Group Communication for the Constrained Application Protocol (CoAP)

draft-ietf-core-groupcomm-bis-03

Esko Dijk, IoTconsultancy.nl
Chonggang Wang, InterDigital
Marco Tiloca, RISE

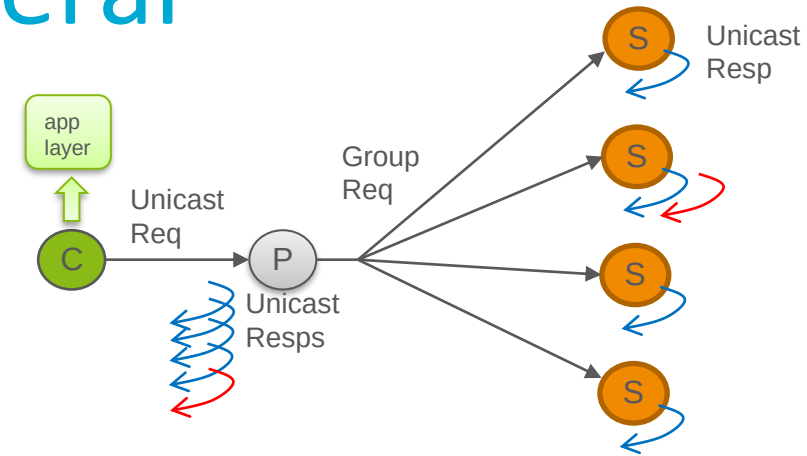
IETF 110, CoRE WG, March 8th, 2021

Goal

- Intended normative successor of experimental RFC 7390 (if approved)
  - As a Standards Track document
  - Obsoletes RFC 7390; Updates RFC 7252 and RFC 7641
- Be standard reference for implementations that are now based on RFC 7390, e.g.:
  - “Eclipse Californium 2.0.x” (Eclipse Foundation)
  - “Implementation of CoAP Server & Client in Go” (OCF)
- What’s in scope?
  - CoAP group communication over UDP/IP, including latest developments
  - (Observe/Blockwise/Security ...)
  - Caching and re-validation of responses
  - Unsecured CoAP or group-OSCORE-secured communication
  - Principles for secure group configuration
  - Use cases (appendix)

Updates in -03 – General

- Multiple CoAP responses to the same group request from the **same server** ( )
  - Handling moved from the CoAP layer to the application
  - Based on interop experience; the application has more context to decide what to do

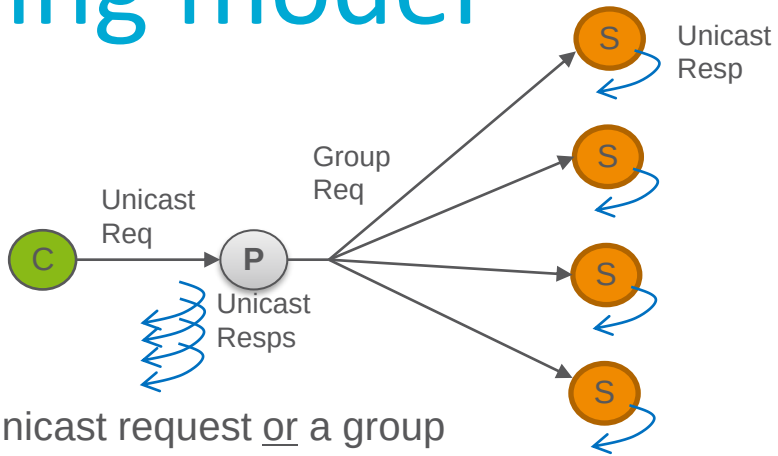


Updates in -03 – General

- Revised guidelines on **forward-proxies** and related issues
- Added guidelines and issues on **reverse-proxies**
  - Stand-in for the whole group of servers, optionally also for each individual server
  - Same and additional issues, compared to a forward-proxy
    - The client may need additional configuration to handle multiple responses
    - The proxy may need additional configuration on the duration of group exchanges
    - The client should get an error, if reusing a Token while a group exchange is still ongoing
- The signaling protocol of *draft-tiloca-core-groupcomm-proxy* is referenced
  - Addresses all known issues with both forward-proxies and reverse-proxies

Updates in -03 – Caching model

- Caching of responses at proxies (P)
  - Two types of cache entries:



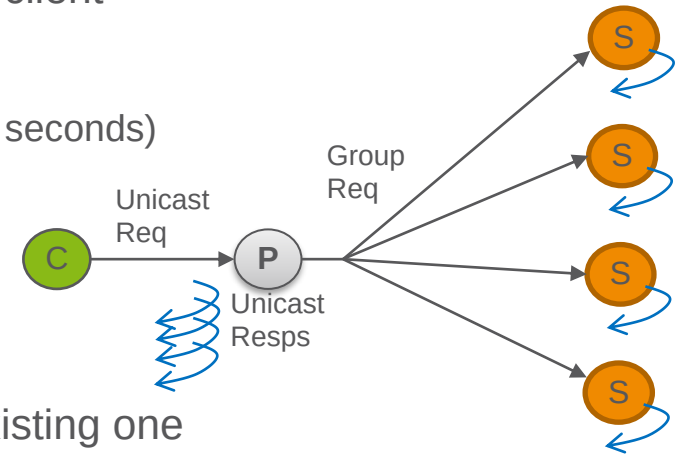
- "**Individual**" cache entry
  - Populated with the response from one server (to a unicast request or a group request)
  - Hit by a matching unicast request intended to that server

