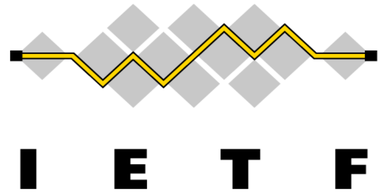
IETF 108

Constrained RESTful Environments WG (core)

Chairs:
Jaime Jiménez <jaime.jimenez@ericsson.com>
Marco Tiloca <marco.tiloca@ri.se>

Mailing list:
core@ietf.org

Jabber:
core@jabber.ietf.org



- We assume people have read the drafts
- Meetings serve to advance difficult issues by making good use of face-to-face communications
- Note Well: Be aware of the IPR principles, according to RFC 8179 and its updates

- Blue sheets
- Jabber Scribe(s)
- Note Taker(s)

Note Well

Any submission to the IETF intended by the Contributor for publication as all or part of an IETF Internet-Draft or RFC and any statement made within the context of an IETF activity is considered an "IETF Contribution". Such statements include oral statements in IETF sessions, as well as written and electronic communications made at any time or place, which are addressed to:

- **The IETF plenary session**
- **The IESG, or any member thereof on behalf of the IESG**
- **Any IETF mailing list, including the IETF list itself, any working group or design team list, or any other list functioning under IETF auspices**
- **Any IETF working group or portion thereof**
- **Any Birds of a Feather (BOF) session**
- **The IAB or any member thereof on behalf of the IAB**
- **The RFC Editor or the Internet-Drafts function**

All IETF Contributions are subject to the rules of RFC 5378 and RFC 8179.

Statements made outside of an IETF session, mailing list or other function, that are clearly not intended to be input to an IETF activity, group or function, are not IETF Contributions in the context of this notice. Please consult RFC 5378 and RFC 8179 for details.

A participant in any IETF activity is deemed to accept all IETF rules of process, as documented in Best Current Practices RFCs and IESG Statements.

A participant in any IETF activity acknowledges that written, audio and video records of meetings may be made and may be available to the public.

https://www.ietf.org/about/note-well/

I E T F

All times are in UTC

Tuesday (100 min)

- 14:10–14:20 Intro, Agenda, Status
- 14:20–14:25 Resource Directory
- 14:25–14:30 Echo-Request-Tag
- 14:30–14:40 CoRE Applications
- 14:40–14:50 Dynlink
- 14:50–15:10 SenML
- 15:10–15:25 Blockwise for DOTS
- 15:25–15:35 AIF
- 15:35–15:50 Flextime (EDHOC+OSCORE)

All times are in UTC

Friday (100 min)

- 14:10–14:15 Intro, Agenda
- 14:15–14:25 CoRECONF
- 14:25–15:30 Group Communication
- 15:30–15:50 Flextime

Agenda Bashing

Intro

Practicalities

- Use the queue request on Meetecho
- Use of queuing at core@jabber.ietf.org
  - **`mic:`** to ask for relaying a question
- This meeting is recorded
- Bluesheets are automatically filled

Published Documents

senml-etch-07 → RFC 8790 !! ✓

published 2020-06

senml-more-units-06 → RFC 8798 !! ✓

published 2020-06

IESG Processing

- draft-ietf-core-resource-directory-25

  In IESG Evaluation (Telechat: 20-08-13)

- draft-ietf-core-stateless-06

  In IESG Evaluation::Revised I-D Needed

In Post-WGLC processing

- draft-ietf-core-dev-urn-07

  On AD Evaluation::Revised I-D Needed

- draft-ietf-core-echo-request-tag-10

  On Shepherd’s Writeup

- draft-ietf-core-oscore-groupcomm-09

  WGLC comments to process

In WGLC

- draft-ietf-core-yang-cbor-13
- draft-ietf-core-comi-10
- draft-ietf-core-sid-14
- draft-ietf-core-yang-library-02

Ends on the 29th of July

Resource Directory

Resource Directory

draft-ietf-core-resource-directory

Zach Shelby, Michael Koster, Carsten Bormann, Peter van der Stok,
Christian Amsüss

2020-07-28

Since IETF107

Processing Secdir and *Genart* reviews

- Major change on Security Policies

  "how" $\rightarrow$ "what"

  "rule" $\rightarrow$ "options"

- Suggest Echo for amplification mitigation, and client identity in simple registration
- Minor clarifications and editorial changes

Echo-Request-Tag

Echo, Request-Tag, and Token Processing

draft-ietf-core-echo-request-tag

Christian Amsüss, John Mattson, Göran Selander

2020-07-28

Since IETF104

Addressing WGLC- and post-WGLC reviews (Francesca, Marco, Klaus)

- 7252 update: RECOMMEND Echo for amplification mitigation
  Give numbers: $\leq 3\times$ request (from QUIC)
  practically: 152 octets free
- Echo: Allow short values
- Echo: Proxies may process it provided they don't deteriorate freshness
- RT: Don't claim to solve to all stateless proxy blockwise problems
- Structure: Introductions grouped with topics
- Improved privacy and security considerations
- Suggest concrete option numbers (twice...)
- Various wording

Problem Details

Problem details

draft-ietf-core-problem-details-01

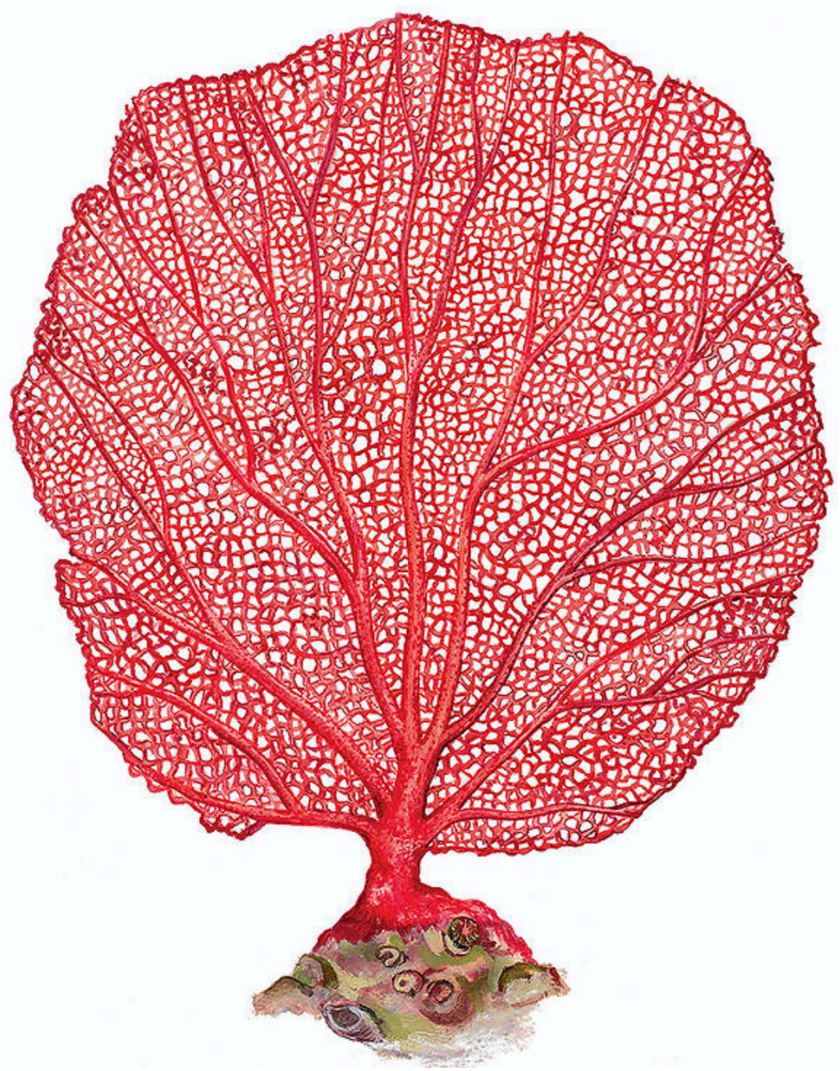
Editorial

- Rebuilding content from the ground up:
  - incrementally include what we are sure about
  - leave out what we are not yet sure about

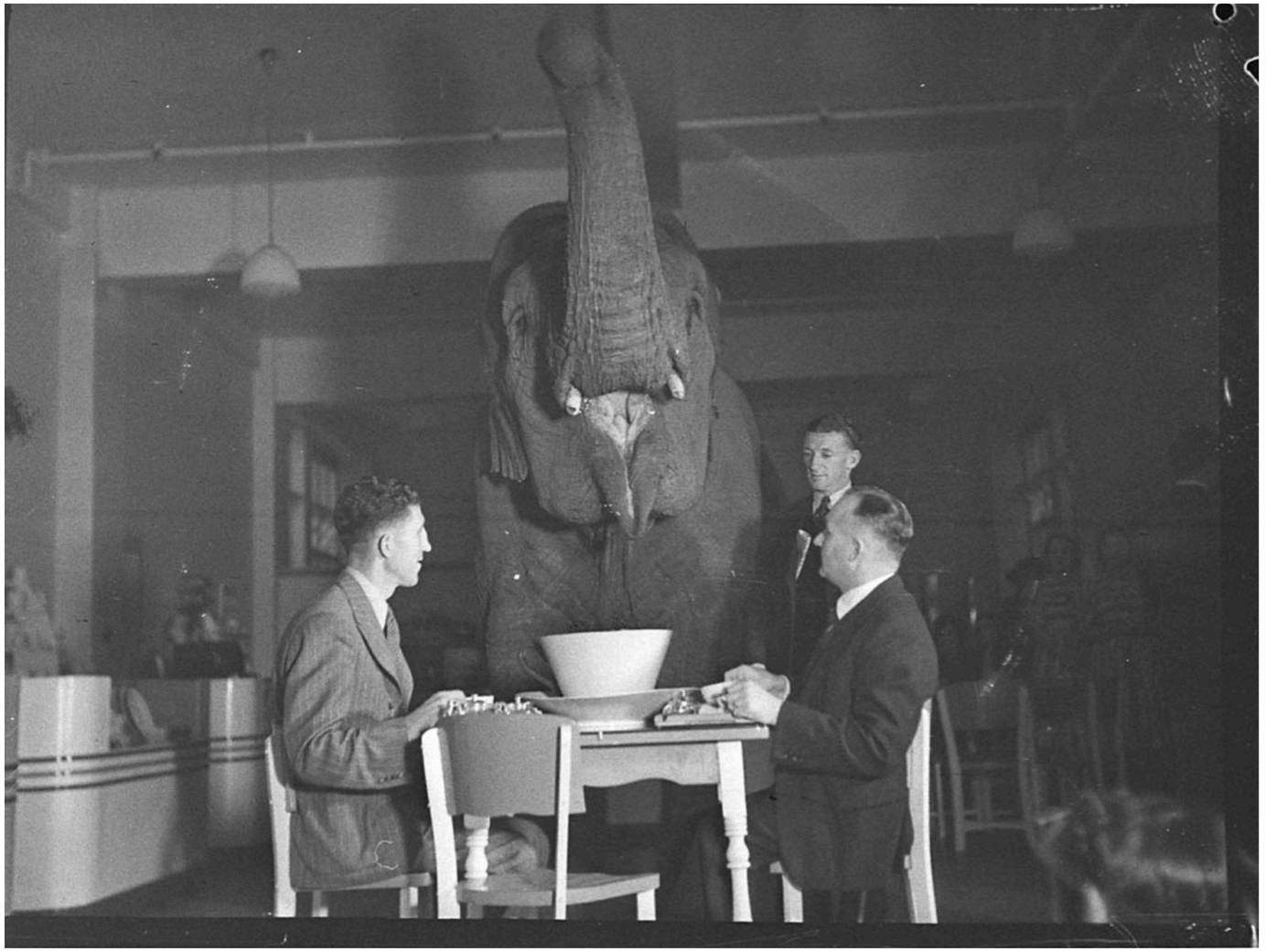
(re)structure

- Base - very small set of common fields
- Feature - extensions per use case (e.g.: tracing, log pipeline, 3rd party error reporting, etc.)

CoRAL all the way



Naming protocol elements



Requirements

- https://github.com/core-wg/core-problem-details/issues/9
  - size/compactness
  - low barrier to entry
  - stability
  - private-public transitions
  - popularity/familiarity to REST API developers
  - we don't consider any existing solution that doesn't prevent collisions

Survey

- Very broad survey of ID schemes
  - IANA (URI, ASN.1, YANG) , LwM2M, Bluetooth, IEEE, ISBN, DOI, DNS, W3C DID, software packages (Go, .NET, Javascript), Twitter Snowflake, etc.
- Work in progress…

Dynlink

draft-ietf-core-dynlink

IETF 108

Dynlink Changelog from versions -08 to -11

- Draft restructuring: introducing the observe attributes first, then dynamic links and then the binding table implementation
- Incorporating feedback received for updates, corrections and clarifications

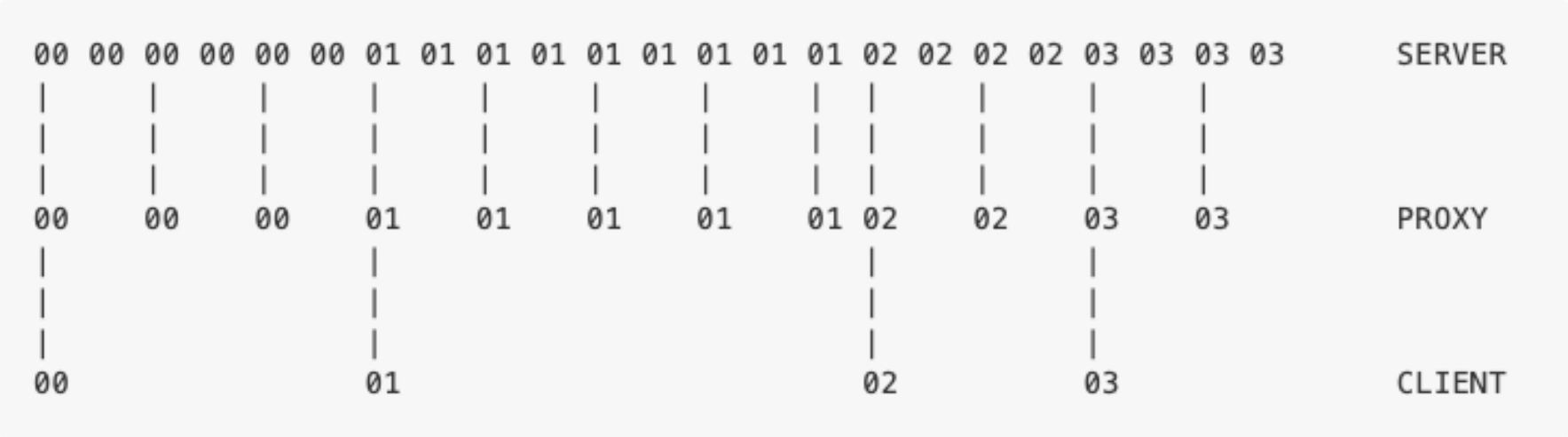
Dynlink Ongoing Discussions (1/3)

- Draft -11 currently discusses notifications arising from setting the observe attributes in terms of reporting values and message transfers
- Substantial editorial changes will be performed to alter the language in order to reflect notifications as RESTful state changes and state transfer

Dynlink Ongoing Discussions (2/3)

- The correct behaviour of pmax is affected by the existence of proxies

We set a conditional observation with `pmax=2` seconds.



