Proxy Operations for CoAP Group Communication

draft-tiloca-core-groupcomm-proxy-06

Marco Tiloca, RISE
Esko Dijk, IoTconsultancy.nl

CoRE WG interim meeting, May 25th, 2022

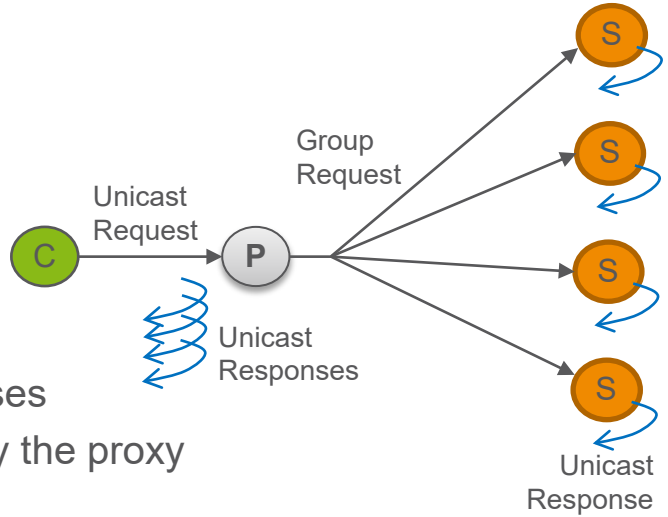
Recap

- **CoAP supports group communication, e.g., 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, e.g., 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, chain of proxies and HTTP-CoAP proxies

- **Proxy is explicitly configured to support group communication**
  - Clients are allowed-listed on the proxy, and identified by the proxy

Message forwarding



- **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
  - with the new CoAP option **Multicast-Timeout**, 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 version -05 (1/3)

- **Last presentation, of version -05, at the CoRE interim on 2021-10-27**
- **Version -06 submitted before IETF 113 (but not presented yet)**
- **"Multicast-Timeout" Option**
  - Renamed from "Multicast-Signaling", as suggested by Carsten
  - Max length of uint reduced to 4 bytes, as suggested by Christian
- **"Response-Forwarding" Option**
  - Updated semantics on “srv_port” port number - null or absent (swapped)
  - null → same port as destination port number of the group request
  - absent → default port number for transport protocol used in the group request ~ 5683

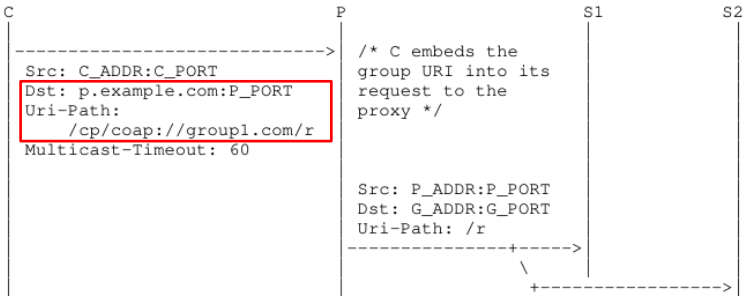
Updates since version -05 (2/3)

- **Improved processing on a reverse-proxy (Section 6.2)**
  - Proxy may rely on a default timeout for accepting responses
  - Client may omit the "Multicast-Timeout" Option to use the default timeout
  - Clients need to be aware of this configuration, which is expected if they are registered and allow-listed at the proxy

- **Placeholder notes on response revalidation**
  - Between proxy and servers, when Group OSCORE is used end-to-end
  - Revalidation might be enabled through an outer ETag for the proxy, but …
  - … cacheable OSCORE had to be used in the first place
    - https://datatracker.ietf.org/doc/draft-amsuess-core-cachable-oscore/
  - Then use of outer ETag can be defined in draft-amsuess-core-cachable-oscore

Updates since version -05 (3/3)

- **Added one more example with a reverse proxy (Appendix A.1)**
  - Only 1 address required at the proxy, rather than
    - 1 address per group
    - 1 address per server in the group
  - Request target expressed in URI-path (RFC8075)
  - Scalable with number of groups and group size



- **HTTP-CoAP proxies**
  - Added processing for HTTP-CoAP reverse-proxy (Section 9.10)
  - Placeholder TODO notes on using streamed delivery of responses using the Transfer-Coding "Chunked“ (RFC 7230), as suggested by Christian (Section 9.9)
    - Yes, it’s doable! To be turned into full text together with an example

- **Clarifications and editorial improvements**

Features at a glance (v -06)

