Observe Notifications as CoAP Multicast Responses

draft-tiloca-core-observe-multicast-notifications-02

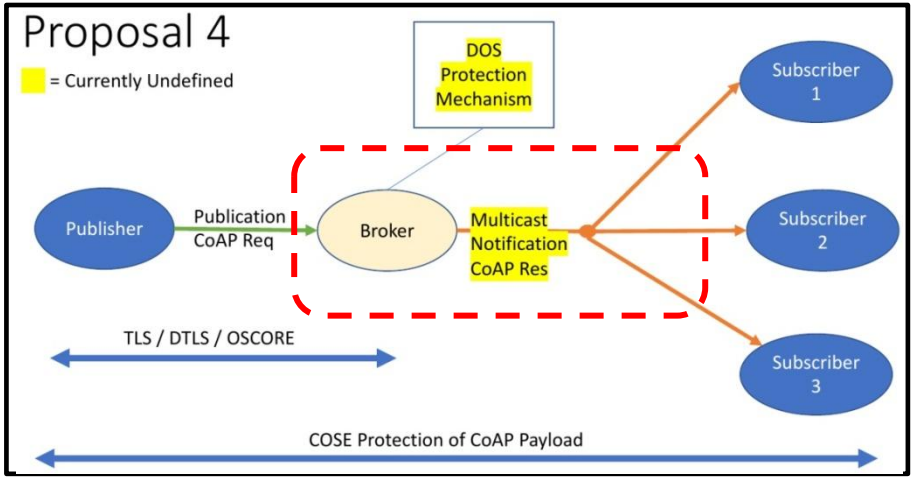
Marco Tiloca, RISE
Rikard Höglund, RISE
Christian Amsüss
Francesca Palombini, Ericsson

IETF CoRE WG, Virtual Interim, April 8th, 2020

Recap

- Observe notifications as multicast responses
  - Many clients observe the same resource on a server S
  - Improved performance due to multicast delivery
  - Multicast responses are not defined yet. Token binding? Security?

- Practical use case
  - Pub-Sub scenario
  - Many clients subscribe to a same topic on the Broker
  - Better performance
  - Subscribers are clients only



From the Hallway Discussion @ IETF 104

Contribution

- Define Observe notifications as multicast responses

- Management and enforcement of a common Token space
  - The Token space belongs to the group
  - The group entrusts the management to the server
  - All clients in a group observation use the same Token value

- Use of Group OSCORE to protect multicast notifications
  - The server aligns all clients of an observation on a same *external_aad*
  - All notifications for a resource are protected with that *external_aad*

Rationale

- The server can start a group observation for a resource, e.g. :
  1. With no observers yet, a traditional registration request comes from a first client
  2. With many traditional observations, all clients are shifted to a group observation

- Phantom observation request
  - Generated inside the server, it does not hit the wire
  - Like if sent by the group, from the multicast IP address of the group
  - Multicast notifications are responses to this phantom request

- The server sends to new/shifted clients an ***error response*** with:
  - Serialization of the phantom request
  - IP multicast address where notifications are sent to
  - current representation of the target resource

Updates from -04

- New section on congestion control
  - Requested by Carsten at IETF 106
  - Building on *core-groupcomm-bis* and RFC 7641
- Encoding of the informative error response
  - New content format *informative-response+cbor*
  - New registry for parameter of informative response
  - Separate registry for parameters of phantom request
- Parameter meaning
  - *src_addr*, *src_port*, *dst_addr*, *dst_port* : addressing information
  - *coap_msg* : serialization of the phantom request (i.e. UDP payload)
  - *notif_num* : latest used observe number, as baseline for the client
  - *res* , *res_ct*: current resource representation and its content-format

Informative error response

```
Payload: { ph_req : {
             src_addr : bstr(M_ADDR),
             src_port : 65500,
             dst_addr : bstr(SERVER_ADDR),
             dst_port : 7252,
             coap_msg : bstr(PH_REQ.CoAP)
           },
           notif_num : 10,
           res : bstr("1234"),
           res_ct : 0
         }
```

