=tag:example.com,sensor

Res: from [2001:db8::1]:5683
Content-Format: application/link-format
Payload:
</sensors/temp>;if="tag:example.com,sensor",
<coap+tcp://[2001:db8::1]>;rel=has-proxy;anchor="/"


Req: to [2001:db8::1]:5683 on TCP
Code: GET
Proxy-Scheme: coap
Uri-Path: /sensors/temp
Observe: 0

Res: 2.05 Content
Observe: 0
Payload:
39.1°C
```

Figure 1: Follow-up request through a has-proxy relation

Note that generating this discovery file needs to be dynamic based on its available addresses; only if queried using a link-local source address, it may also respond with a link-local address in the authority component of the proxy URI.

Unless the device makes resources discoverable at coap+tcp://[2001:db8::1]/.well-known/core or another discovery mechanism, clients may not assume that coap+tcp://[2001:db8::1]/sensors/temp is a valid resource (let alone has any relation to the other resource on the same path). The server advertising itself like this may reject any request on CoAP-over-TCP unless they contain a Proxy-Scheme option.

Clients that want to access the device using CoAP-over-TCP would send a request by connecting to 2001:db8::1 TCP port 5683 and sending a GET with the options Proxy-Scheme: coap, no Uri-Host or -Port options (utilizing their default values), and the Uri-Paths "sensors" and "temp".

2.2. Security context propagation

If the originally requested URI R or the application requirements demand a security mechanism is used, the client MUST only use the proxy T if the proxy can provide suitable credentials. (The hosting URI C is immaterial to these considerations).

Credentials are usable if either:

* The credentials are good for the intended use of R.

  For example, if the application uses the host name and a public key infrastructure and R is coap://example.com/ the proxy accessed as coap+tcp://[2001:db8::1] still needs to provide a certificate chain for the name example.com to one of the system's trust anchors. If, on the other hand, the application is doing a firmware update and requires any certificate from its configured firmware update issuer, the proxy needs to provide such a firmware update certificate.

* The credentials are suitable as a general trusted proxy for the system.

  This applies only to security mechsnisms that are terminated in proxies (i.e. (D)TLS and not OSCORE).

  For a client to trust a proxy to this extent, it must have configured knowledge which proxies it may trust. Such configuration is generally only possible if the application's security selection is based on the host name (as the client's intention to, as in the above example, obtain a firmware update, can not be transported to the proxy).

  This option is unlikely to be useful in same-host proxies, but convenient in scenarios like in Section 4.

2.3. Choice of transports

It is up to the client whether to use an advertised proxy transport, or (if multiple are provided) which to pick.

Links to proxies may be annotated with additional metadata that may help guide such a choice; defining such metadata is out of scope for this document.

Clients MAY switch between advertised transports as long as the document describing them is fresh; they may even do so per request. (For example, they may perform individual requests using CoAP-over-UDP, but choose CoAP-over-TCP for requests with large expected responses).

2.4. Selection of a canonical origin

While a server is at liberty to provide the same resource independently on different transports (i.e. to create aliases), it may make sense for it to pick a single scheme and authority under which it announces its resources. Using only one address helps

proxies keep their caches efficient, and makes it easier for clients to avoid exploring the same server twice from different angles.

When there is a predominant scheme and authority through which an existing service is discovered, it makes sense to use these for the canonical addresses.

Otherwise, it is suggested to use the coap or coaps scheme (given that these are the most basic and widespread ones), and the most stable usable name the host has.

2.5. Advertisement through a Resource Directory

In the Resource Directory specification [I-D.ietf-core-resource-directory], protocol negotiation was anticipated to use multiple base values. This approach was abandoned since then, as it would incur heavy URI aliasing.

Instead, devices can submit their has-proxy links to the Resource Directory like all their other metadata.

A client performing resource lookup can ask the RD to provide available (same-host-)proxies in a follow-up request by asking for ?anchor=the-discovered-host&rel=has-proxy. The RD may also volunteer that information during resource lookups even though the has-proxy link itself does not match the search criteria.

[It may be useful to define RD parameters for use with lookup here, which'd guide which available proxies to include.]

3. Elision of Proxy-Scheme and Uri-Host

A CoAP server may publish and accept multiple URIs for the same resource, for example when it accepts requests on different IP addresses that do not carry a Uri-Host option, or when it accepts requests both with and without the Uri-Host option carrying a registered name. Likewise, the server may serve the same resources on different transports. This makes for efficient requests (with no Proxy-Scheme or Uri-Host option), but In general is discouraged [aliases].

To make efficient requests possible without creating URI aliases that propagate, the "has-unique-proxy" specialization of the has-proxy relation is defined.

If a proxy is unique, it means that it unconditionally forwards to the server indicated in the link context, even if the Proxy-Scheme and Uri-Host options are elided.

[The following two paragraphs are both true but follow different approaches to explaining the observable and implementable behavior; it may later be decided to focus on one or the other in this document.]

While this creates URI aliasing in the requests as they are sent over the network, applications that discover a proxy this way should not "think" in terms of these URIs, but retain the originally discovered URIs (which, because Cool URIs Don't Change[cooluris], should be long-term usable). They use the proxy for as long as they have fresh knowledge of the has-(unique-)proxy statement.

In a way, advertising has-unique-proxy can be viewed as a description of the link target in terms of SCHC [I-D.ietf-lpwan-coap-static-context-hc]: In requests to that target, the link source's scheme and host are implicitly present.

A client MAY use a unique-proxy like a proxy and still send the Proxy-Scheme and Uri-Host option; such a client needs to recognize both relation types, as relations of the has-unique-proxy type are a specialization of has-proxy and typically don't carry the latter (redundant) annotation. [To be evaluated -- one one hand, supporting it this way means that the server needs to identify all of its addresses and reject others. Then again, is a server that (like many now do) fully ignore any set Uri-Host correct at all?]

Example:

