Proxy Operations for CoAP Group Communication

draft-tiloca-core-groupcomm-proxy-02

Marco Tiloca, RISE
Esko Dijk, IoTconsultancy.nl

IETF 109, CoRE WG, November 17th, 2020

Rationale

- **Background** – CoAP supports group communication over IP multicast
  - *draft-ietf-core-groupcomm-bis* discusses also 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

Recap groupcomm-proxy

- In the unicast request addressed to the proxy, the client indicates:
  - To be interested in and capable of handling multiple responses
  - For how long the proxy should collect and forward back responses
  - Use the new CoAP option **"Multicast-Signaling"**, removed by the proxy
- In each response to above, the proxy includes the server address
  - Use the new CoAP option **"Response-Forwarding"**
  - The client can distinguish the responses and the different servers
  - The client can contact an individual server (directly, or via the proxy)
- Group OSCORE can be used for e2e security between client and servers
- DTLS or OSCORE can be used between Client and Proxy (Appendix A)

Updates from -02

- Editorial re-organization of text for Observation
  - Now as dedicated subsections, throughout the protocol workflow
  - The proxy keeps forwarding notifications back, until the observation terminates
- Revised security considerations
- Updated semantics and usage of the new CoAP options
- Added support for a chain of proxies
  - Same principles, extended through multiple hops

Multicast-Signaling option

- Only in C → P requests
  - Presence: explicit claim of support and interest from the client
  - Value: T′ s, i.e. for how long the proxy should forward back responses
  - The proxy removes the option, before forwarding the request to the servers

No.	C	U	N	R	Name	Format	Length	Default
TBD1		x	-		Multicast-Signaling	uint	0-5	(none)

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

- Now the value T′ can also be 0
  - Still ok to forward the request to the servers, no interest in proxy responses
  - SHOULD be used with the No-Response Option, with value 26

- Issues or comments?

Response-Forwarding option

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

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

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

```
srv_info = [
  srv_addr : #6.260(bstr),   ; IP address of the server
  ? srv_port : uint,         ; Port number of the server
]
```

- Address and port in the value
  - If port is omitted, assume the dst port of group URI in the Request – most common

- It used to be an absolute URI, with scheme & hostname
  - Pro: now it's a smaller option, less parsing, handy for constrained clients
  - Con: excludes scenarios where Proxy inserts DNS hostname. **Can we live with it?**

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 Option, 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
- N**on-last** proxies do **not** alter or remove the Response-Forwarding option

OSCORE between Client and Proxy

- Can co-exist with Group OSCORE between client and servers
- Can be used between each pair of hops, until the last proxy
- Some class **U** options are treated as class **E**
  - Proxy-URI, Proxy-Scheme, Uri-Host, Uri-Port
  - OSCORE, if Group OSCORE is used end-to-end
  - Multicast-Signaling and Response-Forwarding from this document
- More options may come. **Any general rule to identify them?**

OSCORE between Client and Proxy

- Proposal: process an option X as class E rather than U if:

- X is intended (also) for the recipient hop and its processing
  - E.g., Uri-Host option, Multicast-Signaling option, …

OR

- X is intended for the final endpoint, but more instances will be added as intended for the recipient hop and its processing
  - E.g., OSCORE option, when Group OSCORE is used end-to-end

- **Accurate enough? Anything simpler?**

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
  - This version adds also support for a chain of proxies

- Next steps
  - Define HTTP headers for HTTP/CoAP Cross-Proxies
  - Enable a HTTP client to talk to a CoAP group

- Need for reviews
  - Promised: Christian, Carsten, Francesca

Thank you!

Comments/questions?

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

Backup

Issues with proxies

- *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 processing at the proxy and 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

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