- Letting pmax influence the server's Max-Age is the current working consensus

Dynlink Ongoing Discussions (3/3)

- There is a proposal from OMA LwM2M to support 2 new attributes, epmin and epmax:

The "Minimum Evaluation Period" (epmin) and "Maximum Evaluation Period" (epmax) values can be used to configure the device to perform reporting evaluations. After the expiry of epmin, the device MAY immediately perform an evaluation per the "Notification Conditions" above. After the expiry of epmax, the device MUST perform an evaluation per the "Notification Conditions". If both the epmin and epmax attributes are defined, the epmin must be less than the epmax."

- The discussion can be found at https://github.com/core-wg/dynlink/issues/18
- Comments from the working group?

Dynlink Next Steps

- Draft -12 will reflect editorial changes to reflect state transfers when conditional observation attributes are used on a resource
- Future drafts will address pmax, epmin and epmax
- A small discussion group will convene after IETF 108 using Jitsi. Please let authors know if you would like to receive an invitation to join

SenML

SenML Features and Versions

draft-ietf-core-senml-versions-00

Carsten Bormann

IETF 108, 2020-07-28, in the cloud

RFC 8428, SenML: Version 10

- RFC 8428 SenML evolution path: allows for version upgrade
- Default version: 10 (accounting for previous development versions)
- Can set higher: [{**“bver":11**, “v”:4711}, …]
- Semantics to be defined by RFC updating RFC 8428

Objective: extensibility

- Over time, new specifications will add features to SenML
- Version number is a unitary declaration:
  implementation of certain features is needed by the receiver to process SenML pack

- Version number N+1 includes all features of version number N (total order)
  - Except for features that are **deprecated**

Version numbers are stupid

- Well, they work well for document revisions and software releases
- Not so great for protocols and other interface specifications
- Long discussion in T2TRG:
  Version numbers force creating a total order on a set of new features
- Better: declare individual features
  - Could do with must-understand fields: `bfeature1_: true`
  - But maybe can leverage the version number?

Proposal: interpret version number as bits

- A number can be used as a bit array
- Version 10 = 1010_2, i.e. features 1 and 3 ($2^1 + 2^3 = 10$)
- Add bits for additional features
- Proposed feature 4: use of Secondary Units ($2^4 = 16$)
  Version number with that additional feature would thus be 26
- Feature code can go up to 52 (53-bit integers in JSON):
  48 remaining now (after secondary units)

53: wasn’t that an evil number?

- Yes.
- But it could be all we need:
  - As the number of features that can be registered has a hard limit (48 codes left at the time of writing), the designated expert is specifically instructed to maintain a frugal regime of code point allocation, keeping code points available for SenML Features that are likely to be useful for non-trivial subsets of the SenML ecosystem.
  - Quantitatively, the expert could for instance steer the allocation to not allocate more than 10 % of the remaining set per year.

draft-ietf-core-senml-versions-00

- Defines the feature system:
  New Registry under the SenML registry
  Reserving feature code 0..3 for “10 = 1010_2”
  Specification required, frugality mandate to designated expert
- **Updates** the RFC 8428 version number to use that system
- Registers **feature code 4**: Use of secondary units

- Now WG draft, submitted 2020-05-13
  - Referenced from RFC 8798 (senml-more-units)
- No technical changes from 2020-03-06 draft-bormann-core-senml-versions-01

Next steps

- Need more reviews!
  This is just about the interpretation for one field…
- Proposal: Process these reviews, check if we are done, WGLC

SenML Data Value Content-Format Indication

draft-ietf-core-senml-data-ct-02

Carsten Bormann

IETF 108

Content-Format indication

- SenML Records can contain (binary) "data values" in a "vd" field

```
[
 {"bn":"urn:dev:ow:10e2073a01080063:", "n":"temp", "v":7.1},
  {"n":"open", "vb":false},
  {"n":"nfc-reader", "vd":"aGkgCg"}
]
```

- This draft: new Content-Format indication ("ct") field to indicate the Content-Format of the data in the SenML Record

Example SenML Record with data value and Content-Format indication

```
{"n":"nfc-reader", "vd":"gmNmb28YKg", "ct":60}
```

CBOR CoAP Content Format

base64(

```
82          # array(2)
  63        # text(3)
    666F6F  # "foo"
  18 2A     # unsigned(42)
```

)

Updates for draft-ietf-senml-data-ct-02

- must-understand-ct-field (“ct_”) proved inscrutable
  - Interaction with normal field (“ct”) in resolution process is unmanageable
  - So we got rid of it again (thanks for the suggestion, Klaus)
  - Would have been nice to have, but don’t know how to do it

- Authors believe this is now ready for WGLC

Blockwise for DOTS

New CoAP Block-Wise Transfer Options For Faster Transmission

draft-bosh-core-new-block-04

IETF CoRE Meeting, 28th July 2020

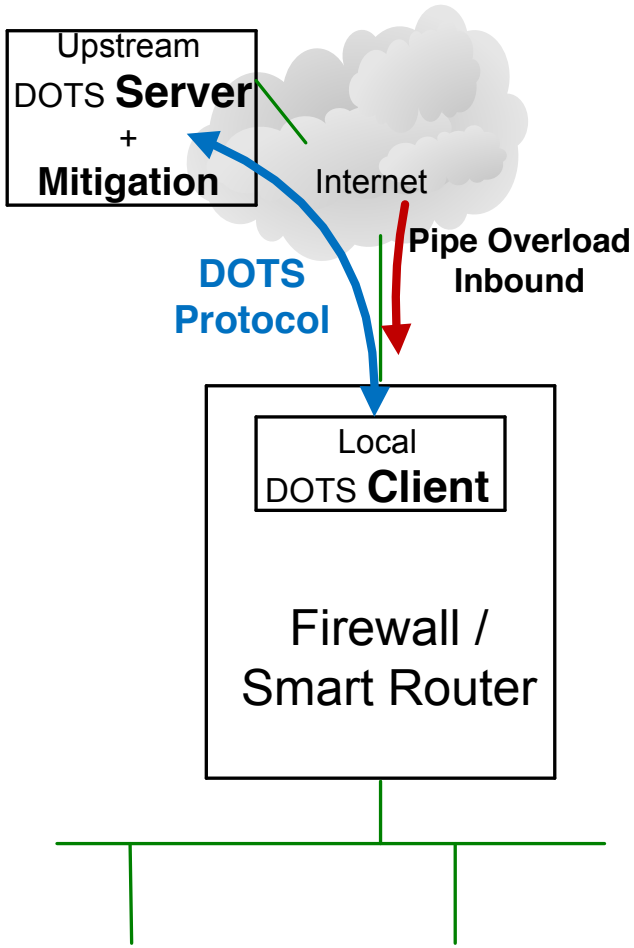
Mohamed Boucadair

Jon Shallow

Agenda

- Handling blocks in lossy environments: The DOTS Case
- The Solution
  - Block3 Option
  - Block4 Option
- Next Steps

Sample Target Deployment



- DDoS Open Threat Signalling (DOTS)
- DOTS: App – CBOR – CoAP – DTLS – IP
- Client requests mitigation (NON)
- Server updates with simple DOTS mitigation status (NON)
- Inbound Pipe Overload
  - Clients can still request mitigations
  - Mitigation should be able to control pipe overload
- See RFC8782 for more details

DOTS-inferred CoAP Requirements

- Need to transfer body with ***multiple payloads***
- Handle packet loss
  - ***Need recovery*** mechanism
- Cannot rely on getting responses (***Pipe overload***)
  - Request/Response model lock-step fails
- ***Fast*** transfer for large bodies
  - Lock-step model slows things down (RTT)
  - Packet interchange reduction
- ***Utilize*** other CoAP options where possible
- ***Maintain existing CoAP*** ethos/methodology

Solution

- BLOCK3 with BLOCK1 characteristics
- BLOCK4 with BLOCK2 characteristics

Number	C	U	N	R	Name	Format	Length	Default
TBA1	x				Block3	uint	0-3	(none)
TBA2	x			x	Block4	uint	0-3	(none)

- OSCORE Class E and Class U

BLOCK3

- BLOCK3 has BLOCK1 characteristics
- *Individual payload 2.xx response not required*
  - *2.31 (Continue) not used*
- Use of NON (recommended) or increase of NSTART
- Requires CoAP Option Request-Tag unique per body
- Each request payload has unique Token
- New TBA3 4.xx (Missing Payloads) to indicate missing payloads
- Every MAX_PAYLOAD (default 10) pause/check guard (ACK_TIMEOUT or CON)
- Body subject to PROBING_RATE

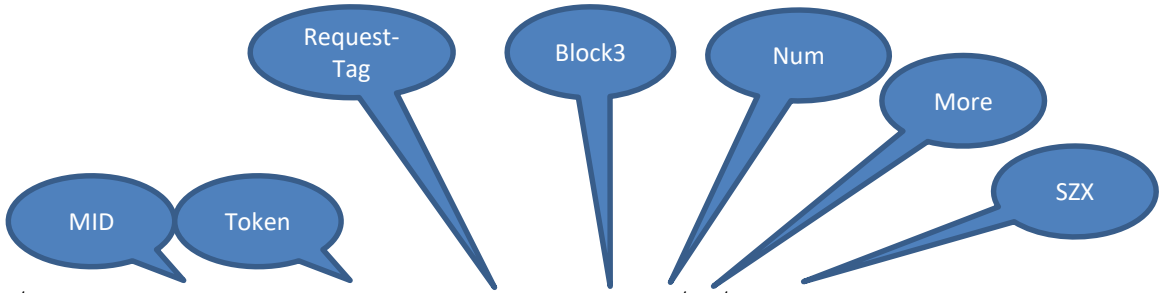
Congestion Control

Response TBA3 4.xx (Missing Payloads)

- CBOR encoded diagnostic response
- Content-Format “application/missing-blocks+cbor-seq”
- CDDL

```
TBA3-payload = (request-tag, missing-block-list)
; A copy of the opaque Request-Tag value
request-tag = bstr
missing-block-list = [1 * missing-block-number]
; A unique block number not received
missing-block-number = uint
```

BLOCK3 Example




```
CoAP        CoAP
Client      Server
  | Success  |
  +--------->| NON PUT /path M:0x01 T:0xf0 RTag=10 B3:0/1/1024
  +--------->| NON PUT /path M:0x02 T:0xf1 RTag=10 B3:1/1/1024
  +--------->| NON PUT /path M:0x03 T:0xf2 RTag=10 B3:2/1/1024
  +--------->| NON PUT /path M:0x04 T:0xf3 RTag=10 B3:3/0/1024
  |<---------+ NON 2.04      M:0xf1 T:0xf3

  | Failure  |
  +--------->| NON PUT /path M:0x05 T:0xe0 RTag=11 B3:0/1/1024
  +---->X    | NON PUT /path M:0x06 T:0xe1 RTag=11 B3:1/1/1024
  +---->X    | NON PUT /path M:0x07 T:0xe2 RTag=11 B3:2/1/1024
  +--------->| NON PUT /path M:0x08 T:0xe3 RTag=11 B3:3/0/1024
  | Retry    |
  |<---------+ NON TBA3      M:0xf2 T:0xe3 [Missing 1,2 for RTag=11]
  +--------->| NON PUT /path M:0x09 T:0xe4 RTag=11 B3:1/1/1024
  +---->X    | NON PUT /path M:0x0a T:0xe5 RTag=11 B3:2/1/1024
  | Retry    |
  |<---------+ NON TBA3      M:0xf3 T:0xe4 [Missing 2 for RTag=11]
  +--------->| NON PUT /path M:0x0b T:0xe6 RTag=11 B3:2/1/1024
  |<---------+ NON 2.04      M:0xf4 T:0xe6
```

BLOCK4

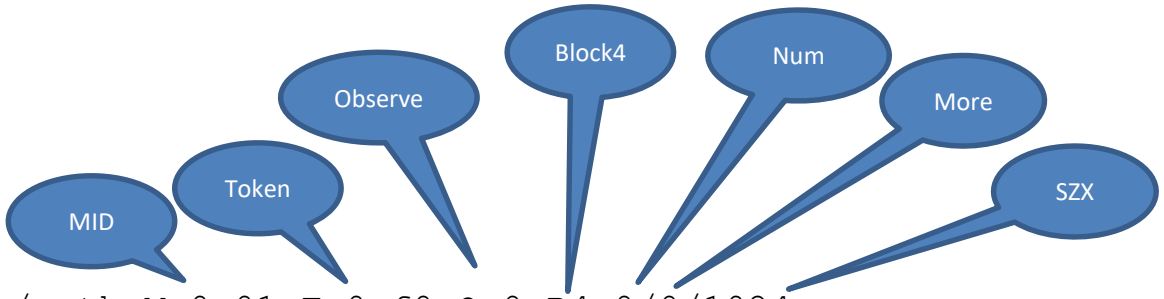
- BLOCK4 has BLOCK2 characteristics
- GET with BLOCK4 triggers BLOCK4 response instead of BLOCK2 if needed
  - Missing blocks indicated by GET with multiple BLOCK4
- Use of NON (recommended) or increase of NSTART
- Requires CoAP Option ETag unique per body
- Every MAX_PAYLOAD (default 10) pause/check guard (ACK_TIMEOUT or CON)
- Body subject to PROBING_RATE