```
Req: to [ff02::fd]:5683 on UDP
Code: GET
Uri-Path: /.well-known/core
Uri-Query: if=tag:example.com,sensor

Res: from [2001:db8::1]:5683
Content-Format: application/link-format
Payload:
</sensors/>;if="tag:example.com,collection",
<coap+tcp://[2001:db8::1]>;rel=has-proxy;anchor="/"


Req: to [2001:db8::1]:5683 on TCP
Code: GET
Uri-Path: /sensors/

Res: 2.05 Content
Content-Format: application/link-format
Payload:
</sensors/temperature>;if="tag:example.com,sensor"
```

Figure 2: Follow-up request through a has-unique-proxy relation. Compared to the last example, 5 bytes of scheme indication are saved during the follow-up request.

It is noteworthy that when the URI reference /sensors/temperature is resolved, the base URI is coap://device0815.example.com and not its coap+ws counterpart -- as the request is implicitly forwarded there, which both the client and the server are aware of. However, this detail is of little practical importance: A simplistic client that uses coap+ws://device0815.proxy.rd.example.com as a base URI will still arrive at an identical follow-up request with no ill effect, as long as it only uses the wrongly assembled URI for dereferencing resources, the security context is the same, and it does not (for example) pass it on to other devices.

4. Third party proxy services

A server that is aware of a suitable cross proxy may use the has-proxy relation to advertise that proxy. If the protocol used towards the proxy provides name indication (as CoAP over TLS or WebSockets does), or by using a large number of addresses or ports, it can even advertise a (more efficient) has-unique-proxy relation. This is particularly interesting when the advertisements are made available across transports, for example in a Resource Directory.

How the server can discover and trust such a proxy is out of scope for this document, but generally involves the same kind of links.

The proxy may advertise itself without the origin server's involvement; in that case, the client needs to take additional care (see Section 7.2).

```
Req: GET http://rd.example.com/rd-lookup?if=tag:example.com,sensor

Res:
Content-Format: application/link-format
Payload:
<coap://device0815.example.com/sensors/>;if="tag:example.com,collection"
<coap+wss://device0815.proxy.rd.example.com>;rel=has-unique-proxy;anchor


Req: to device0815.proxy.rd.example.com on WebSocket
Host (indicated during upgrade): device0815.proxy.rd.example.com
Code: GET
Uri-Path: /sensors/

