Constrained RESTful Environments WG (core)

Chairs:

Andrew McGregor <andrewmcgr@gmail.com>

Carsten Bormann <cabo@tzi.org>

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 3979 and its updates

✓Blue sheets
✓Scribe(s):
http://tools.ietf.org/wg/core/minutes

Note Well

This summary is only meant to point you in the right direction, and doesn't have all the nuances. The IETF's IPR Policy is set forth in BCP 79; please read it carefully.

The brief summary:

❖**By participating with the IETF, you agree to follow IETF processes.**

❖**If you are aware that a contribution of yours (something you write, say, or discuss in any IETF context) is covered by patents or patent applications, you need to disclose that fact.**

❖**You understand that meetings might be recorded, broadcast, and publicly archived.**

For further information, talk to a chair, ask an Area Director, or review the following:

BCP 9 (on the Internet Standards Process)

BCP 25 (on the Working Group processes)

BCP 78 (on the IETF Trust)

BCP 79 (on Intellectual Property Rights in the IETF)

Milestones (from WG charter page)

http://datatracker.ietf.org/wg/core/charter/

Document submissions to IESG:

- Done CoAP protocol specification with mapping to HTTP Rest API to IESG
- Feb 2013 Blockwise transfers in CoAP to IESG
- Feb 2013 Observing Resources in CoAP to IESG
- Apr 2013 Group Communication for CoAP to IESG
- Dec 2099 HOLD (date TBD) Constrained security bootstrapping specification to IESG

draft-ietf-core-coap approved by IESG ✓

2013-07-11

eight YES ballots

draft-ietf-core-coap-18

- Approved by the IESG 2013-07-11
- Now in RFC editor queue
  - will stay there for a while:
    - MISSREF*A*R(1G)

```
REF  draft-ietf-tls-oob-pubkey   NOT-RECEIVED
     draft-mcgrew-tls-aes-ccm-ecc     NOT-RECEIVED
```

- Recent Changes
  - Accept Option is now critical
  - Size1 is imported from -block for 4.13 errors
    - well, first Size was split into Size1 and Size2
  - Lots of small clarifications and small editorial fixes
  - Clarify that we really focus on ECC with P-256 curve (MTI)

We are now in a phase change

- From
  - Oh I have this cool idea, how about that
- To
  - I have a deployment with a problem to solve
  - Here is how we solved it

Today

All times are in time-warped CEST

- 13:00–13:10 Intro
- 13:10–13:20 DICE preview
- 13:20–14:10 Access Control/Authorization in CoAP
- 14:10–14:25 Groupcomm (AR)
- 14:25–14:40 HTTP mapping (SL)
- 14:40–14:50 Service Discovery (ZS)
- 14:50–14:60 Core Interfaces (ZS)
- 14:60 Links-JSON (CB)
- 14:60 “If we have time”
  - 14:60 Core-Entities (FV)
  - 14:60 (Content-Format) Parameters (YD)
  - 14:60 Group Authentication (QM)

} WG docs

Between the slots

- CoAP/LWIG work (➔ Matthias Kovatsch)
- Observe in Tue 9-11 slot (➔ Klaus Hartke)

- Meet up after this meeting in the front

- DICE BOF, Wed 1510–1610
  - DTLS In Constrained Environments

Thursday

- 13:00–13:05 Intro
- 13:05–13:11 External updates (OMA, cc work)
- 13:11–14:01 open issues in -block and -observe
- –14:01 conditional observe (if we have time)
- 14:01–14:41 Alternative Transports
- 14:41–15:00 “If we have time”, continued
  - 14:41–14:51 Sleepy nodes update
  - 14:51–14:60 Spillover from Monday

Group 1: Security DICE BOF preview Authorization

DICE BOF preview

The Problem

- CoAP is moving towards mass deployment
  - DTLS v1.2 is the chosen security mechanism
  - Suitable range of security modes & ciphers
  - This was exactly the right choice!
- However, DTLS v1.2 has several drawbacks
  - Handshake overhead is unnecessarily high
  - DTLS handshake state-machine is complex (TCP + TLS)
  - Not clear what sub-protocols, options and modes are needed
  - No support for IP multicast, which CoAP is often used with
- What if we just do nothing?
  - Alternative, likely broken, security mechanisms will be invented
  - Or worse, deployments without security, e.g. for multicast

The Scope

- The DICE working group would initially:
  - Define a minimal DTLS profile
  - Provide requirements for the design of TLS v1.3
  - Define use of group keys with the DTLS record layer

- Explicitly out of scope:
  - The WG would not change TLS standards
    - Any TLS related changes will go to the TLS WG
  - Group key management
  - Specification of new cipher suites

Related Work

- Profiling Work Item Strawman

  http://tools.ietf.org/html/draft-keoh-dtls-profile-iot-00

- Group Communication Security Work Item Strawman

  http://www.ietf.org/id/draft-keoh-dtls-multicast-security-00.txt

- Existing work

  http://www.ietf.org/id/draft-keoh-lwig-dtls-iot-01.txt

  http://www.ietf.org/id/draft-hartke-core-codtls-02.txt

  http://www.ietf.org/id/draft-tschofenig-lwig-tls-minimal-03.txt

  http://www.ietf.org/id/draft-keoh-tls-multicast-security-00.txt

  http://www.ietf.org/id/draft-ietf-tls-oob-pubkey-07.txt

  http://www.ietf.org/id/draft-jennings-core-transitive-trust-enrollment-01.txt

  http://tools.ietf.org/html/draft-schmitt-two-way-authentication-for-iot-00

  http://tools.ietf.org/html/draft-greevenbosch-dice-authent-author-revoc-00

  http://tools.ietf.org/html/draft-greevenbosch-tls-ocsp-lite-00

Authorization

CoAP doesn't do AAA

- We delegate authentication to DTLS
  - Authenticate peer endpoint in DTLS handshake
  - Several crypto options are available
    - Don't just think HTTPS-style PKI here
- Authorization is done "on top of CoAP"
- Let's not talk about the third A

Authorization

- Authentication tells us who the other endpoint is (subject)
- Authorization tells us what the other endpoint is allowed to do
  - down to the level of resources (objects) and methods (permissions) on them
- Also named **Access Control**

Why discuss this in CoRE?

- CoAP protocol doesn’t do authorization
  - DTLS supplies the identifiers for that
- Authorization may need some exchanges before it can be established
- Interoperability?

“No new protocols”

- Use DTLS for authentication and most of “the crypto stuff”
- Use CoAP for information transfer
- Use [JSON, _____] for the data structures
- Define a couple of data structures and how to use them (OK, that’s a protocol)

Access Control Framework for Constrained Environments

draft-selander-core-access-control-00

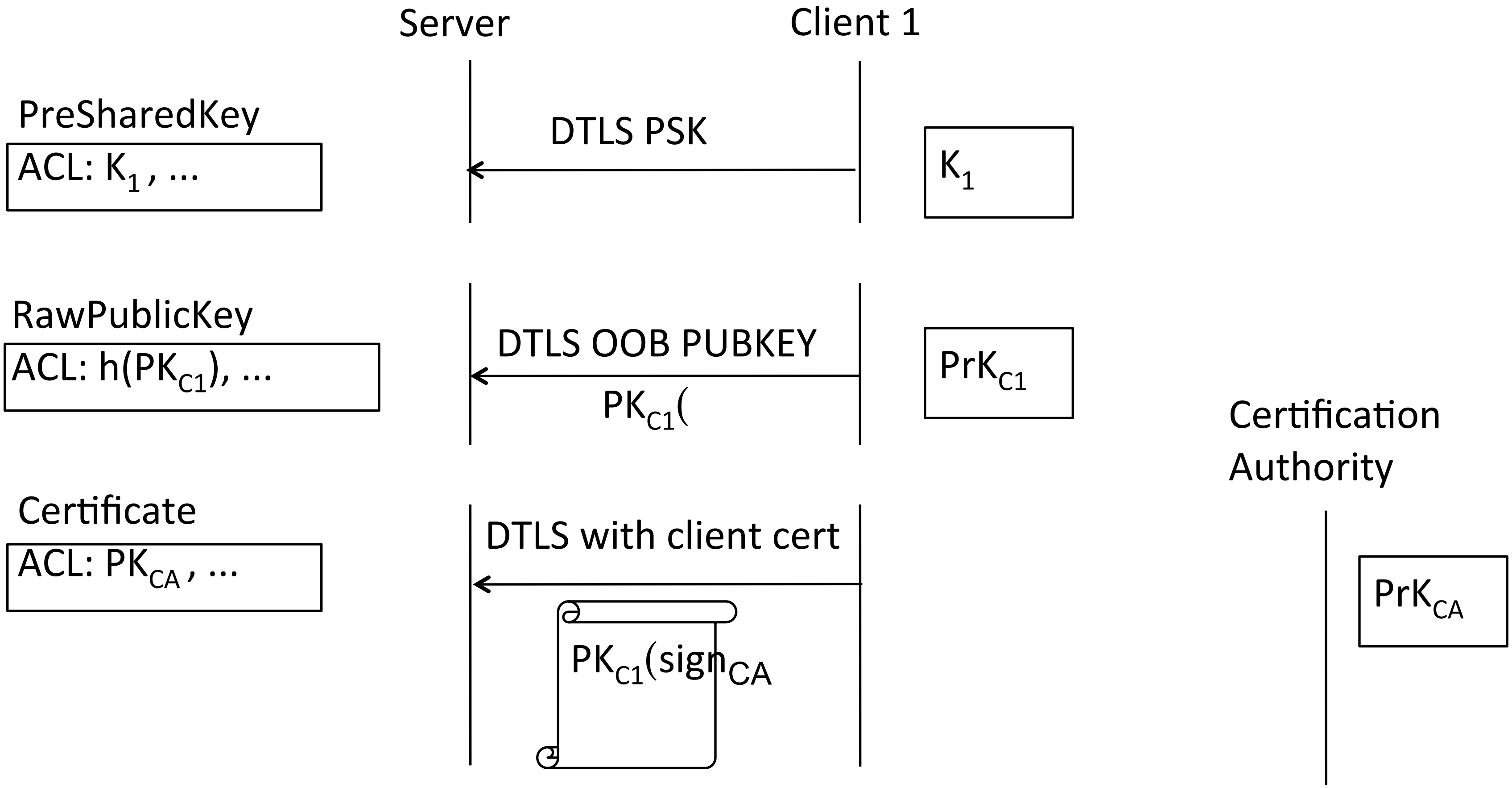
Göran Selander, Ericsson Research, Stockholm
Mohit Sethi, Ericsson Research, Helsinki
Ludwig Seitz, SICS Swedish ICT, Lund

Content of draft

A token-based Access Control Framework for CoRE

- Requirements for AC in constrained environments
- AC Framework
  - Rationale, roles & message flow
  - Assertion transfer options
  - Key provisioning schemes (alternative “Security Modes”)
  - Extended Access Control Lists
- Applications (profiles) of the ACF
  - Assertion profiles (XACML-SAML profile)
  - Message protection profiles (communication / object security)

Background: CoAP Security Modes



Server

Client 1

PreSharedKey

ACL: K_1 , ...

DTLS PSK

K_1

RawPublicKey

ACL: h(PK_{C1}), ...

DTLS OOB PUBKEY

PK_{C1}(

PrK_{C1}

Certification Authority

Certificate

ACL: PK_{CA} , ...

DTLS with client cert

PK_{C1}(sign$_{CA}$

PrK_{CA}

Why/what/how standardize ACF

Why (not just all-or-nothing AC)?

- Some applications require more granular or flexible access control
- Security standardization is a means to support good security practice.

What (should be standardized)?

- assertion formats for different use cases (separate profile of ACF)
- transfer of assertion from client to resource server (part of ACF)
- client-server message protection, including secure transfer of assertions (separate profile of ACF)
- security modes, i.e. key provisioning schemes (part of ACF)

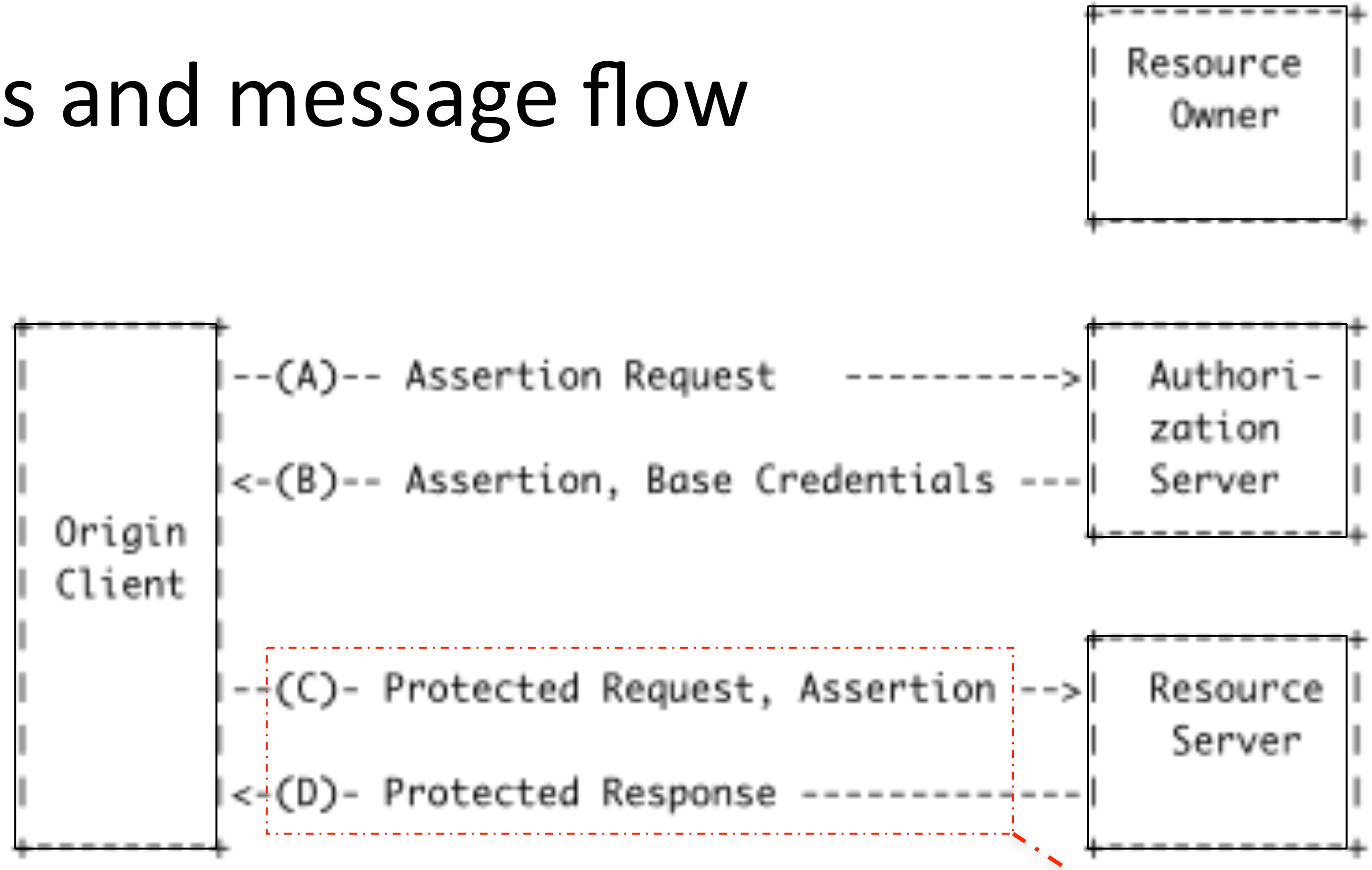
How (to encompass requirements of different use cases)?

- By means of assertion profiles and message protection profiles.

