Proxy Operations for CoAP Group Communication

draft-tiloca-core-groupcomm-proxy-05

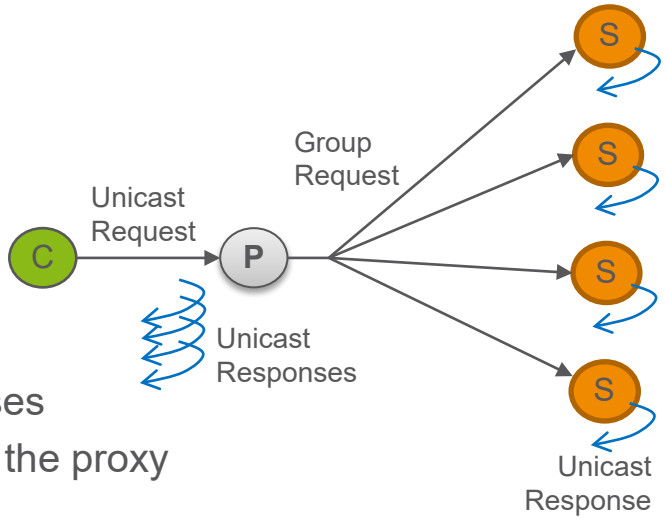
Marco Tiloca, RISE
Esko Dijk, IoTconsultancy.nl

CoRE WG interim meeting, October 27th, 2021

Recap

- **CoAP supports group communication over IP multicast**
  - Section 3.5 of *draft-ietf-core-groupcomm-bis* discusses issues when using a proxy
  - The proxy forwards a request to the group of servers, over IP multicast
  - Handling responses and relaying them back to the client is not trivial
- **Contribution – Definition of proxy operations for CoAP group communication**
  - Addressed all issues in *draft-ietf-core-groupcomm-bis*
  - Signaling protocol between client and proxy, with two new CoAP options
  - Individual responses from the CoAP servers relayed back to the client
  - Support for forward-proxies, reverse-proxies and a chain of proxies
- **The proxy is explicitly configured to support group communication**
  - Clients are allowed-listed on the proxy, and identified by the proxy

How it works



- **In the <u>unicast</u> request addressed to the proxy, the client indicates:**
  - To be interested / capable of handling multiple responses
  - For how long the proxy should collect and forward responses
  - In the new CoAP option **Multicast-Signaling**, removed by the proxy
- **In each response to the group request, the proxy includes the server address**
  - In the new CoAP option **Response-Forwarding**
  - The client can distinguish responses and different servers
  - The client can later contact an individual server (directly, or again via the proxy)
- **Group OSCORE can be used for end-to-end security between client and servers**
- **Security between Client and Proxy, especially to identify the Client**
  - (D)TLS or OSCORE (see *draft-tiloca-core-oscore-capable-proxies*)

Updates since IETF 110

- **Caching model at the proxy fully specified in this document, see Section 7**
  - It was mostly in *draft-ietf-core-groupcomm-bis* , agreed to move it here
  - Freshness & validation model, aligned to *groupcomm-bis*, with many inputs from Christian
    - The proxy fully serves from its cache only if fully aware of the servers in the group
    - Client-Servers and Proxy-Servers validation based on ETag, as in *groupcomm-bis*
    - Client-Proxy validation is possible, using the new Group-ETag Option
  - Dedicated section on caching of responses protected end-to-end with Group OSCORE
    - Based on *draft-amsuess-core-cachable-oscore*

- **Clarified rationale of response forwarding**
  - The proxy may receive >1 responses to the same group request from the same server
  - Those are also forwarded “as they come”, possibly updating cache entries
  - The client application has better context to deal with the >1 responses

Updates since IETF 110

- **Removed appendix about OSCORE between client and proxy**
  - Agreed that this method has broader applicability and more use cases
  - Now defined in a separate document: *draft-tiloca-core-oscore-capable-proxies*

- **Latest main addition: support for HTTP-CoAP proxies**
  - Build on first steps taken in Section 10 of RFC7252 and in RFC8075
  - If Group OSCORE is used end-to-end, the mapping from Section 11 of RFC8613 is used
  - Adapted version of the approach for a CoAP-CoAP proxy
    - Defined HTTP header fields corresponding to the new CoAP options
    - A single HTTP "batch" response is sent to the client, including *N* HTTP responses
      - Outer Content-Type: multipart/mixed
      - Part Content-Type: application/http (*N* instances)
      - Content-Type of each part's body: <Content-Type of the server's response>