Res: 2.05 Content
Content-Format: application/link-format
Payload:
</sensors/temperature>;if="tag:example.com,sensor"
```

Figure 3: HTTP based discovery and CoAP-over-WS request to a CoAP resource through a has-unique-proxy relation

4.1. Generic proxy advertisements

A third party proxy may advertise its availability to act as a proxy for arbitrary CoAP requests.

[TBD: Specify a mechanism for this; <coap+ws://myself>;rel=has-proxy;anchor="coap://*" for all supported protocols appears to be an obvious but wrong solution.]

The considerations of Section 7.2 apply here.

5. Client picked proxies

When a resource is accessed through an "actual" proxy (i.e., a host between the client and the server, which itself may have a same-host proxy in addition to that), the proxy's choice of the upstream server is originally (i.e., without the mechanisms of this document) either configured (as in a "chain" of proxies) or determined by the request URI (where a proxy picks CoAP over TCP for a request aimed at a coap+tcp URI).

A proxy that has learned, by active solicitation of the information or by consulting links in its cache, that the requested URI is available through a same-host proxy, or that has learned of advertised URI aliasings, may use that information in choosing the upstream transport, and to use responses obtained through one transport to satisfy requests on another.

For example, if a host at coap://h1.example.com has advertised </res>,<coap+tcp://h1.example.com>;rel=has-proxy;anchor="/", then a proxy that has an active CoAP-over-TCP connection to h1.example.com can forward an incoming request for coap://h1.example.com/res through that CoAP-over-TCP connection with a suitable Proxy-Scheme on that connection.

If the host had marked the proxy point as <coap+tcp://h1.example.com>;rel=has-unique-proxy, then the proxy could elide the Proxy-Scheme and Uri-Host options, and would (from the original CoAP caching rules) also be allowed to use any fresh cache representation of coap+tcp://h1.example.com/res to satisfy requests for coap://h1.example.com/res.

6. Related work and applicability to related fields

6.1. On HTTP

The mechanisms introduced here are similar to the Alt-Svc header of [RFC7838] in that they do not create different application-visible addresses, but provide dispatch through lower transport implementations.

Unlike in HTTP, the variations of CoAP protocols each come with their unique URI schemes. Thus, origin URIs can be used without introducing a distrinction between protocol-id and scheme.

To accomodate the message size constraints typical of CoRE environments, and accounting for the differences between HTTP headers and CoAP options, information is delivered once at discovery time.

Using the has-proxy and has-unique-proxy with HTTP URIs as the context is NOT RECOMMENDED; the HTTP provisions the Alt-Svc header and ALPN are preferred.

6.2. Using DNS

As pointed out in [RFC7838], DNS can already serve some of the applications of Alt-Svc and has-unique-proxy by providing different CNAME records. These cover cases of multiple addresses, but not different ports or protocols.

While not specified for CoAP yet (and neither being specified here),

[which is an open discussion point for CoRE -- should we? Here? In a separte DNS-SD document?]

DNS SRV records (possibly in combination with DNS Service Discovery [RFC6763]) can provide records that could be considered equivalent

to has-unique-proxy relations. If _coap._tcp, _coaps._tcp, _coap._udp, _coap+ws._tcp etc. were defined with suitable semantics, these can be equivalent:

``` _coap._udp.device.example.com SRV 0 0 device.example.com 61616 device.example.com AAAA 2001:db8::1

<coap://[2001:db8::1]>;rel=has-unique-proxy;anchor="coap://device.example.com" ```

It would be up to such a specification to give details on what the link's context is; unlike the link based discovery of this document, it would either need to pick one distinguished context scheme for which these records are looked up, or would introduce aliasing on its own.

### 6.3. Using names outside regular DNS

Names that are resolved through different mechanisms than DNS, or names which are defined within the scope of DNS but have no universally valid answers to A/AAAA requests, can be advertised using the relation types defined here and CoAP discovery.

