Proxy Operations for CoAP Group Communication

draft-tiloca-core-groupcomm-proxy-03

Marco Tiloca, RISE
Esko Dijk, IoTconsultancy.nl

IETF 110, CoRE WG, March 8th, 2021

Recap

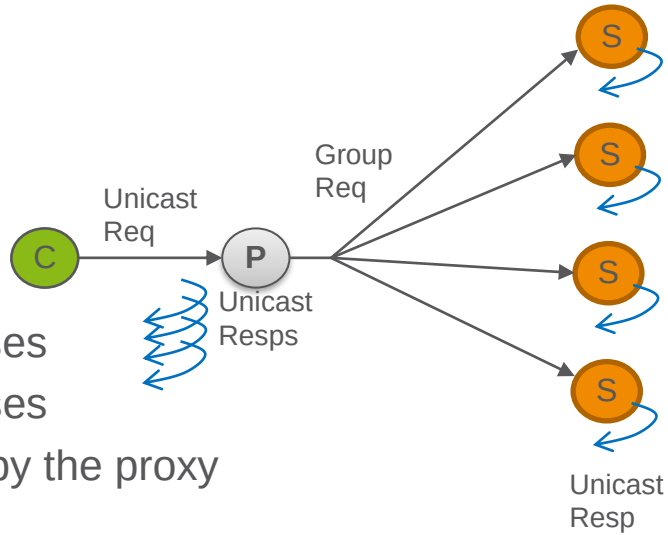
- CoAP supports group communication over IP multicast
  - Section 3.4 of *draft-ietf-core-groupcomm-bis* discusses issues when using a proxy
  - The proxy forwards a request to the group of servers, over IP multicast
  - Handling responses and forwarding them back to the client is not trivial

- Contribution – Description 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
  - Responses individually forwarded back to the client

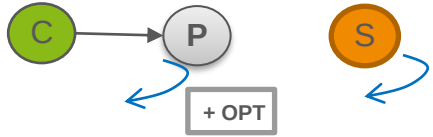
- The proxy is explicitly configured to support group communication
  - Clients are allowed-listed on the proxy, and identified by the proxy

How it works

- In the unicast request addressed to the proxy, the client indicates:
  - To be interested / capable of handling multiple responses
  - How long the proxy should collect and forward responses
  - The new CoAP option **Multicast-Signaling**, removed by the proxy
- In each response to a 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 contact an individual server (directly, or again via the proxy)
- Group OSCORE can be used for e2e security between client and servers
- OSCORE can be used between Client and Proxy (Appendix A), or DTLS



Updates from -03



- Only for P → C responses
  - Value: addressing information about the server (from the original response)
  - The proxy adds the option, before forwarding the response to the client
  - Presence: the client can distinguish responses and origin servers

No.	C	U	N	R	Name	Format	Length	Default
TBD2			-		Response-Forwarding	(*)	9-24	(none)

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

Response-Forwarding Option

- Format of option value aligned with [1]
  - Serialization of a '*tp_info*' array
  - Thanks to Christian for the suggestion!

```
tp_info = [
      tp_id : 1,              ; UDP as transport protocol
   srv_host : #6.260(bstr),   ; IP address where to reach the server
 ? srv_port : uint / null     ; Port number where to reach the server
]
```

- Assumed '*srv_port*' values when '*tp_id*' = 1 (UDP)
  - If present with value Null → default CoAP port number 5683
  - If not present → same as destination port number of the proxied group request

[1] *draft-tiloca-core-observe-multicast-notifications-05*

Updates from -03

- New registrations, for '*tp_id*' different than 1 (UDP)
  - "CoAP Transport Information" Registry [1]
  - Useful here already, when using a reverse-proxy
- "CoAP Transport Information" Registry [1]
- Section 3.1
  - Encoding of elements in "Srv Addr" and "Req Info"
- Section 3.2
  - Default value to use, if 'srv_port' has value Null