- "**Aggregated**" cache entry
  - Populated with all the responses to a group request, from any server in the group
  - Hit by a matching group request intended to all servers

Updates in -03 – Caching model

› As it receives responses to a group request, the proxy:

1. Forwards each response from the origin server S to the client
2. Adds each response to the individual cache entry for S
   – Same lifetime as Max-Age of the response (or default to 60 seconds)
3. Adds the response to a list L



› After forwarding back all the responses, the proxy:

1. Creates an aggregated cache entry, or cleans up the existing one
2. Copies the responses from the list L to the cache entry
3. Set the cache entry lifetime to the smallest Max-Age of the added responses
4. Set the cache entry as active

Updates in -03 – Caching model

› When it receives a response to a unicast request, the proxy:

1. Forwards back the response from the origin server S to the client
2. Creates an Individual cache entry for S, or updates the existing one
   - Same lifetime as Max-Age of the response (or default to 60 seconds)
3. Looks for existing Aggregated cache entries, such that:
   - They would produce a hit, if receiving a group request matching the forwarded unicast request
4. In each found Aggregated cache entry:
   - Store the response, possibly overwriting a currently stored one
   - Set the lifetime of the cache entry to *min*(current entry lifetime, Max-Age of the response)

› Same when the proxy sends requests to the servers, to refresh its cache

Updates in -03 – Validation model

› *Section 8.2.1 of RFC 7252 left this for further study*

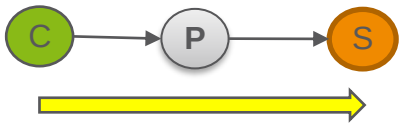
› **Between Client and Servers**

The Multi-ETag Option

No.	C	U	N	R	Name	Format	Length	Default
TBD1				x	Multi-ETag	(*)	any	(none)

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

› New Multi-Etag option
  – Only for group requests
  – One instance per server in group to revalidate against



› Option value: CBOR sequence of 1+M elements
  – First element: addressing information of the server, encoded as in *groupcomm-proxy*
  – The following M elements are entity-tag values, as CBOR byte strings

› A server processes only the Multi-Etag option pertaining to itself, unlike ETag
  – What follows uses ETag, as in RFC 7252

Updates in -03 – Validation model

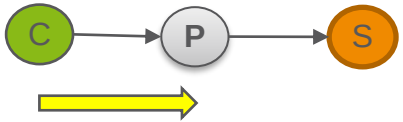
- **Between Client and Proxy**
- New Group-Etag Option
  - Only for Aggregated cache entries
  - For group requests and related responses

No.	C	U	N	R	Name	Format	Length	Default
TBD2				x	Group-ETag	opaque	1-8	(none)

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

The Group-ETag Option

- Option value: an entity-tag value, as CBOR byte string
  - Basically, a version number of the Aggregated cache entry (maintained by the proxy)



- A 2.05 (Content) response may include one Group-ETag Option
- In a GET/FETCH group request
  - One option instance per e-tag value to revalidate against the proxy's Aggregated cache entries
- A 2.03 (Valid) response revalidates all responses in the Aggregated cache entry
  - MUST include one Group-Etag Option indicating the revalidated responses set

Caching/validation with e2e security

- **Caching at a proxy**
  - Possible, by using deterministic requests
  - Limited to (REST) safe requests with no side-effects on resource at the servers
  - See *draft-amsuess-core-cachable-oscore*

- **Response re-validation**
  - Possible between origin client and origin servers, with Multi-ETag options
    - Caveat: different set of Multi-ETag options → Different deterministic request
    - Different deterministic requests → Different cache entries at the proxy
    - Trade off between flexibility for the client and caching efficiency at the proxy
  - Not possible between proxy and origin servers, with Multi-ETag options
  - Not possible between origin client and proxy, with Group-ETag options

Open point – Github issue #11

› **What's the most appropriate place for below new items?**

1. General mechanics & rules on cacheability of responses at proxies
   – Appropriate to be in this document?

2. Validation of individual responses, with the new Multi-ETag option
   – Appropriate to be in this document? Or in a separate dedicated document?

3. Validation of a set of response cached at the proxy, with the new Group-ETag option
   – Appropriate to be in this document? Or instead in *draft-tiloca-core-groupcomm-proxy*?

Next steps

- Address comments from John [1] – Thanks!
  - More reviews would be good! Promised @IETF108: Christian
- Address open point from today (issue #11) and other Github issues [2]
- Test selected functions in CoAP implementations
  - E.g. “Observe + multicast” extension of RFC 7641
  - Report results

[1] https://mailarchive.ietf.org/arch/msg/core/xy3ImeWkbqziBhqs4NCGwNP6R7U/

[2] https://github.com/core-wg/groupcomm-bis/issues

Thank you!

Comments/questions?

https://github.com/core-wg/groupcomm-bis/

Motivation (backup slide)

- RFC 7390 was published in 2014
  - CoAP functionalities available by then were covered
  - No group security solution was available to indicate
  - It is an Experimental document (started as Informational)

- What has changed?
  - More CoAP functionalities have been developed (Block-Wise, Observe)
  - RESTful interface for membership configuration is not really used
  - Group OSCORE provides group end-to-end security for CoAP

- Practical considerations
  - Group OSCORE clearly builds on RFC 7390 normatively
  - However, it can refer RFC 7390 only informationally