Requirements for AC in CoRE

- General AC/security requirements
- Requirements for constrained environments, e.g.
  - No additional messages
  - Keep message sizes small
- Granularity
  - GET/PUT/POST/DELETE
  - Allow access control policies to depend on *local conditions*
- Flexibility
  - Easy to set and change authorized clients and access rights
- Compatibility with existing standards
  - Avoid duplicating existing work, e.g. XACML, SAML

Roles and message flow



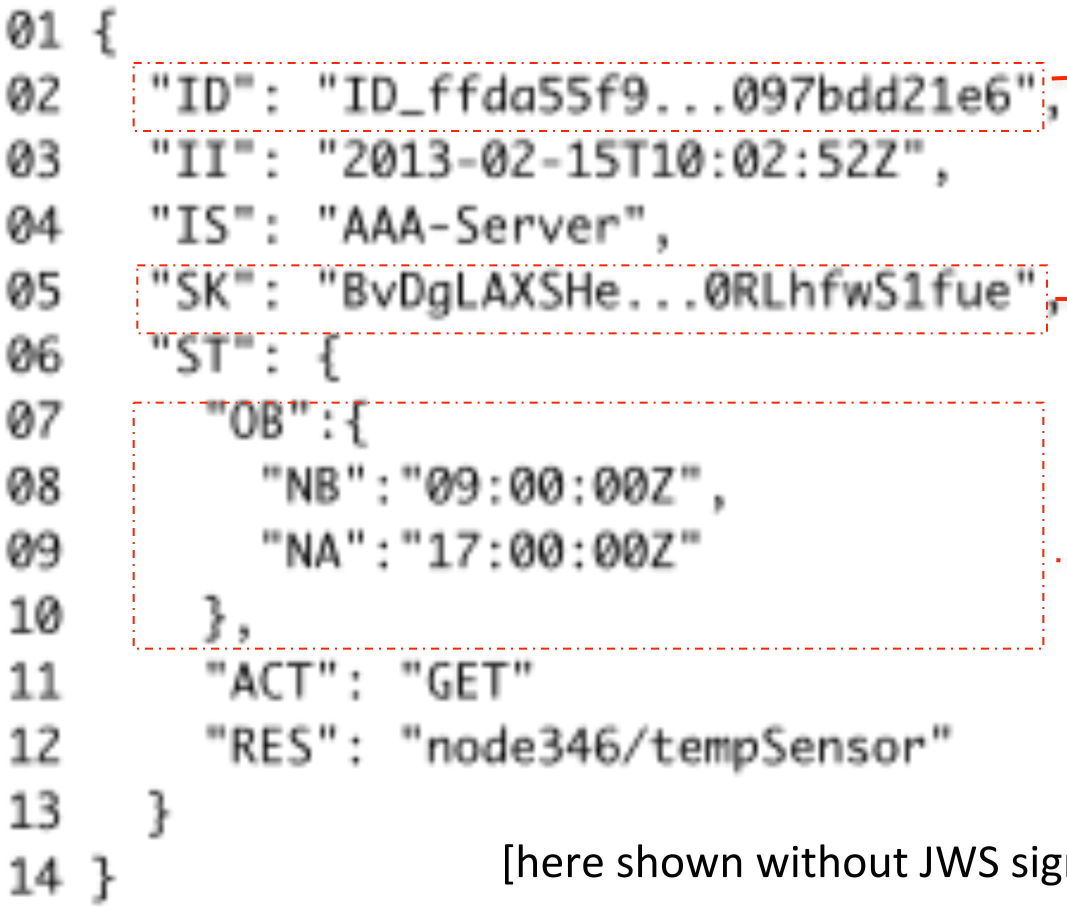
```
                                                   +----------+
                                                   | Resource |
                                                   |  Owner   |
                                                   |          |
                                                   +----------+

+--------+                                         +----------+
|        |--(A)-- Assertion Request   ----------->|  Authori- |
|        |                                         |  zation  |
|        |<-(B)-- Assertion, Base Credentials ---|  Server  |
| Origin |                                         +----------+
| Client |
|        |                                         +----------+
|        |--(C)- Protected Request, Assertion -->|  Resource |
|        |                                         |  Server  |
|        |<-(D)- Protected Response ------------|          |
+--------+                                         +----------+
```

CoAP

Different ways to transfer the assertion in (C)-(D)

- In CoAP (query part of URI, new CoAP option)
- In DTLS handshake (e.g TLS Authorization Extensions [RFC5878])

Assertion profile: Compact SAML-XACML



```
{
  "ID": "ID_ffda55f9...097bdd21e6",
  "II": "2013-02-15T10:02:52Z",
  "IS": "AAA-Server",
  "SK": "BvDgLAXSHe...0RLhfwS1fue",
  "ST": {
    "OB":{
      "NB":"09:00:00Z",
      "NA":"17:00:00Z"
    },
    "ACT": "GET"
    "RES": "node346/tempSensor"
  }
}
```

Possible Nonce N

E.g. public key of Origin Client

XACML obligation carries local conditions

[here shown without JWS signature]

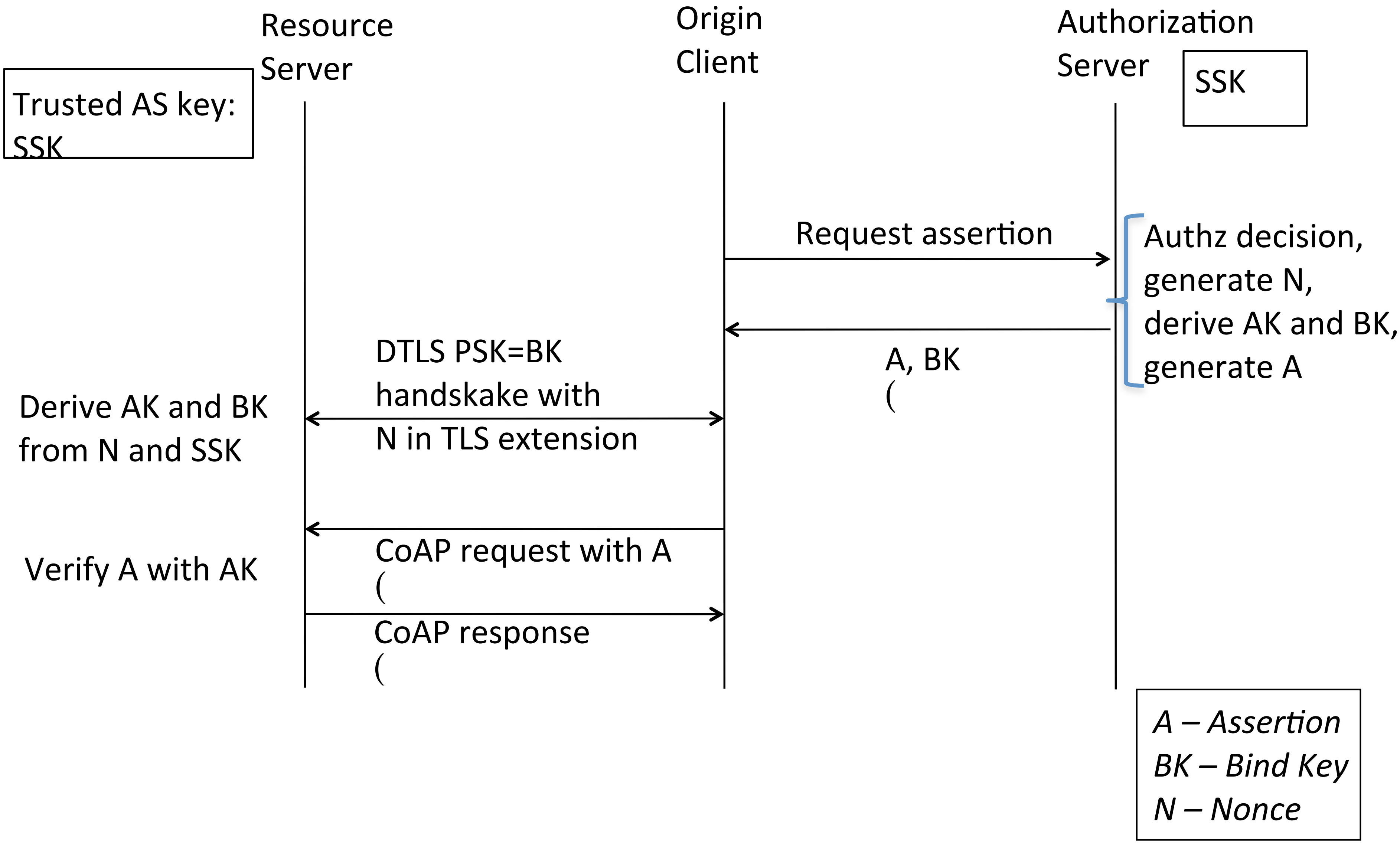
Implementation

- Object secured payload, assertion in CoAP option, PSK
- Arduino Mega 2560
  - 16 MHz, 256 kB Flash, 8 kB SRAM, 4 kB EEPROM.
  - Custom CoAP, AES, HMAC-SHA256
- Processing the CoAP messages on the device, including authorization handling, required 7.3 kB of static memory
- Details in http://soda.swedish-ict.se/5523/
- Conclusions from this setting:
  - Object security based authentication and authorization is feasible
  - Communication security (DTLS) less feasible (RAM, latency, ...)

Client-Server Message Protection

- Different alternatives to secure CoAP
  - Communication Security (e.g. DTLS as specified in CoAP)
  - Object Security (currently unspecified)
- CoAP security mechanism impacts the protection of assertion during transfer, and access control is dependent on authentication
  - Nevertheless, assertion format should ideally be independent on CoAP security mechanism
- ACF offers new key provisioning schemes (“Security Modes”)
  - ACF PK: RS and AS has exchanged public keys.
  - ACF SSK: RS and AS share a secret key.
  - In either case, AS may provide base credentials to the Origin Client

Example of DTLS with ACF SSK



Resource Server

Origin Client

Authorization Server

Trusted AS key: SSK

SSK

Request assertion

Authz decision, generate N, derive AK and BK, generate A

A, BK
(

DTLS PSK=BK handskake with N in TLS extension

Derive AK and BK from N and SSK

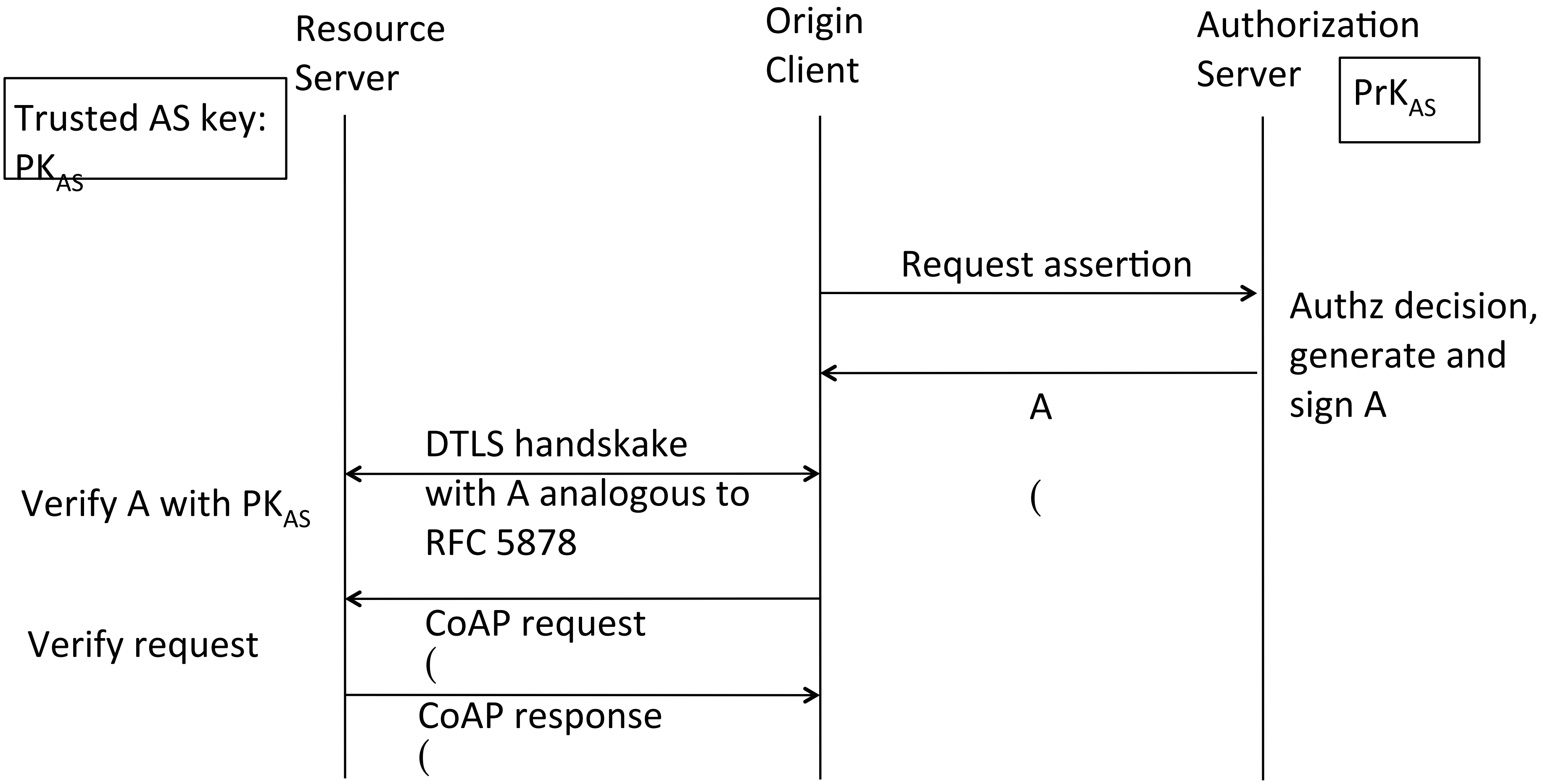
CoAP request with A
(

Verify A with AK

CoAP response
(

A – Assertion
BK – Bind Key
N – Nonce

Example of DTLS with ACF PK



Resource Server

Origin Client

Authorization Server

Trusted AS key: PK_{AS}

PrK_{AS}

Request assertion

Authz decision, generate and sign A

A

(

DTLS handskake with A analogous to RFC 5878

Verify A with PK_{AS}

CoAP request

(

Verify request

CoAP response

(

Transferring assertion in DTLS or in CoAP

- In DTLS (e.g. RFC 5878)
  - + Authentication and authorization in one go
  - -Brings in application logic into DTLS
  - -Still need to verify that each CoAP request is authorized
- In CoAP (URI query or CoAP Option)
  - + DTLS freed from application logic
  - + Same CoAP solution for comm. sec. and object sec.
  - -In DTLS, need to carry some authentication information related to assertion (e.g. Nonce)
    - + However, a Nonce in ClientHello allows for new DoS mitigation mechanisms

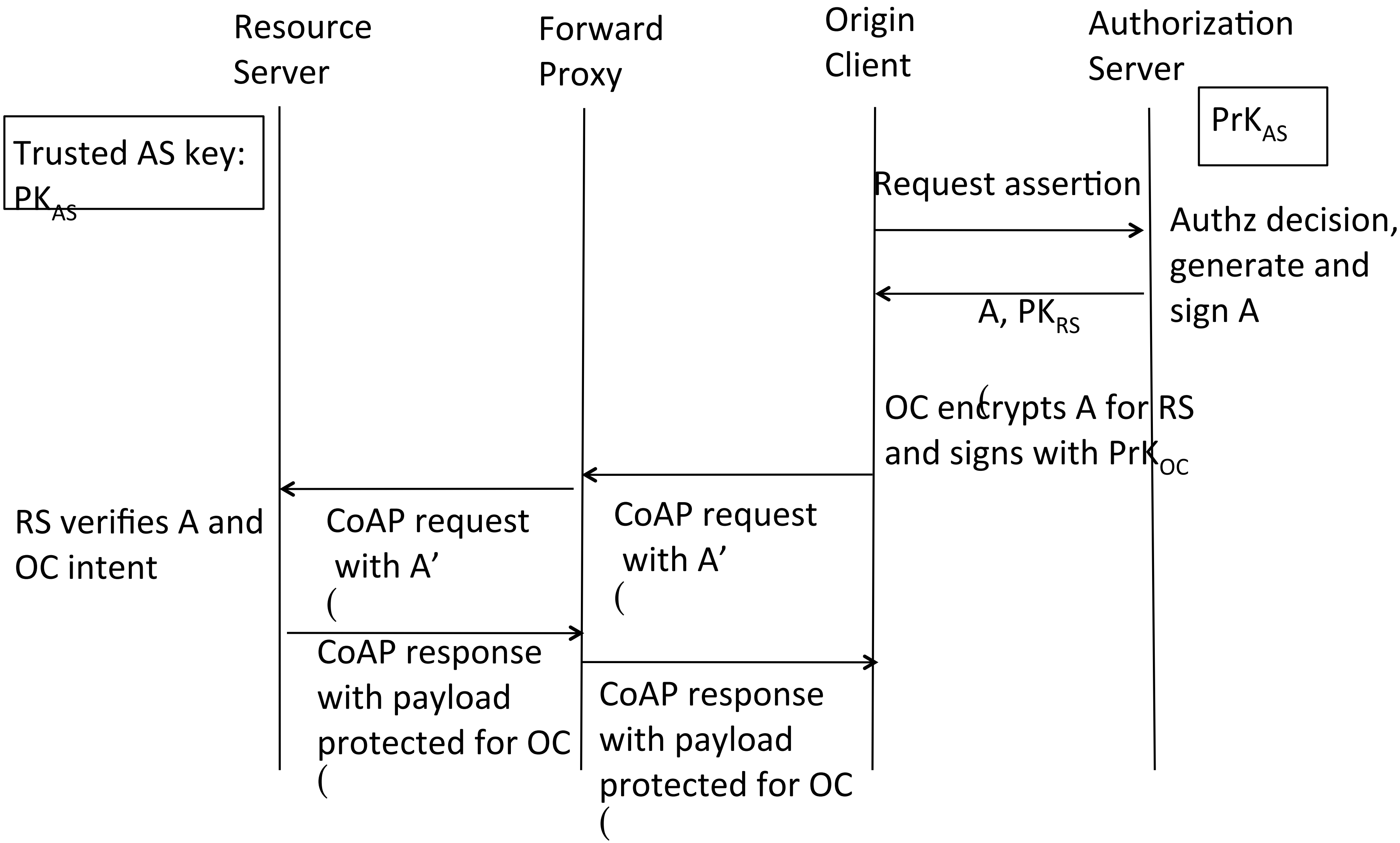
Extended ACLs

- Category A (shared key/identity/trust anchor)
  - Allowed to set up DTLS
  - "Root" access to resources
- Category B (shared key/identity/trust anchor)
  - Allowed to set up DTLS
  - Access to resource according to assertion (if any)

Features:

- Backward compatibility with CoAP RFC (= Category A)
- Distinctinguish requests which require assertions
- Enables reduced privileges for proxies

Example of Object security and ACF PK



Resource Server

Forward Proxy

Origin Client

Authorization Server

Trusted AS key: PK_{AS}

PrK_{AS}

Request assertion

Authz decision, generate and sign A

A, PK_{RS}

OC en(crypts A for RS and signs with PrK_{OC}

CoAP request with A'
(

CoAP request with A'
(

RS verifies A and OC intent

CoAP response with payload protected for OC
(

CoAP response with payload protected for OC
(

Object security based message protection

Object security for both authentication and authorization

+ Pure object based security solution instead of mixed (DTLS with assertion) reduces

+ No handshake latencies

+ Solution may piggy-back assertion (e.g. add client signature of assertion)

+ Application layer security end-to-end irrespective of proxies

-Need similar security protocol considerations as DTLS

  - + The AS is a TTP

Summary

- Outline and some details of an Access Control Framework for CoRE, based on a first set of requirements
- Compatible with AC standards
- Enables policies depending on local conditions
- Analysis of different assertion transfer options
- Alternative Security Modes for CoAP
- Both DTLS and Object Security message protection worth pursuing