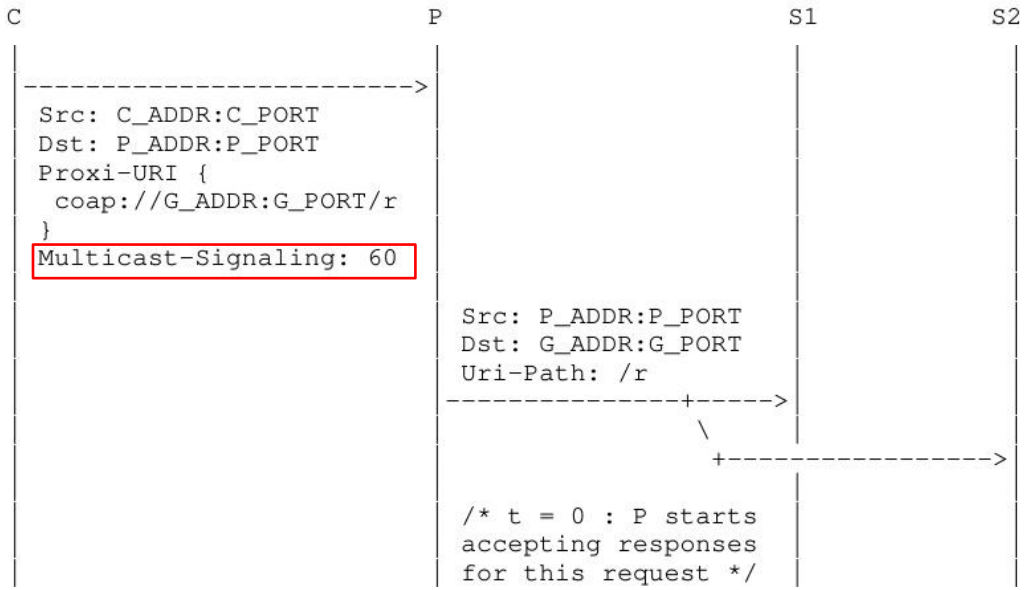
Transport Protocol	Description	Value	Srv Addr	Req Info	Reference
UDP secured with DTLS	UDP with DTLS is used as per RFC8323	2	tp_id srv_host srv_port	token cli_host ?cli_port	[This document]
TCP	TCP is used as per RFC8323	3	tp_id srv_host srv_port	token cli_host ?cli_port	[This document]
TCP secured with TLS	TCP with TLS is used as per RFC8323	4	tp_id srv_host srv_port	token cli_host ?cli_port	[This document]
WebSockets	WebSockets are used as per RFC8323	5	tp_id srv_host srv_port	token cli_host ?cli_port	[This document]
WebSockets secured with TLS	WebSockets with TLS are used as per RFC8323	6	tp_id srv_host srv_port	token cli_host ?cli_port	[This document]