Updates from -04

- Appendix A - Alternative ways to retrieve a phantom request

```
Request:

    GET </ps/topics?rt=oic.r.temperature>
    Accept: CoRAL

Response:

    2.05 Content
    Content-Format: CoRAL

    rdf:type <http://example.org/pubsub/topic-list>
    topic    </ps/topics/1234> {
        dst_addr h"ff35003020010db8..1234"
        src_port 5683
        dst_addr h"20010db80100..0001"
        dst_port 5683
        coap_msg h"120100006464b431323334"
    }
```

- Pub-Sub
  - The phantom request is part of the topic metadata
  - A subscriber gets it already upon topic discovery
  - Early listening for multicast observations

- Sender introspection
  - Useful for debugging upon intercepting notifications
  - Query the server on a dedicated interface

```
Request:

    GET </.well-known/core/mc-sender?token=6464>

Response:

    2.05 Content
    Content-Format: application/informative-response+cbor

    {'ph_req': {
        'dst_addr': h"ff35003020010db8..1234"
        'src_port': 5683
        'dst_addr': h"20010db80100..0001"
        'dst_port': 5683
        'coap_msg': h"120100006464b431323334"
    }}
```

Updates from -04

- Cancellation of group observation
  - The server sends to itself a phantom cancellation request
  - A multicast 5.03 response follows, with no payload

- When? Not enough clients are still active
  - Proposal: rough counting of alive clients, with a poll for interest
  - New CoAP options for successful multicast notifications

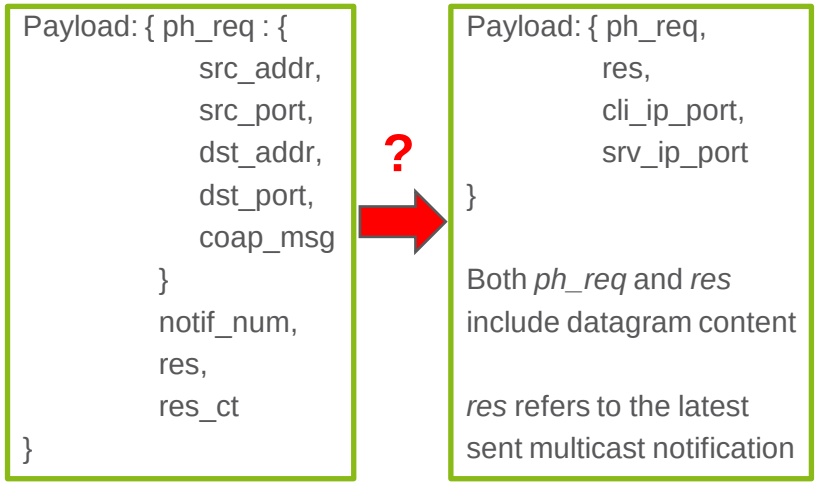
- Server current rough estimate: *n*
  - Expected confirmations *m* < *n*
  - Option value: *q* = ceil (*n* / *m*)
  - Each client picks a random *c* : [0, *q*)
  - If *c* == 0, the client sends a registration request (Non; with No-Response)
  - The server receives *r* of such requests, than *n* ← (*r* * *q*)

No.	C	U	N	R	Name	Format	Len.	Default
TBD		x			Multicast-Response-Feedback-Divider	uint	0-8 B	(none)

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

Open points

- Informative error response in CoRAL
  - Early version already in Appendix A
- Considerations on the rough counting of alive clients
  - When stop waiting for confirmations? Leisure time + some transmit time …
  - Good practices and checks to be sure avoiding Smurf attacks
- Alternative encoding of the informative request
  - Now the info on the current resource is split
  - Serialize it as the phantom request in coap_msg?
  - Pro: use the native Observe numbers



Summary

- Multicast notifications to all clients observing a resource
- Latest additions
  - Media type and encoding for the error response
  - Cancellation of group observation, based on rough counting of clients
  - Alternative ways to retrieve a phantom request
- Open points to address
  - Considerations and parameter tuning for the client rough counting
  - Encoding within the error response (full notification vs. resource representation)
  - Error response in CoRAL (already sketched in the Appendix)
  - Error response using the format from *core-coap-problem* ?
- Need for document reviews

Thank you!