In figure Figure 4, a server using a cryptographic name as described in [I-D.amsuess-t2trg-rdlink] is discovered and used.

```
Req: to [ff02::fd]:5683 on UDP
Code: GET
Uri-Path: /.well-known/core
Uri-Query: rel=has-proxy&anchor=coap://nbswy3dpo5xxe3denbswy3dpo5xxe3de.

Res: from [2001:db8::1]:5683
Content-Format: application/link-format
Payload:
<coap://[2001:db8::1]>;rel=has-unique-proxy;anchor="coap://nbswy3dpo5xxe

Req: to [2001:db8::1]:5683 on TCP
Code: GET
OSCORE: ...
Uri-Path: /sensors/temp
Observe: 0

Res: 2.05 Content
OSCORE: ...
Observe: 0
Payload:
39.1°C
```
```

Figure 4: Obtaining a sensor value from a local device with a global name

7. Security considerations

7.1. Security context propagation

Clients need to strictly enforce the rules of Section 2.2. Failure to do so, in particular using a thusly announced proxy based on a certificate that attests the proxy's name, would allow attackers to circumvent the client's security expectation.

The option to accept credentials suitable for a general trusted proxy is in place for (D)TLS protected scenarios, in which cross-protocol end-to-end protection is not available. Whether a client will recognize certificates for general trusted proxies at all depends on the original proxy setup's security considerations (of [RFC7252] Section 11.2 and [RFC2616] Section 15.7).

7.2. Traffic misdirection

Accepting arbitrary proxies, even with security context propagation performed properly, would attackers to redirect traffic through systems under their control. Not only does that impact availability, it also allows an attacker to observe traffic patterns.

This affects both OSCORE and (D)TLS, as neither protect the participants' network addresses.

Other than the security context propagation rules, there are no hard and general rules about when an advertised proxy is a suitable candidate. Aspects for consideration are:

* When no direct connection is possible (e.g. because the resource to be accessed is served as coap+tcp and TCP is not implemented in the client, or because the resource's host is available on IPv6 while the client has no default IPv6 route), using a proxy is necessary if complete service disruption is to be avoided.

  While an adversary can cause such a situation (e.g. by manipulating routing or DNS entries), such an adversary is usually already in a position to observe traffic patterns.

* A proxy advertised by the device hosting the resource to be accessed is less risky to use than one advertised by a third party.

  Note that in some applications, servers produce representations based on unverified user input. In such cases, and more so when

multiple applications share a security context, the advertisements' provenance may need to be considered.

7.3. Protecting the proxy

A widely published statement about a host's availability as a proxy can cause many clients to attempt to use it.

This is mitigated in well-behaved clients by observing the rate limits of [RFC7252], and by ceasing attempts to reach a proxy for the Max-Age of received errors.

Operators can further limit ill-effects by ensuring that their client systems do not needlessly use proxies advertised in an unsecured way, and by providing own proxies when their clients need them.

7.4. Implementing proxies

Proxies that are trusted (i.e., that terminate (D)TLS connections and have an own server certificate) need to consider the same aspects as clients for their client-side interface as all other clients.

Proxies can often process data from different security contexts. When they do, care needs to be taken to not apply has-proxy statements across security contexts. (This consideration is not specific to proxies, but comes up more frequently there).

8. IANA considerations

8.1. Link Relation Types

IANA is asked to add two entries into the Link Relation Type Registry last updated in [RFC8288]:

Relation Name	Description	Reference
has-proxy	The link target can be used as a proxy to reach the link context.	RFCthis
has-unique-proxy	Like has-proxy, and using this proxy implies scheme and host of the target.	RFCthis

Table 1: New Link Relation types

9. References

9.1. Normative References

[RFC7252]

Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://doi.org/10.17487/RFC7252>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://doi.org/10.17487/RFC8288>.

9.2. Informative References

[aliases] W3C, "Architecture of the World Wide Web, Section 2.3.1 URI aliases", n.d., <https://www.w3.org/TR/webarch/#uri-aliases>.

[cooluris] BL, T., "Cool URIs don't change", n.d., <https://www.w3.org/Provider/Style/URI>.

[I-D.amsuess-core-coap-over-gatt]
Amsüss, C., "CoAP over GATT (Bluetooth Low Energy Generic Attributes)", Work in Progress, Internet-Draft, draft-amsuess-core-coap-over-gatt-01, 2 November 2020, <https://datatracker.ietf.org/doc/html/draft-amsuess-core-coap-over-gatt-01>.