[1] *draft-tiloca-core-observe-multicast-notifications-05*

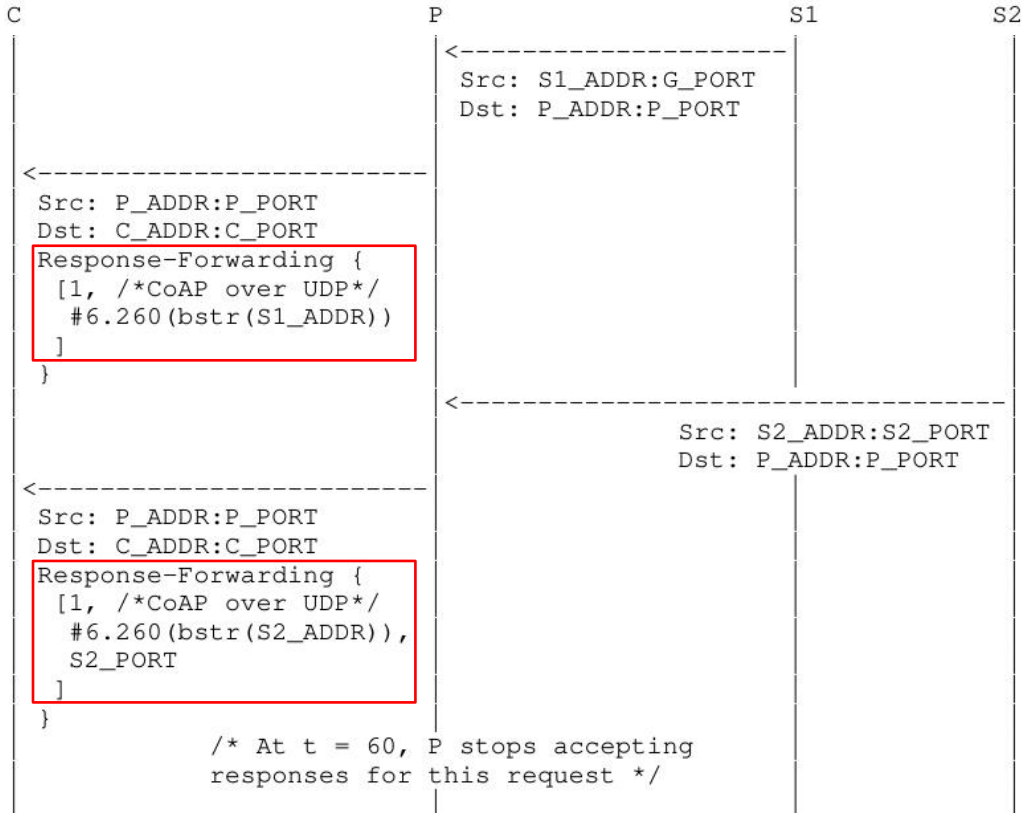
Updates from -03

- Support for reverse-proxies added
  - Thanks to Christian also for this suggestion!
- A client aware of server being a reverse-proxy:
  - MUST use the Multicast-Signaling Option (under discussion: issue #19)
  - Client acts similar to the case of the forward-proxy
- The reverse-proxy:
  - Processes the new options like a forward-proxy
  - Possibly “reveals itself” as a reverse-proxy
    - If it receives a group request without Multicast-Signaling option, and one is required, then …
    - it returns a 4.00 response, including an empty Multicast-Signaling option
- No difference for the servers

Example with forward-proxy (1/2)

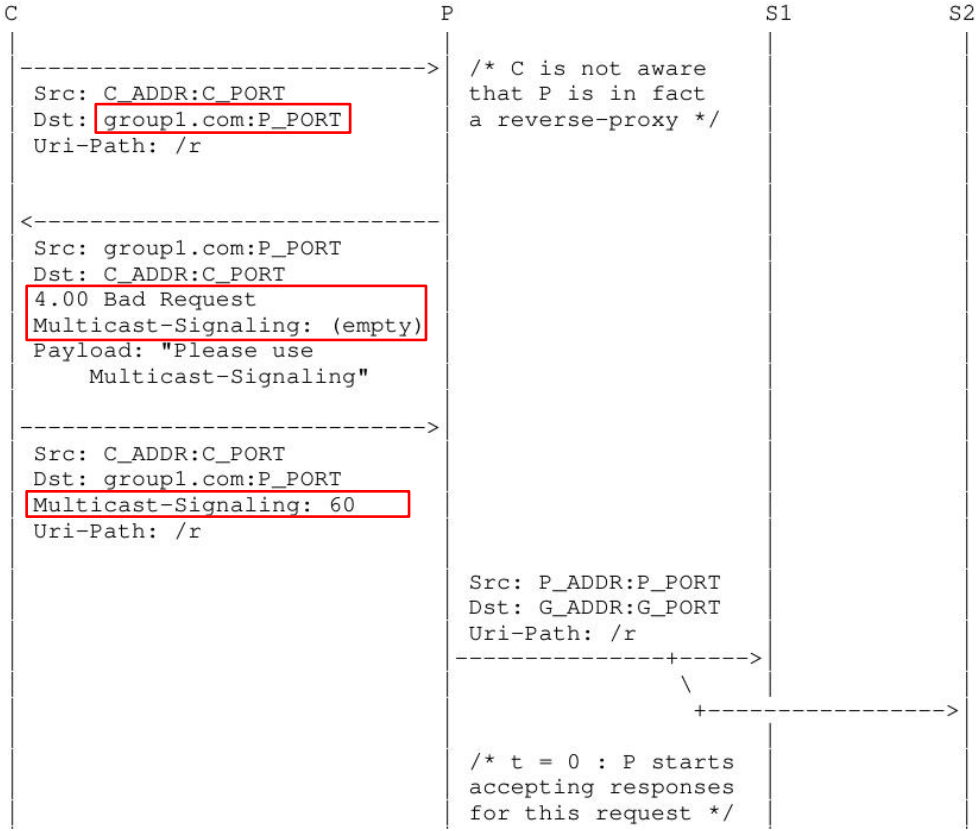


Example with forward-proxy (2/2)



