Observe Notifications as CoAP Multicast Responses

draft-tiloca-core-observe-multicast-notifications-04

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

IETF 109, CoRE WG, November 17th, 2020
What, Why

› Observe notifications as **multicast responses**
  - Many clients observe the same resource on a server
  - Improved performance due to multicast delivery
  - Multicast responses are not defined yet – Token binding, security, …

› Example of relevant use case
  - Pub-Sub scenario, also discussed at IETF 104
  - Many subscribers to a same topic on the Broker
  - Better performance
  - Subscribers can remain clients only
How

- Define multicast responses, in particular Observe notifications

- Token space managed by the server
  - The Token space **belongs** to the group (clients)
  - The group **entrusts** the management to the server
  - All clients in a group observation use the same Token value

- 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`
Phantom request and error response

› The server requests the observation on its own, e.g. when:
   1. a first traditional registration request comes from a first client
   2. some threshold is crossed – clients can be shifted to a group observation

› Consensus on Token & external_aad , by using a 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 clients a 5.03 “informative response” with:
   – The serialization of the phantom request
   – The IP multicast address where notifications are sent to
   – The serialization of the latest multicast notification (i.e., the current resource status)
Updates from -04

- Improved and extensible encoding of the informative error response
  - Transport-independent information, for the phantom request and the latest notification
  - Common transport-specific information; detailed specification for CoAP over UDP

```
informative_response_payload = { 
  1 => bstr, ; phantom request (transport-independent information) 
  2 => bstr, ; latest notification (transport-independent information) 
  3 => array ; transport-specific information 
}
```

```
tp_info = [ 
  tp_id : 1, ; UDP as transport protocol 
  token : bstr, ; Token of phantom request and multicast notifications 
  srv_addr : #6.260(bstr), ; Src. address of multicast notifications 
  srv_port : uint, ; Src. port of multicast notifications 
  cli_addr : #6.260(bstr), ; Dst. address of multicast notifications 
  ? cli_port : uint ; Dst. port of multicast notifications 
]
```

**OLD**

```
Payload: { 
  ph_req : bstr(PH_REQ.CoAP), 
  lastnotif : bstr(LAST_NOTIF.CoAP) 
  cl_addr : bstr(GROUP_ADDR), 
  cl_port : GROUP_PORT, 
  srv_addr : bstr(SERVER_ADDR), 
  srv_port : SERVER_PORT, 
}
```

**NEW**

```
Payload: { 
  ph_req : bstr(0x01 | OPT), 
  lastnotif : bstr(0x25 | OPT | 0xff | PAYLOAD), 
  tp_info : [1, 0x7b, bstr(GRP_ADDR), GRP_PORT] 
}
```
Updates from -04

- Improved rough counting of active clients, by poll for interest
  - New CoAP option in successful multicast notifications 📥Now the length is max 1 byte!

- Server current rough estimate: COUNT
  - \( N = \max(COUNT, 1) \)
  - \( M \) desired confirmations
  - \( L = \lceil \log_2(N / M) \rceil \)
  - Option value: \( Q = \max(L, 0) \)
  - Each client picks a random value \( I : [0, 2^Q) \)
  - If \( I == 0 \), the client sends a re-registration request
    - Non Confirmable; w/ No-Response; w/ the new Option having empty value
  - The server receives \( R \) of such requests; meanwhile, the estimate has become \( COUNT' \)
  - \( F = R \times (2^Q) \); then \( COUNT \leftarrow COUNT' + ((F - N) / D) \), with \( D > 0 \) as dampener

- Pseudo-code in Appendix B, also optimized for constrained clients
Updates from -04

› Added support for intermediary proxies
  – The proxy (next to the server) directly listens to the IP multicast address
  – The original Token of the phantom request has to match at the proxy
  – The proxy forwards multicast notifications back to each client
    › The proxy uses the Token values offered by the clients

› Without end-to-end security (Section 8)
  – The proxy can retrieve the phantom request from the informative response
  – The informative response is still forwarded back to each new client

› With end-to-end security (Section 9)
  – The informative response is also protected with OSCORE or Group OSCORE
  – The proxy **cannot** retrieve the Phantom request from the informative response
  – Each client has to explicitly provide the Phantom request to the proxy
Updates from -04

› The client sends the rebuilt Phantom request as addressed to the proxy
  – The request itself already provides the transport-specific information

› The sent request includes a new CoAP option Listen-To-Multicast-Responses
  – This provides the transport-independent information
  – Value: serialized CBOR array, i.e. ‘tp_info’ from the informative response

<table>
<thead>
<tr>
<th>No.</th>
<th>C</th>
<th>U</th>
<th>N</th>
<th>R</th>
<th>Name</th>
<th>Format</th>
<th>Len.</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>Listen-To-Multicast-Responses</td>
<td>(*)</td>
<td>3-1024</td>
<td>(none)</td>
</tr>
</tbody>
</table>

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

tp_info = {
  tp_id : 1,  ; UDP as transport protocol
  token : bstr,  ; Token of phantom request and multicast notifications
  srv_addr : #6.260(bstr),  ; Src. address of multicast notifications
  srv_port : uint,  ; Src. port of multicast notifications
  cli_addr : #6.260(bstr),  ; Dst. address of multicast notifications
  cli_port : uint  ; Dst. port of multicast notifications
}

› Now the proxy has all it needs to receive multicast notifications

› Appendix C and Appendix D include interaction examples
Open points

› Proxy example fixes

› Negotiation: What to do when there is no common way?

