CoAP Protocol Indication

draft-amsuess-core-transport-indication-01

which you may know from having piggybacked on resource-directory-extensions on IETF110

Christian Amsüss

2021-07-28, IETF111

A brief history of CoAP schemes

2014	RFC7252	`coap` for UDP
2015–2017	`coap-tcp-tls`$\leq$`08`	`coap+tcp` for TCP
2017	`coap-tcp-tls-09`	`coap` for TCP or UDP
	`coap-tcp-tls`$\geq$`10`	`coap+tcp` for TCP

A brief history of CoAP schemes

2014	RFC7252	`coap` for UDP
2015–2017	`coap-tcp-tls`$\leq$`08`	`coap+tcp` for TCP
2017	`coap-tcp-tls-09`	`coap` for TCP or UDP
	`coap-tcp-tls`$\geq$`10`	`coap+tcp` for TCP

From No-Objection ballots

...these scheme registrations [...] present an "antipattern" ...

This runs counter to the principle that a URI identifies a resource ...

I am perplexed that no concrete mechanism for UDP/TCP failover is provided ...

It was known not to be easy

From Bill's 2014 (IETF 89) presentation

Situating Transport Information in CoAP URI

Transport Information	Req 4.1.1	Req 4.1.2	Req 4.1.3	Req 4.1.4
Scheme				
Authority				
Rootless Path				

From Section 4.1, draft-silverajan-core-coap-alternative-transports-04:

- Req 4.1.1: Conformance to RFC3986 syntax and algorithms
- Req 4.1.2: Preserving transport info when relative references are encountered
- Req 4.1.3: Avoiding URI aliasing with multiple transports
- Req 4.1.4: Avoiding heavy DNS reliance

URI aliasing pain points

`coaps://[2001:db8::1]/cfe` $\stackrel{?}{=}$ `coaps+tcp://[2001:db8::1]/cfe`

Pick one side:

- Multiple entries in discovery
- Multiple entries in caches
- Transports stay unused
- Devices can not connect[1]

Proxies can't pick transports according to their abilities.

No established terminology to describe URI aliasing.

[1] CoAP-over-UDP was never described as mandatory-to-implement

Tools we have

- Easy resource metadata

  `<coap+tcp://[2001:db8::1]>;rel=...`

  (with some indirection to make site-wide statements)

Tools we have

- Easy resource metadata

  ```
  <coap+tcp://[2001:db8::1]>;rel=...
  ```

  (with some indirection to make site-wide statements)

- Cheap proxying

  ```
  CON GET
  Observe: 0
  Uri-Path: "cfe"
  +Proxy-Scheme: "coap"
  ```

  (with triviality bonus points for implementations ignoring the 'critical' flag)

Putting it together

```
</cfe>;rt="tag:...:coffemachine";rel=hosts;anchor="/",
<coap+tcp://[2001:db8::1]>;rel=has-proxy;anchor="/"
```

Goals (1-2/5)

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.

Server implementation: Just accept provided Proxy-Scheme options.

Client implementation: Ignore, or use indicated protocol and add Proxy-Scheme (and, if needed, Uri-Host) option.

Message overhead kills

```
 CON GET
 Observe: 0
 Uri-Path: "cfe"
+Proxy-Scheme: "coap"
```

$\sim$ 5 bytes per request. More if host names are involved.

Goals (3/5)

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

`rel=has-unique-proxy` additionally means you can skip Proxy-Scheme and Uri-Host

Proxy interaction

Goals (4-5/5)

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.

...which I'll be happy to elaborate on in hallway discussions.

Just As With Any Proxy.

OK, there's more in the text, but that's the gist.

Problematic with third-party protocol translation services:
What's done by (D)TLS users here? Do they use proxies at all?
Are all-valid certificates common there? Do we want to endorse them?

Take-home message

- It can probably be just this simple.
- No URI aliasing introduced in applications.

Questions? Comments? Interest?

Backup slide / FAQ

Didn't we want to do this with DNS?

We[2] still can, just need to phrase the equivalent statements in DNS.

Straw man for "`coap://device.example.com` has CoAP-over-TCP running on port 1234":

```
_has-coap-proxy._tcp.device.example.com SRV 0 0 device.example.com 1234
device.example.com AAAA 2001:db8::1
```

How does this relate to HTTP's `Alt-Svc`?

Generally similar; links instead of headers (as common in CoAP), and no need for protocol-id because we have schemes already.

[2] Whoever wants to use it will need to volunteer as coauthor.