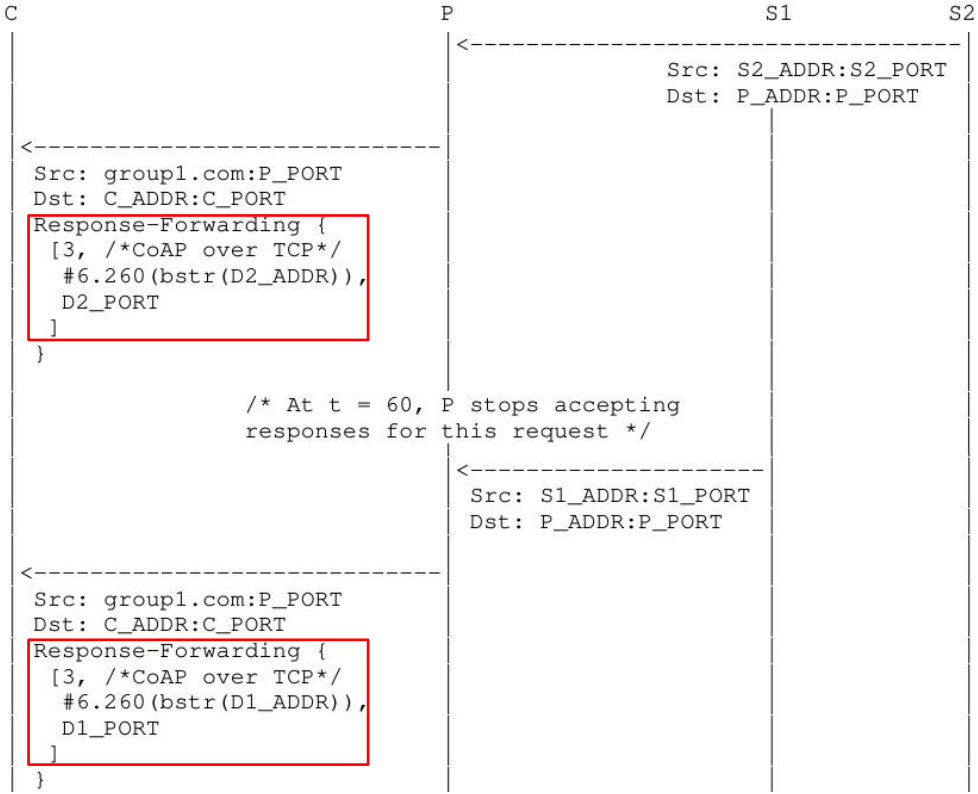
Example with reverse-proxy (1/3)

- 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



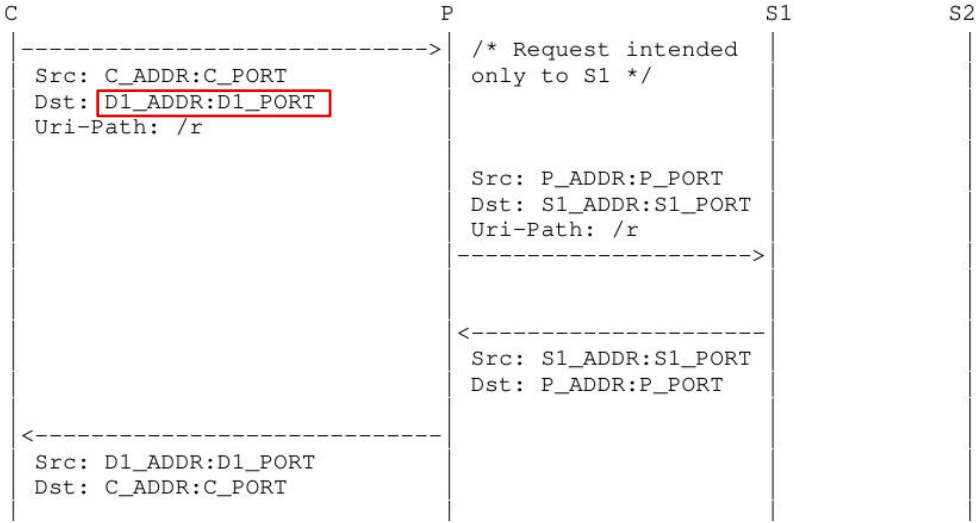
Example with reverse-proxy (2/3)

- 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



OSCORE between Client and Proxy

- Can co-exist with Group OSCORE between client and servers

- Some class **U** options are then treated by Proxy as if class **E**
  - Proxy-URI, Proxy-Scheme, Uri-Host, Uri-Port Options
  - OSCORE Option, if Group OSCORE is used end-to-end
  - Multicast-Signaling and Response-Forwarding Options (from this document)

- More options may come → Revised general rule, from Appendix A:
  - *This generally applies to all options that the proxy needs to understand and process in its exchange with the origin client. Further options can be added and treated as class U, e.g. related to routing information.* Accurate and simple enough?