Delegated CoAP Authorization Framework
draft-gerdes-core-dcaf-authorize-00

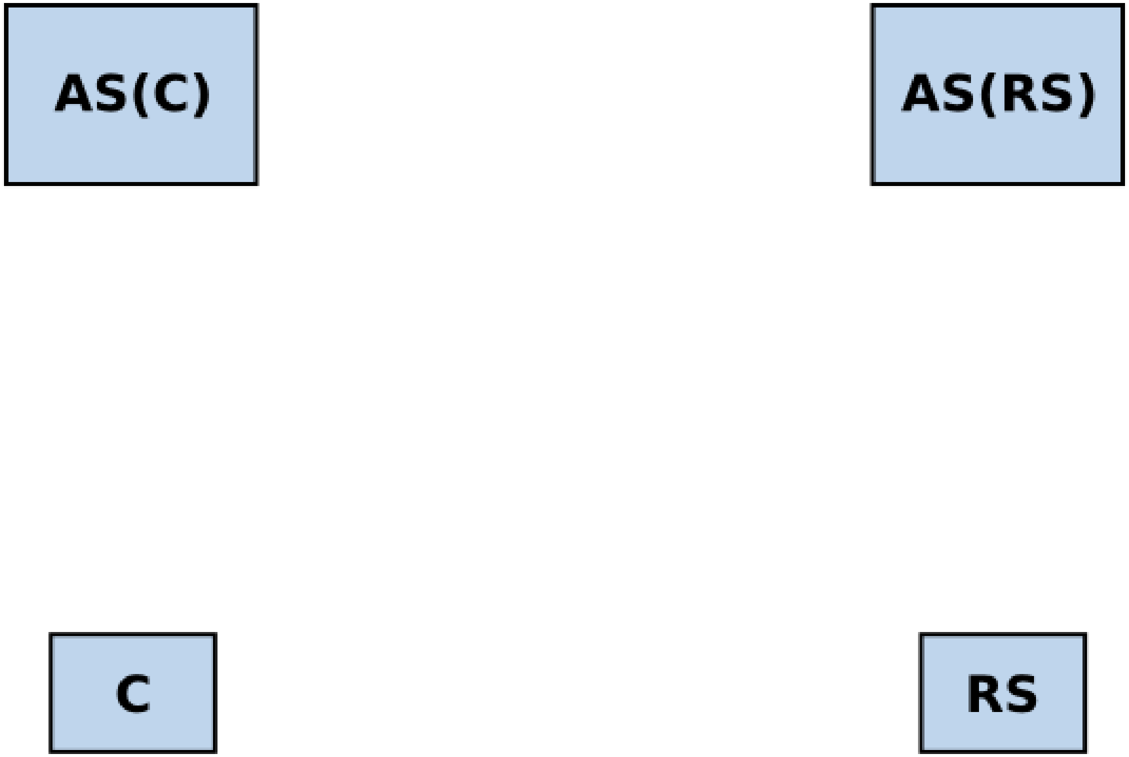
Stefanie Gerdes, Olaf Bergmann, Carsten Bormann

IETF-87, CoRE WG, 29.07.2013

Goals

- **Relieve constrained nodes from managing authentication and authorization**
- Secure exchange of authorization information
- Establish DTLS channel between constrained nodes
- Use only symmetric key cryptography on constrained nodes
- Support of class-1 devices

Architecture



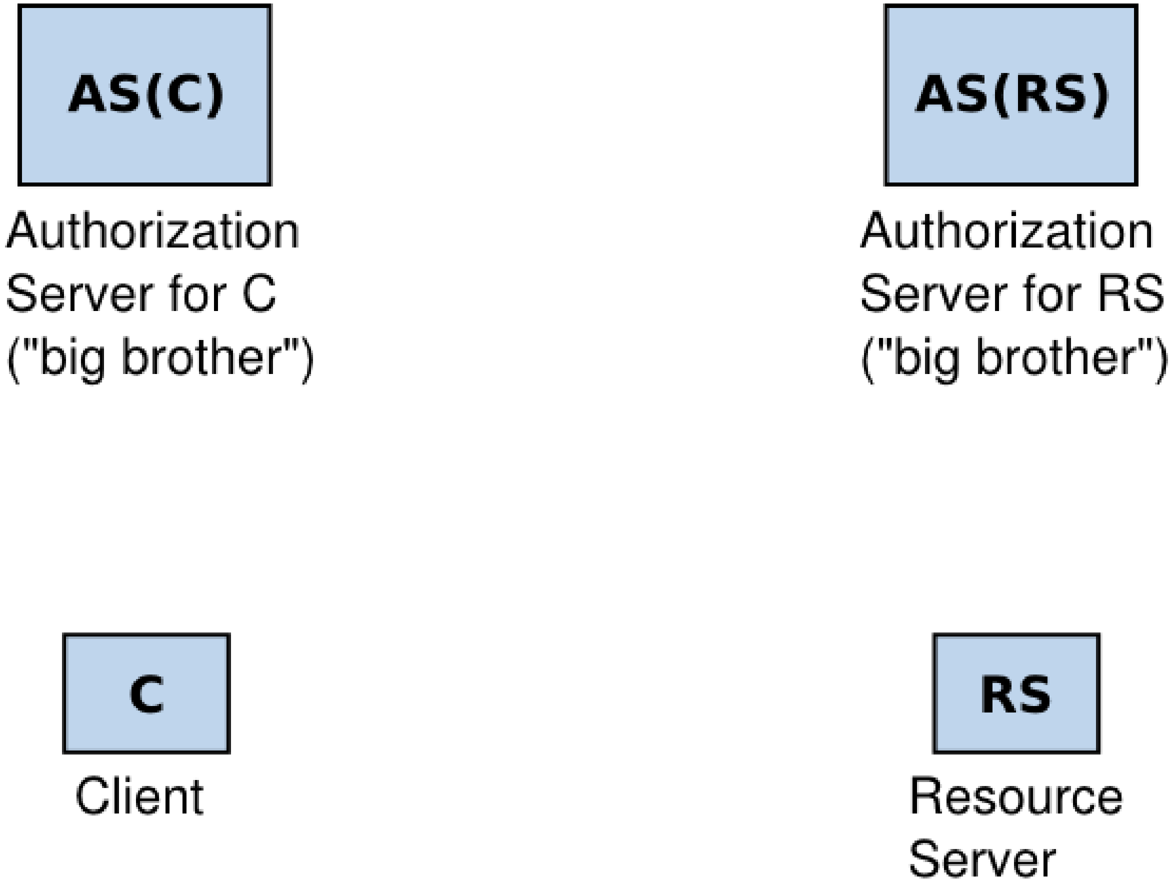
AS(C)

AS(RS)

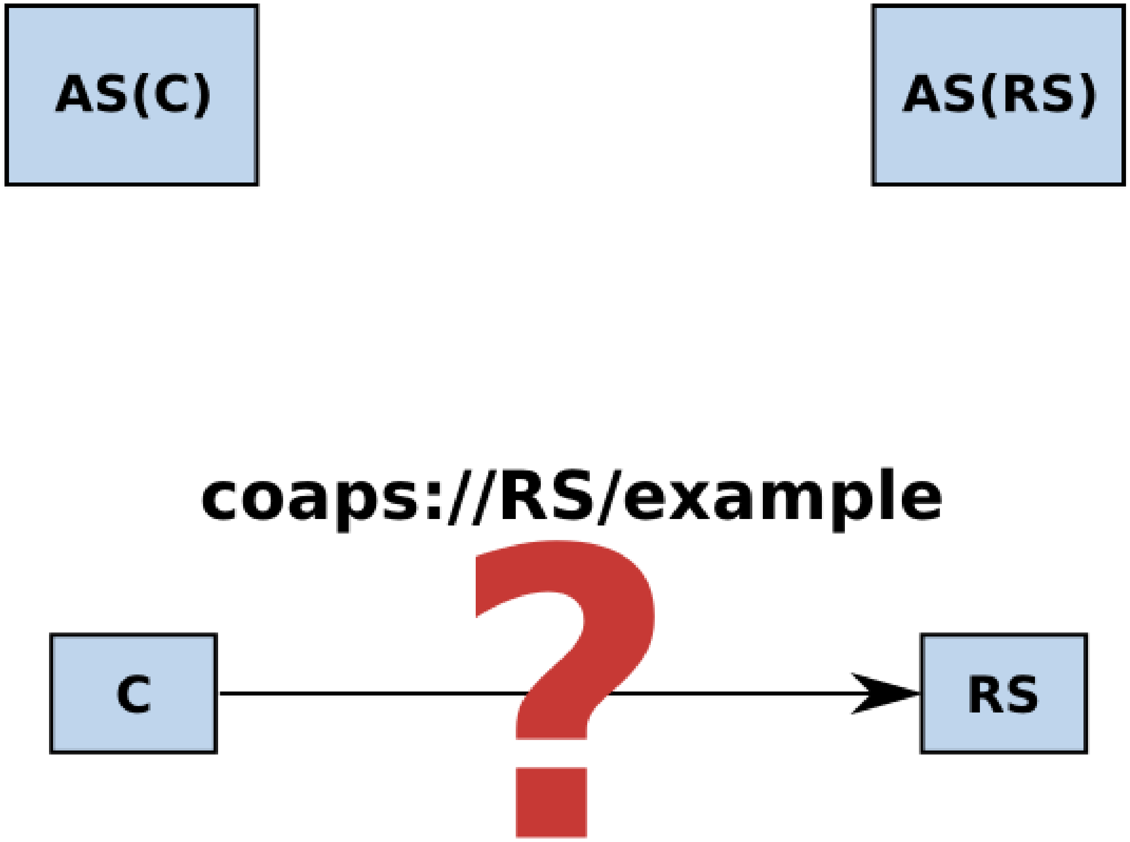
C

RS

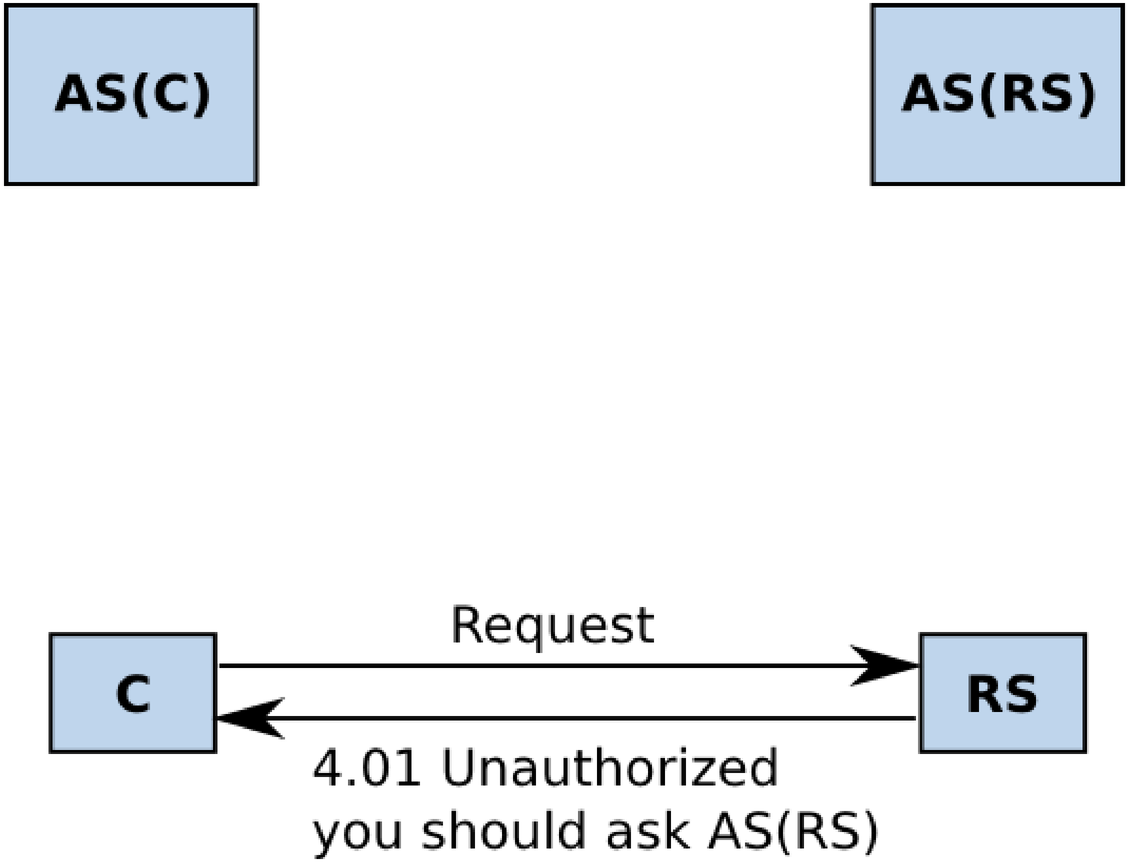
Architecture


AS(C)
Authorization Server for C ("big brother")
AS(RS)
Authorization Server for RS ("big brother")
C
Client
RS
Resource Server

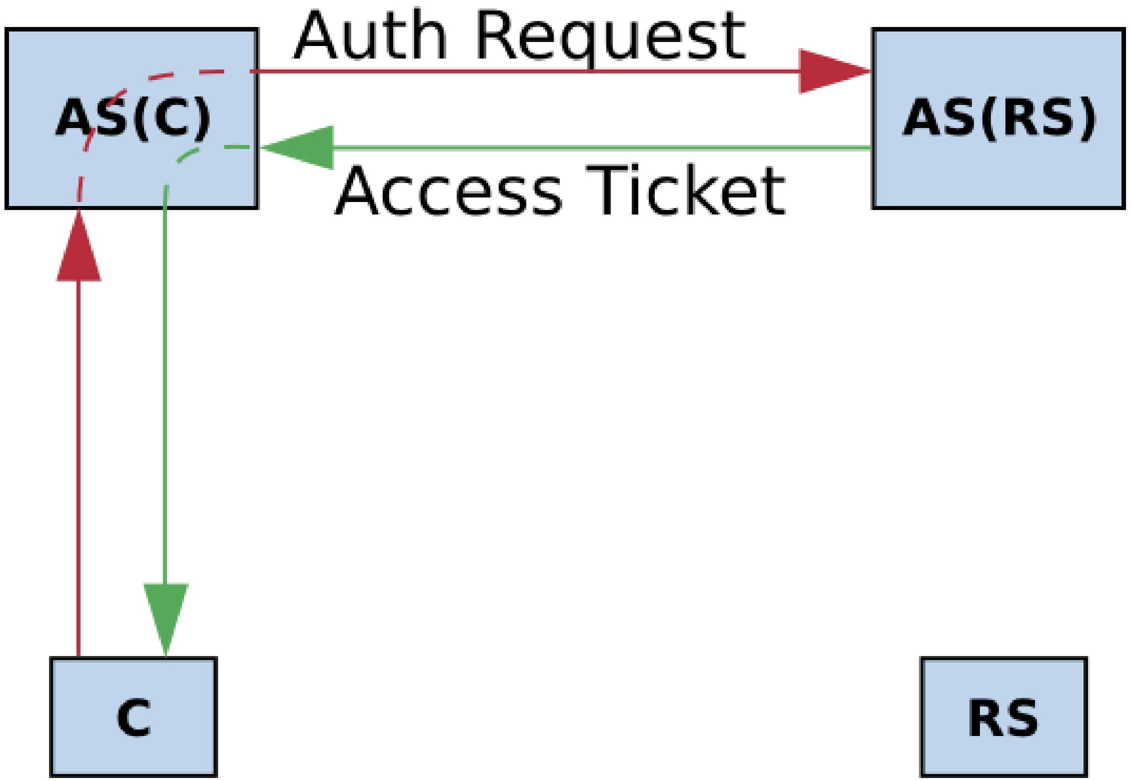
Problem: Securely Access a Resource at RS



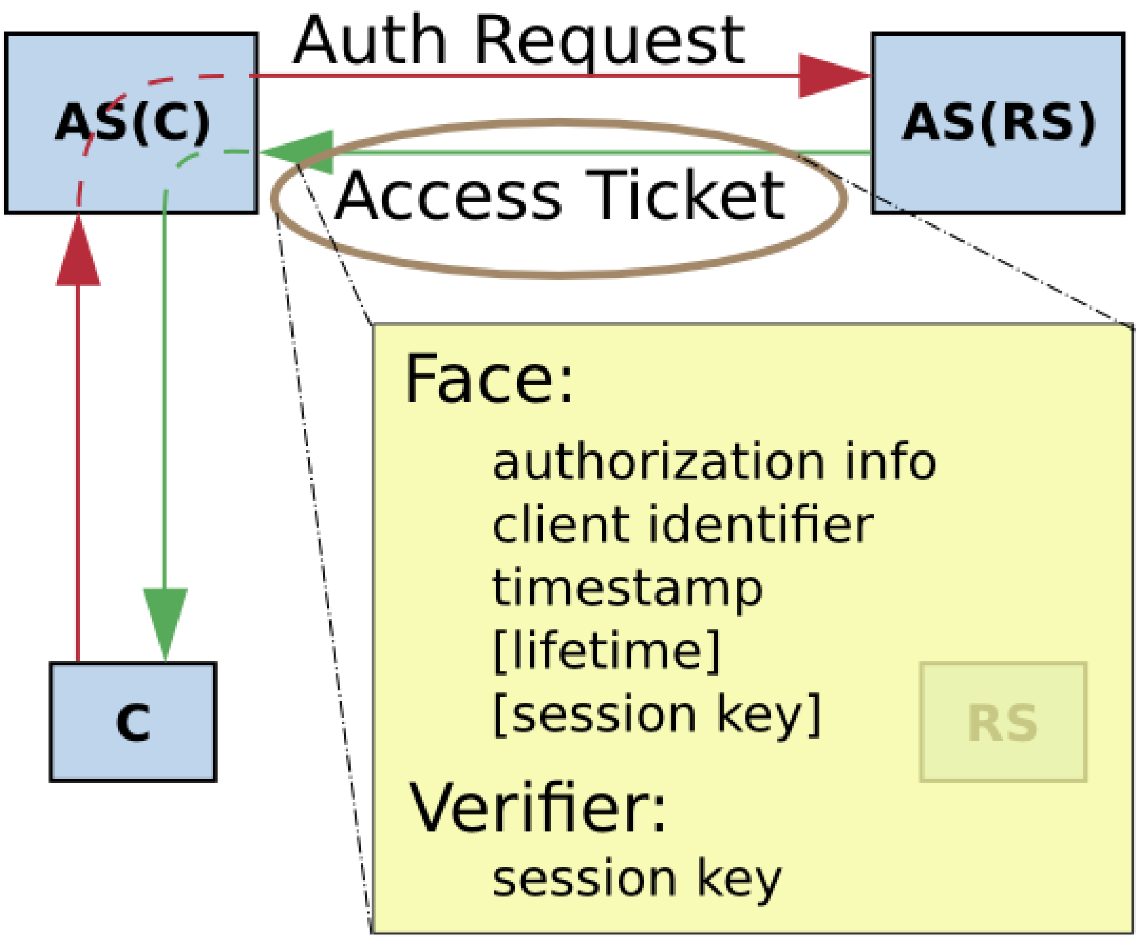
Try NoSec (or RD Lookup)



Contact RS's Big Brother for Authorization



Access Ticket



AS(C)

Auth Request

AS(RS)