Comments/questions?

https://gitlab.com/crimson84/draft-tiloca-core-observe-responses-multicast

Backup

Assumptions

- Clients have previously discovered the resource to access
- The server knows the IP multicast address where to send notifications
- If Group OSCORE is used to secure multicast notifications
  - The server has previously joined the right OSCORE group
- The server provides the clients with other required information

Server side

1. Build a GET phantom request; Observe option set to 0

2. Choose a value T, from the Token space for messages …
   - … coming from the multicast IP address and addressed to target resource

3. Process the phantom request
   - As coming from the group and its IP multicast address
   - As addressed to the target resource

4. Hereafter, use T as token value for the group observation

5. Store the phantom request, with no reply right away

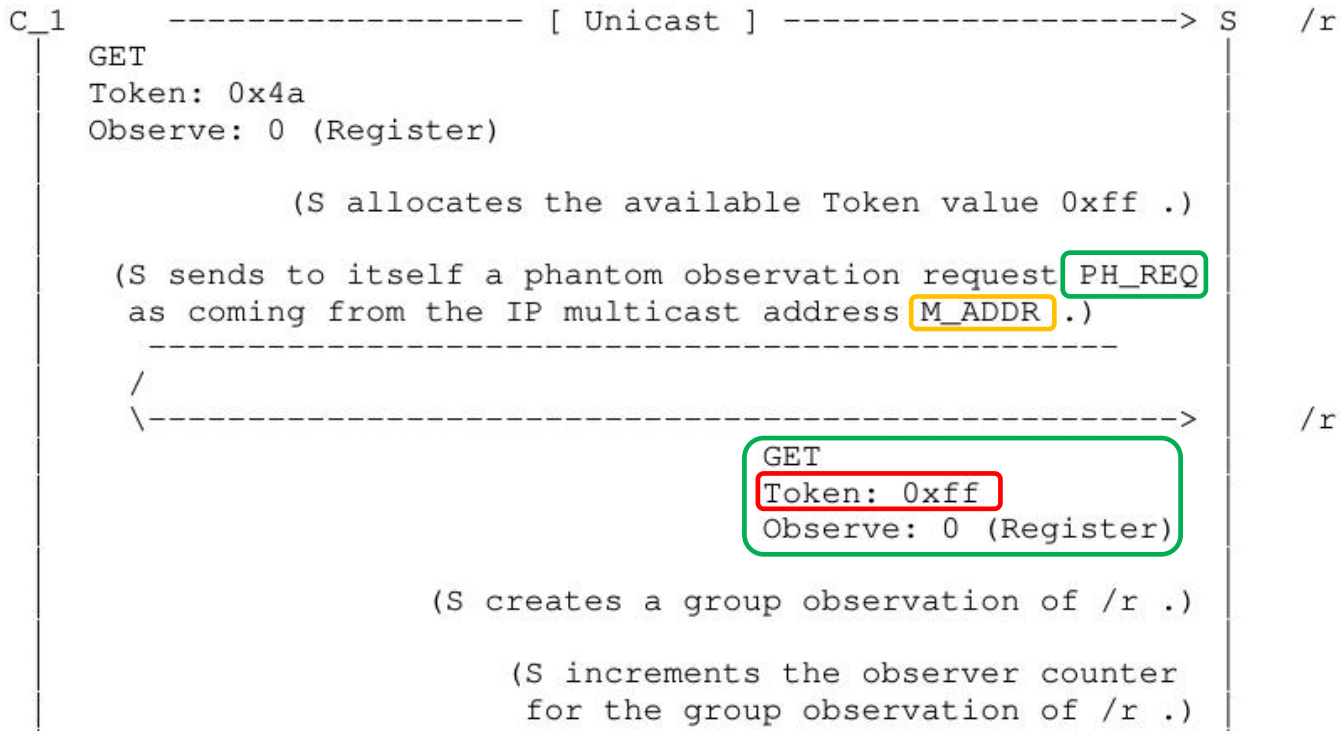
Interaction with clients

- The server sends to new/shifted clients an ***error response*** with
  - '*ph_req*': byte serialization of the phantom request + Multicast IP addres + …
  - ' *res*': current representation of the target resource
  - '*notif_num*' and '*res_ct*': observe counter and content-format for the resource