OSCORE between Client and Proxy

- Nested OSCORE can have a broad applicability
  - A proxy forwarding to a group of CoAP servers – Like in this document
  - A server in a local domain, also acting as (cross-)proxy to servers in external domains
    - E.g., the local-domain server can be the Device Manager in a LWM2M setup

- Nested OSCORE is currently forbidden – There used to be no use case …
  - RFC 8613: *Nested use of OSCORE is not supported: If OSCORE processing detects an OSCORE option in the original CoAP message, then processing SHALL be stopped.*
  - This would require proper amendment, design and analysis.

- Move it from Appendix A of this document to a separate, dedicated document?

Summary

- Proxy operations for CoAP group communication
  - Embedded signaling protocol, using two new CoAP options
  - The proxy forwards individual responses to the client for a signaled time
  - The client can distinguish the origin servers and corresponding responses
- Latest additions
  - Revised encoding of server address information
  - Added support for reverse-proxies
  - Workflow examples, for forward-proxies and reverse-proxies
- Next steps
  - Define HTTP headers for Cross-Proxies → enable a HTTP client to talk to a CoAP group
  - ... and other issues listed
- Need for reviews – Promised: Christian, Carsten

Thank you!

Comments/questions?

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

Backup

Issues with proxies

- From Section 3.4 of *draft-ietf-core-groupcomm-bis*

- Issues when using proxies
  - Clients to be allow-listed and authenticated on the proxy
  - The client may receive multiple responses to a single *unicast* request
  - The client may not be able to distinguish responses and origin servers
  - The proxy does not know when to stop handling responses

- Possible approaches for proxy to handle the responses
  - **Individually forwarded back to the client**
  - Forwarded back to the client as a single aggregated response

Workflow: C -> P

- C prepares a request addressed to P
  - The group URI is included in the Proxi-Uri option or the URI-* options
- C chooses T seconds, as token retention time
  - T < Tr , with Tr = token reuse time
  - T considers the processing time at the proxy and the involved RTTs
- C includes the Multicast-Signaling option, with value T′ < T
- C sends the request to P via unicast
  - C retains the token beyond the reception of a first matching response

Workflow: P -> S

- P identifies C and verifies it is allowed-listed
- P verifies the presence of the Multicast-Signaling option
  - P extracts the timeout value T′
  - P removes the Multicast-Signaling option
- P forwards the request to the group of servers, over IP multicast
- P will handle responses for the following T′ seconds
  - Observe notifications are an exception – they are handled until the Observe client state is cleared.

Workflow: S -> P

- S processes the request and sends the response to P

- P includes the Response-Forwarding option in the response
  - The option value is absolute URI of the server
  - IP address: source address of the response
  - Port number: source port number of the response

Workflow: P -> C

- P forwards responses back to C, individually as they come
- P frees-up its token towards the group of servers after T′ seconds
  - Later responses will not match and not be forwarded to C
  - Observe notifications are the exception
- C retrieves the Response-Forwarding option
  - C distinguishes different responses from different origin servers
  - C is able to later contact a server individually (directly or via the proxy)
- C frees-up its token towards the proxy after T seconds
  - Observe notifications are the exception

Support for chain of proxies (1/2)

- Each proxy forwards the group request to the next hop
  - Nothing changes for the last proxy or for the origin servers
- Each proxy has to allow-list and authenticate the previous hop
- Only the last proxy removes the Multicast-Signaling option altogether
- For each **non-last** proxy:
  - The time indication T′ from Multicast-Signaling is still used for the local timer
  - If T′ > 0, a new value T″ < T′ replaces the value of Multicast-Signaling
- If a good T″ can’t be determined, reply with 5.05 (Proxying not supported)
  - Include Multicast-Signaling, with the minimum acceptable value for T′

Support for chain of proxies (2/2)

- Each proxy forwards the response back to the previous hop
  - Nothing changes for the last proxy or for the origin servers
- Only the last proxy adds the Response-Forwarding option
- Each **non-last** proxy does **not** alter or remove the Response-Forwarding option

OSCORE between Client and Proxy

- P has to authenticate C
  - A DTLS session would work
  - If Group OSCORE is used with the servers
    - P can check the counter signature in the group request
    - P needs to store the clients' public keys used in the OSCORE group
    - P may be induced to forward replayed group requests to the servers

- Appendix A – OSCORE between C and P
  - If Group OSCORE is also used between C and the servers
    1. Protect the group request with Group OSCORE (C<->Servers context)
    2. Protect the result with OSCORE (C<->P context)
       - Some class U options are processed as class E options
    3. Reverse processing for responses