- **All the issues highlighted in Section 3.5 *core-groupcomm-bis* are addressed**
- **Signaling protocol for message forwarding through the proxy**
  - "Multicast-Timeout" Option included by the client in the request to the proxy
  - "Response-Forwarding" Option included by the proxy in the relayed responses
- **Caching of responses at the proxy**
  - Plus response validation between the proxy and the servers in the group
  - Plus response validation between the client and the proxy, with a new CoAP Option "Group-ETag“
  - Note: *core-groupcomm-bis* defines caching at the client and validation between client and servers
- **Support for both forward-proxies and reverse-proxies**
  - CoAP-CoAP proxies, with examples
  - HTTP-CoAP proxies, with examples
  - In a chain of proxies

Relation to other documents

- **Case in point in *draft-bormann-core-responses***
  - Multiple (non-traditional) responses to a same request, coming from members of a CoAP group
  - Use of the "Multicast-Timeout" Option to provide the proxy with a time indication of interest

- **Use case in *draft-tiloca-core-oscore-capable-proxies***
  - In scenarios when:
    - OSCORE is used between client and proxy, also but not only for client authentication; and/or
    - Group OSCORE is used end-to-end between client and servers → OSCORE-in-OSCORE

- **Referred concrete approach to address issues from *draft-ietf-core-groupcomm-bis***
  - In *draft-core-groupcomm-bis*, issues when using proxies are highlighted but not addressed
  - Agreed that concrete approaches are to be defined in separate documents
  - From Carsten's WGLC review [1] of *draft-core-groupcomm-bis*:

"In several places, the document relies heavily on draft-tiloca-core-groupcomm-proxy supplying solutions for what it itself needs to leave open. I believe we should at least have accepted that as a WG document before we pass on draft-ietf-core-groupcomm-bis to the IESG."

[1] https://mailarchive.ietf.org/arch/msg/core/PtqtDE_3PWR-n-o_z9h0HxW2vDI/

Summary

- **Main latest additions**
  - Revised name and semantics of the new CoAP Options
  - Reverse-proxies can rely on a default timeout for relaying responses
  - New example with reverse-proxy needing only one address and using RFC 8075 style proxy request

- **Planned next steps**
  - Align with terminology and concepts from *draft-bormann-core-responses*
  - Use **CRIs** (*draft-ietf-core-href*) for server addressing information in the “Response-Forwarding” Option
  - “Cancellation”: Allow clients to stop the proxy relaying responses early, i.e., before timeout expiration
  - HTTP-CoAP proxies
    - Define and add examples on relaying responses as a stream, with Transfer-Encoding “Chunked”
    - Add security considerations revising those from RFC 8075, for the groupcomm case

- **V -06 has all the main functionalities stable, and a clear relation with other documents**

- **Working Group Adoption ?**

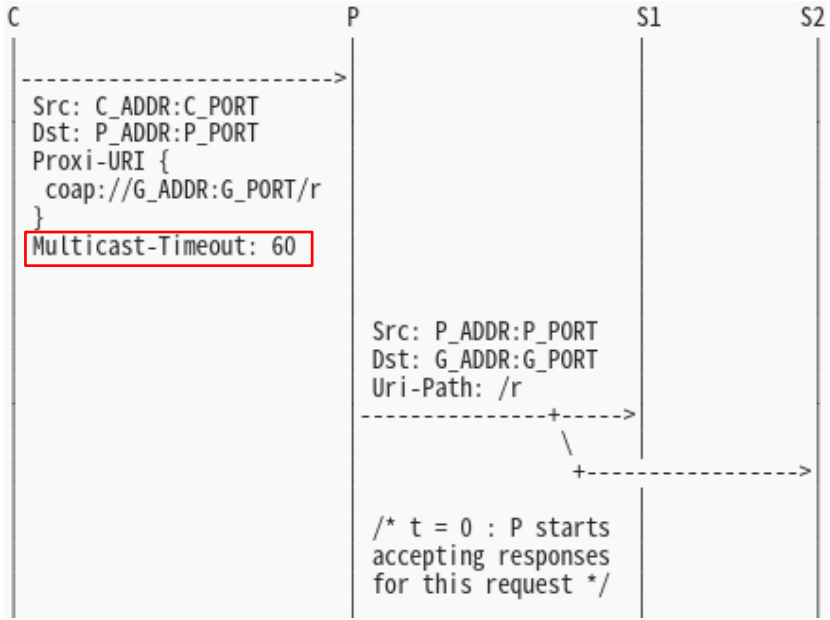
Thank you!