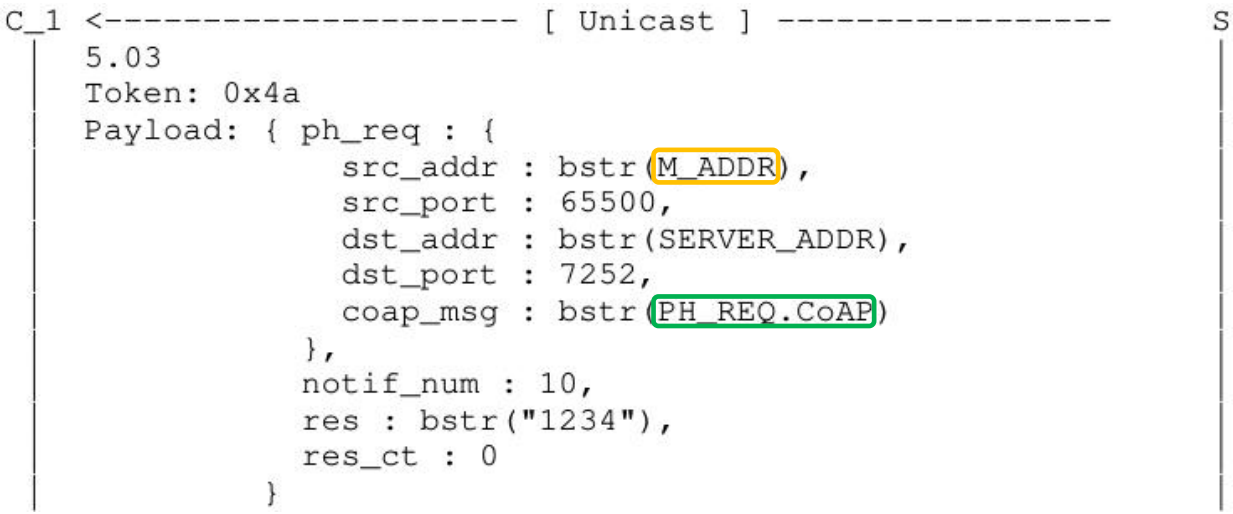
- When the value of the target resource changes
  - The server sends an Observe notification to the IP multicast address
  - The notification has the Token value T of the phantom request

- When getting the error response, a client
  - Configures an observation from an endpoint associated to the multicast IP address
  - Accepts observe notifications with Token value T, sent to that multicast IP address