› ‘lastnotif’ in the informative response
  – Serialization of latest sent multicast notification
  – To be made optional (like an empty ACK)

› OSCORE group possibly self-managed by the server
  – The observation request might even double as joining request of a Monitor member
  – The informative response would include also key material, as in a Joining Response
  – Thoughts? Objections?
Summary

› Latest additions
  – Improved and extensible encoding of the informative error response
  – Improved rough counting of clients; added pseudo-code for constrained clients
  – Added support for intermediary proxies, with or without end-to-end security
  – Re-organization of sections and editorial improvements

› Next steps
  – Address open points
  – Investigate possible optimization with a deterministic phantom request

› Need for document reviews
  – Potential reviewers: Göran, Jaime, Carsten
Thank you!

Comments/questions?

https://gitlab.com/crimson84/draft-tiloca-core-observe-responses-multicast
Backup
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, store (not send) reply for lastnotif
Interaction with clients

- The server sends to new/shifted clients an **error response** with
  - ‘ph_req’: serialization of the phantom request
  - ‘lastnotif’: serialization of the latest sent notification for the target resource
    - ‘token’: the selected Token value T, used for ‘ph_req’ and ‘lastnotif’
    - ‘cli_addr’ and ‘cli_port’: source address/port of the phantom request
    - ‘srv_addr’ and ‘srv_port’: destination address/port of the phantom request

- When the value of the target resource changes:
  - The server sends an Observe notification to the IP multicast address ‘cli_addr’
  - The notification has the Token value T of the phantom request

- When getting the error response, a client:
  - Configures an observation for an endpoint associated to the multicast IP address
  - Accepts observe notifications with Token value T, sent to that multicast IP address
C1 registration

C_1 ---------------- [ Unicast ] -----------------> S /r

GET
Token: 0x4a
Observe: 0 (Register)
<Other options>

(S allocates the available Token value 0x7b.)
(S sends to itself a phantom observation request PH_REQ as coming from the IP multicast address GRP_ADDR.)

/ \-----------------------------------------------------------> /r

GET
Token: 0x7b
Observe: 0 (Register)
<Other options>

(S creates a group observation of /r.)

(S increments the observer counter for the group observation of /r.)
C1 registration

C_1 <---------------------- [ Unicast ] ---------------------- S

5.03
Token: 0x4a
Content-Format: application/informative-response+cbor
<Other options>
Payload: {
    ph_req : bstr(0x01 | OPT),
    last_notif : bstr(0x25 | OPT | 0xff | PAYLOAD),
    tp_info : [1, 0x7f, bstr(SRV_ADDR), SRV_PORT,
               bstr(GRP_ADDR), GRP_PORT]
}
C2 registration

C_2          [ Unicast ]          \rightarrow S  \ /

GET
Token: 0x01
Observe: 0 (Register)
<Other options>

(S increments the observer counter for the group observation of /r.)

C_2          [ Unicast ]          \leftarrow S  

5.03
Token: 0x01
Content-Format: application/informative-response+cbor
<Other options>
Payload: {
   ph_req   :  bstr(0x01 | OPT),
   last_notif : bstr(0x25 | OPT | 0xff | PAYLOAD),
   tp_info   : [1, 0x7b, bstr(SRV_ADDR), SRV_PORT,
                  bstr_GRP_ADDR), GRP_PORT]
}

(The value of the resource /r changes to "5678".)
Multicast notification

C_1
+ ------------------ [ Multicast ] ------------------ S
C_2 (Destination address/port: GRP_ADDR/GRP_PORT)

2.05
Token: 0x7b
Observe: 11
Content-Format: application/cbor
<Other options>
Payload: : "5678"

› 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-alg-crv’ : countersignature curve of the algorithm
- ‘cs-key-kty’ : countersignature key type
- ‘cs-key-crv’ : countersignature curve of the key
- ‘cs-kenc’ : countersignature key encoding
- ‘alg’ : AEAD algorithm
- ‘hkdf’ : HKDF algorithm
C1 registration w/ security

C_1

---

[ Unicast w/ OSCORE ]
---

| 0.05 (FETCH) |
| Token: 0x4a |
| OSCORE: {kid: 1 ; piv: 101 ; ...} |
| <Other class U/I options> |
| 0xff |
| Encrypted payload |
| 0x01 (GET), |
| Observe: 0 (Register), |
| <Other class E options> |
|

(S allocates the available Token value 0x7b.)

(S sends to itself a phantom observation request as coming from the IP multicast address as coming from the IP multicast address GRP_ADDR.)

---

0.05 (FETCH)

Token: 0x7b

OSCORE: {kid: 5 ; piv: 501 ; kid context: 57ab2e; ...}

<Other class U/I options>
0xff
Encrypted payload {0x01 (GET), Observe: 0 (Register), <Other class E options>}

<Counter signature>

(S steps SN_5 in the Group OSCORE Sec. Ctx : SN_5 <= 502)

(S creates a group observation of /r.)

(S increments the observer counter for the group observation of /r.)
C1 registration w/ security

C_1 <----------------- [ Unicast w/ OSCORE ] --------------- S

2.05 (Content)
Token: 0x4a
OSCORE: { piv: 301; ... }
<Other class U/I options>
0xff
Encrypted_payload {
  5.03 (Service Unavailable),
  Content-Format: application/informative-response+cbor,
  <Other class E options>,
  0xff,
  CBOR_payload {
    ph_req : bstr(0x05 | OPT | 0xff | PAYLOAD | SIGN),
    last_notif : bstr(0x25 | OPT | 0xff | PAYLOAD | SIGN),
    tp_info : [1, [0x7b], bstr(SRV_ADDR), SRV_PORT,
               [bstr(GRP_ADDR), GRP_PORT]],
    join_uri : "coap://myGM/ace-group/myGroup",
    sec_gp : "myGroup"
  }
}
C2 registration w/ security

5: Sender ID (‘kid’) of S in the OSCORE group
501: Sequence Number of S in the OSCORE group when S created the group observation
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