Access Ticket

C

Face:

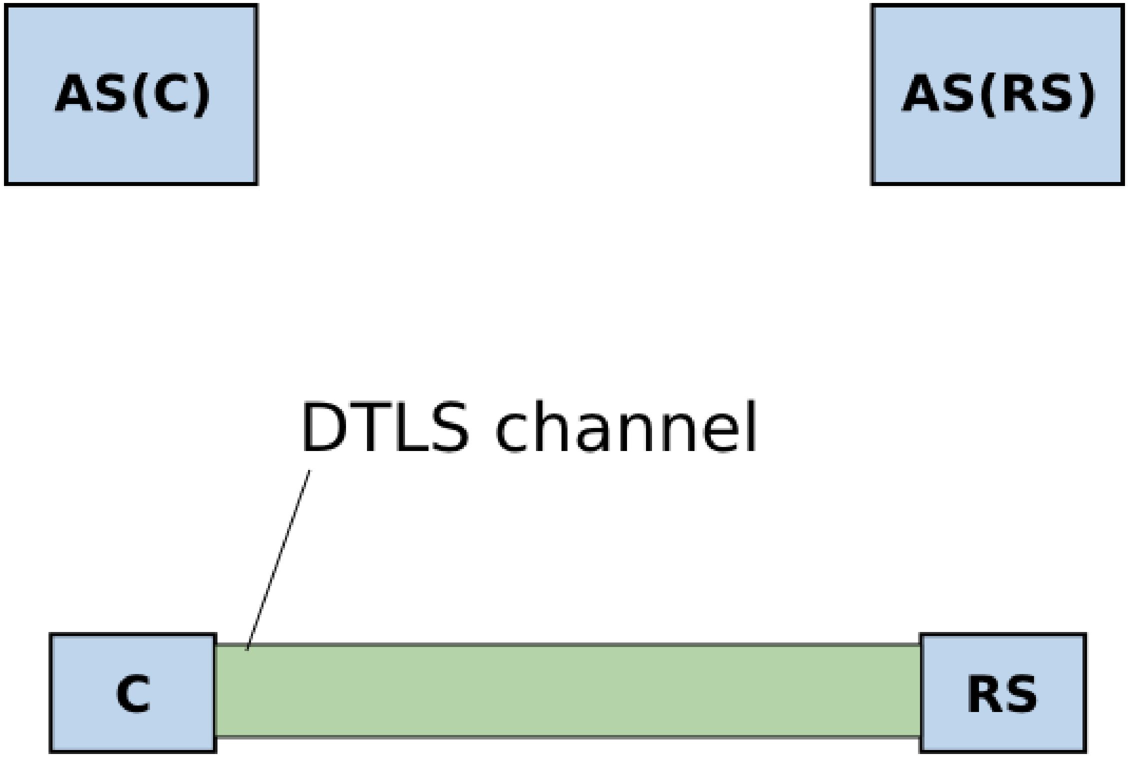
authorization info
client identifier
timestamp
[lifetime]
[session key]

RS

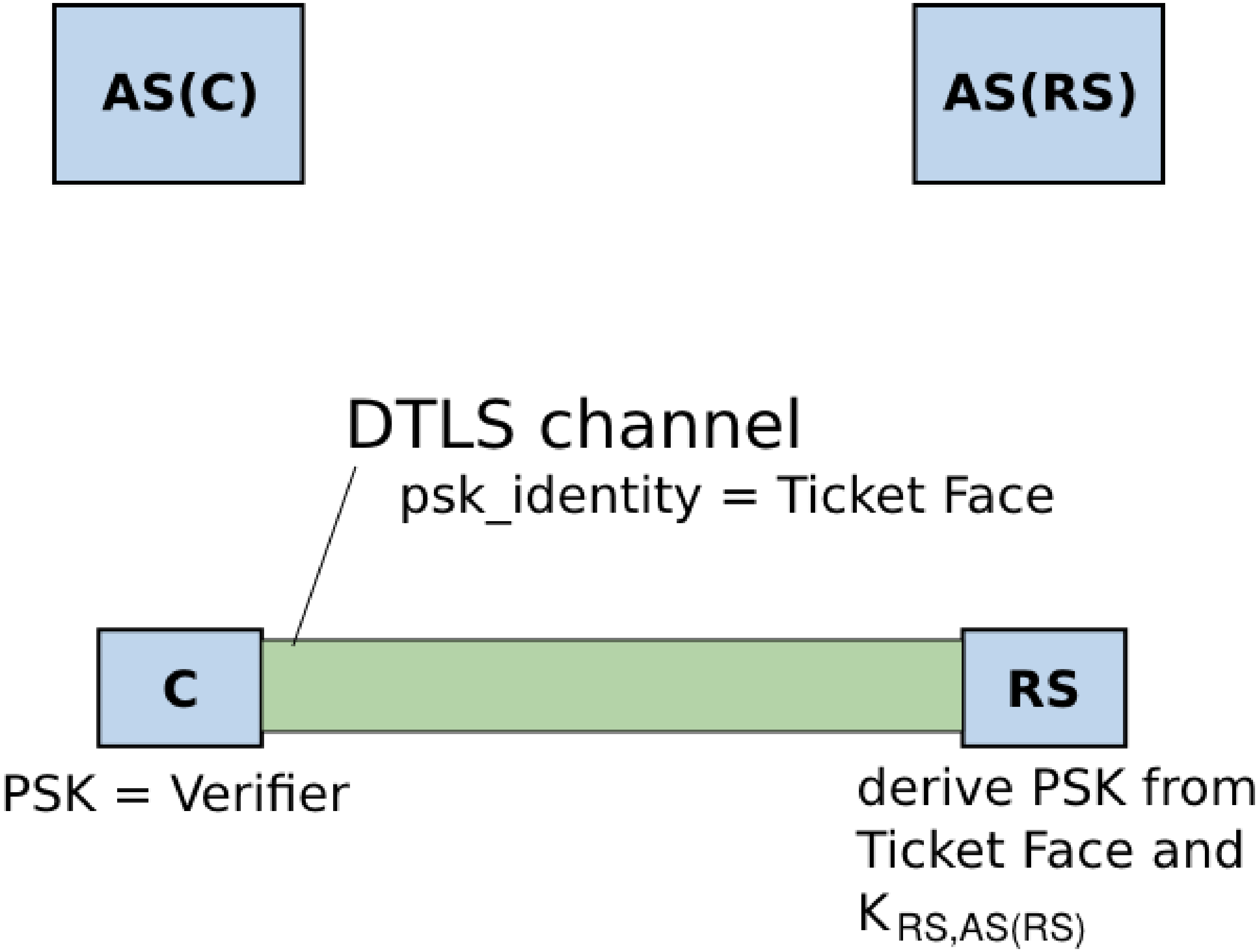
Verifier:

session key

Use Access Ticket to Establish DTLS Channel

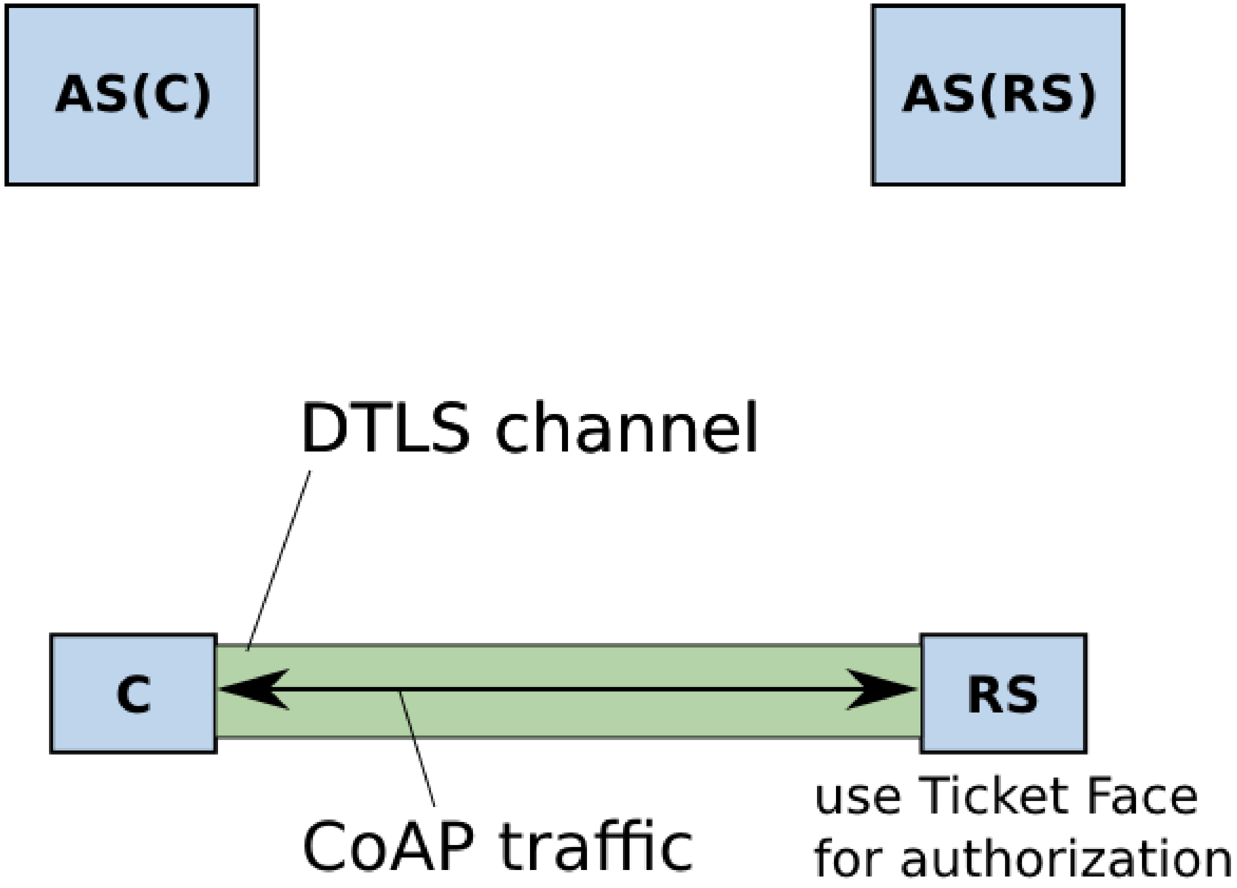


PSK Derivation

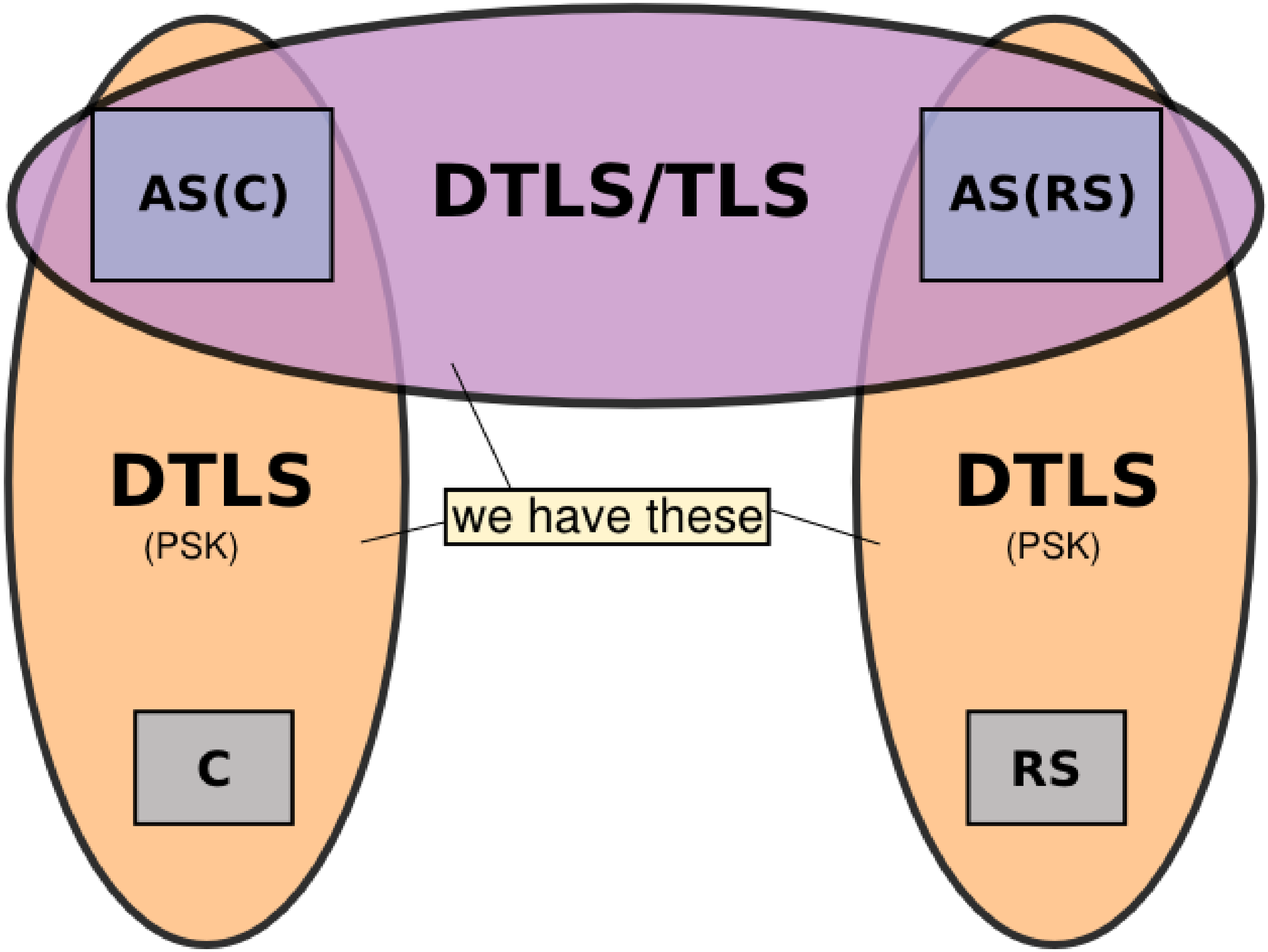


AS(C)
AS(RS)
DTLS channel
psk_identity = Ticket Face
C
RS
PSK = Verifier
derive PSK from Ticket Face and $K_{RS,AS(RS)}$

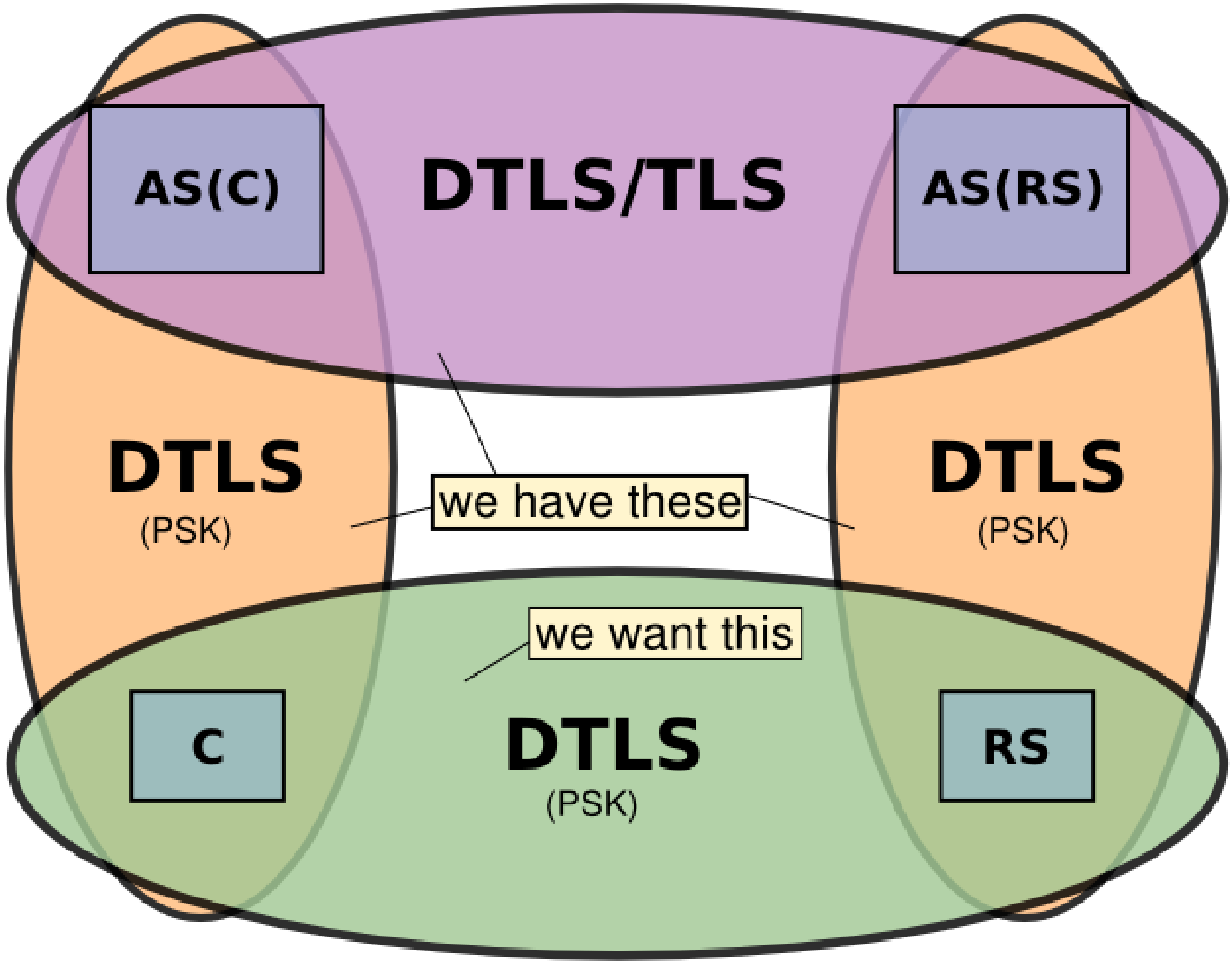
RS Permits Authorized Requests Over DTLS