Congestion Control

BLOCK4 with Observe

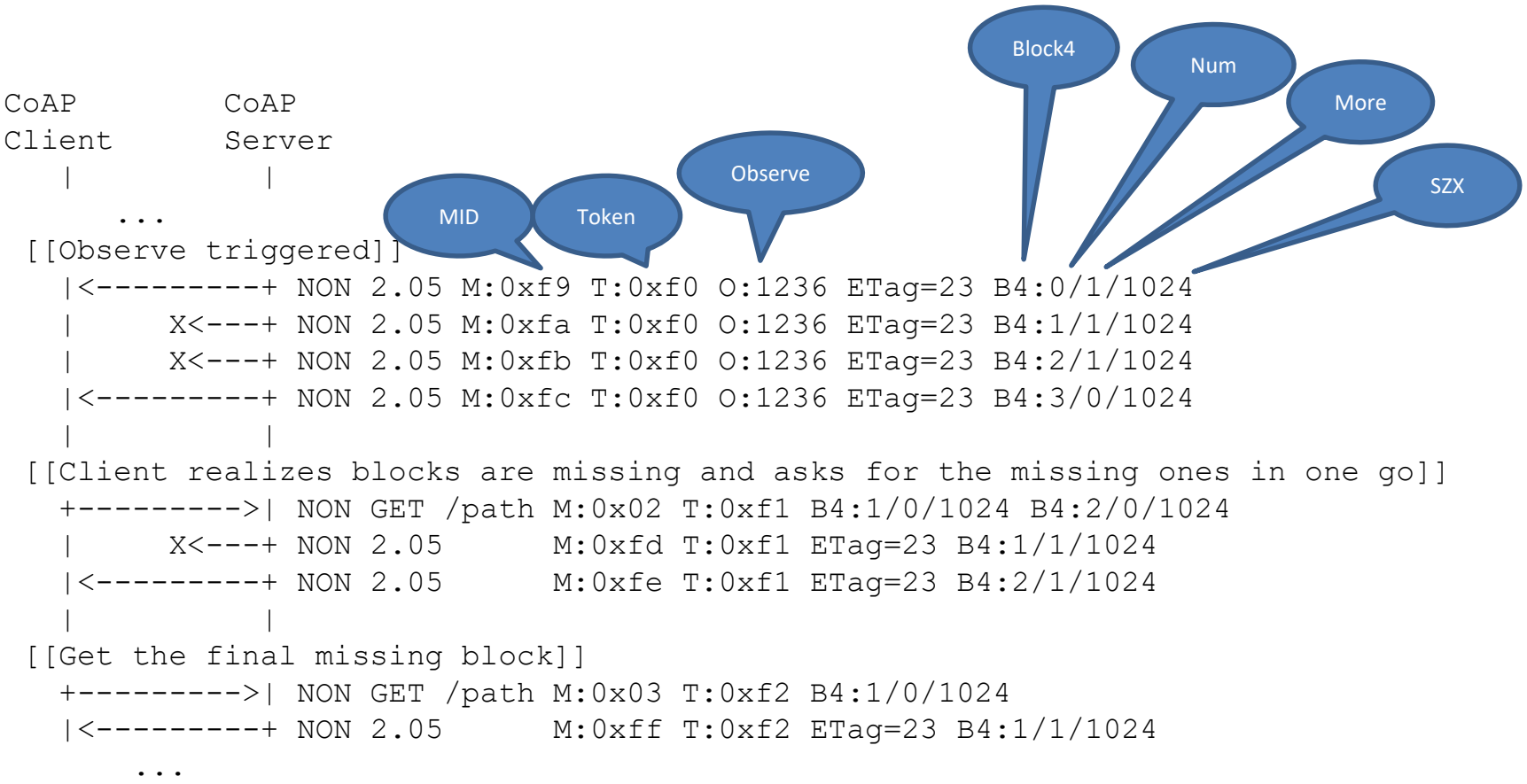
- Same Observe value returned in all of the pseudo responses for the same body
  - Block Number 0 could get dropped
- Same Token is used in the set of pseudo responses
- New Token used for requesting missing blocks
  - Set of retry pseudo responses use new Token
- Next Observe pseudo response uses original Observe setup Token

BLOCK4 Example 1




```
 CoAP          CoAP
Client        Server
  |             |
  +---------->| NON GET /path M:0x01 T:0xf0 O:0 B4:0/0/1024
  |<----------+ NON 2.05 M:0xf1 T:0xf0 O:1234 ETag=21 B4:0/1/1024
  |<----------+ NON 2.05 M:0xf2 T:0xf0 O:1234 ETag=21 B4:1/1/1024
  |<----------+ NON 2.05 M:0xf3 T:0xf0 O:1234 ETag=21 B4:2/1/1024
  |<----------+ NON 2.05 M:0xf4 T:0xf0 O:1234 ETag=21 B4:3/0/1024
     ...
[[Observe triggered]]
  |<----------+ NON 2.05 M:0xf5 T:0xf0 O:1235 ETag=22 B4:0/1/1024
  |<----------+ NON 2.05 M:0xf6 T:0xf0 O:1235 ETag=22 B4:1/1/1024
  |<----------+ NON 2.05 M:0xf7 T:0xf0 O:1235 ETag=22 B4:2/1/1024
  |<----------+ NON 2.05 M:0xf8 T:0xf0 O:1235 ETag=22 B4:3/0/1024
     ...
```

BLOCK4 Example 2



Status & Next Steps

- The draft was discussed in two dedicated interim meetings
  - All comments raised in these interims and also in the mailing list are addressed
  - Simplified design with reuse of existing CoAP options
- Request adoption as a WG Document

Thank You

AIF

Authorization Information Format (AIF)

draft-bormann-core-ace-aif-09

Carsten Bormann

IETF 108

Problem: Convey authorization information

- Authorization ("Access Control") is usually modeled by the *Access Control Matrix* (Lampson 1971), a function mapping a Subject and an Object to a set of Permissions (Rights): M: S x O $\rightarrow 2^R$
- This is often sliced by object into an ACL (Access Control List)
- To know about the authorizations of a client, we slice by subject: "Capability list" or "C-list", C: O $\rightarrow 2^R$
- Binding to subject done outside, e.g. in access grant (in certain kinds of secure channel, or providing some subject authentication verifier, e.g., a Proof of Possession token)

draft-bormann-core-ace-aif

- Represent C-list as an array of pairs:

  AIF-Generic<Toid, Tperm> = [* [Toid, Tperm]]

- For the RESTful case, specialize to:

  AIF-REST = AIF-Generic<path, permissions>

  path = tstr *; URI relative to enforcement point — O*

  permissions = uint .bits methods ; *what methods are allowed* — 2^R

  methods = &(GET: 0 POST: 1 PUT: 2 DELETE: 3 FETCH: 4 PATCH: 5 iPATCH: 6)

- Could define other cases, e.g., for MQTT (outside scope of this spec)

Dynamic permissions in draft-bormann-core-ace-aif-09

- AIF is designed for static resources of IoT devices
- Actions often lead to dynamic “action resources” (pointed to by Location-* response options)
- Idea: Derive permissions from base resource
- methods /= &(Dynamic-GET: 32 Dynamic-POST: 33 Dynamic-PUT: 34
  Dynamic-DELETE: 35
  Dynamic-FETCH: 36 Dynamic-PATCH: 37 Dynamic-iPATCH: 38)
- These permissions say what can be done to resources created from the resource to which they apply (a bit like NFSv4 inheritance)

Status for draft-bormann-core-ace-aif-09

- AIF has been around since 2014 (part of DCAF work);
  was listed as contribution on ACE BOF at IETF 89
- ACE has recently noticed a need to go ahead with standardizing this;
  WG adoption call ends today (**you can still put in your opinion**!)
  - Ben Kaduk:
    What else exists like this? How could AIF be used outside ACE?

- On agenda of ACE meeting tomorrow

CoRECONF

CORECONF

Andy Bierman
Michel Veillette
Peter van der Stok
Alexander Pelov
Ivaylo Petrov

Status yang-cbor -12 -> -13

- Clarified and fixed some examples
- Cleared confusion between *YANG template* and *yang-data extensions*
- Made '**bits'** encoding more efficient in case of big position values
- Added *application/yang-data+cbor* media-type and *application/yang-data+cbor; id=name* content-type definition
- Fixed inconsistency between text and CBOR tag registration table

Status SID -12 -> -14

- Renamed SID to YANG SID
- Clarified early allocation
- Added content type registration of *application/yang-data+cbor; id=sid*
- Clarified that new SIDs are required for change in semantics of nodes
- Do not depend on JSON encoding for definitions
- Made explicit the use of YANG 1.1 for *ietf-sid-file*
- Added missing YANG module related namespace registration
- Removed the default SID range sizes

Status comi -09 -> 10

- Only use the name CORECONF as CoMI does not seem to need it's proper name
- Updated Media-type and Content-type registrations
- Extended the security considerations section
- Updated and clarified some examples

Status yang-library -01 -> -02

- Added missing YANG module related namespace registration
- Extended the security considerations section

Steps forward

- Result from the WGLC and next steps

Group Communication

Group Communication for the Constrained Application Protocol (CoAP)

draft-ietf-core-groupcomm-bis-01

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

IETF 108 - CoRE WG, July 31st, 2020

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 …)
  - Unsecured CoAP or group-OSCORE-secured communication
  - Principles for secure group configuration
  - Use cases (appendix)

Overview of -01 updates

- Mostly addressed Jim's review at [1] – Thanks!
- Clarifications on group membership for client-only nodes
  - Don't have to be in an application group or CoAP group
  - Have to be in the used security group
- Response suppression
  - No need to talk of "legitimate" requests (was issue #4)
  - Suppress if nothing to say, unless the application requires to respond anyway
- Token reuse
  - Clearer indications and differences compared to the unicast case

[1] https://mailarchive.ietf.org/arch/msg/core/CkoNseJhJgALEs3iOLMqUZhEarI/

Overview of -01 updates

- When proxies are used
  - Clarifications on stop accepting responses to group requests
  - The client has more (app-)context information to judge when stopping
- Multicast scope to use
  - Configure in advance, i.e. not up to the client to decide
- Clarification on cancelling group observations
- Usage of Group OSCORE
  - Mentioned both the group mode and the pairwise mode (was issue #5)
  - Creation/management of OSCORE groups addressed in other documents
  - Updated security considerations; reference to COSE-bis documents

Open Github issues

- The UDP port may change (issue #1)
  - Multicast request → Src: 59101 Dst: 9999
  - Unicast response → Src: 5683 Dst: 59101
- The outcome of the thread at [2] seems to converge to:
  - Both source address and source port number of the response are irrelevant to the successful processing at the client
- Planned update
  - The source port number of the response can differ from the destination port number of the request. A client MUST be able to handle this.
  - Issues with that?

[2] https://mailarchive.ietf.org/arch/msg/core/d2CJN0g-ksq9uf0hDqBCRqcHz5g/

Open Github issues

- Requirements for response suppression (issue #2)
  - Operate on Response Code Class, instead of Response Code
  - Planned to switch to Response Code Class, as NoResponse does
  - Issues with that?

- Use URI-Host for naming application groups (issue #3)
  - *“If encoded in the CoAP group URI, the information typically gets removed in the CoAP request sent over the wire. Then the receiving server cannot use it.”*
  - *”It can be added to an outgoing CoAP request (with the group URI already resolved to IP address). Then it influences the choice of application group, because each virtual server will have a different set of resources hosted.”*
  - Should the draft explicitly admit it?

More open points

- Client support for admin-local scope
  - It's not in RFC 7252, but it's in RFC 7390 for discovery use cases.
  - Keep it?
- Mapping of application groups and security groups, see [3]
  - Many app groups using one sec group is fine.
  - One app group using many sec groups is "delicate".
  - **Case A**: the sec groups use different algorithms/parameters → A server joins all of them; a client joins any that it supports. This looks ok.
  - **Case B**: the sec groups express different access control properties → This is problematic and a trouble for applications; better rely on resource properties.
  - Proposal: include Case A as relevant example; not recommend Case B

[3] https://mailarchive.ietf.org/arch/msg/core/4JtUVaB-XG_g0i_8v8CEMGyNdO8/

Next steps

- Work on open Github issues

- Address open points
  - Two left from Jim's review of -00
  - Others also raised today

- Interop of selected functions in CoAP implementations
  - "Observe + multicast" – Locally tested, w/ and w/o Group OSCORE
  - Limited usage of Blockwise (first multicast request with Block2)

Thank you!

Comments/questions?

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

Group OSCORE - Secure Group Communication for CoAP

draft-ietf-core-oscore-groupcomm-09

Marco Tiloca, RISE
Göran Selander, Ericsson
Francesca Palombini, Ericsson
Jiye Park, Universität Duisburg-Essen

IETF 108, CoRE WG, July 31st, 2020

Update since the April meeting

- Version -09 submitted in June
  - Addressed open points raised in April
  - Addressed remaining points from Jim's and Christian's reviews
- WGLC on -09, ended the 20th of July
  - Comments from Jim [1] and Peter [2] – Thanks!
- 2nd interop during this Hackathon
- New discussion item on separate pairwise space for PIVs

[1] https://mailarchive.ietf.org/arch/msg/core/VMhrAPEt4TE8jahatVd1EoDzdMI/

[2] https://mailarchive.ietf.org/arch/msg/core/tOHaMpTrWJ2CfsX2E5IGS8qpt-U/

Main updates in -09

- Two different operating modes
  - **Group mode** – Main and usual mode
    - MUST be supported
    - Encryption with group keying material; signature included
  - **Pairwise mode**
    - MAY be supported – If so, use for unicast requests (e.g., Block-wise, Echo, …)
    - Encryption with derived pairwise keying material; no signature

- New Group Flag bit in the OSCORE option
  - Set to 1 if the message is protected in group mode
  - Set to 0 if the message is protected in pairwise mode (aligned with OSCORE)

Main updates in -09

- Pairwise key derivation
  - Same construction from 3.2.1 of RFC 8613
  - **Pairwise key = HKDF(Sender/Recipient Key, DH Shared Secret, info, L)**
    - Sender Key of the sender node, i.e. Recipient Key of the recipient side
    - Static-static DH shared secret, from one's private key and the other's public key
  - Compatible with ECDSA and EdDSA (after coordinate remapping)

- Major editorial revision of Section 2 "Security Context"
  - Improved presentation of Common/Sender/Recipient context
  - Derivation of keys for the pairwise mode explained here
  - Update and loss of the Security Context (e.g., in case of rekeying and reboot)

- Usage of update registries and COSE capabilities from COSE-bis

Report from IETF 108 Hackathon

- Tests with RISE and August Cellars implementations

- Successful interop tests
  - Communication in group mode
  - Derivation of pairwise keys

- Successful local tests
  - Communication in pairwise mode

Main points from WGLC

- Information is now replicated in the Security Context
  - Sufficient to keep ‘Counter Signature Parameters’
  - Delete ‘Counter Signature Key Parameters’ as redundant.
  - Issues with that?