HTTP-CoAP proxy example

```
POST https://proxy.url/hc/?target_uri=coap://G_ADDR:G_PORT/ HTTP/1.1
Content-Length: <REQUEST_TOTAL_CONTENT_LENGTH>
Content-Type: text/plain
Multicast-Signaling: 60

Body: Do that!
```

- C → P : HTTP unicast group request
  - P converts it to a CoAP group request
  - Forwarded to **coap://G_ADDR:G_PORT**
- P accepts responses for **60 s**
- S1 → P : CoAP response
  - Converted to HTTP and stored
- S2 → P : CoAP response
  - Converted to HTTP and stored

… … … TIMEOUT!

```
HTTP/1.1 200 OK
Content-Length: <BATCH_RESPONSE_TOTAL_CONTENT_LENGTH>
Content-Type: multipart/mixed; boundary=batch_foo_bar

--batch_foo_bar
Content-Type: application/http

HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: <INDIVIDUAL_RESPONSE_1_CONTENT_LENGTH>
Response-Forwarding: coap://S1_ADDR

Body: Done!
--batch_foo_bar
Content-Type: application/http

HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: <INDIVIDUAL_RESPONSE_2_CONTENT_LENGTH>
Response-Forwarding: coap://S2_ADDR:S2_PORT

Body: More than done!
--batch_foo_bar
```

- P prepares one HTTP “batch” response
  - Include the different individual responses, one for each replying server
- P → C : HTTP “batch” response
- C extracts the individual HTTP responses from the “batch” response

Summary

- **Latest additions**
  - Clarified rationale of response forwarding at the proxy
  - Caching model at the proxy (both freshness and validation), on different legs
  - OSCORE use with proxy / intermediary moved out to a separate document
  - Use of HTTP-CoAP proxies → An HTTP client can talk to a CoAP group, also with Group OSCORE

- **Next steps**
  - Use CRIs (*draft-ietf-core-href*) for server addressing information in the Response-Forwarding Option
  - Add more examples with CoAP-CoAP reverse-proxies
  - HTTP-CoAP proxies
    - See if individual responses can be relayed as a stream, through Transfer-Encoding: Chunked
    - Add security considerations revising those from RFC8075

- **Need for reviews – Previously promised: Christian, Carsten**

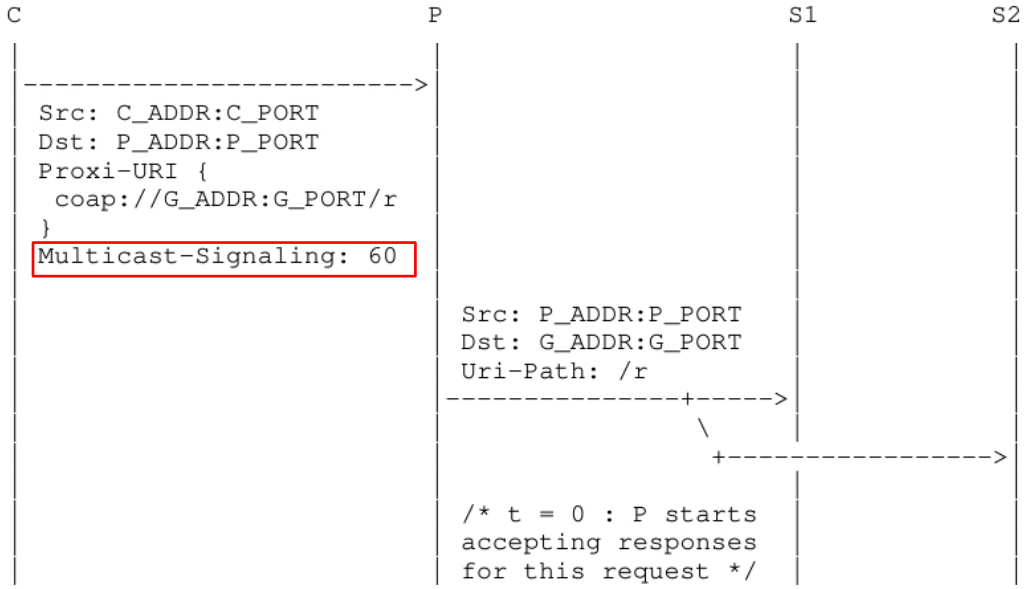
Thank you!