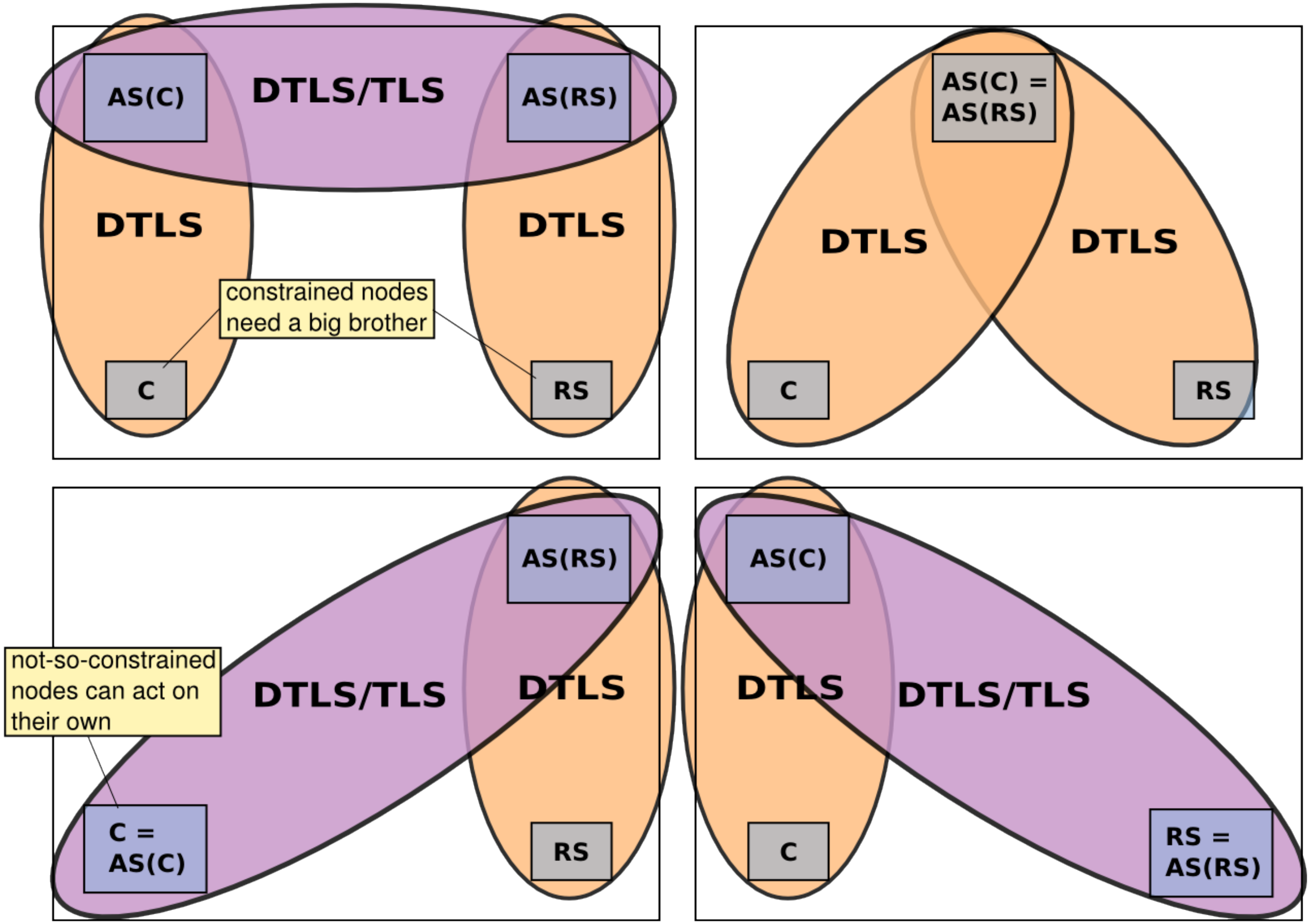
Initial Trust Relationships



Trust: The Complete Picture



Roles



constrained nodes need a big brother

not-so-constrained nodes can act on their own

The DCAF Protocol

- Requires not-so-contrained nodes to do the hard work (possibly including public-key crypto)
- Utilize DTLS to transmit authorization information and access tickets
- Authenticate origin client by its access ticket:
  - RS and AS(RS) share at least one session key
  - AS(RS) creates Ticket Face + Verifier, tells AS(C), C
  - C initiates DTLS handshake with RS
  - Ticket Face is PSK identity, Verifier is PSK
  - RS calculates PSK from Ticket Face
- Knowledge of Verifier authenticates C to RS!
- Knowledge of PSK authenticates RS to C!
- Face contains authorization information valid for the entire session
  - Verifier ensures Face's integrity

Conclusions

- Off-load authentication and authorization from constrained nodes
  - Big brothers can use PKI etc. to authenticate
  - constrained nodes only need symmetric key cryptography
- secure transmission of authorization information
- secure transmission of PSK for DTLS channel C ↔ RS
- requires transitive trust relationship RS→AS(RS)→AS(C)→C

- Questions:
  - Is this approach reasonable?
  - Is this something that the WG should work on?

Group 2:WG docs

Group Communication for CoAP

Akbar Rahman
Esko Dijk

IETF 87, July 2013

http://www.ietf.org/id/draft-ietf-core-groupcomm-011.txt

Summary of Changes (1/7)

- I-D had several updates (Rev. 06 to Rev.11) after IETF-86 (Orlando)

- Changes from ietf-05 to ietf-06:
  - Added a new section on commissioning flow when using discovery services when end devices discover in which multicast group they are allocated (#295).
  - Added a new section on CoAP Proxy Operation (section 3.9) that outlines the potential issues and limitations of doing CoAP multicast requests via a CoAP Proxy (#274).
  - Added use case of multicasting controller on the backbone (#279).
  - Use cases were updated to show only a single CoAP RD (to replace the previous multiple RDs with one in each subnet). This is a more efficient deployment and also avoids RD specific issues such as synchronization of RD information between serves (#280).

Summary of Changes (2/7)

- Changes from ietf-05 to ietf-06 (continued):
  - Added text to section 3.6 (Configuring Group Membership in Endpoints) that clarified that any (unicast) operation to change an endpoint's group membership must use DTLS-secured CoAP.
  - Clarified relationship of this document to [I-D.ietf-core-coap] in section 2.2 (Scope).
  - Removed IPSec related requirement, as IPSec is not part of [I-D.ietf-core-coap] anymore.
  - Editorial reordering of subsections in section 3 to have a better flow of topics. Also renamed some of the (sub)sections to better reflect their content. Finally, moved the URI Configuration text to the same section as the Port Configuration section as it was a more natural grouping (now in section 3.3) .
  - Editorial rewording of section 3.7 (Multicast Request Acceptance and Response Suppression) to make the logic easier to comprehend (parse).
  - Various editorial updates for improved readability.

Summary of Changes (3/7)

- Changes from ietf-06 to ietf-07:
  - Added an IANA request (in section 7.2) for a dedicated content- format (Internet Media type) for the group management JSON format called 'application/coap-group+json' (#299).
  - Clarified semantics (in section 3.6) of group management JSON format (#300).
  - Added details of IANA request (in section 7.1) for a new CORE Resource Type called 'core.gp'.
  - Clarified that DELETE method (in section 3.6) is also a valid group management operation.
  - Various editorial updates for improved readability.

Summary of Changes (4/7)

- Changes from ietf-07 to ietf-08:
  - Updated text in section 3.6 (Configuring Group Membership in Endpoints) to make it more explicit that the Internet Media Type is used in the processing rules (#299).
  - Addressed various comments from Peter van der Stok (#296).
  - Various editorial updates for improved readability including defining all acronyms.

Summary of Changes (5/7)

- Changes from ietf-08 to ietf-09:
  - Cleaned up requirements language in general. Also, requirements language are now only used in section 3 (Protocol Considerations) and section 6 (Security Considerations). Requirements language has been removed from other sections to keep them to a minimum (#271).
  - Addressed final comment from Peter van der Stok to define what "IP stack" meant (#296). Following the lead of CoAP-17, we know refer instead to "APIs such as IPV6_RECVPKTINFO [RFC3542]".
  - Changed text in section 3.4 (Group Methods) to allow multicast POST under specific conditions and highlighting the risks with using it (#328).
  - Various editorial updates for improved readability.

Summary of Changes (6/7)

- Changes from ietf-09 to ietf-10:
  - Added a fourth option in section 3.3 on ways to obtain the URI path for a group request.
  - Clarified use of content format in GET/PUT requests for Configuring Group Membership in Endpoints (in section 3.6).
  - Changed reference "draft-shelby-core-resource-directory" to "draft-ietf-core-resource-directory".
  - Clarified (in section 3.7) that ACKs are never used for a multicast request (from #296).
  - Clarified (in section 5.2/5.2.3) that MPL does not support group membership advertisement.
  - Adding introductory paragraph to Scope (section 2.2).
  - Wrote out fully the URIs in table section 3.2.
  - Reworded security text in section 7.2 (New Internet Media Type) to make it consistent with section 3.6 (Configuring Group Membership).
  - Fixed formatting of hyperlinks in sections 6.3 and 7.2.

Summary of Changes (7/7)

- Changes from ietf-10 to ietf-11:
  - Added text to section 3.8 (Congestion Control) to clarify that a "CoAP client sending a multicast CoAP request to /.well-known/core SHOULD support core-block" (#332).
  - Various editorial updates for improved readability.

Configuring Group Membership in Endpoints (1/2)

- Would like to reconfirm with WG the approach for configuring Group Membership In Endpoints (Section 3.6 and the IANA Section)
  - Optional (unicast) RESTful interface (GET/PUT/POST/DELETE) to set Group configuration resource in Endpoints by commissioning tool:
    - New “core.gp” resource type (rt) CoRE Parameters Registry for designating the Group Configuration resource
    - New “application/coap-group+json” Internet Media Type for the Group configuration resource content
      - JSON-based content format
      - IP multicast address or Hostname (FQDN)

Configuring Group Membership in Endpoints (2/2)

- Example:
  - Req: POST /gp (Content-Format: application/coap-group+json)
    [{ "n": "floor1.west.bldg6.example.com",
    "ip": "ff15::4200:f7fe:ed37:14cb" }]
  - Res: 2.04 Changed

- SHOULD use DTLS-secured CoAP communication between Endpoint and commissioning tool

Next Steps

- Any other updates that the WG would like to see?
- Is the I-D ready for WGLC?

Best Practices for HTTP-CoAP Mapping Implementation

Angelo Castellani, Salvatore Loreto, Akbar Rahman, Thomas Fossati, Esko Dijk

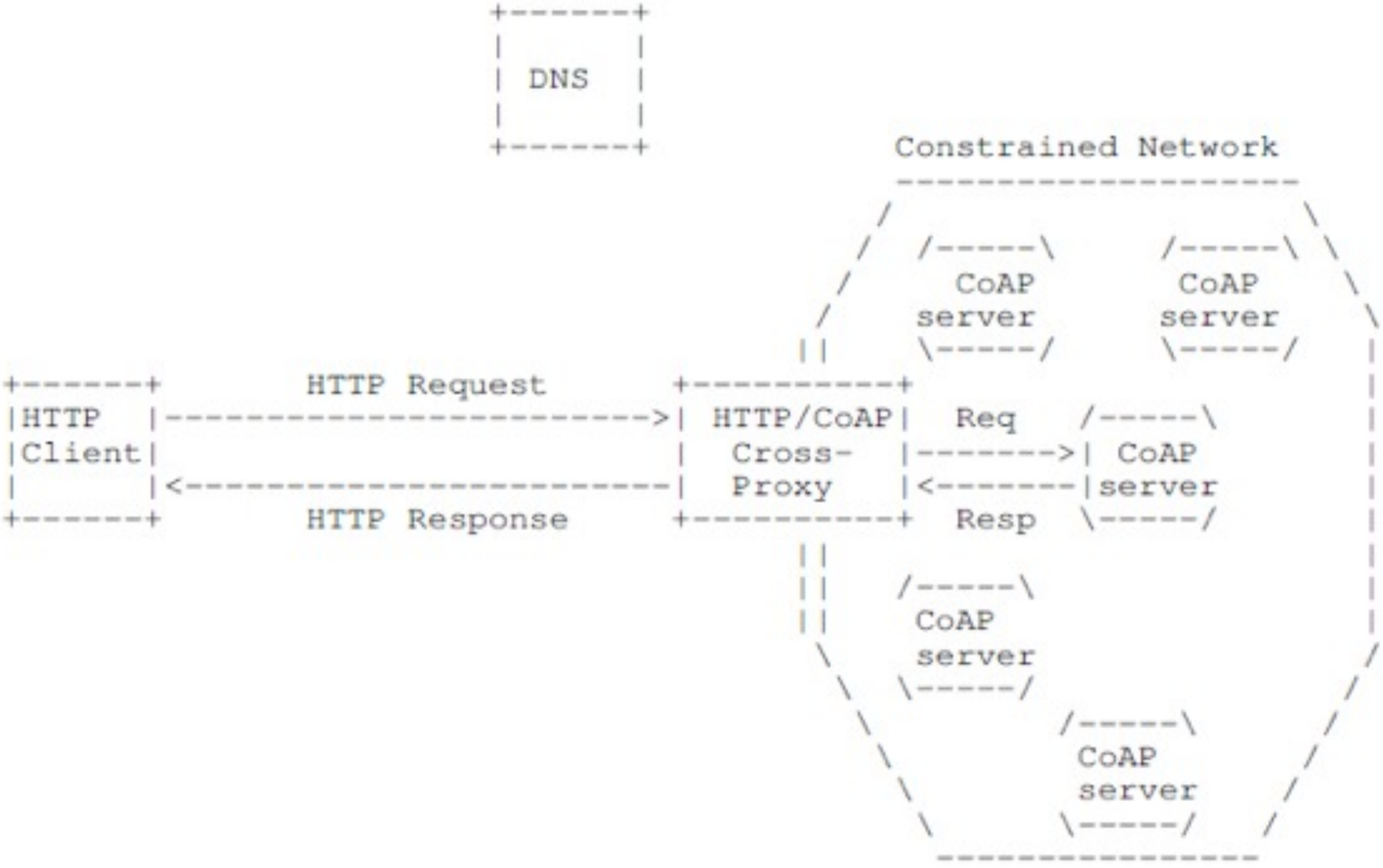
IETF 87, July 2013

http://www.ietf.org/id/draft-ietf-core-http-mapping-01.txt

Main Changes (from IETF Orlando)

- Added (for Reverse Proxy) HTTP to CoAP URI Mapping:
  - Requirements
  - Proposals
    - Solution #1: Carsten's Mapping Proposal
    - Solution #2: Adding IPv6 Literals to Carsten's Mapping Proposal
    - Solution #3: Split CoAP URI in Query Arguments
    - Solution #4: CoAP URI as Single Query Parameter
    - Solution #5: Split CoAP URI in Path Components
  - Solution Comparison Matrix

Reverse Cross-Protocol Proxy Deployment Scenario



Requirements (1/2)

- REQ1: Syntactic correctness of the HTTP URI (i.e. handle percent-encoding when needed)
- REQ2: HTTP URI must be able include all elements of a CoAP URI (e.g. coap(s) scheme, hostname, host literals IPv4/IPv6, port, path, query components, characters allowed in CoAP URI)
- REQ3: The mapping operation must produce a string that can be directly used by a proxy as input to the process of Section 6.4. of [I-D.ietf-core-coap]
- REQ4: HTTP URI should be easily readable/writable by humans, if possible (i.e. easy to read the CoAP URI embedded in it, e.g.: avoid multiple levels of percent encoding, etc.)

Requirements (2/2)

- REQ5: HTTP cache friendliness of the mapping solution, i.e. maximize the caching of CoAP resources by HTTP intermediaries (a solution where entire CoAP URI is provided as a single query parameter is bad in this respect)
- REQ6: Normalised form: preferably there should be only one normal/ default way to encode the URI, so that we do not end up with multiple different cache entries for the same CoAP resource in intermediaries
- REQ7: HTTP URI should be as short as possible

Solution #1: Carsten's Mapping Proposal

- This is the mapping proposal originally defined in [I-D.bormann-core-cross-reverse-convention]
- URI template: TBD
- Example:
  - http://proxy.example.com/.well-known/core-translate/1.2.3.4_4567/foo/bar?a=3
  - Maps to coap://1.2.3.4:4567/foo/bar?a=3
- Notes:
  - How to include IPv6 literals was not defined in [I-D.bormann-core-cross-reverse-convention]
  - The CoAP scheme is derived from HTTP scheme (http or https). The "_{Port}" part is optional

Solution #2: Adding IPv6 Literals to Carsten's Mapping Proposal (1/2)

- Adding IPv6 literals support to [I-D.bormann-core-cross-reverse-convention]
- URI template: “/.well-known/core-translate/{authority-encoded}/ {path}?{query}”
- Example 1:
  - http://proxy.example.com/.well-known/core-translate/%5B2001:db8::1%5D:4567/foo/bar?a=3
  - Maps to coap://[2001:db8::1]:4567/foo/bar?a=3
- Example 2:
  - http://proxy.example.com/.well-known/core-translate/server.coap.example.com:4567/foo/bar?a=3
  - Maps to coap://server.coap.example.com:4567/foo/bar?a=3

Solution #2: Adding IPv6 Literals to Carsten's Mapping Proposal (2/2)

- Example 3:
  - http://proxy.example.com/.well-known/core-translate/ server.coap.example.com/foo/bar?a=3
  - Maps to coap://server.coap.example.com/foo/bar?a=3
- Example 4:
  - http://proxy.example.com/.well-known/core-translate/ 1.2.3.4:4567/ foo/bar?a=3
  - Maps to coap://1.2.3.4:4567/foo/bar?a=3

Solution #3: Split CoAP URI in Query Arguments (1/2)

- This proposal splits the CoAP URI in parts and puts parts in to separate query arguments of the HTTP URI
- URI template: “/.well-known/core-translate/ host={host} &port={port}&path={path}?{query}”
  - Note: The query parts "host", "port", "path" and "query" are all optional in the URI
  - TBD: discuss order of query arguments; and what to do with duplicates

Solution #3: Split CoAP URI in Query Arguments (2/2)

- Example:
  - http://proxy.example.com/.well-known/core- translate?host= %5B2001:db8::1%5D&port=4567&path=/foo/bar?a=3&b=5
  - Maps to coap://[2001:db8::1]:4567/foo/bar?a=3&b=5

Solution #4: CoAP URI as Single Query Parameter

- Inspired by certain web services that put HTTP callback URIs in URI- query parameters