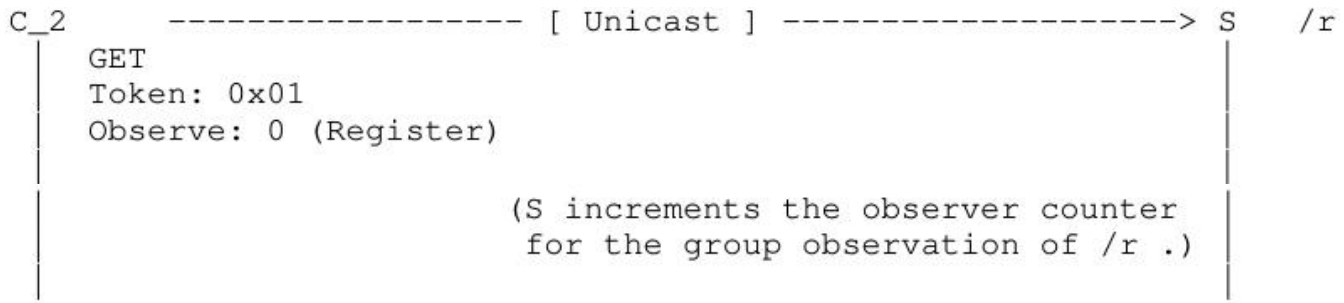
C1 registration



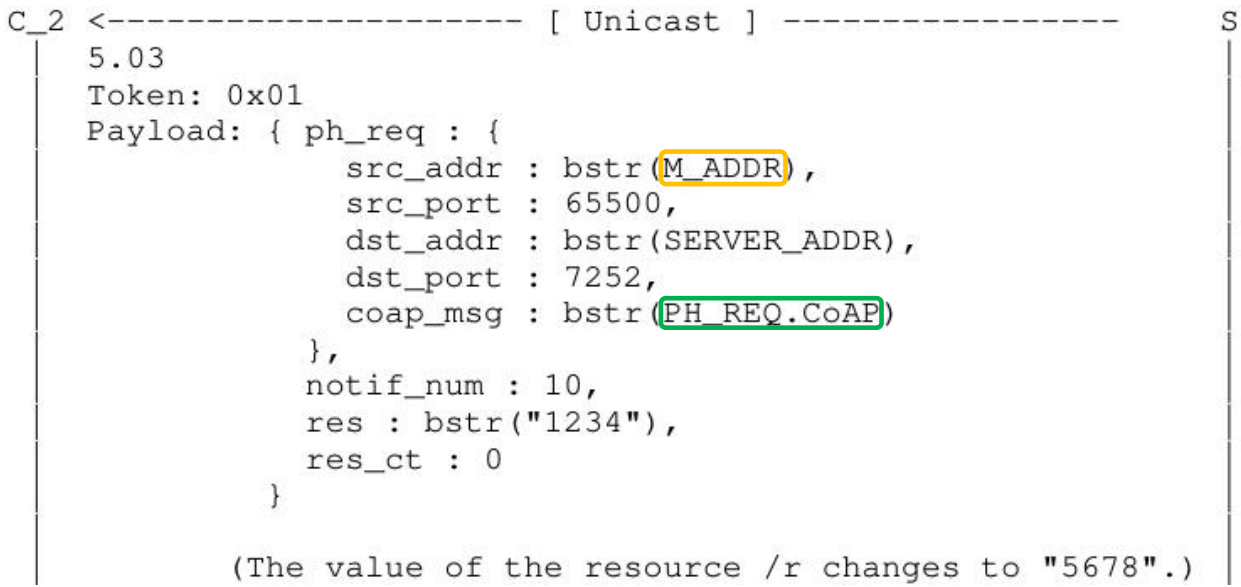
C1 registration



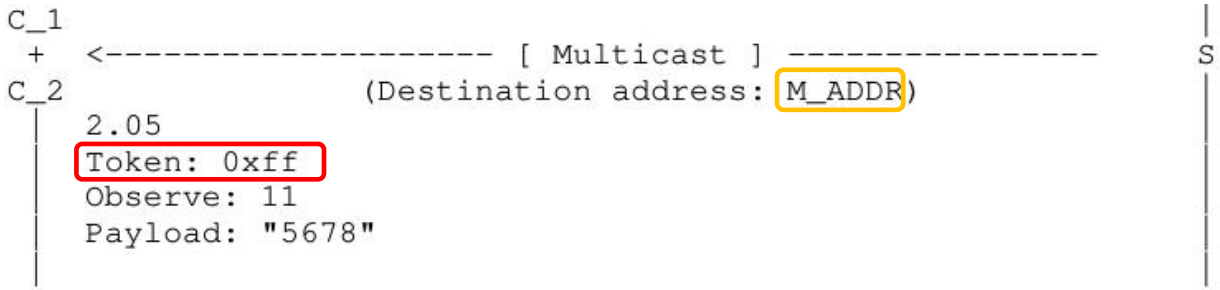
C2 registration



C2 registration



Multicast notification



- Same Token value of the Phantom Request
- Enforce binding between
  - Every multicast notification for the target resource
  - The (group) observation that each client takes part in