Comments/questions?

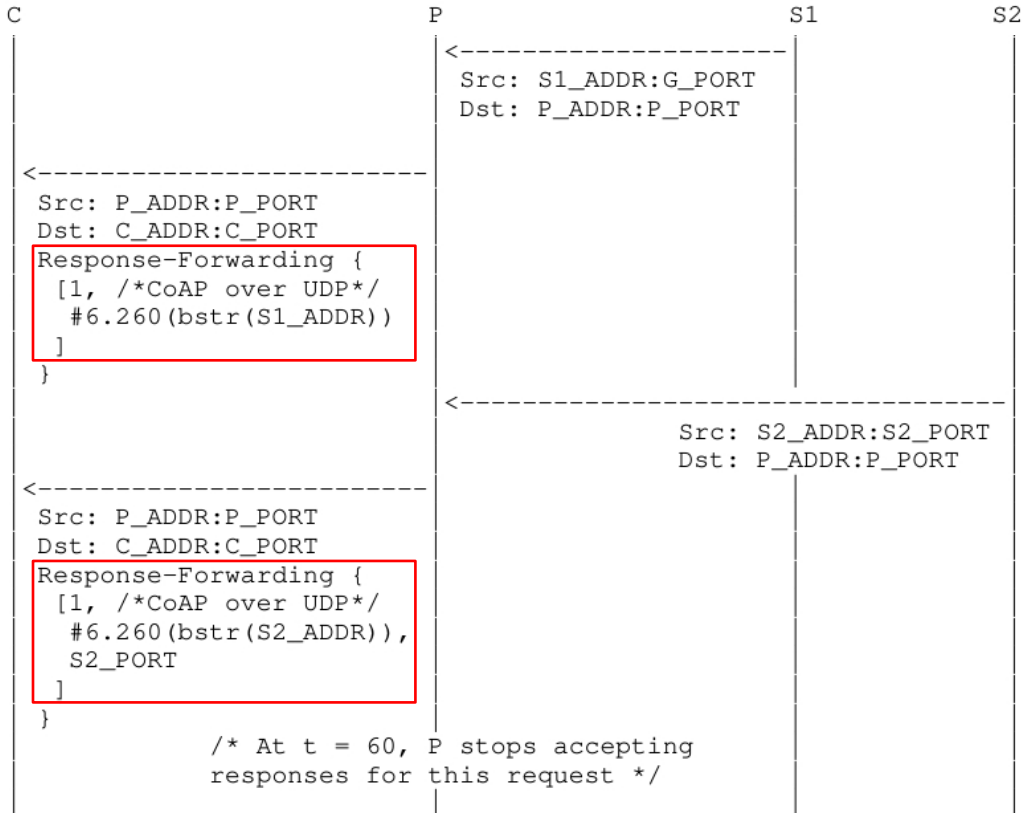
https://gitlab.com/crimson84/draft-tiloca-core-groupcomm-proxy

Backup

Example with forward-proxy (1/2)