- Curve remapping in the pairwise mode, for DH secret derivation
  - Current text Ed25519 (MTI) → Montgomery for X25519 (MTI if supporting pairwise mode)
  - **Jim**: *consider remapping to the short-Weierstrass curve instead*
  - Mention just as possible alternative? Or have Wei25519 and ECDH25519 as MTI?

- Wrap-around of Sender Sequence Number (SSN)
  - **Jim**: *is the wrap-around of the SSN or of the PIV?*
  - It should really be the SSN, which is used as PIV. Anything missing to clarify?

Main points from WGLC

- Support for Observe, across group rekeying
  - Now the client and server store the 'kid' of the original Observe request
  - That value is the 'request_kid' in the external_aad of notifications, also after rekeying
  - **Jim**: *should we store also the kid context?*
  - No need to, it's not part of the 'external_aad'. Keep as is?

- New Context established → Reset the Sender Sequence Number to 0 ?
  - Now it's not reset, unless the application decides differently
  - **Jim**: *having it reset simplifies the detection of group rekeying*
  - Reset also Replay Windows and Observe Numbers of ongoing observations
  - Change to reset by default? Can the application do differently?

Separate SSN spaces

- Right now: every node has a single SSN space
  - Used for PIVs both in group mode and pairwise mode

- New proposal from Jim: **two separate SSN spaces**
  - One SSN for the group mode
  - For each associated recipient
    - One pairwise SSN – NEW
  - For each associated client
    - One group Replay Window
    - One pairwise Replay Window – NEW

Separate SSN spaces

- Pros
  - Less frequent exhaustion of SSN values
  - Reuse of OSCORE code for the pairwise mode

- Cons
  - Higher storage (extra SSNs and Replay Windows)
  - Might result in greater communication overhead (fresh PIV in some responses)

- Issues
  1. The server might have to use its fresh PIV (no reusage of request PIV)
     - E.g., when request and response are protected in different modes
  2. Separate synchronization of the two spaces for servers
     - The synch method using Echo needs some adaptation (see Appendix E.3)

Separate SSN spaces - Issue #1

1. C → S : Request in Group Mode
   - kid: SID_C ; piv: $gPIV_C$
   - Nonce built from { SID_C , $gPIV_C$ } ; Key: gK_C
2. S → C : Response in Pairwise Mode
   - kid: SID_S ; piv: NONE
   - Nonce built from { SID_S , $gPIV_C$ } ; Key: pK_{SC}
3. C → S : Request in Pairwise Mode
   - kid: SID_C ; piv: $pPIV_{CS}$
   - Nonce built from { SID_C , $pPIV_{CS}$ } ; Key: pK_{CS}
4. S → C : Response in Pairwise Mode
   - kid: SID_S ; piv: NONE
   - Nonce built from { SID_S , $pPIV_{CS}$ } ; Key: pK_{SC}

Request and response are protected in different modes

AND

The server reuses the request PIV (PIV reflection)

If $gPIV_C$ == $pPIV_{CS}$, in (1) and (3)

Nonce reusage with pK_{SC} , in (2) and (4)

{ SID_S , $gPIV_C$ } == { SID_S , $pPIV_{CS}$ }

Separate SSN spaces - Issue #2

1. C → S : Request in group mode
   - With client's group PIV
2. S → C : Response in pairwise mode
   - With server's pairwise PIV and Echo option
   - S stores <kid, gid, piv> from the request at (1)
3. C → S : Request in pairwise mode
   - With client's pairwise PIV and Echo option
   - Should also include the client's group PIV

Where?

a) In a new CoAP option

b) In the payload, next to the ciphertext
   - Length signaled in the OSCORE option

- Need to integrity protect?
- How for (b)? Use the external_aad ?
  - It deviates from OSCORE format
  - Not ideal for code reuse

› Need more discussion, especially with implementers
   - Weigh pros/cons and performance tradeoffs

› Opinions about separate SSN spaces?

Next steps

- Addressing WGLC comments in version -10
  - Jim
  - Peter

- More discussion on separate PIVs for the pairwise mode

- More interop tests in pairwise mode

Thank you!

Comments/questions?

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

Discovery of OSCORE Groups with the CoRE Resource Directory

draft-tiloca-core-oscore-discovery-06

Marco Tiloca, RISE
Christian Amsüss
Peter van der Stok

IETF 108, CoRE WG, July 31st, 2020

Recap

- A newly deployed device:
  - May not know the OSCORE groups and their Group Manager (GM)
  - May have to wait GMs to be deployed or OSCORE groups to be created

- Use web links for discovery – typically through the Resource Directory (RD)
  - Discover an OSCORE group and retrieve information to join it
  - Practically, discover the links to join the OSCORE group at its GM
  - CoAP Observe supports early discovery and changes in group information

- Use resource lookup, to retrieve:
  - The name of the OSCORE group
  - A link to the resource at the GM for joining the group

Updates overview

- Addressed review of -05 from Jim – Thanks!
  - https://mailarchive.ietf.org/arch/msg/core/h62d2c2mYmG43ykz52KvbbEpgDc/
  - Some new open points (later slides)

- Revised terminology about groups
  - Now better aligned with *draft-ietf-core-groupcomm-bis*

- Clarified limitation of Link-Format as non typed
  - We can't signal an algorithm that has string value "-10" in the COSE registry
  - No such problem if we use CoRAL

Updates overview

- Fairhair/BACnet example
  - Removed the double registration
  - Removed registration of membership to application groups
    - Feature not defined in the RD document; we don't want to introduce it here
    - Common practice in some deployments; it can be in a separate document
  - Clarified that it's just an example, with no prescriptive intentions

- Added some text on one application group using many security groups
  - As of now, general reference to application policies
  - To be refined, based on the outcome of [1] related to *draft-ietf-core-groupcomm-bis*
  - Further discussion required: Which security groups must a participant join?

[1] https://mailarchive.ietf.org/arch/msg/core/4JtUVaB-XG_g0i_8v8CEMGyNdO8/

Updates overview

- Examples in CoRAL
  - Now moved to the document body
  - Next to the Link-Format examples
    - Registration
    - Update with re-registration
    - Lookup #1, Lookup #2

- New Appendix A
  - Full Fairhair/BACnet example in CoRAL

- This version -06 has now full support for both Link-Format and CoRAL RD

```
Request: Joining node -> RD

Req: GET coap://rd.example.com/rd-lookup/res
  ?rt=core.osc.mbr&sec-gp=feedca570000
Accept: TBD123456 (application/coral+cbor)
Observe: 0

Response: RD -> Joining node

Res: 2.05 Content
Observe: 24
Content-Format: TBD123456 (application/coral+cbor)

Payload:
#using <http://coreapps.org/core.oscore-discovery#>
#using reef = <http://coreapps.org/reef#>
#using iana = <http://www.iana.org/assignments/relation/>

#base <coap://[2001:db8::ab]/>
reef:rd-item </group-oscore/feedca570000> {
    reef:rt "core.osc.mbr"
    sec-gp "feedca570000"
    app-gp "group1"
    cs_alg -8
    cs_alg_crv 6
    cs_key_kty 1
    cs_key_crv 6
    cs_kenc 1
    iana:authorization-server <coap://as.example.com/token>
}
```

Open points

- When registering an OSCORE group to the RD
  - Possible to register related link to an Authorization Server (AS)
  - The AS is associated to the GM of the OSCORE group

```
Request: GM -> RD

Req: POST coap://rd.example.com/rd?ep=gm1
Content-Format: 40
Payload:
</group-oscore/feedca570000>;ct=41;rt="core.osc.mbr";
                              sec-gp="feedca570000";app-gp="group1";
                              cs_alg="-8";cs_alg_crv="6";
                              cs_key_kty="1";cs_key_crv=6";
                              cs_kenc="1",
<coap://as.example.com/token>;
      rel="authorization-server";
      anchor="coap://[2001:db8::ab]/group-oscore/feedca570000"

Response: RD -> GM

Res: 2.01 Created
Location-Path: /rd/4521
```

- Jim: not sure it should be the GM to register the “rel” link to the AS
- Who else can that be? It’s about accessing resources at the GM.
  - The GM also knows about that AS already when the group is created

Open points

- When registering an OSCORE group to the RD
  - The GM indicates the names of the application groups using the OSCORE group
  - Now we don't say how the GM knows the application groups

```
Request: GM -> RD

Req: POST coap://rd.example.com/rd?ep=gm1
Content-Format: 40
Payload:
</group-oscore/feedca570000>;ct=41;rt="core.osc.mbr";
                              sec-gp="feedca570000";app-gp="group1";
                              cs_alg="-8";cs_alg_crv="6";
                              cs_key_kty="1";cs_key_crv=6";
                              cs_kenc="1",
<coap://as.example.com/token>;
      rel="authorization-server";
      anchor="coap://[2001:db8::ab]/group-oscore/feedca570000"

Response: RD -> GM

Res: 2.01 Created
Location-Path: /rd/4521
```

- Suggestion from Jim in the "CoRAL and forms" discussion [2].
  - Related to the GM admin interface in *draft-tiloca-ace-oscore-gm-admin*
  - When creating the OSCORE group at the GM, indicate also the application groups

[2] https://mailarchive.ietf.org/arch/msg/core/BoYGYmEpJMUS8bk4PNHOEaFFcdU/

Open points

- We now use a resource type
  - rt = “core.osc.mbr”
  - Group-membership resource of an OSCORE Group Manager
- Should we have also an if= ?

```
Request: GM -> RD

Req: POST coap://rd.example.com/rd?ep=gm1
Content-Format: 40
Payload:
</group-oscore/feedca570000>;ct=41;rt="core.osc.mbr";
                              sec-gp="feedca570000";app-gp="group1";
                              cs_alg="-8";cs_alg_crv="6";
                              cs_key_kty="1";cs_key_crv=6";
                              cs_kenc="1",
<coap://as.example.com/token>;
      rel="authorization-server";
      anchor="coap://[2001:db8::ab]/group-oscore/feedca570000"

Response: RD -> GM

Res: 2.01 Created
Location-Path: /rd/4521
```

- Probably it does not matter that much, but …
- Compare *draft-ietf-ace-key-groupcomm*:
  - The group’s parent uses `if=ace.group`

Summary and next steps

- Addressed Jim’s review
- Revised CoRAL examples in the document body
- Next steps
  - Close open points from Jim’s review
  - Bridge with *ace-oscore-gm-admin* - The GM knows the names of application groups
- Need for reviews

Thank you!

Comments/questions?

https://gitlab.com/crimson84/draft-tiloca-core-oscore-discovery

Backup

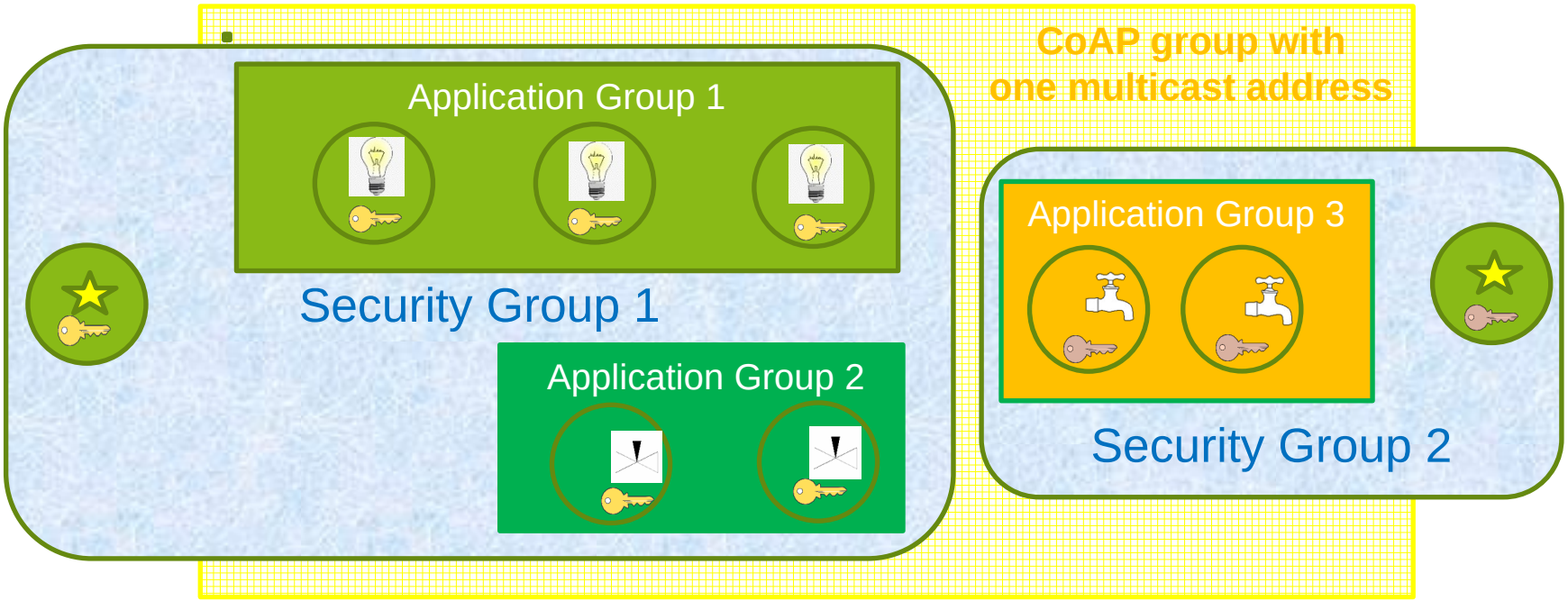
Application/CoAP/Security Groups

- Application group
  - Defined in {RD} and reused as is
  - Set of CoAP endpoints sharing a pool of resources
  - Registered and looked up just as per Appendix A of {RD}