- URI template: TBD

- Example:
  - http://proxy.example.com/.well-known/core-translate?uri=coap%3A%2 F %2F%5B2001%3Adb8%3A%3A1%5D%3A4567%2Ffoo%2Fbar%3Fb %3Dbefore_colon% 253Aafter_colon
  - Maps to coap://[2001:db8::1]:4567/foo/bar?b=before_colon %3Aafter_colon

Solution #5: Split CoAP URI in Path Components (1/2)

- URI template: /.well-known/core-translate/{scheme}/{+host}/{port}/{+path_abempty}/{+query}

- Where:
  - scheme is "coap" or "coaps" or the empty string;
  - host matches the production defined in [RFC3986] Sec. 3.2.2. (need to percent-encode '[' and ']' in IP-literal);
  - port matches the production defined in [RFC3986] Sec. 3.2.3.;
  - path_abempty matches the production defined in [RFC3986] Sec. 3.3. (need to percent-encode any '/' occurrence);
  - query matches the production defined in [RFC3986] Sec. 3.4. (need to percent-encode any '/' and '?' occurrence);

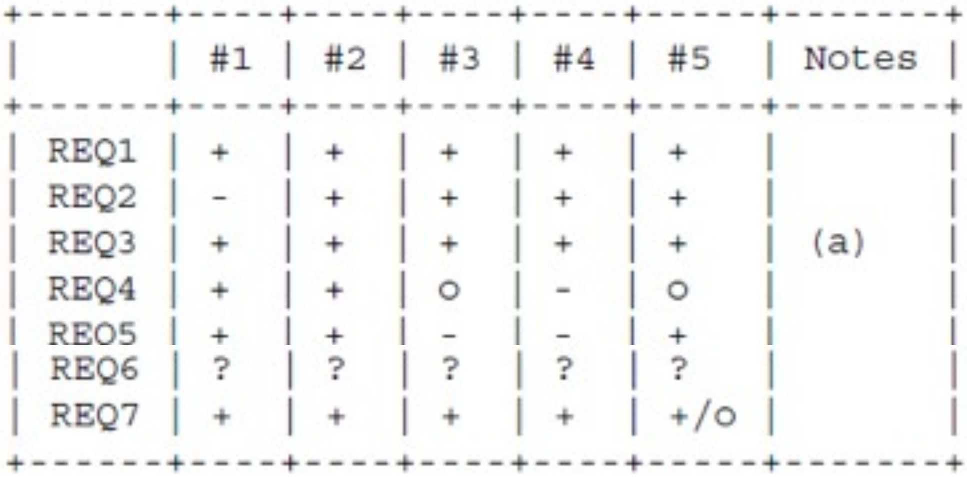
Solution #5: Split CoAP URI in Path Components (2/2)

- CoAP URI is reconstructed as per [RFC3986] Sec. 5.3. carrying out the following substitutions before going through the algorithm:
  - if scheme is the empty string, make it "coap";
  - if port is the empty string, make it "5683";
  - TBD (possibly not needed): if path-abempty is the empty string, make it "/";

- Example:
  - http://proxy.example.com/.well-known/core-translate/coap/server.coap.example.com/4567/foo%2Fbar/a=3
  - coap://server.coap.example.com:4567/foo/bar?a=3

Solution Comparison Matrix

- The following table compares the HTTP to CoAP URI solutions to the given requirements



	#1	#2	#3	#4	#5	Notes
REQ1	+	+	+	+	+	
REQ2	-	+	+	+	+	
REQ3	+	+	+	+	+	(a)
REQ4	+	+	o	-	o	
REQ5	+	+	-	-	+	
REQ6	?	?	?	?	?	
REQ7	+	+	+	+	+/o	

Legend:
+ Meets the requirement
- Does not meet the requirement
o Partly meets the requirement
? TBD
(a) Details need to be defined for each solution.

Next Steps

- What approach for HTTP to CoAP URI translation does the WG recommend?
    - Solution #1: Carsten's Mapping Proposal
    - Solution #2: Adding IPv6 Literals to Carsten's Mapping Proposal
    - Solution #3: Split CoAP URI in Query Arguments
    - Solution #4: CoAP URI as Single Query Parameter
    - Solution #5: Split CoAP URI in Path Components

CoRE Resource Directory

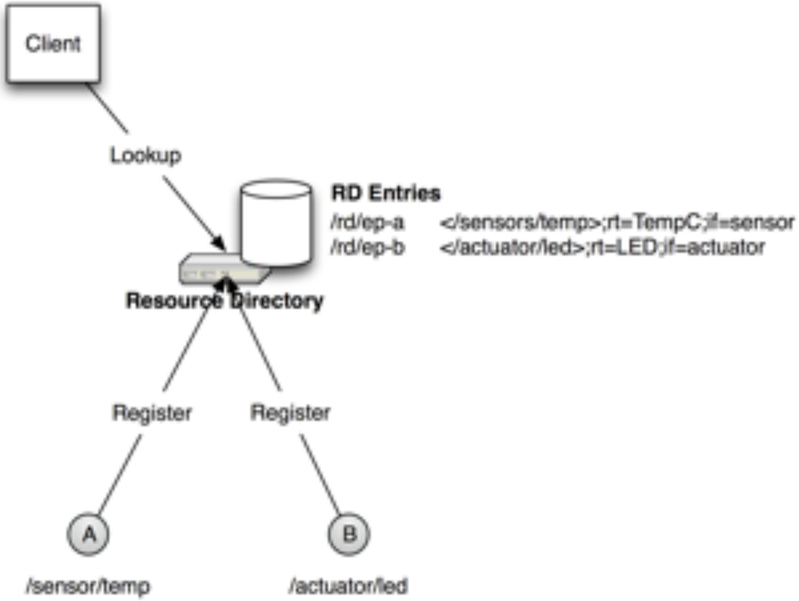
draft-ietf-core-resource-directory-05

Z. Shelby, C. Bormann, S. Krco

CoRE WG, IETF-87 Berlin

Background

- Not a new concept
  - think web search engine or any link directory
- Defines the interfaces to a Resource Directory
- Based on Web Linking framework and the CoRE Link Format
- Generic REST design for use over HTTP and CoAP
- Part of the OMA Lightweight M2M standard
- Has already been deployed
  - In traffic monitoring systems
  - In street lighting systems
  - For vehicular asset tracking
  - By a major Cellular M2M operator


Client
Lookup
RD Entries
/rd/ep-a </sensors/temp>;rt=TempC;if=sensor
/rd/ep-b </actuator/led>;rt=LED;if=actuator
Resource Directory
Register
Register
A
B
/sensor/temp
/actuator/led

Changes since shelby-05

- Submitted as WG document

When are we done?

- We are defining an interface…. let’s keep it simple
- Close the WG adoption comments
- Get (even) more implementation experience
- Access control and security considerations completed
- Integrate a DNS-SD mapping section
- Maintain compatibility with OMA Lightweight M2M
  - Registration, Update and De-registration interfaces

Comments and Known Issues

- Remove the ETag “Validation” feature
- Add a DNS-SD mapping section based on draft-lynn-core-discovery-mapping-02
- More clarification in the Simple Discovery and Discovery sections
- Disallow a GET on the EP entry location e.g. /rd/1234
- Cross-reference the Groupcomm draft WRT the RD group functionality
- Further security considerations on access control
- Improve the lookup function set
  - Some link responses are awkward
  - Separate types of lookups into separate function sets?

draft-ietf-core-interfaces-00

CoRE Interfaces

Zach Shelby, Matthieu Vial

CoRE WG, IETF-87 Berlin

CoRE Interfaces

- Interface patterns for use in typical IoT applications
- Applicable to both CoAP or HTTP
- Set of basic REST interfaces
  - Link list
  - Batch
  - Linked batch
  - Sensor
  - Parameter, Read-only Parameter
  - Actuator
  - Bindings
  - Resource observation query interface
- Example resource organization structure
  - Function sets, made up of sub-resources
    - Sub-resources and their attributes
      - Path, resource type, interface type(s), data type etc.

Simple Interfaces

Interface	if=	Methods
Link List	core.ll	GET
Batch	core.b	GET, PUT, POST (where applicable)
Linked Batch	core.lb	GET, PUT, POST, DELETE (where applicable)
Sensor	core.s	GET
Parameter	core.p	GET, PUT
Read-only Parameter	core.rp	GET
Actuator	core.a	GET, PUT, POST

Function Set Example

Function Set	Root Path	RT	IF
Device Description	/d	simple-dev	core.ll
Sensors	/s	simple-sen	core.b
Actuators	/a	simple-act	core.b

Type	Path	RT	IF	Data Type
Name	/d/name	simple:dev:n	core.p	xsd:string
Model	/d/model	simple:dev:mdl	core.rp	xsd:string

Type	Path	RT	IF	Data Type
Light	/s/light	simple-sen-lt	core.s	xsd:decimal (lux)
Humidity	/s/humidity	simple-sen-hum	core.s	xsd:decimal (%RH)
Temperature	/s/temp	simple-sen-tmp	core.s	xsd:decimal (degC)

Simple Examples

Sensor Interface

```
Req: GET /s/humidity (Accept: text/plain)
Res: 2.05 Content (text/plain)
80

Req: GET /s/humidity (Accept: application/senml+json)
Res: 2.05 Content (application/senml+json)
{"e":[
    { "n": "humidity", "v": 80, "u": "%RH" }],
}
```

Parameter Interface

```
Req: GET /d/name
Res: 2.05 Content (text/plain)
node5
```

Actuator Interface

```
Req: GET /a/1/led
Res: 2.05 Content (text/plain)
0
```

List & Batch Examples

Batch Interface

```
 Req: GET /s
   Res: 2.05 Content (application/senml+json)
   {"e":[
       { "n": "light", "v": 123, "u": "lx" },
       { "n": "temp", "v": 27.2, "u": "degC" },
       { "n": "humidity", "v": 80, "u": "%RH" }],
   }
```

Link List Interface

```
 Req: GET /d (Accept:application/link-format)
   Res: 2.05 Content (application/link-format)
   </d/name>;rt="simple-dev-n";if="core.p",
   </d/model>;rt="simple-dev-mdl";if="core.rp”
```

Linked Batch Interface

```
 Req: POST /l (Content-type: application/link-format)
   </s/light>,</s/temp>
   Res: 2.04 Changed
```

Changes since shelby-04

- Submitted as WG document

When are we done?

- Remember, this is Informational
- Set of useful REST interface paradigms for IoT
  - The current scope is already broad enough?
- Close comments from WG adoption
- Complete security considerations

Comments and Known Issues

- Improve the scope and purpose of the document
- Clarify that the Observe query parameters are one way of achieving such functionality
- Align with OMA Lightweight Object design as an example
- Security considerations need work
- Should we add a Collection interface type?
  - POST/DELETE to add/remove sub-resources

- 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 3979 and its updates

✓Blue sheets
✓Scribe(s)

Note Well

This summary is only meant to point you in the right direction, and doesn't have all the nuances. The IETF's IPR Policy is set forth in BCP 79; please read it carefully.

The brief summary:

❖**By participating with the IETF, you agree to follow IETF processes.**

❖**If you are aware that a contribution of yours (something you write, say, or discuss in any IETF context) is covered by patents or patent applications, you need to disclose that fact.**

❖**You understand that meetings might be recorded, broadcast, and publicly archived.**

For further information, talk to a chair, ask an Area Director, or review the following:

BCP 9 (on the Internet Standards Process)

BCP 25 (on the Working Group processes)

BCP 78 (on the IETF Trust)

BCP 79 (on Intellectual Property Rights in the IETF)

Group 4:
Related Work Reports

CoRE AA Status Summary

Stefanie Gerdes

IETF-87, CoRE WG, 01.08.2013

Basic Goals

- Authentication and Authorization
- Support class-1 and class-2 devices
- Support multiple crypto schemes (PK, PSK)

Documents (Proposed)

In Flux

- Overview draft
- Communication / channel security draft
- Object security draft
- Authorization Representation draft

Topics

- Use cases
- Common Terminology
- Requirements
- Trust Model
- Roles
- Object / channel security
- Revocation
- Key management
- Crypto schemes (PK, PSK)
- Authorization information / assertions representation
- Threat Model

Further steps

- Start working on that in CoRE
- Talk to other working groups with useful input / related topics
  - OAuth
  - JOSE
- Input is appreciated, especially on use cases and requirements

OMA Lightweight M2M Overview

(A new standard using lots of IETF specs including DTLS, CoAP, Block, Observe, Resource Directory, SenML…)

Zach Shelby

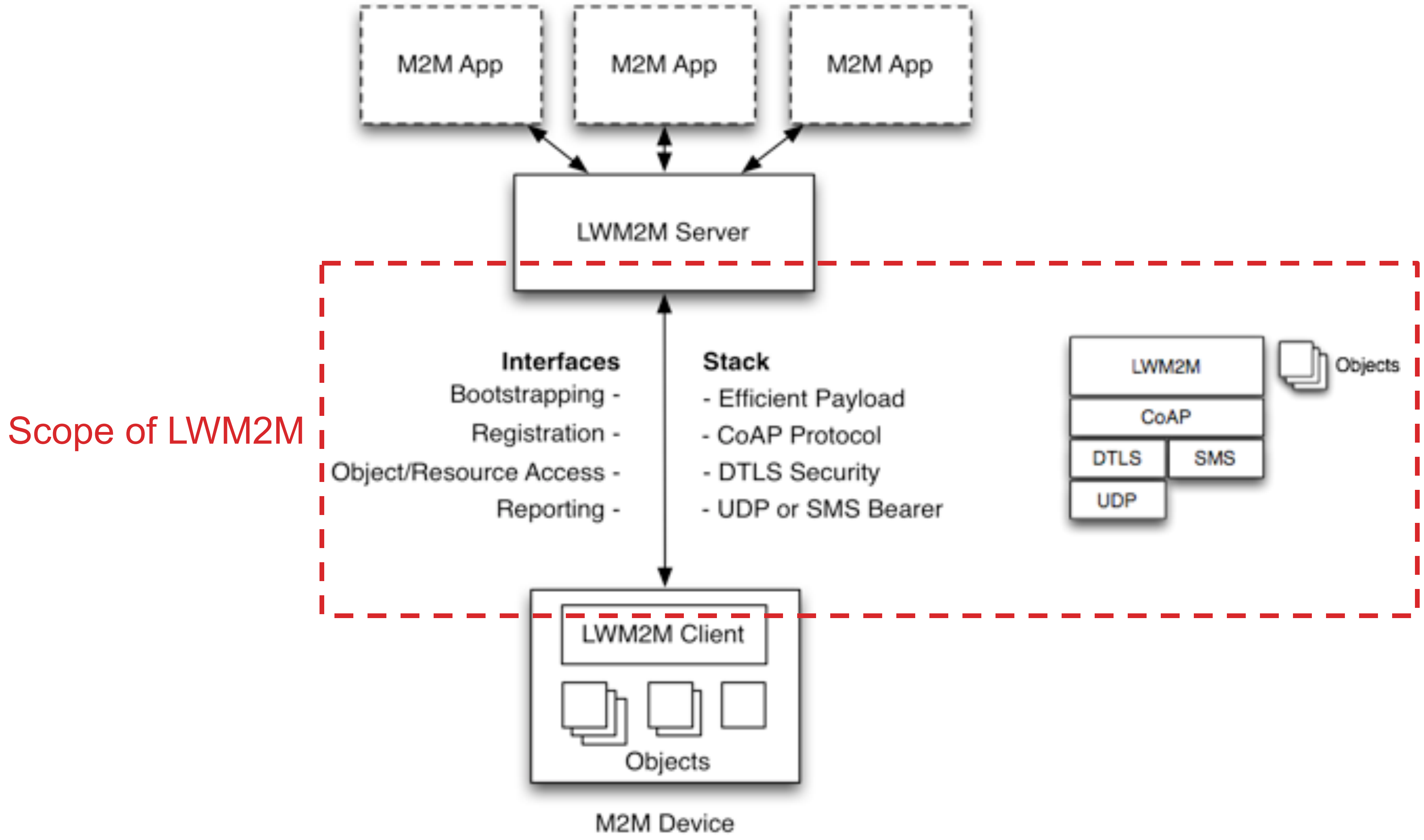
CoRE WG @ IETF-87 Berlin

OMA Lightweight M2M

- Open Mobile Alliance is well known for Device Management (DM)
- OMA Lightweight M2M is a new standard from the alliance
  - Focused on constrained Cellular and sensor network M2M devices
  - Driven by leading operators and vendors
- Scope
  - Interfaces, protocol & security between Device and Server
  - Object and Resource model
  - Bootstrap, Device, Access Control, Connectivity and Firmware Objects
- Draft Specifications are available:

  http://member.openmobilealliance.org/ftp/Public_documents/DM/LightweightM2M/Permanent_documents/OMA-TS-LightweightM2M-V1_0-20130717-D.zip
- Timeline
  - Consistency review completed June 2013
  - Candidate Approval expected 3Q/2013

Architecture



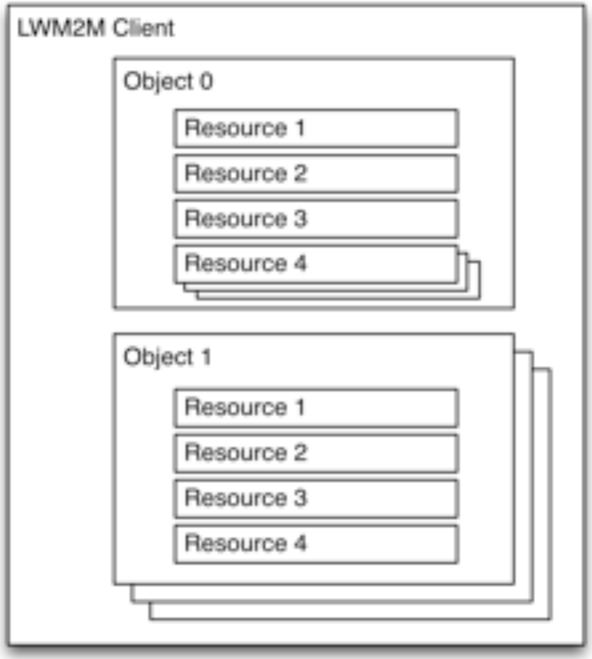
Object Model

- A Client has one or more Object Instances
- An Object is a collection of Resources
- A Resource is an atomic piece of information
  - Read, Written or Executed
- Resources can have multiple instances
- Objects, Resources and Instances are identified by a 16-bit Integer
- Objects/Resources are accessed with simple URIs:

```
/{Object ID}/{Object Instance}/{Resource ID}
e.g. /12/1/3
```