Comments/questions?

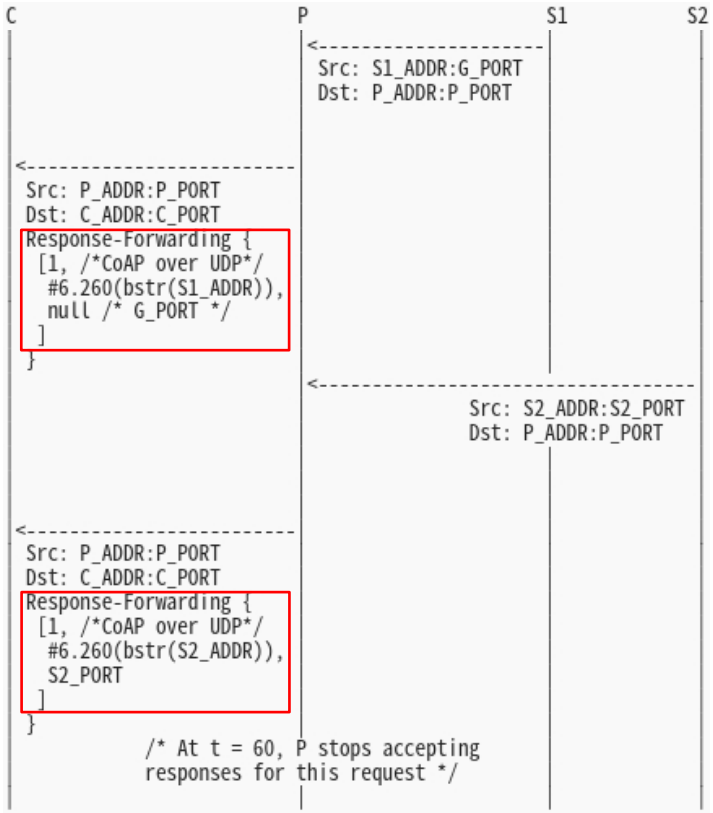
https://gitlab.com/crimson84/draft-tiloca-core-groupcomm-proxy

Backup

Example with forward-proxy (1/2)

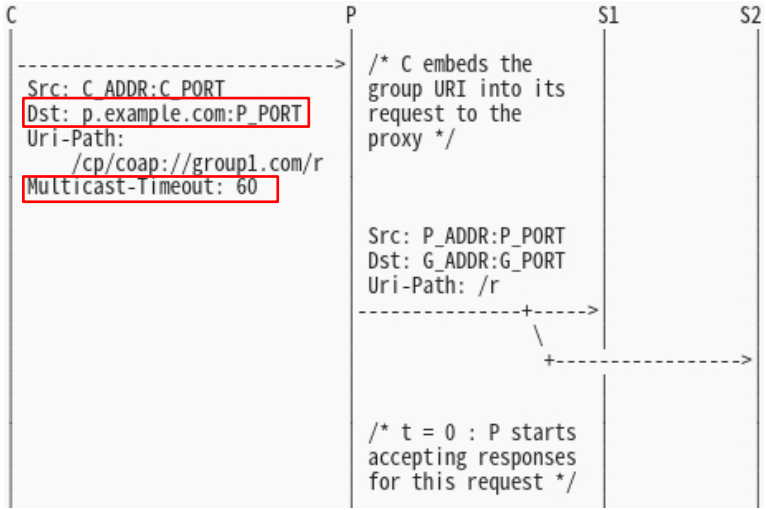


Example with forward-proxy (2/2)