- CoAP Group
  - Defined in *draft-ietf-core-groupcomm-bis*
  - Set of CoAP endpoints listening to the same IP multicast address
  - The IP multicast address is the ‘base’ address of the link to the application group

- (OSCORE) Security Group
  - Set of CoAP endpoints sharing a common security material (e.g. OSCORE Ctx)
  - A GM registers the group-membership resources for accessing its groups

Application vs. Security Groups



Client of application group

Different key sets

Resources for given function

Alg/key related parameters

- New optional parameters for a registered group-membership resource
  - (*)(**) *cs_alg* : countersignature algorithm, e.g. “EdDSA”
  - (*) *cs_alg_crv* : countersignature curve (if applicable), e.g. “Ed25519”
  - (*) *cs_key_kty* : countersignature key type, e.g. “OKP”
  - (*) *cs_key_crv* : countersignature curve (if applicable), e.g. “Ed25519”
  - (*) *cs_kenc* : encoding of public keys, e.g. “COSE_Key”
  - (**) *alg* : AEAD algorithm
  - (**) *hkdf* : HKDF algorithm

- Benefits for a joining node, when discovering the OSCORE group
  - (*) No need to ask the GM or to have a trial-and-error when joining the group
  - (**) Decide whether to join the group or not, based on supported the algorithms

Registration

- The GM registers itself with the RD
  - MUST include all its join resources, with their link attributes
  - New ‘rt’ value “core.osc.mbr”

```
Request: GM -> RD

Req: POST coap://rd.example.com/rd?ep=gm1
Content-Format: 40
Payload:
</group-oscore/feedca570000>;ct=41;rt="core.osc.mbr";
                             sec-gp="feedca570000";app-gp="group1";
                             cs_alg="-8";cs_alg_crv="6";
                             cs_key_kty="1";cs_key_crv=6";
                             cs_kenc="1",
<coap://as.example.com/token>;
      rel="authorization-server";
      anchor="coap://[2001:db8::ab]/group-oscore/feedca570000"

Response: RD -> GM

Res: 2.01 Created
Location-Path: /rd/4521
```

Discovery (1/2)

- The device performs a resource lookup at the RD
  - Known information: name of the Application Group, i.e. “group1”
  - Need to know: OSCORE Group Identifier; Join resource @ GM; Multicast IP address
  - ‘*app-gp*’ ✉ Name of the Application Group, acting as tie parameter in the RD

```
Request: Joining node -> RD

Req: GET coap://rd.example.com/rd-lookup/res
  ?rt=core.osc.mbr&app-gp=group1

Response: RD -> Joining node


Res: 2.05 Content
Payload:
<coap://[2001:db8::ab]/group-oscore/feedca570000>;rt="core.osc.mbr";
    sec-gp="feedca570000";app-gp="group1";
    cs_alg="-8";cs_alg_crv="6";cs_key_kty="1";
    cs_key_crv=6";cs_kenc="1";anchor="coap://[2001:db8::ab]"
```

Discovery (2/2)

- The device performs an endpoint lookup at the RD
  - Still need to know the Multicast IP address
  - ‘ep’ // Name of the Application Group, value from ‘*app-gp*’
  - ‘base’ // Multicast IP address used in the Application Group

```
Request: Joining node -> RD

Req: GET coap://rd.example.com/rd-lookup/ep
  ?et=core.rd-group&ep=group1

Response: RD -> Joining node

Res: 2.05 Content
Payload:
</rd/501>;ep="group1";et="core.rd-group";
          base="coap://[ff35:30:2001:db8::23]"
```

Observe Notifications as CoAP Multicast Responses

draft-tiloca-core-observe-multicast-notifications-03

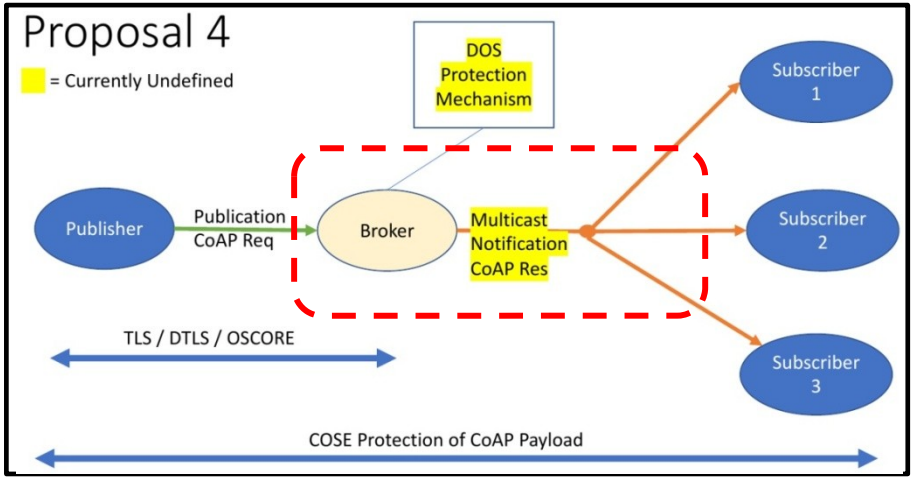
Marco Tiloca, RISE
Rikard Höglund, RISE
Christian Amsüss
Francesca Palombini, Ericsson

IETF 108, CoRE WG, July 31st, 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?

- Example 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

Proposed approach

- Define Observe notifications as multicast responses

- Token space from a group to a particular 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 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

- Consensus on token / external_aad by creating 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

- To the unicast request, the server sends a 5.03 ***error response*** with:
  - Serialization of the phantom request
  - IP multicast address where notifications are sent to
  - Serialization of the latest multicast notification (i.e. current resource status)

Updates overview

Informative error response

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

- Revised encoding of the error response
- Parameter meaning
  - *ph_req* : serialization of the phantom request
  - *last_notif* : serialization of the latest sent multicast notification
  - *cl_addr* , *cl_port*: source address/port of the phantom request
    → Destination address/port of the multicast notifications
  - *srv_addr* , *srv_port*: destination address/port of the phantom request
- *'last_notif'* gives clients:
  - The current representation of the target resource
  - A baseline for the Observe number of following multicast notifications
  - May become optional – opinions?
- When creating the observation, the server creates and stores a first *'last_notif'*

Updates overview

- Improved rough counting of active clients
  - Poll for interest, using a new CoAP option in successful multicast notifications

- Server current rough estimate: *N*
  - Expected confirmations *M* < *N*
  - Option value: *Q* = ceil (*N* / *M*)
  - Each client picks a random *I* : [0, *Q*)
  - If *I* == 0, the client sends a re-registration request
    - Non Confirmable; w/ No-Response; w/ the new Option having empty value
    - Given explicit indications to prevent Smurf attacks
  - The server receives *R* of such requests; *X* new clients have registered in the meanwhile
    - Added a server timeout, building on RFC 7252 and *core-groupcomm-bis* parameters
  - Then *N* := (*R* * Q) + *X*

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,

- The new Appendix A describes the algorithm in pseudo-code

Updates overview

- Alternative ways to retrieve a phantom request
  - Revised examples in Appendix B
  - Pub-Sub (phantom request as part of topic metadata)
  - Sender introspection of intercepted notifications
- Congestion control
  - Added text about broadcast storm
- Clarifications on Group OSCORE
  - The group mode is the one to use

```
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> {
        ph_req h"120100006464b431323334"
        last_notif h"120100006464b431324321"
        cli_addr h"ff35003020010db8..1234"
        cli_port 5683
        srv_addr h"20010db80100..0001"
        srv_port 5683
    }
```

```
Request:

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

Response:

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

    {
        'ph_req': h"120100006464b431323334"
        'last_notif' : h"120100006464b431324321"
        'cli_addr': h"ff35003020010db8..1234"
        'cli_port': 5683
        'srv_addr': h"20010db80100..0001"
        'srv_port': 5683
    }
```

Summary

- Multicast notifications to all clients observing a resource

- Latest additions
  - Improved encoding of error response
  - Improved rough counting of clients
  - Clarifications and editorial revision

- Next steps
  - Cover a scenario where a Proxy is used
  - Align concepts with draft-amsuess-core-cachable-oscore

- Need for document reviews

Thank you!

Comments/questions?

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

Backup

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

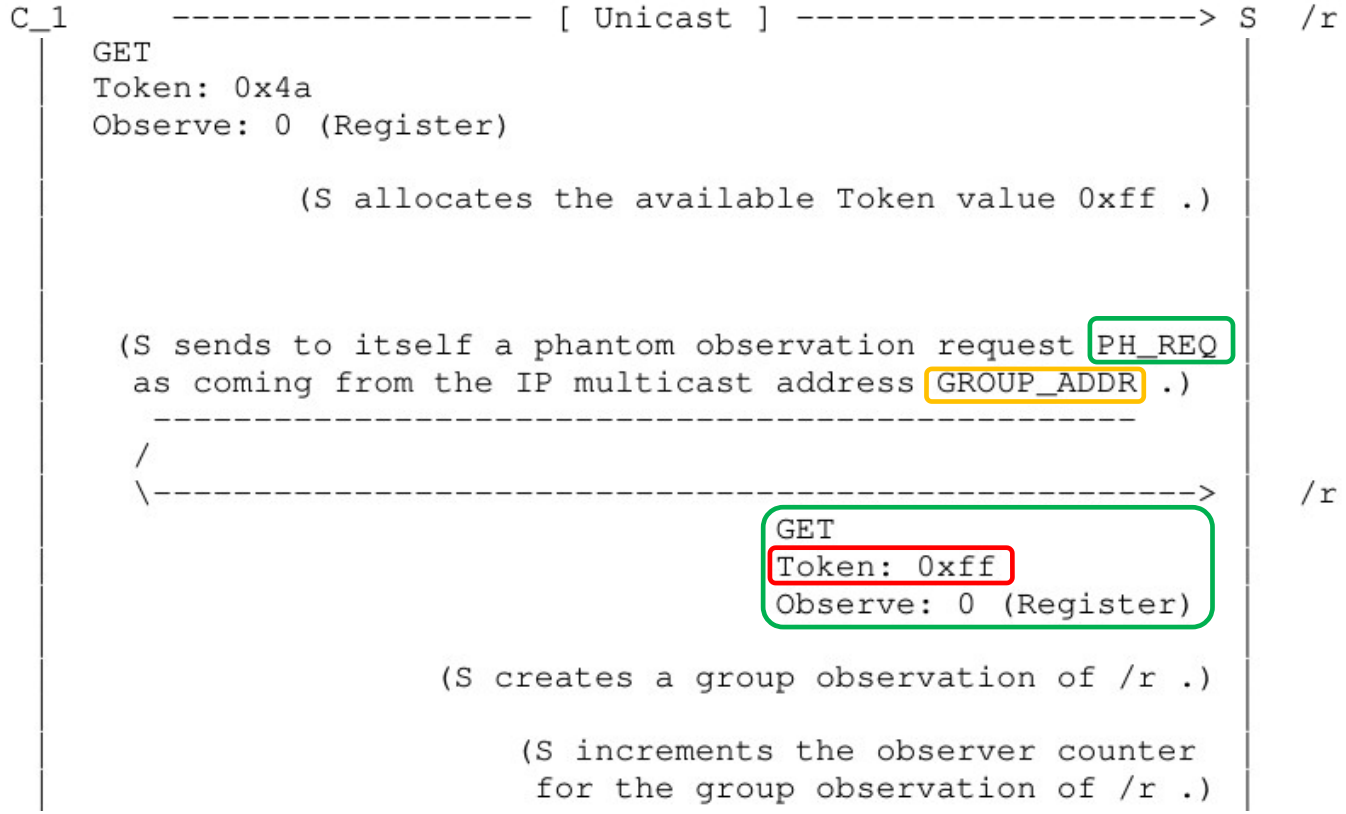
Interaction with clients

- The server sends to new/shifted clients an ***error response*** with
  - '*ph_req*': serialization of the phantom request
  - '*last_notif*': serialization of the latest sent notification for the target resource
  - '*cli_addr*' and '*cli_port*': source address/port of the phantom request
  - '*srv_addr*' and '*srv_port*': destination address/port of the phantom request

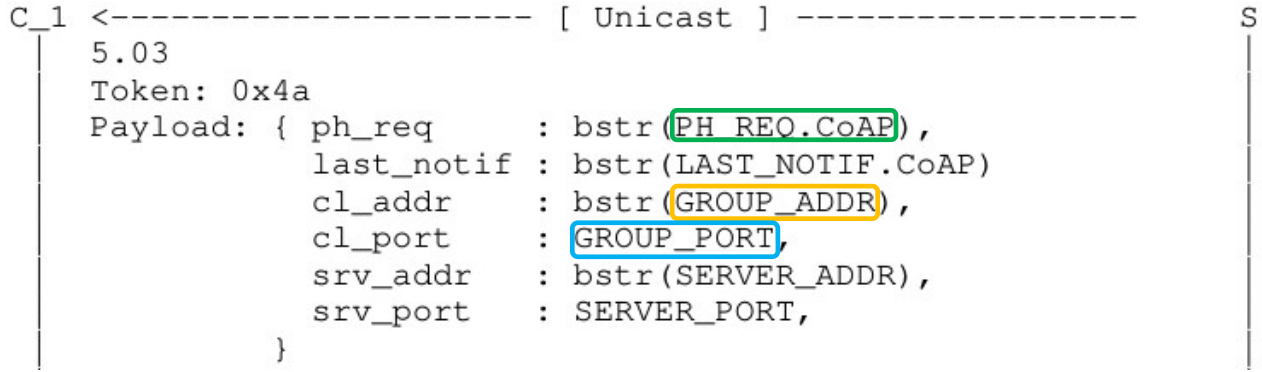
- When the value of the target resource changes:
  - The server sends an Observe notification to the IP multicast address '*cli_addr*'
  - The notification has the Token value T of the phantom request