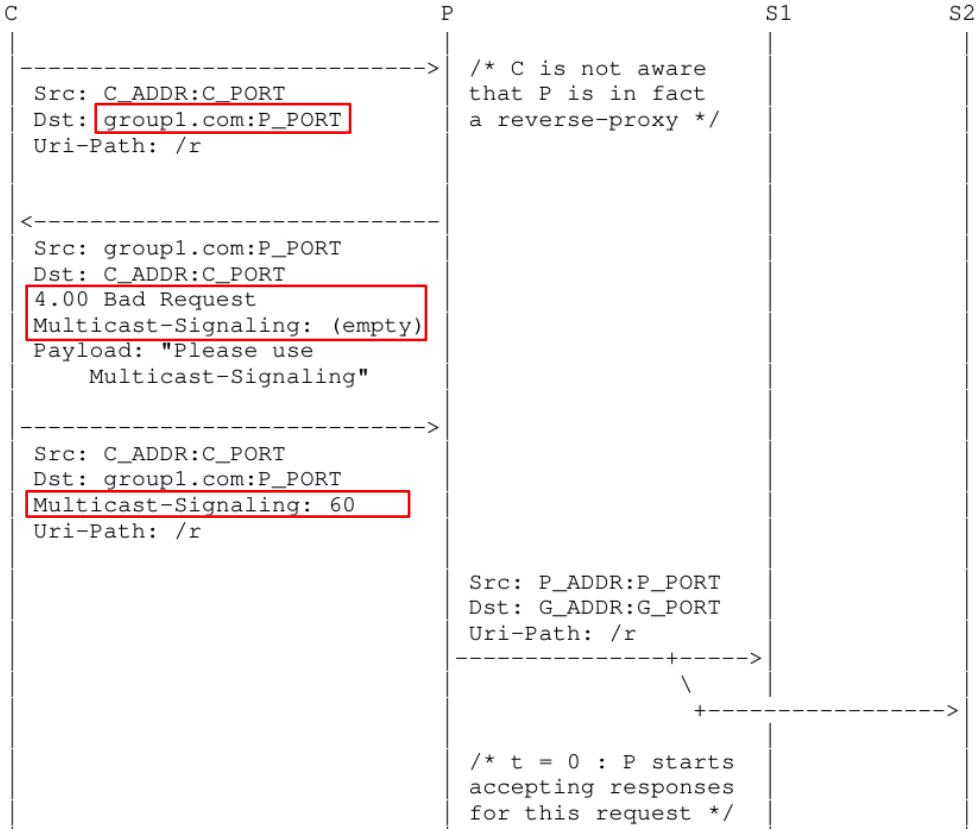


Example with forward-proxy (2/2)



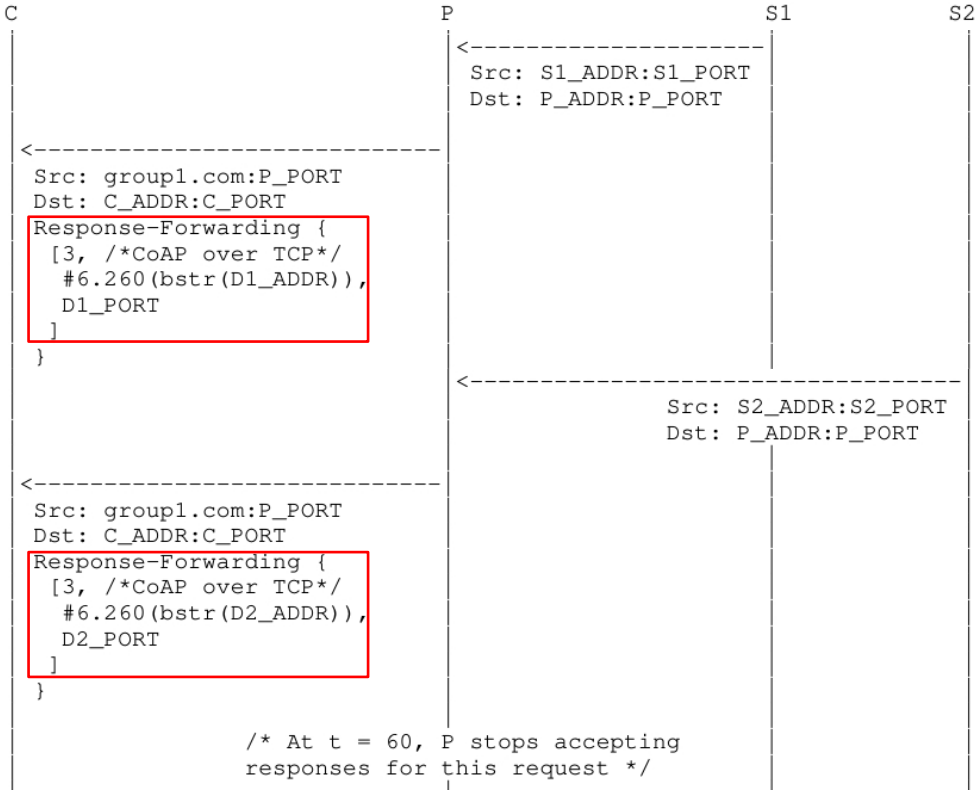
Example with reverse-proxy (1/3)

- C→P: CoAP over TCP
- group1.com resolves to the address of P
- The proxy hides the group as a whole and the individual servers



Example with reverse-proxy (2/3)

- C→P: CoAP over TCP
- group1.com resolves to the address of P
- The proxy hides the group as a whole and the individual servers
- Dx_ADDR:Dx_PORT is mapped to address and port of server Sx



Example with reverse-proxy (3/3)

- C→P: CoAP over TCP
- group1.com resolves to the address of P
- The proxy hides the group as a whole and the individual servers
- Dx_ADDR:Dx_PORT is mapped to address and port of server Sx