[I-D.amsuess-t2trg-rdlink]
Amsüss, C., "rdlink: Robust distributed links to constrained devices", Work in Progress, Internet-Draft, draft-amsuess-t2trg-rdlink-01, 23 September 2019, <https://datatracker.ietf.org/doc/html/draft-amsuess-t2trg-rdlink-01>.

[I-D.bormann-t2trg-slipmux] Bormann, C. and T. Kaupat, "Slipmux: Using an UART interface for diagnostics, configuration, and packet transfer", Work in Progress, Internet-Draft, draft-bormann-t2trg-slipmux-03, 4 November 2019, <https://datatracker.ietf.org/doc/html/draft-bormann-t2trg-slipmux-03>.

[I-D.ietf-core-resource-directory] Amsüss, C., Shelby, Z., Koster, M., Bormann, C., and P. V. D. Stok, "CoRE Resource Directory", Work in Progress, Internet-Draft, draft-ietf-core-resource-directory-28, 7 March 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-core-resource-directory-28>.

[I-D.ietf-lpwan-coap-static-context-hc] Minaburo, A., Toutain, L., and R. Andreasen, "Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-lpwan-coap-static-context-hc-19, 8 March 2021, <https://

datatracker.ietf.org/doc/html/draft-ietf-lpwan-coap-static-context-hc-19>.

[I-D.silverajan-core-coap-protocol-negotiation] Silverajan, B. and M. Ocak, "CoAP Protocol Negotiation", Work in Progress, Internet-Draft, draft-silverajan-core-coap-protocol-negotiation-09, 2 July 2018, <https://datatracker.ietf.org/doc/html/draft-silverajan-core-coap-protocol-negotiation-09>.

[lwm2m] OMA SpecWorks, "White Paper - Lightweight M2M 1.1", n.d., <https://omaspecworks.org/white-paper-lightweight-m2m-1-1/>.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <https://doi.org/10.17487/RFC2616>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://doi.org/10.17487/RFC6690>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://doi.org/10.17487/RFC6763>.

[RFC7838] Nottingham, M., McManus, P., and J. Reschke, "HTTP Alternative Services", RFC 7838, DOI 10.17487/RFC7838, April 2016, <https://doi.org/10.17487/RFC7838>.

[RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, DOI 10.17487/RFC8323, February 2018, <https://doi.org/10.17487/RFC8323>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://doi.org/10.17487/RFC8613>.

Appendix A. Change log

Since -00:

*Added introduction

*Added examples

*Added SCHC analogy

*Expanded security considerations

*Added guidance on choice of transport, and canonical addresses

*Added subsection on interaction with a Resource Directory

*Added comparisons with related work, including rdlink and DNS-SD sketches

*Added IANA considerations

*Added section on open questions

Appendix B. Open Questions / further ideas

*OSCORE interaction: [RFC8613] Section 4.1.3.2 requirements place OSCORE use in a weird category between has-proxy and has-unique-proxy (because if routing still works, the result will be correct). Not sure how to write this down properly, or whether it's actionable at all.

Possibly there is an inbetween category of "The host needs the Uri-Host etc. when accessed through CoAP, but because the host does not use the same OSCORE KID across different virtual hosts, it's has-unique-proxy as soon as you talk OSCORE".

*Self-uniqueness:

A host that wants to indicate that it doesn't care about Uri-Host can probably publish something like </>;rel=has-unique-proxy to do so.

This'd help applications justify when they can elide the Uri-Host, even when no different protocols are involved.

*Advertising under a stable name:

If a host wants to advertise its host name rather than its IP address during multicast, how does it best do that?

Options, when answering from 2001:db8::1 to a request to ff02::fd are:

<coap://myhostname/foo>,...,
<coap://[2001:db8::1]>;rel=has-unique-proxy;anchor="coap://myhostname"

which is verbose but formally clear, and

```
</foo>,...,
<coap://[2001:db8::1]>;rel=has-unique-proxy;anchor="coap://myhostname"
```

which is compatible with unaware clients, but its correctnes with respect to canonical URIs needs to be argued by the client, in this sequence

- understanding the has-unique-proxy line,

- understanding that the request that went to 2001:db8::1 was really a Proxy-Scheme/Uri-Host-elided version of a request to coap://myhostname, and then

- processing any relative reference with this new base in mind.

(Not that it'd need to happen in software in that sequence, but that's the sequence needed to understand how the /foo here is really coap://myhostname/foo).

Appendix C. Acknowledgements

This document heavily builds on concepts explored by Bill Silverajan and Mert Ocak in [I-D.silverajan-core-coap-protocol-negotiation], and together with Ines Robles and Klaus Hartke inside T2TRG.

Author's Address

Christian Amsüss
Hollandstr. 12/4
1020
Austria

Phone: +43-664-9790639
Email: christian@amsuess.com