- 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

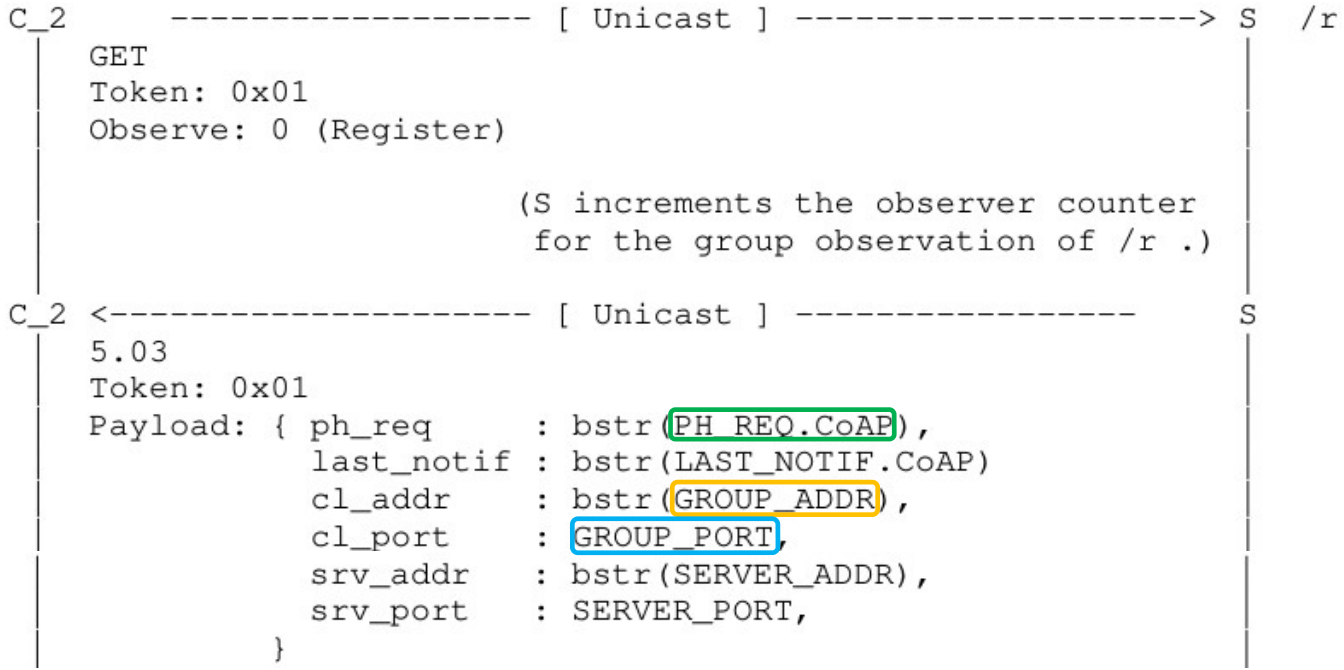
C1 registration



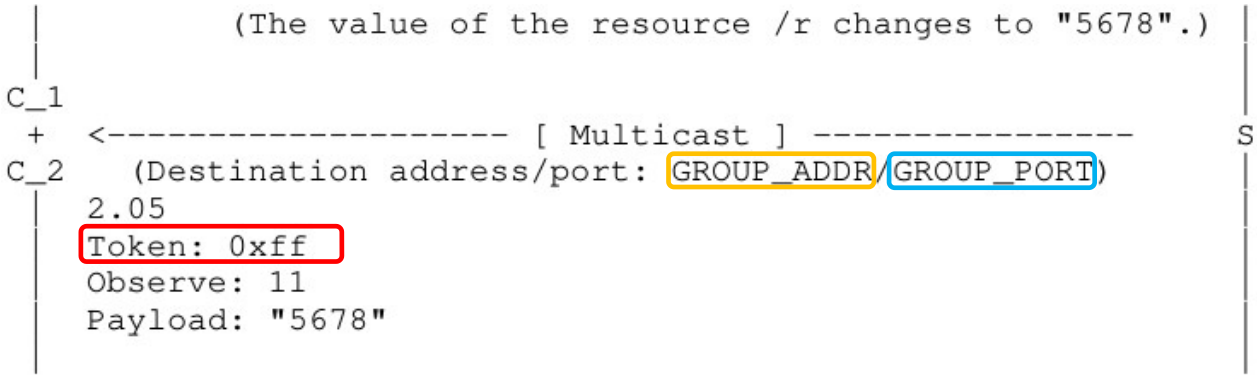
C1 registration



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

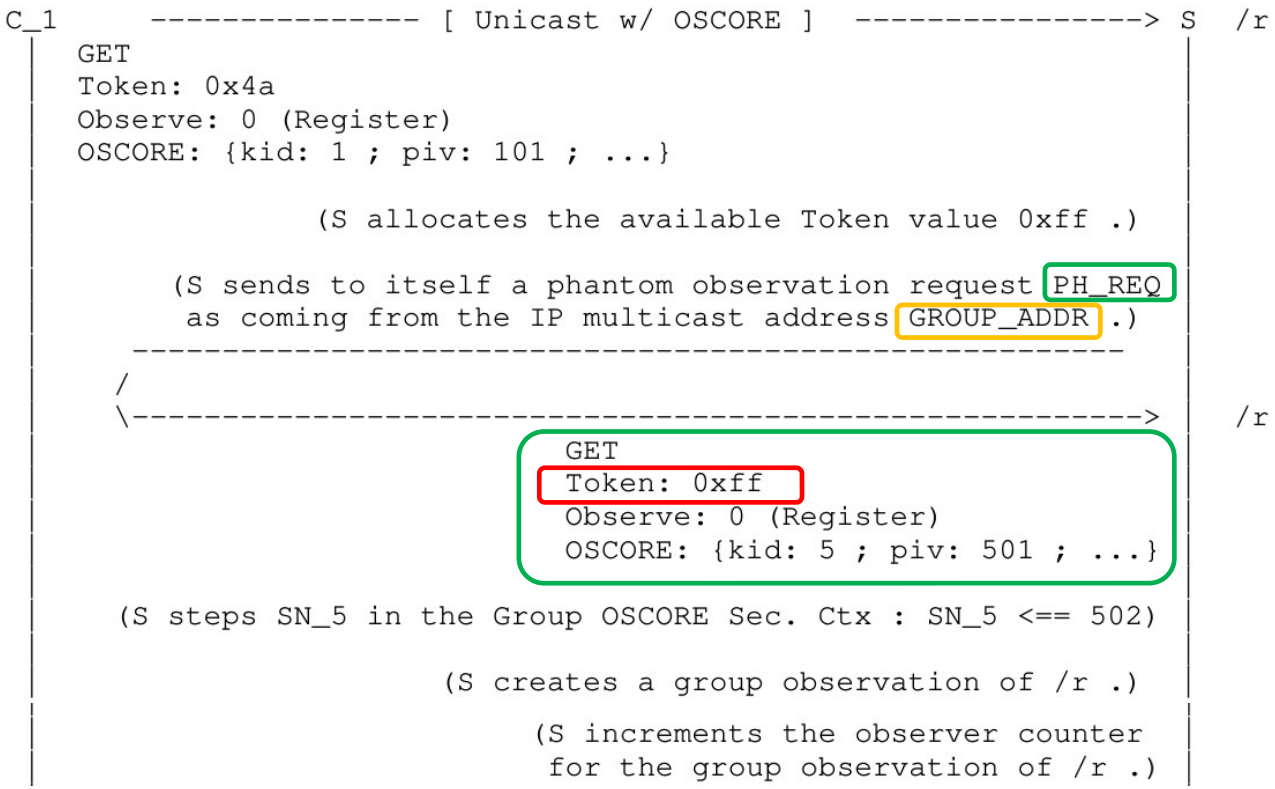
- In the error response, the server can ***optionally*** specify also:
  - '*join-uri*' : link to the Group Manager to join the OSCORE group
  - '*sec-gp*' : name of the OSCORE group
  - '*as-uri*' : link to the ACE Authorization Server associated to the Group Manager
  - '*cs-alg*' : countersignature algorithm
  - '*cs-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

MUST

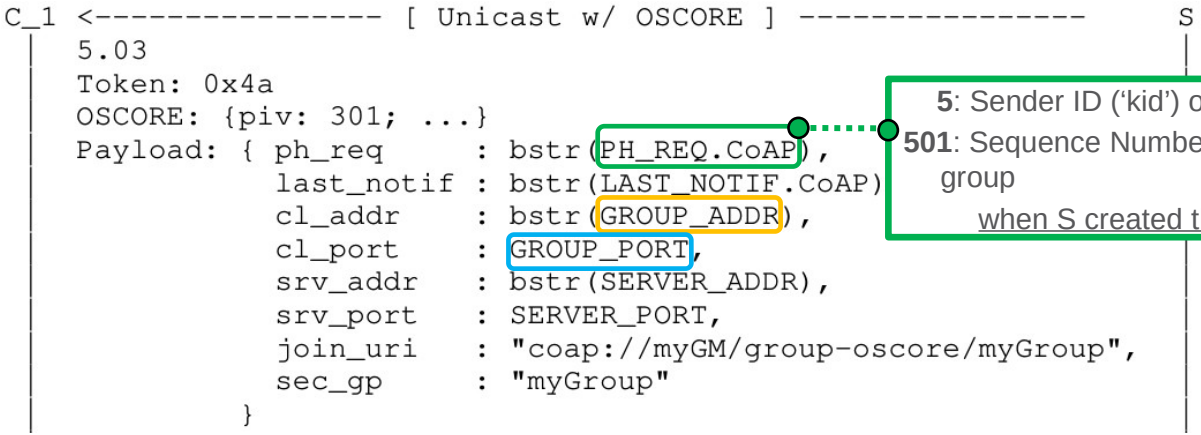
MAY

- 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



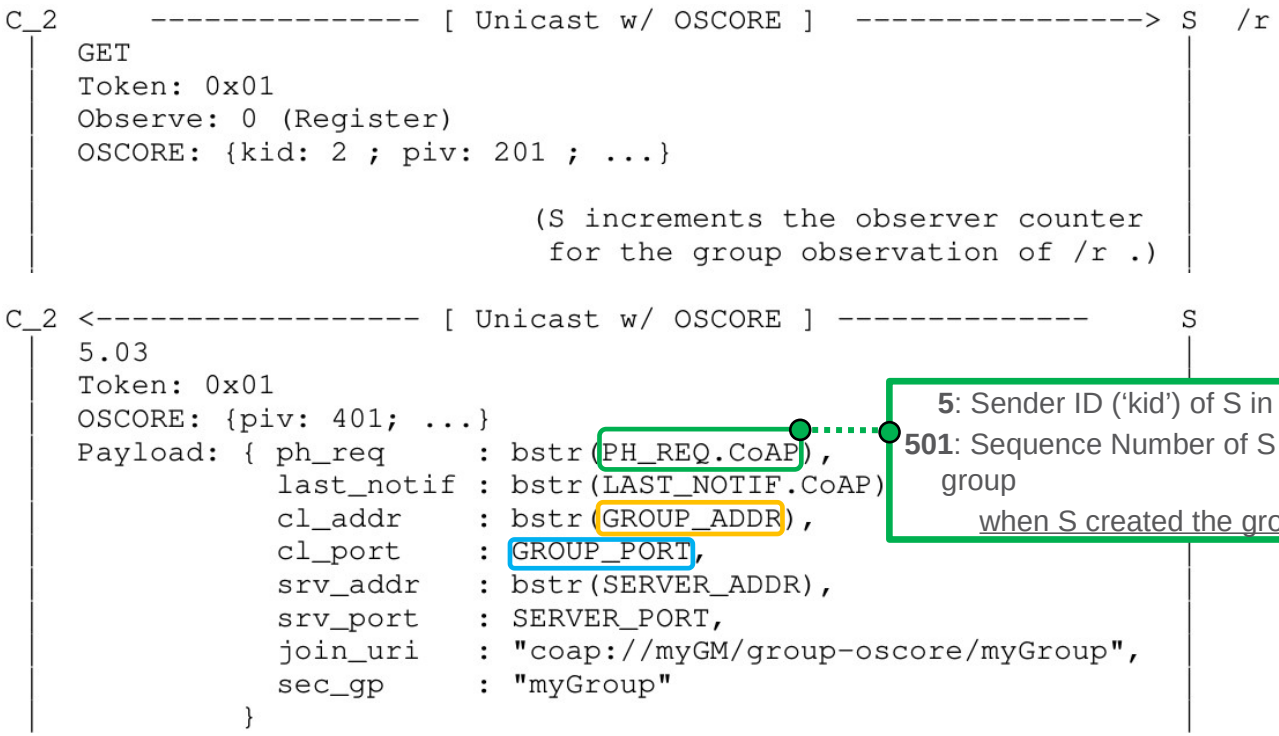
C1 registration w/ security



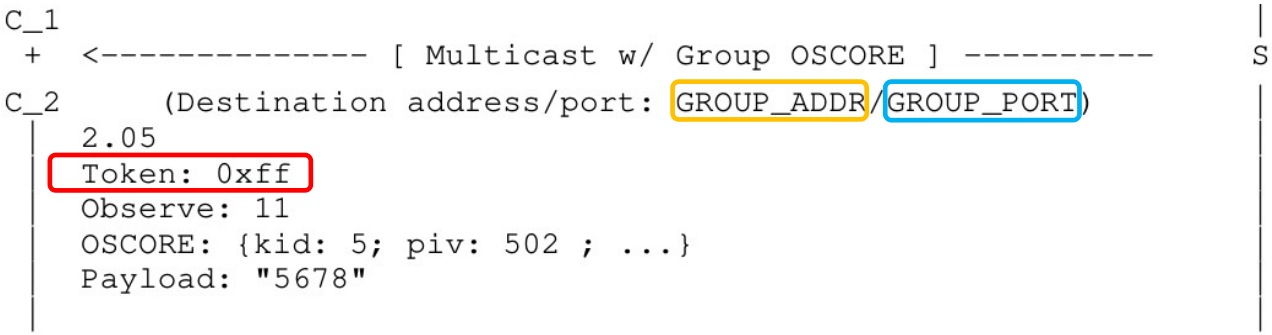
5: Sender ID ('kid') of S in the OSCORE group

501: Sequence Number of S in the OSCORE group when S created the group observation

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

Observe Notifications as CoAP Multicast Responses

draft-tiloca-core-observe-multicast-notifications-03

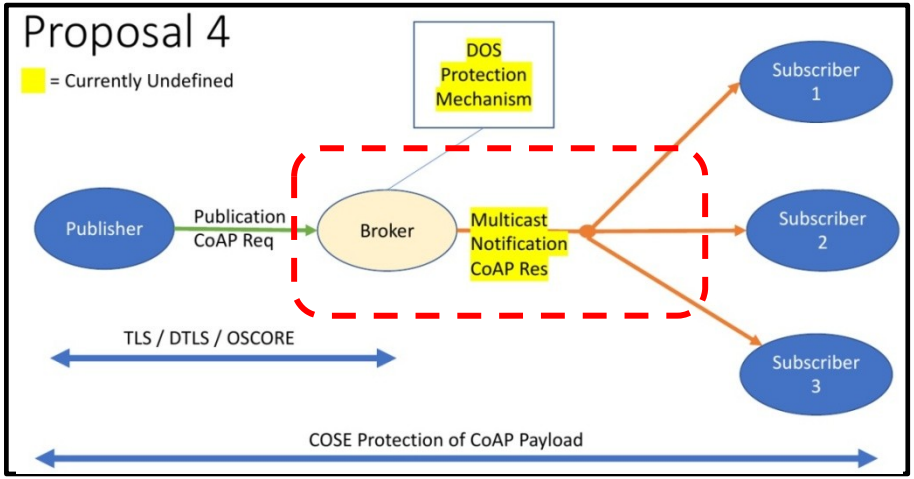
Marco Tiloca, RISE
Rikard Höglund, RISE
Christian Amsüss
Francesca Palombini, Ericsson

IETF 108, CoRE WG, July 31st, 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?