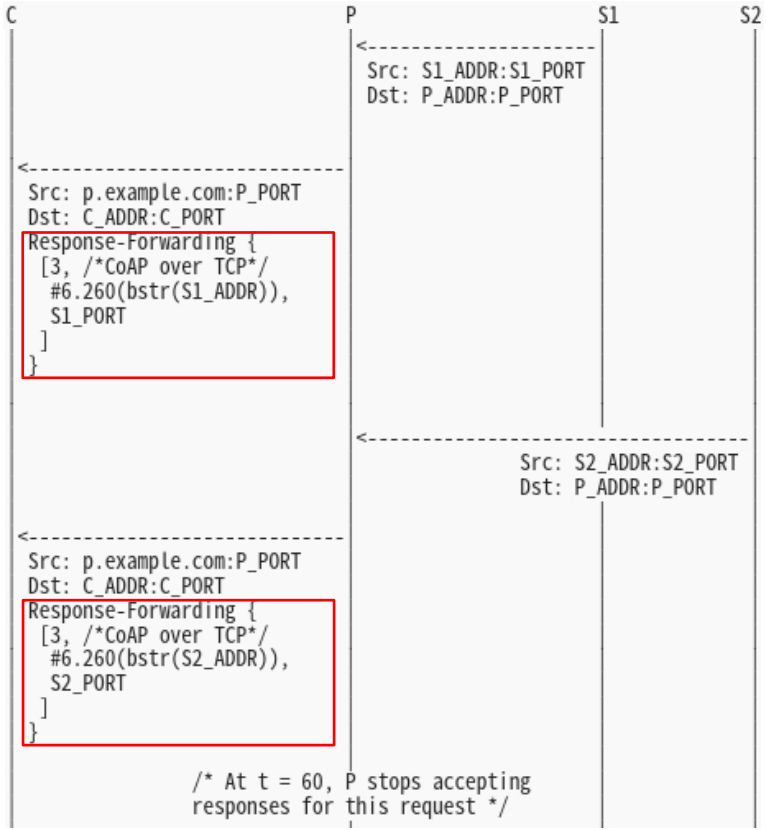
Example #1 with reverse-proxy (1/3)

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



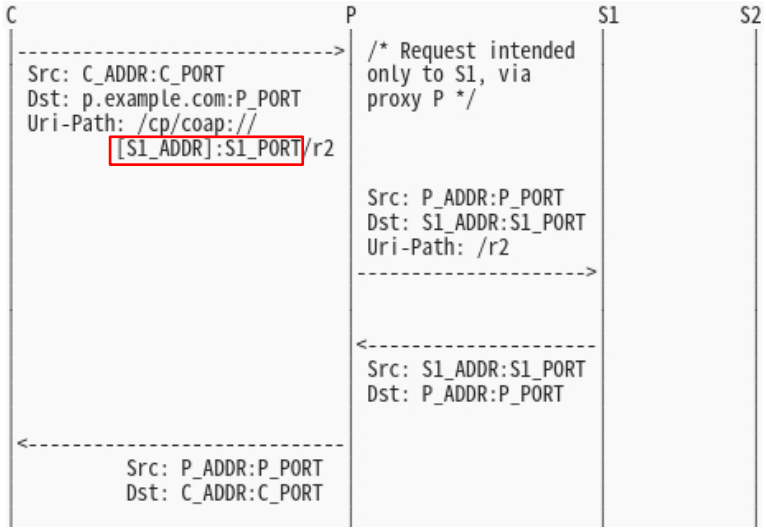
Example #1 with reverse-proxy (2/3)

- C→P: CoAP over TCP
- p.example.com resolves to the address of P
- group1.com resolves to the multicast address of the group
- 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



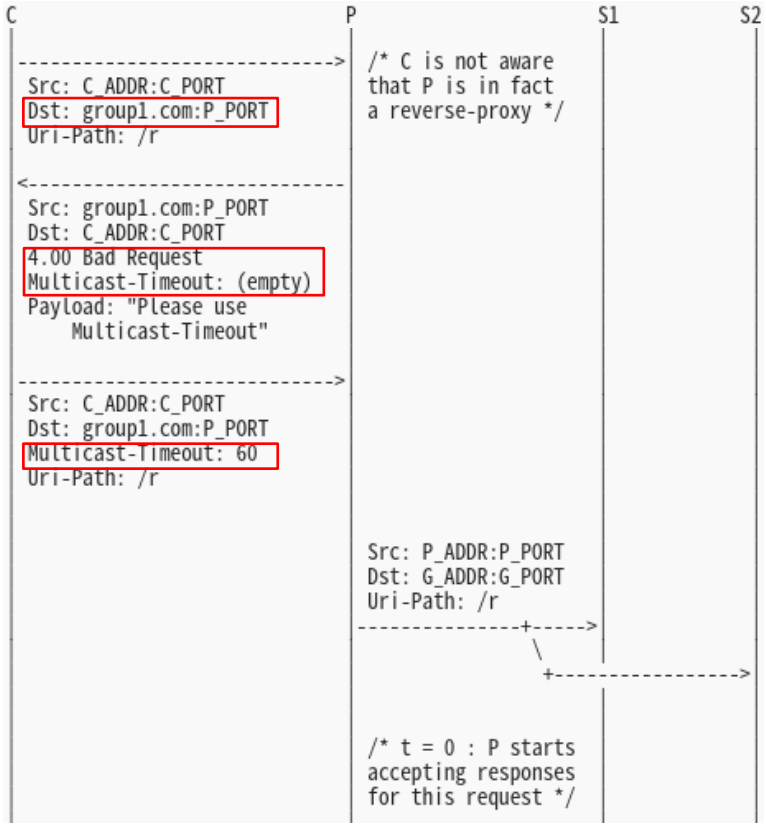
Example #1 with reverse-proxy (3/3)