Security with Group OSCORE

- The phantom request is protected with Group OSCORE
  - ***x*** : the Sender ID ('kid') of the Server in the OSCORE group
  - ***y*** : the current SN value ('piv') used by the Server in the OSCORE group
  - Note: the Server consumes the value ***y*** and does not reuse it as SN in the group

- To secure/verify all multicast notifications, the OSCORE *external_aad* is built with:
  - 'req_kid' = ***x***
  - 'req_piv' = ***y***

- The phantom request is still included in the informative response
  - Each client retrieves ***x*** and ***y*** from the OSCORE option

Security with Group OSCORE

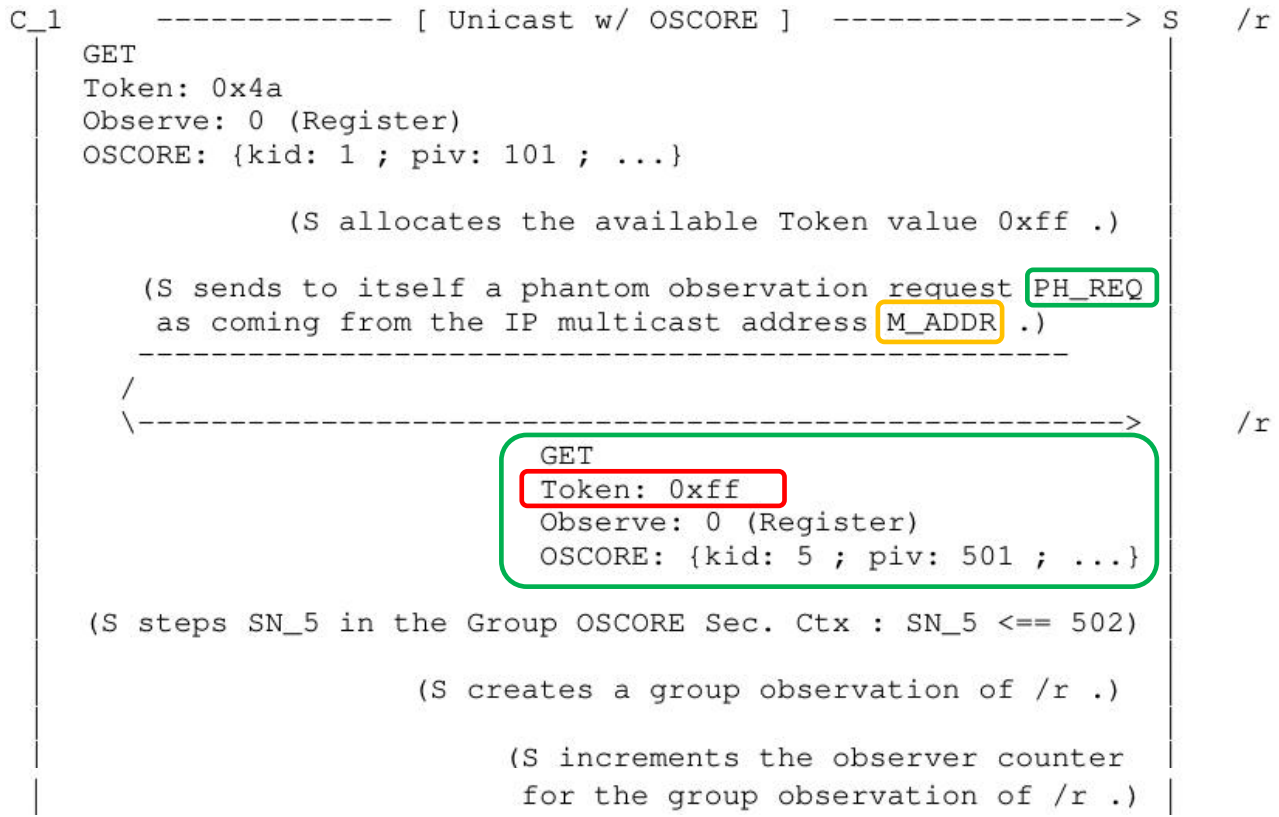
- In the error response, the server can ***optionally*** specify also:
  - '*join-uri*' : link to the Group Manager to join the OSCORE group
  - '*sec-gp*' : name of the OSCORE group
  - '*as-uri*' : link to the ACE Authorization Server associated to the Group Manager
  - '*cs-alg*' : countersignature algorithm
  - '*cs-crv*' : countersignature curve
  - '*cs-kty*' : countersignature key type
  - '*cs-kenc*' : countersignature key encoding
  - '*alg*' : AEAD algorithm
  - '*hkdf*' : HKDF algorithm