- Example 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

Proposed approach

- Define Observe notifications as multicast responses

- Token space from a group to a particular 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 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

- Consensus on token / external_aad by creating 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

- To the unicast request, the server sends a 5.03 ***error response*** with:
  - Serialization of the phantom request
  - IP multicast address where notifications are sent to
  - Serialization of the latest multicast notification (i.e. current resource status)

Updates overview

Informative error response

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

- Revised encoding of the error response
- Parameter meaning
  - *ph_req* : serialization of the phantom request
  - *last_notif* : serialization of the latest sent multicast notification
  - *cl_addr* , *cl_port*: source address/port of the phantom request
    → Destination address/port of the multicast notifications
  - *srv_addr* , *srv_port*: destination address/port of the phantom request
- *'last_notif'* gives clients:
  - The current representation of the target resource
  - A baseline for the Observe number of following multicast notifications
  - May become optional – opinions?
- When creating the observation, the server creates and stores a first *'last_notif'*

Updates overview

- Improved rough counting of active clients
  - Poll for interest, using a new CoAP option in successful multicast notifications

- Server current rough estimate: *N*
  - Expected confirmations *M* < *N*
  - Option value: *Q* = ceil (*N* / *M*)
  - Each client picks a random *I* : [0, *Q*)
  - If *I* == 0, the client sends a re-registration request
    - Non Confirmable; w/ No-Response; w/ the new Option having empty value
    - Given explicit indications to prevent Smurf attacks
  - The server receives *R* of such requests; *X* new clients have registered in the meanwhile
    - Added a server timeout, building on RFC 7252 and *core-groupcomm-bis* parameters
  - Then *N* := (*R* * Q) + *X*

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,

- The new Appendix A describes the algorithm in pseudo-code

Updates overview

- Alternative ways to retrieve a phantom request
  - Revised examples in Appendix B
  - Pub-Sub (phantom request as part of topic metadata)
  - Sender introspection of intercepted notifications
- Congestion control
  - Added text about broadcast storm
- Clarifications on Group OSCORE
  - The group mode is the one to use

```
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> {
        ph_req h"120100006464b431323334"
        last_notif h"120100006464b431324321"
        cli_addr h"ff35003020010db8..1234"
        cli_port 5683
        srv_addr h"20010db80100..0001"
        srv_port 5683
    }
```

```
Request:

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

Response:

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

    {
        'ph_req': h"120100006464b431323334"
        'last_notif' : h"120100006464b431324321"
        'cli_addr': h"ff35003020010db8..1234"
        'cli_port': 5683
        'srv_addr': h"20010db80100..0001"
        'srv_port': 5683
    }
```

Summary

- Multicast notifications to all clients observing a resource

- Latest additions
  - Improved encoding of error response
  - Improved rough counting of clients
  - Clarifications and editorial revision

- Next steps
  - Cover a scenario where a Proxy is used
  - Align concepts with draft-amsuess-core-cachable-oscore

- Need for document reviews

Thank you!

Comments/questions?

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

Backup

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

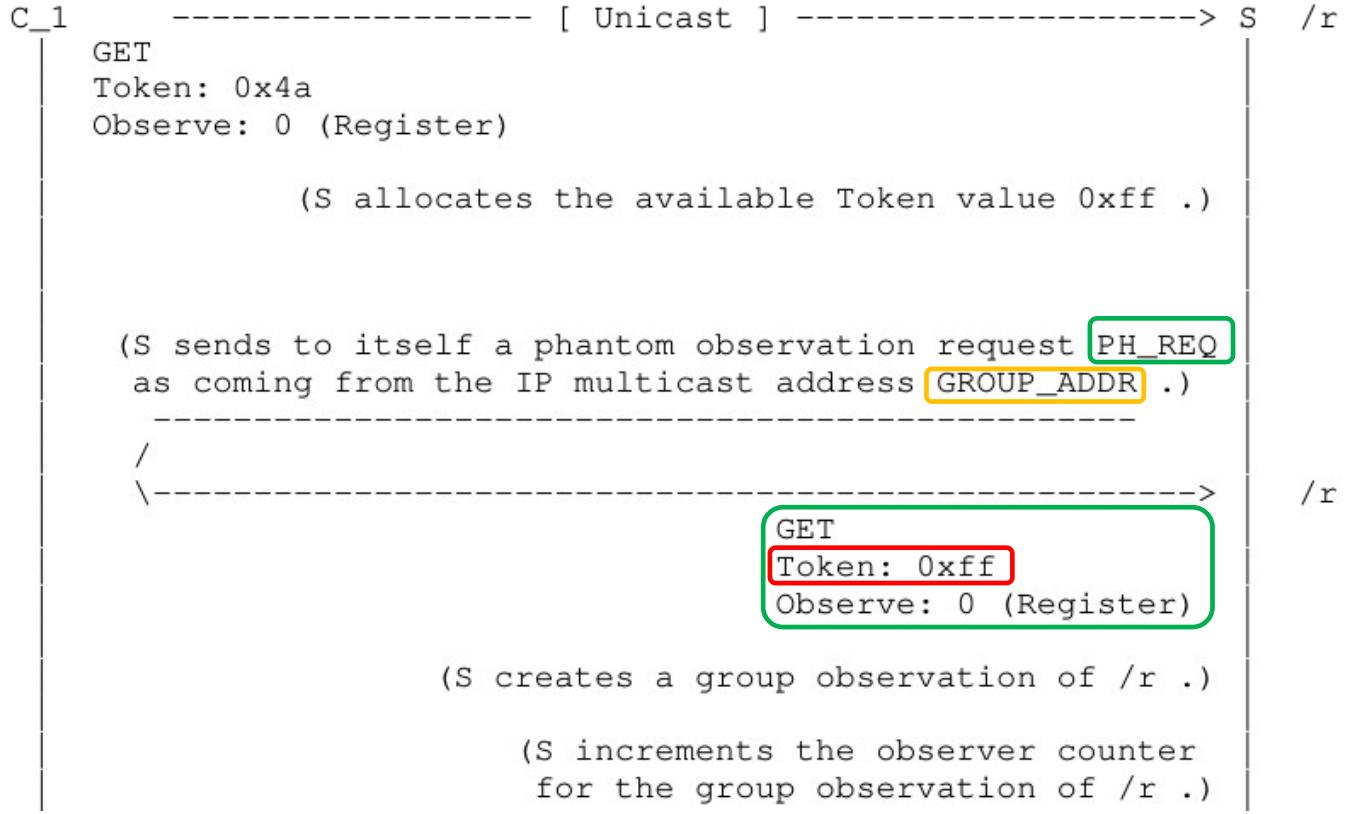
Interaction with clients

- The server sends to new/shifted clients an ***error response*** with
  - '*ph_req*': serialization of the phantom request
  - '*last_notif*': serialization of the latest sent notification for the target resource
  - '*cli_addr*' and '*cli_port*': source address/port of the phantom request
  - '*srv_addr*' and '*srv_port*': destination address/port of the phantom request

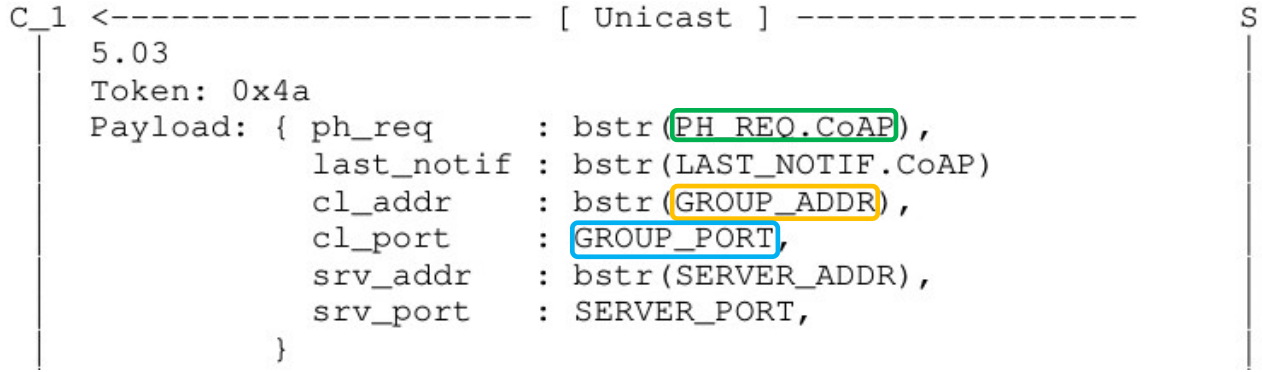
- When the value of the target resource changes:
  - The server sends an Observe notification to the IP multicast address '*cli_addr*'
  - The notification has the Token value T of the phantom request