- C→P: CoAP over TCP
- p.example.com resolves to the address of P
- group1.com resolves to the multicast address of the group
- 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



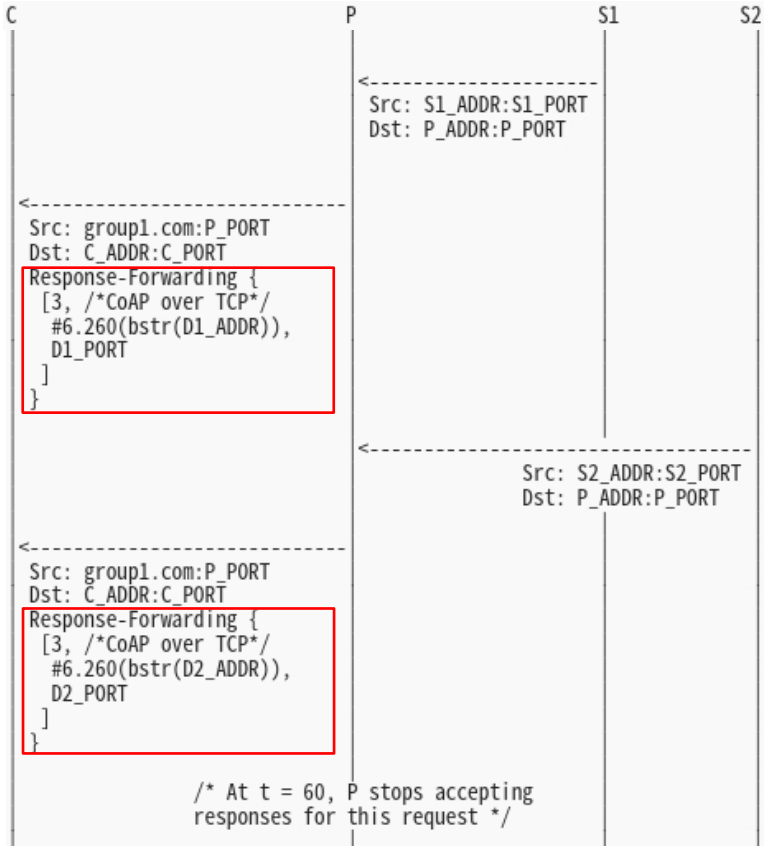
Example #2 with reverse-proxy (1/3)

- 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



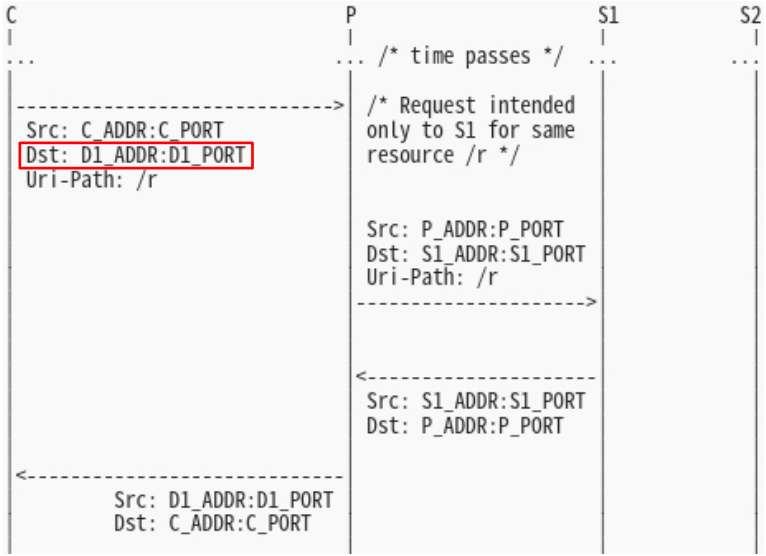
Example #2 with reverse-proxy (2/3)

- 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 #2 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



Example with HTTP-CoAP proxy

```
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-Timeout: 60
```

- 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!

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

```
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:G_PORT

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--
```

- C extracts the individual HTTP responses from the “batch” response