- Yes, we made the root configurable

```
e.g. /lwm2m/12/1/3
```



Announcing: Internet of Things Plugtest

- When?
  - November 20-22nd, 2013
- Where?
  - Las Vegas
- Who?
  - Implementers world-wide
- How Much?
  - Free!
- Tests:
  - CoAP (Mandatory)
  - Block, Observe (Optional)
  - OMA Lightweight M2M (Optional)

ipso Alliance

ETSI World Class Standards

oma Open Mobile Alliance

Advanced CoAP Congestion Control: Preliminary Results & Work in Progress

August Betzler, Carles Gomez, Ilker Demirkol, Josep Paradells – Universitat Politècnica de Catalunya

august.betzler@entel.upc.edu

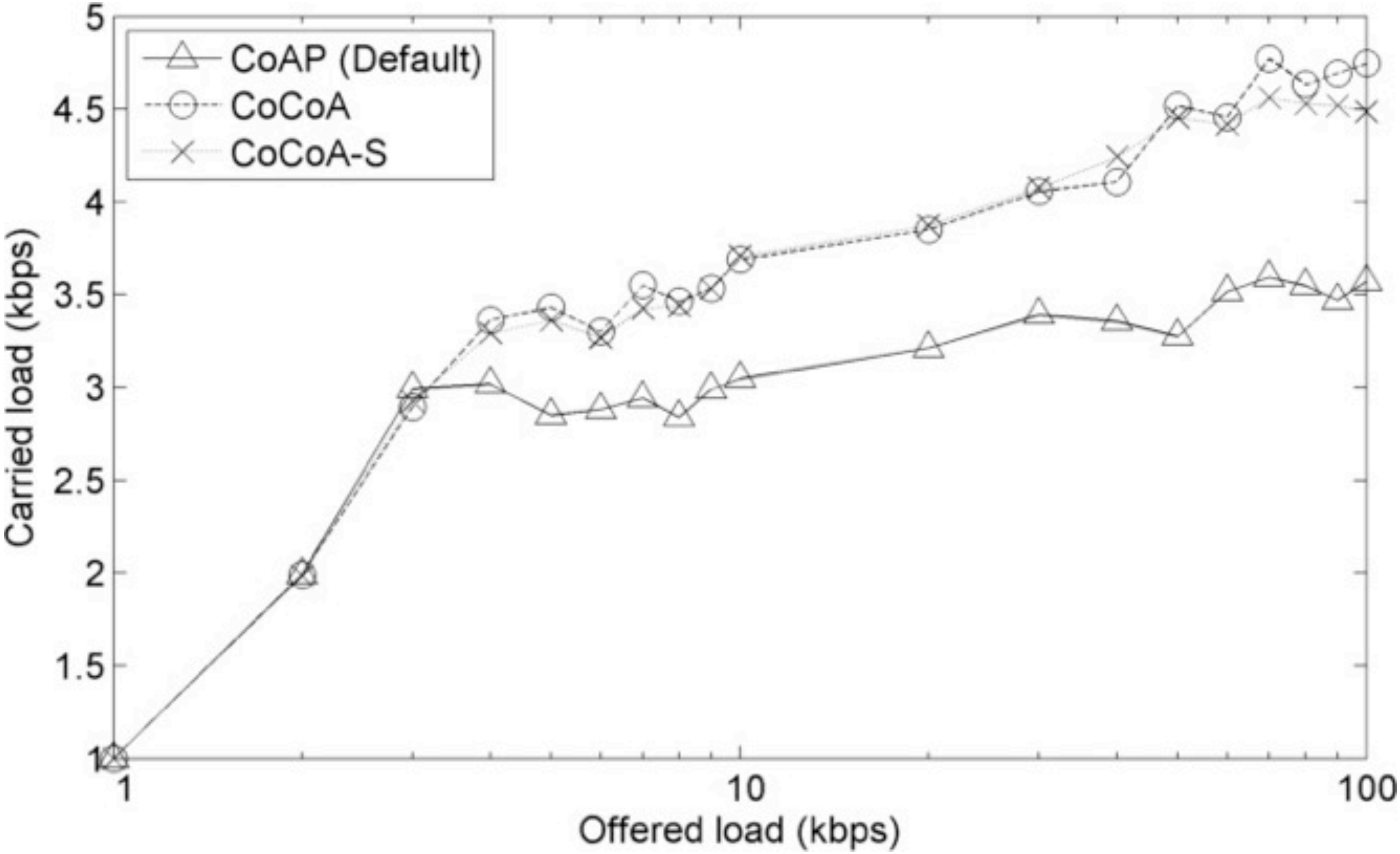
Carsten Bormann – Universität Bremen TZI

cabo@tzi.org

CoAP Congestion Control Principles

- Default CoAP chooses RTO for a transaction from a fixed interval:
  - RTO = [ACK_TIMEOUT, ACK_TIMEOUT*ACK_RANDOM_FACTOR]
  - Binary exponential backoff upon RTO expiration
- However, advanced congestion control algorithms may use RTT information to calculate the RTO adaptively.
  - E.g. CoCoA [draft-bormann-core-cocoa-00].

Simulation Results: Grid Topology



Betzler et al., “Congestion Control in Reliable CoAP Communication“, MSWIM 2013

Items of Work in Progress

- Weighting of weak/strong estimator
- Variable Backoff Factor after RTO expires:
  - Depending on current RTO estimate
- Dithering of RTO when using estimated RTO
- Should a minimum RTO below 1 s be allowed?
- NSTART=1 too conservative?
- Influence of MAC layer on performance
  - MAC Acknowledgements
- Benefits beyond congestion control :
  - Response Time

Group 2: WG docs

-block

-block-12

- **Solves #331 for core document (Size1/Size2 split)**
- **Still open:**
  - **#211 (provisional responses)**
  - **#253 (more flexible control of initiative)**
    - Do we really need to address these?
- **Still to do: editorial improvements**

- **WGLC2?**

Observing Resources in CoAP

draft-ietf-core-observe

IETF 87

Klaus Hartke

All tickets closed

… except for two tiny details:

- Cancellation
- Liveliness

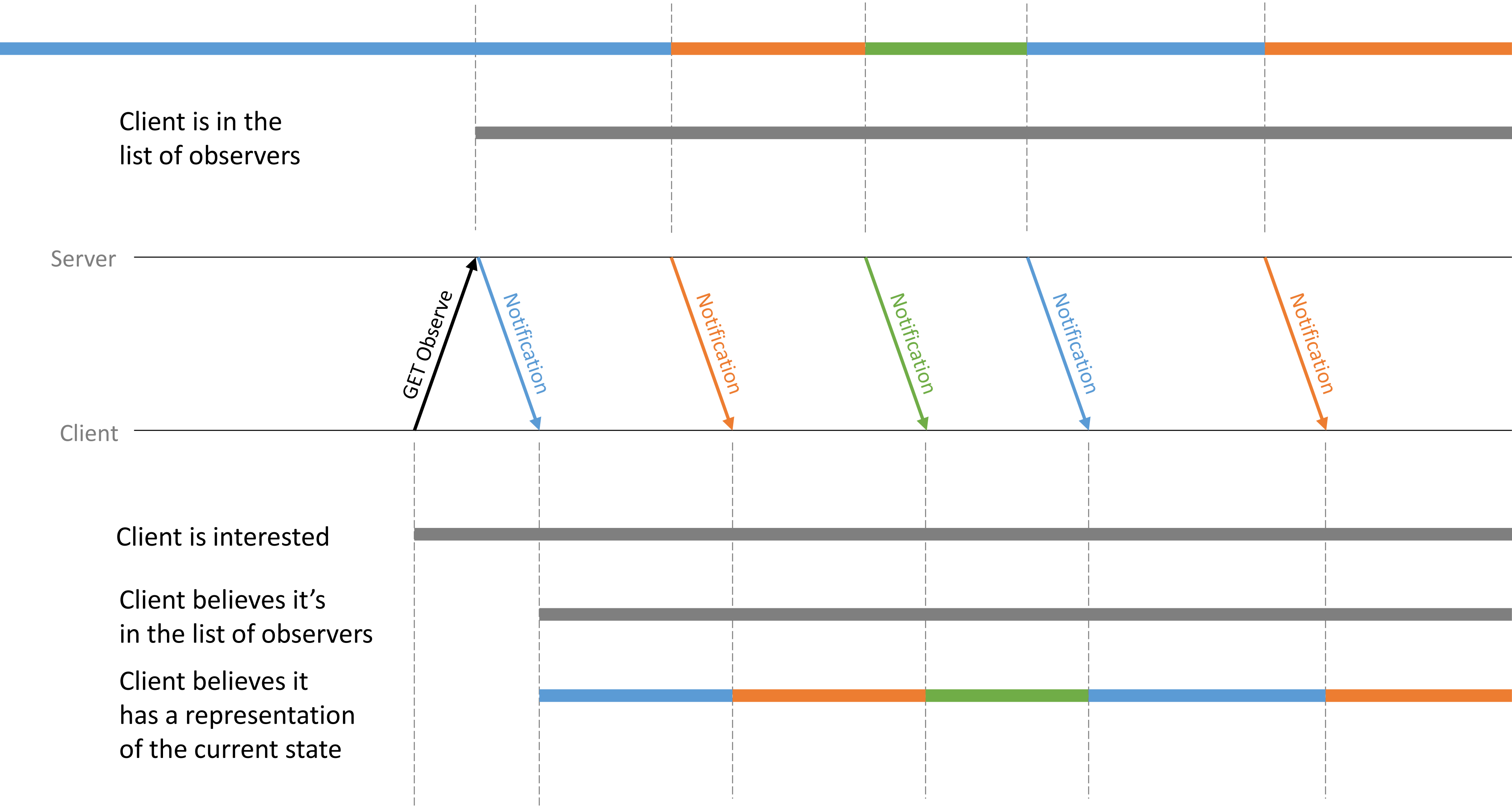
Cancellation

An entry is removed from the list of observers when

- the server sends a non-2.xx notification
- the server reboots and loses the state
- the client actively rejects a notification, or
  the last attempt to transmit a confirmable notification times out
  - “garbage collection”
- ~~the client makes a GET request to the resource~~

Do we need a way for a client to remove its entry eagerly?