- 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

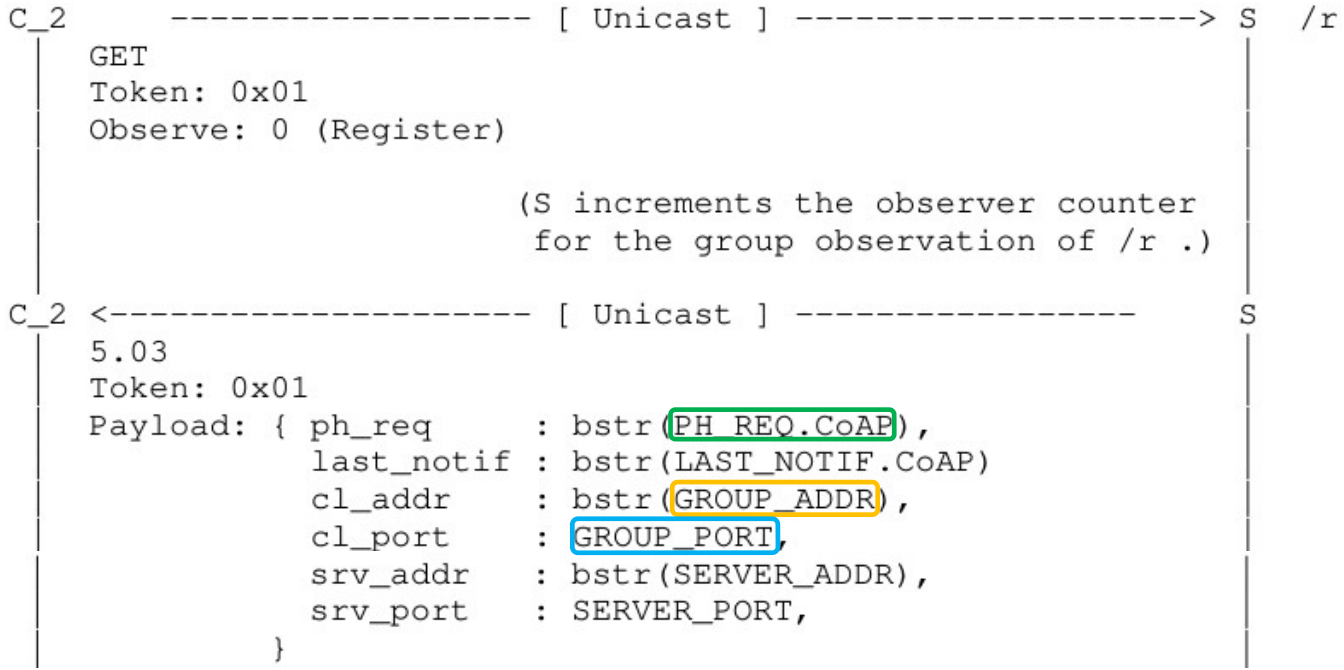
C1 registration



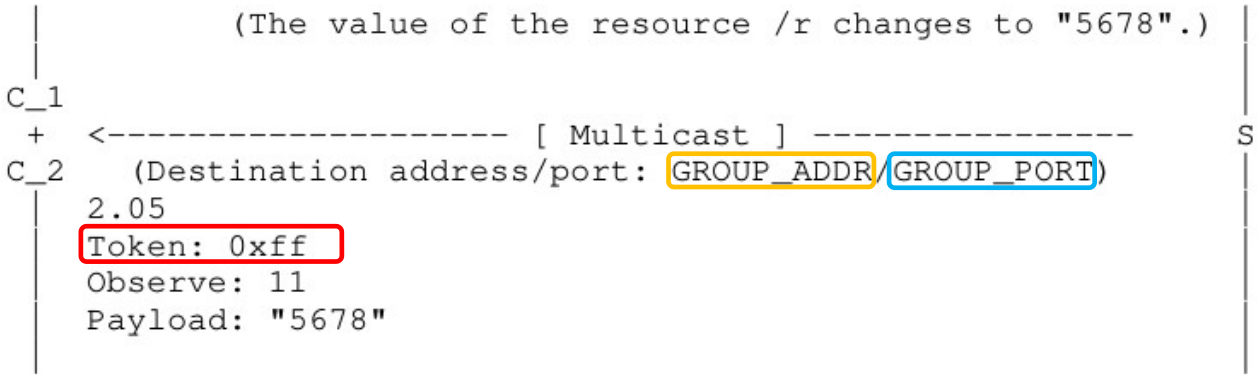
C1 registration



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

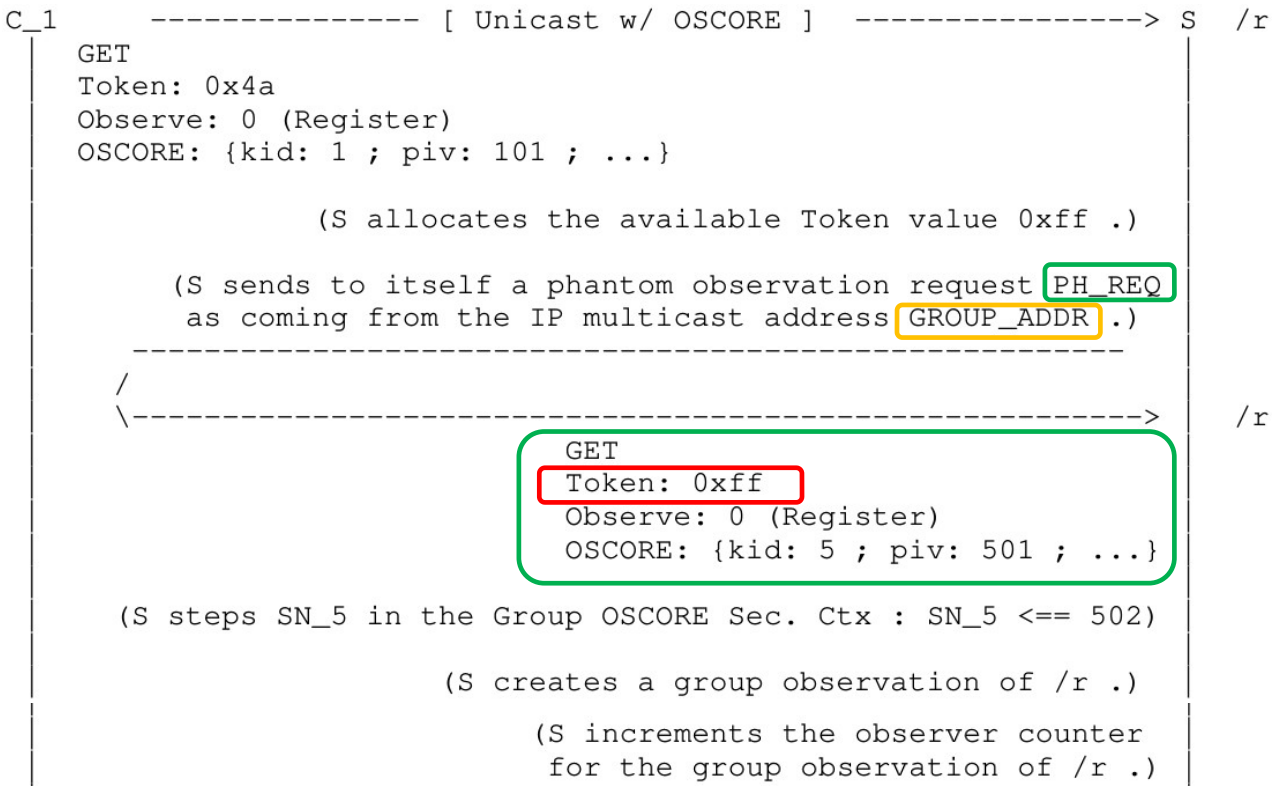
- In the error response, the server can ***optionally*** specify also:
  - '*join-uri*' : link to the Group Manager to join the OSCORE group
  - '*sec-gp*' : name of the OSCORE group
  - '*as-uri*' : link to the ACE Authorization Server associated to the Group Manager
  - '*cs-alg*' : countersignature algorithm
  - '*cs-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

MUST

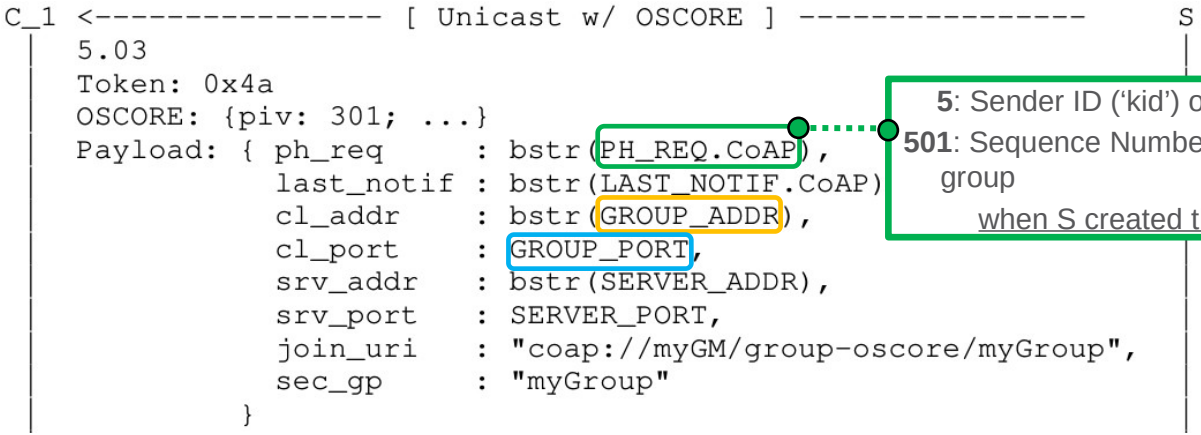
MAY

- 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



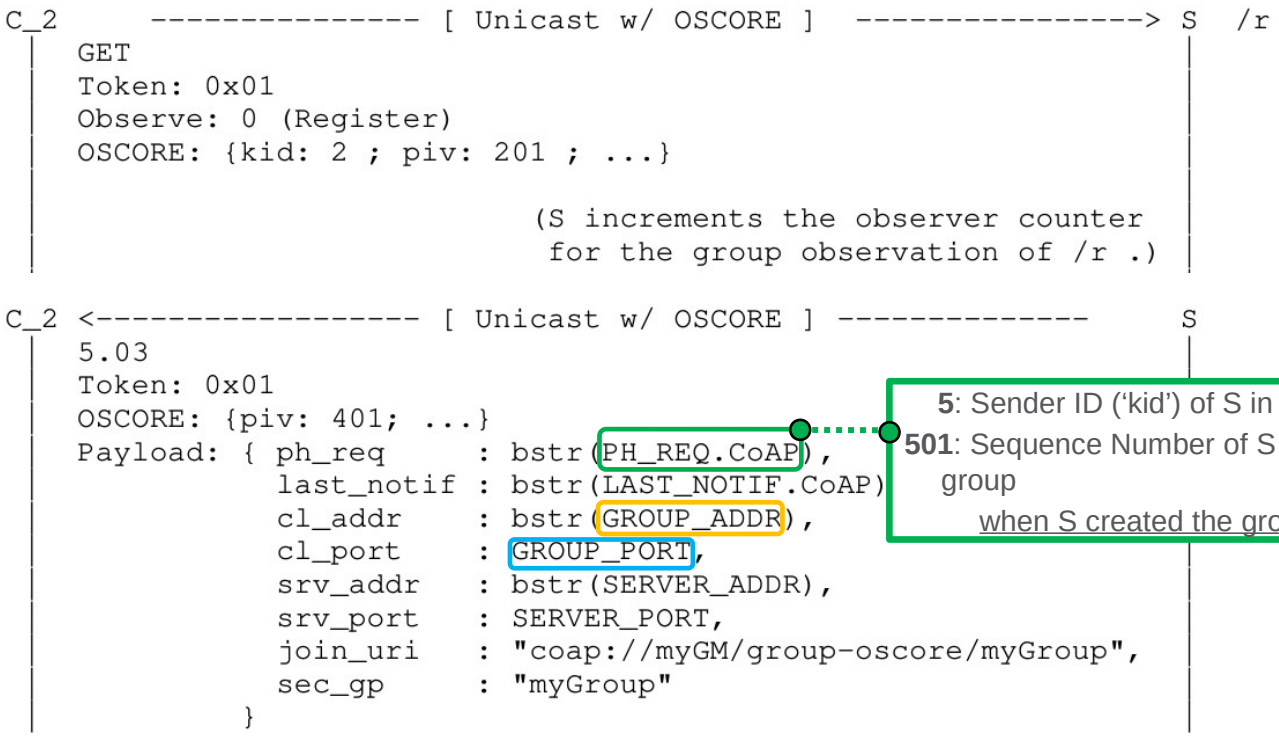
C1 registration w/ security



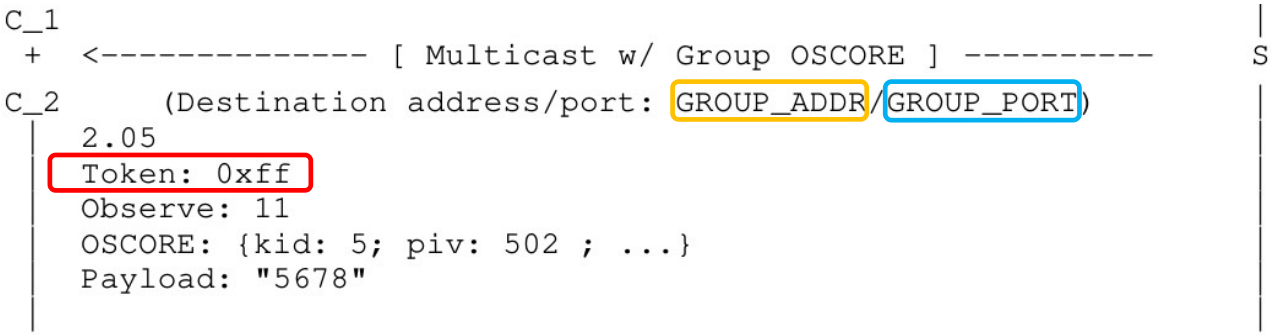
5: Sender ID ('kid') of S in the OSCORE group

501: Sequence Number of S in the OSCORE group when S created the group observation

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

Proxy Operations for CoAP Group Communication

draft-tiloca-core-groupcomm-proxy-01

Marco Tiloca, RISE
Esko Dijk, IoTconsultancy.nl

IETF 108, CoRE WG, July 31st, 2020

Recap

- CoAP supports group communication over IP multicast
  - *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

Contribution

- Description of proxy operations for CoAP group communication
  - Addressed all issues in *draft-ietf-core-groupcomm-bis*
  - Signaling protocol 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

- Version -01 addresses Christian's review [1] – Thanks!
  - Revised properties and usage of the two CoAP options
  - "Nested OSCORE" (Appendix A), if OSCORE is used between Client and Proxy

[1] https://mailarchive.ietf.org/arch/msg/core/AwYqnQu703V5RGR43JQxRslkYsw/

Rationale

- In the 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

- In a response to a group request, the proxy includes the server address
  - The client can distinguish the responses and the different servers
  - The client can contact an individual server (directly, or via the proxy)

- Group OSCORE for e2e security between client and servers

Multicast-Signaling option

No.	C	U	N	R	Name	Format	Length	Default
TBD1		x	-		Multicast-Signaling	uint	1-5 B	(none)

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

- Used only in requests
  - Presence: explicit claim of support and interest from the client
  - Value: indication to the proxy on how long to handle unicast responses
- The proxy removes the option, before forwarding the request

Response-Forwarding option

No.	C	U	N	R	Name	Format	Length	Default
TBD2			-		Response-Forwarding	string	8-20 B	(none)

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

- Used only in responses
  - Presence: allows the client to distinguish responses and originator servers
  - Value: absolute URI of the server (address and port from the response)
- The proxy adds the option, before forwarding the response to the client

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

“Nested OSCORE”

- 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

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
- Next steps
  - Cover the case with a chain of proxies
  - Define HTTP headers for Cross-Proxies
- Need for reviews

Thank you!

Comments/questions?

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

Cachable OSCORE

draft-amsuess-core-cachable-oscore

Christian Amsüss, Marco Tiloca

2020-07-31

Background

`multicast-notifications`

Comparison with ICNs

OSCON

Caching and OSCORE

POST / 2.01
KID and PIV in request } uncachable

. . . and it's only one client anyway

For every complex problem, there is a solution...

that is simple, neat and ~~wrong~~ insufficient

Group OSCORE
FETCH / 2.05
magically hit cache

} verification fails

Consensus request

- Pick request sender KID and PIV
- Trust in the request[1]

The ideal candidate to generate a Consensus Request is the server: "Ticket Requests"

[1] It'd be a pity if someone requested `/whom-i-know`, and gave you the response claiming they requested `/whom-to-trust`

Ticket Request example

```
Client                    Proxy                Server

 enc(GET /a, C:1) ------------------------->
 <--------- enc(Try enc(GET /a, S:1), S/C:1)

 enc(GET /a, S:1) ------>
                      (cache hit)
 <--- enc(2.05 data, S:2)
```

Assuming pre-existing multicast setup

`multicast-notifications`'s Phantom Requests are Ticket Requests

1. Great for observations
2. Great for large representations[2]
3. Not so great for everything else

[2] Unless outer-block mode is used. Which you want. In which case see 3.

Magically hitting the cache key

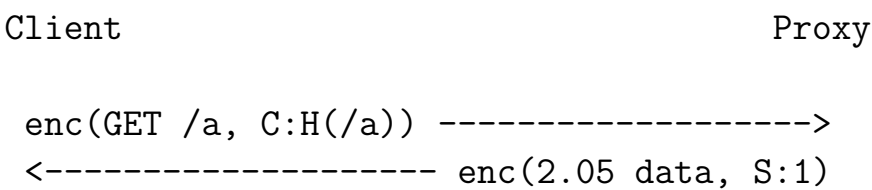
```
Client                                                Proxy

 enc(GET /a, C:1), H(/a) ------------------>
 <- enc(2.05 data, S:2) Resp-For enc(GET /a, S:1)
```

... provided `H(/a)` is derived the same for every request

(actually it's rather hashing the complete plaintext|AAD)

Now that we all agree...



[3] Also very nice for B.2 mode

Now that we all agree...

```
Client                                              Proxy

 enc(GET /a, C:H(/a)) ------------------->
 <--------------------- enc(2.05 data, S:1)
```

- Hash over all input to encryption (incl. AAD)
- PartIV too short for sufficient hash – ID-Detail[3]
- In group it's encrypt-and-sign – deterministic client with private key known to group members

[3] Also very nice for B.2 mode

Questions

- Practicality
- Cryptography
- Interest in CoRE

Thank you!
Comments/questions?