MUST

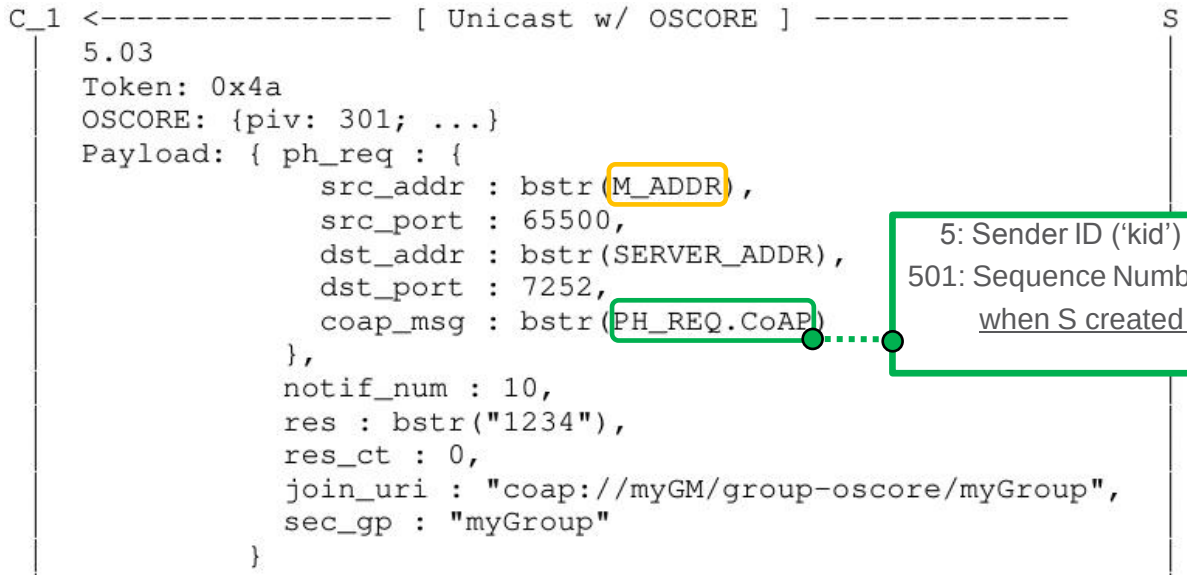
MAY

- Clients can still discover the OSCORE group through other means
  - E.g., using the CoRE Resource Directory, as in *draft-tiloca-core-oscore-discovery*

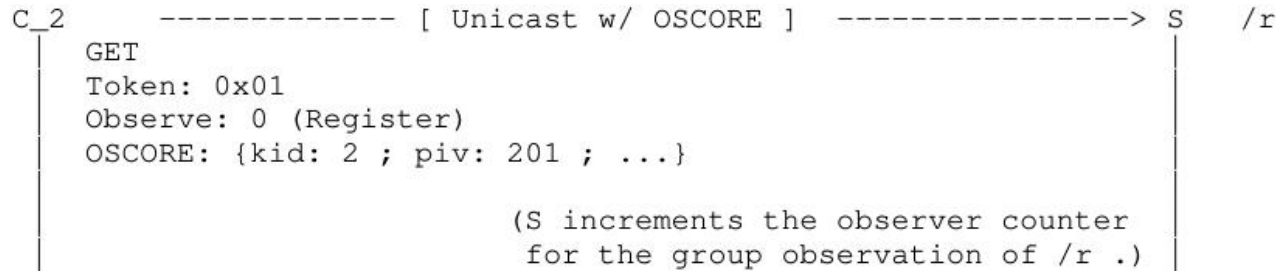
C1 registration w/ security



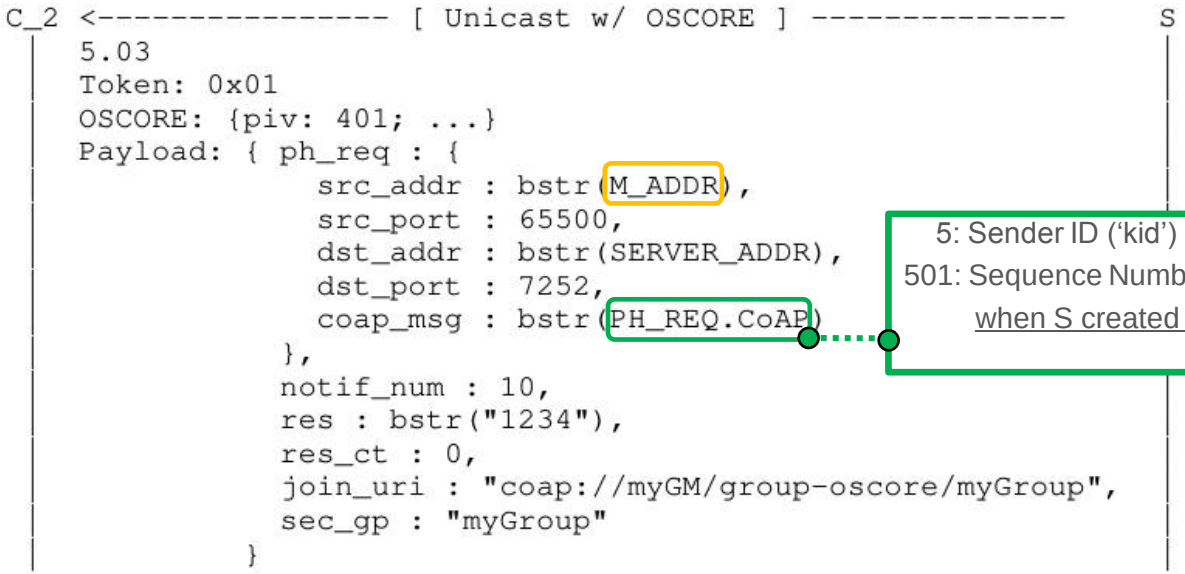
C1 registration w/ security



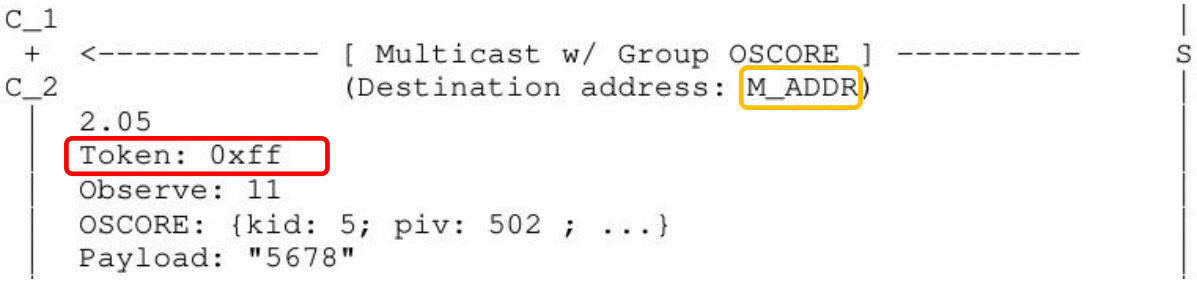
C2 registration w/ security



C2 registration w/ security



Multicast notification w/ security



- When encrypting and signing the multicast notification:
  - The OSCORE *external_aad* has 'req_kid' = 5 and 'req_iv' = 501
  - Same for all following notifications for the same resource
- Enforce secure binding between
  - Every multicast notification for the target resource
  - The (group) observation that each client takes part in