Liveliness



Client is in the list of observers

Server

GET Observe

Notification

Notification

Notification

Notification

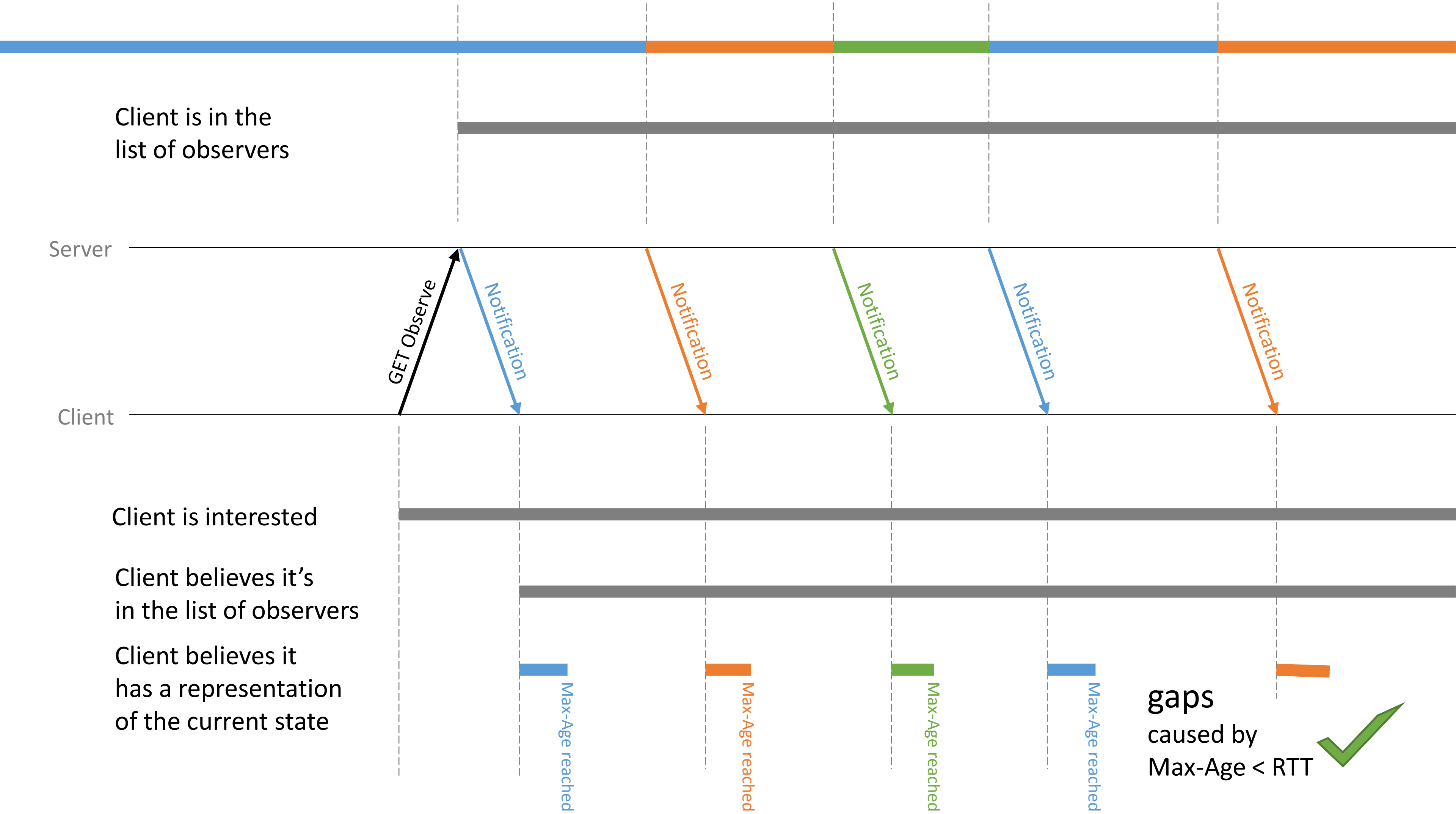
Notification

Client

Client is interested

Client believes it’s in the list of observers

Client believes it has a representation of the current state



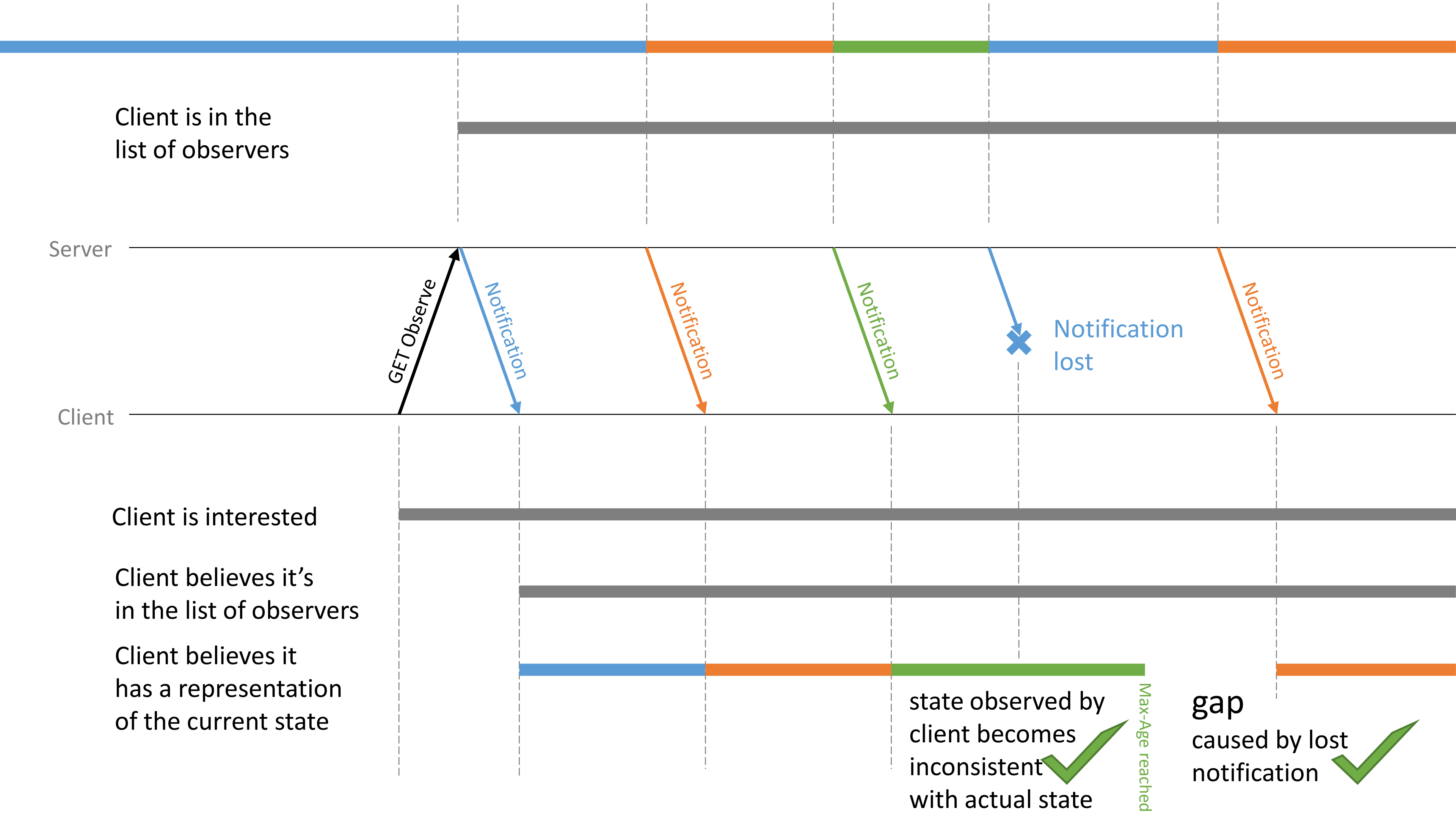
Client is in the
list of observers

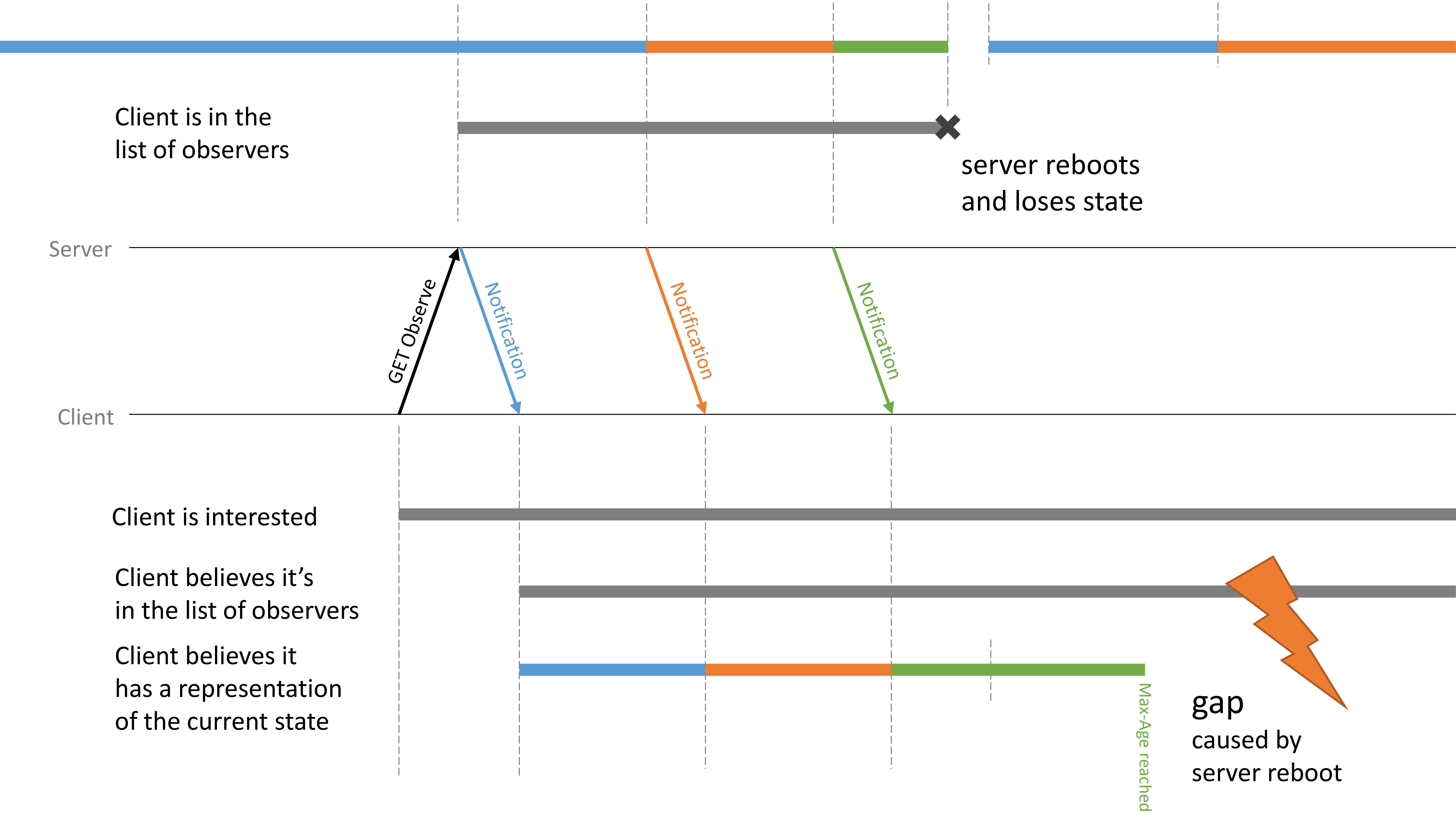
Client is interested

Client believes it’s
in the list of observers

Client believes it
has a representation
of the current state

gaps
caused by
Max-Age < RTT ✓


Client is in the list of observers
Server
Client
GET Observe
Notification
Notification
Notification
Notification lost
Notification
Client is interested
Client believes it's in the list of observers
Client believes it has a representation of the current state
state observed by client becomes inconsistent with actual state
Max-Age reached
gap
caused by lost notification



Client is in the
list of observers

server reboots
and loses state

Client is interested

Client believes it's
in the list of observers

Client believes it
has a representation
of the current state

gap
caused by
server reboot

Liveliness (ii)

- The client assumes that it is in the list of observers
- Incoming notifications confirm this assumption
- The absence of notifications requires the client to validate its assumption eventually
- There is no need to validate the assumption while the client still has a fresh representation
- The client cannot validate its assumption without communicating with the server
- The server could provide a hint when the client should validate, but it’s really up to the client to decide how long it is comfortable with potentially not being in the list of observers

Liveliness (iii)

Token reuse

- Client can re-register any time using the same token
- Server updates/replaces entry in the list when token is still there; Server sends representation to the client
- Server adds a new entry if the token is not in the list anymore; Server sends representation to the client

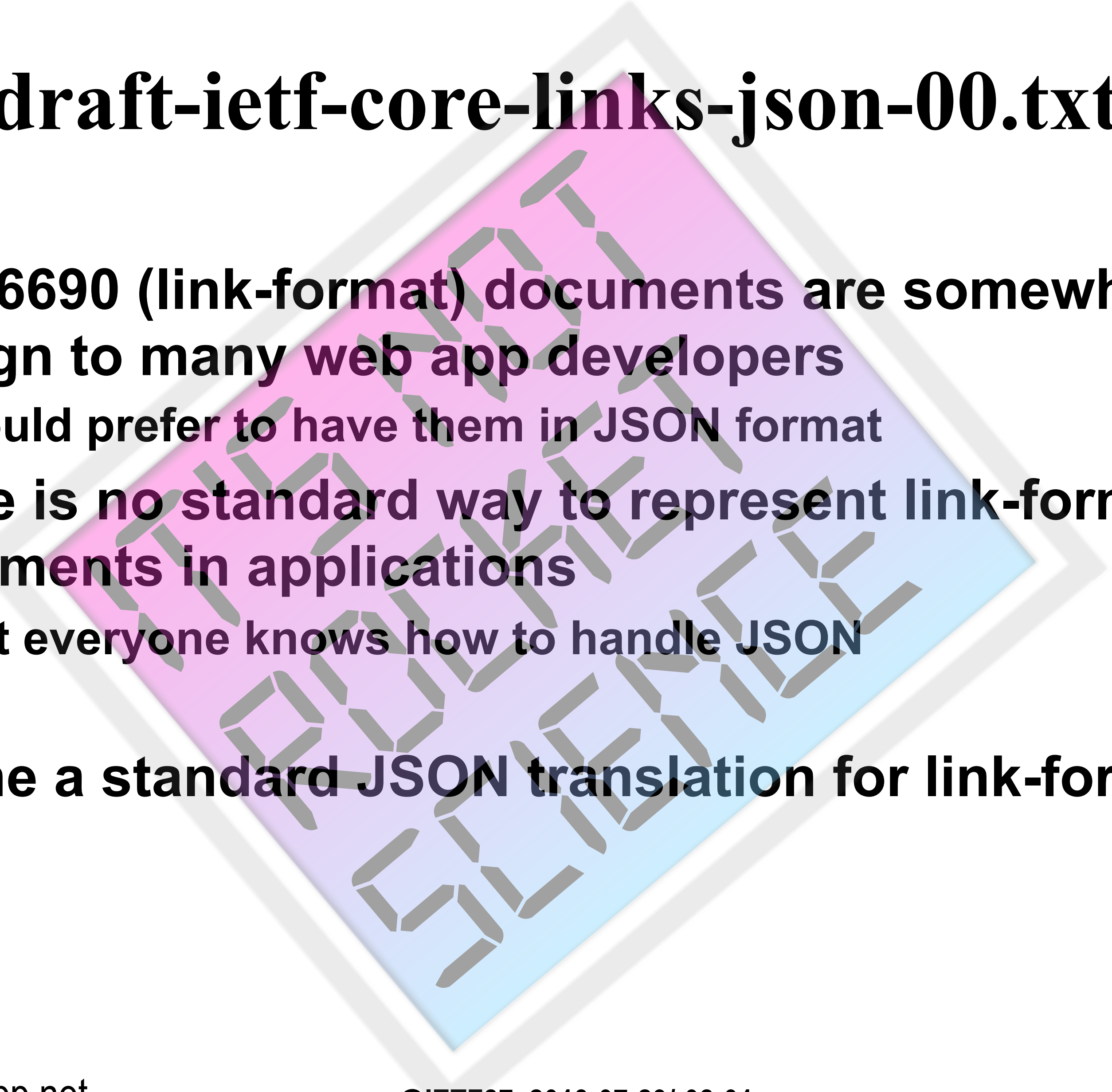
Ping/Pong

- Client can send a “ping” with the token of its original request
- Server sends a “positive pong” if the token is still in the list; Client needs to issue a GET if it needs a fresh representation
- Server sends a “negative pong” if the token is not in the list; Client needs to re-register if it’s still interested

draft-ietf-core-links-json-00.txt

- RFC 6690 (link-format) documents are somewhat foreign to many web app developers
  - would prefer to have them in JSON format
- There is no standard way to represent link-format documents in applications
  - but everyone knows how to handle JSON

→ Define a standard JSON translation for link-format



```
</sensors>;ct=40;title="Sensor Index",
</sensors/temp>;rt="temperature-c";if="sensor",
</sensors/light>;rt="light-lux";if="sensor",
<http://www.example.com/sensors/t123>
  ;anchor="/sensors/temp";rel="describedby",
</t>;anchor="/sensors/temp";rel="alternate"
```

➔

```
[{"href":"/sensors","ct":"40","title":"Sensor Index"},
  {"href":"/sensors/temp","rt":"temperature-c","if":"sensor"},
  {"href":"/sensors/light","rt":"light-lux","if":"sensor"},
  {"href":"http://www.example.com/sensors/t123",
   "anchor":"/sensors/temp","rel":"describedby"},
  {"href":"/t","anchor":"/sensors/temp","rel":"alternate"}]
```

Group 5:
Alternate Transports

The need for alternate transports

- Non-IP transports: e.g., SMS
- Alternate IP transport protocols: e.g., TCP, Websockets
- Combine non-IP and IP: e.g. SMS request, UDP response

Issues

- Encapsulation
  - Delimiting in stream transports (e.g., TCP)
  - Do we need CON/NON/ACK/RST for acknowledged transports?
  - Data transparency (Base64url…)
- URIs
  - Identifying the desired transport(s)
  - Enabling transport selection
  - Having multiple transports for the *same* resource
- Return Path
  - How to indicate the desired recipient for the response

Endpoint identification

- **UDP: IP-Address + Port**
  - **Generalize to Address + Port + Transport (for TCP…)**
- **SMS: MSISDN**
- **Websockets: WS (WSS) URI**

- **Use indirection? DNS/SRV? DHTs?**

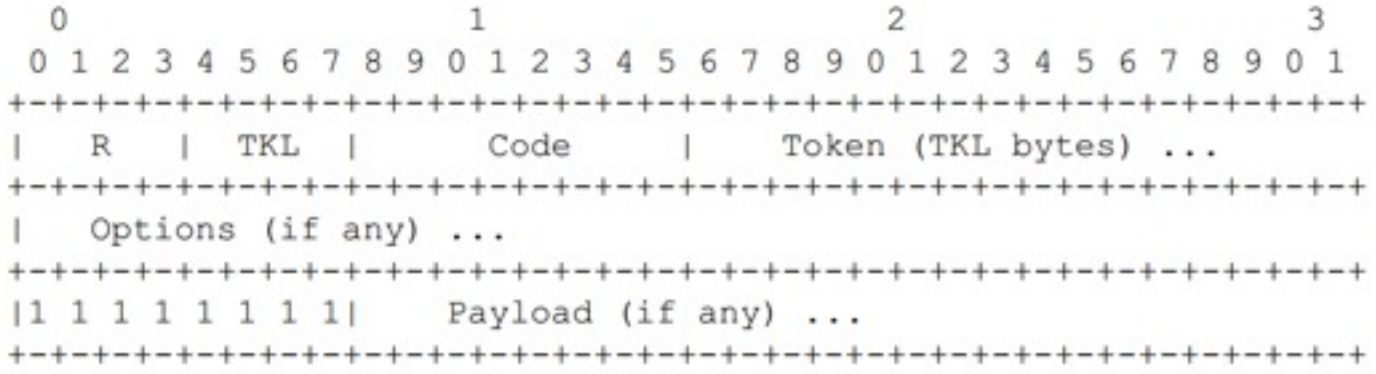
- **Do this in a way that can be used both in URIs and for Return-Path?**

Encapsulation questions 1/3

- Whether a (stream) transport needs delimiters to indicate start/end of a CoAP message
- Whether reliability and congestion control mechanisms of CoAP are required
- Whether explicit CoAP message length needs to be communicated
- Whether masking of CoAP message is required
  - *WebSocket requires masking of the data from Client to Server data*

Encapsulation questions 2/3

- Are all fields of the CoAP header always required
  - Protocol: if communicated already by used transport
  - Type: if reliability is provided by used transport
  - Message ID: if retransmissions and duplication handling are covered by used transport
- E.g. in case of WebSocket proposal, the CoAP message format proposal looks like below: version and type changed to reserved, message ID is elided.

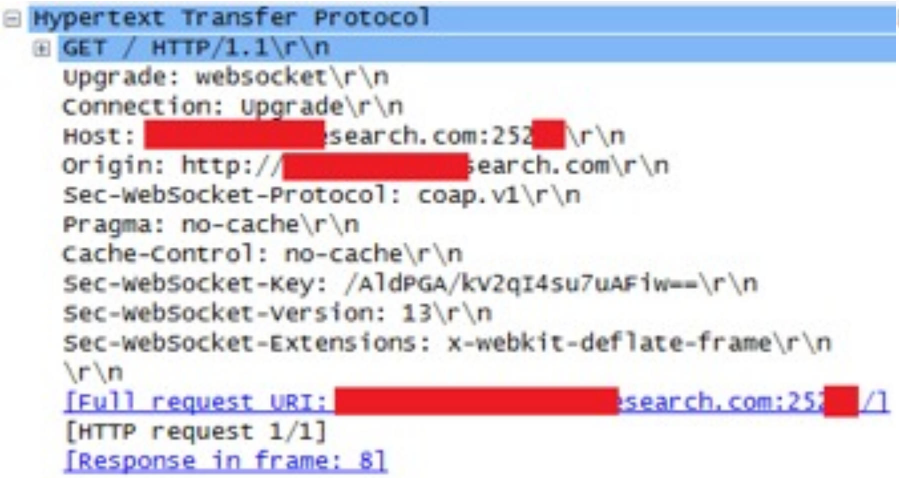


```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   R   |  TKL  |      Code     |    Token (TKL bytes) ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Options (if any) ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |1 1 1 1 1 1 1 1|    Payload (if any) ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- CoAP messages can be arbitrarily large on some transports (for UDP 1152 byte maximum message size is recommended) -> multiplexing of CoAP message to avoid head of line blocking may thus be required
  - CoAP blockwise transfer, or e.g. WebSocket multiplexing extensions (*draft-ietf-hybi-websocket-multiplexing-11*)

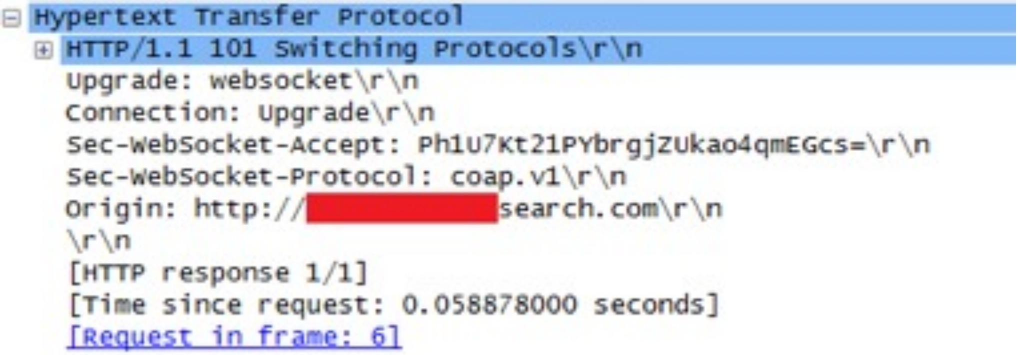
Encapsulation questions 3/3 – WebSocket example, using WebSocket framing on top of TCP

1) WebSocket handshake request



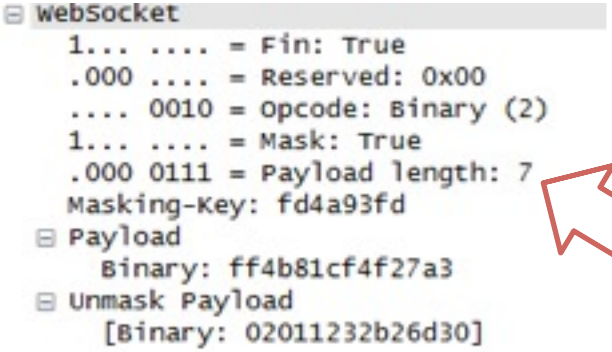
```
⊟ Hypertext Transfer Protocol
  ⊞ GET / HTTP/1.1\r\n
    Upgrade: websocket\r\n
    Connection: Upgrade\r\n
    Host: [redacted]search.com:252[redacted]\r\n
    Origin: http://[redacted]earch.com\r\n
    Sec-WebSocket-Protocol: coap.v1\r\n
    Pragma: no-cache\r\n
    Cache-Control: no-cache\r\n
    Sec-WebSocket-Key: /AldPGA/kV2qI4su7uAFiw==\r\n
    Sec-WebSocket-Version: 13\r\n
    Sec-WebSocket-Extensions: x-webkit-deflate-frame\r\n
    \r\n
    [Full request URI: [redacted]search.com:25[redacted]/]
    [HTTP request 1/1]
    [Response in frame: 8]
```

2) WebSocket handshake reply



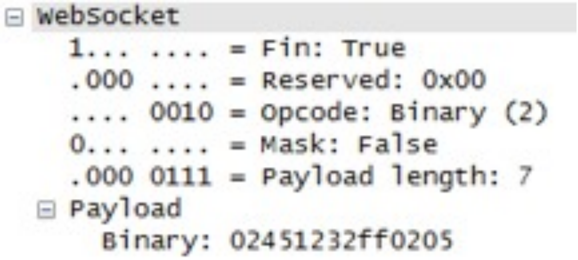
```
⊟ Hypertext Transfer Protocol
  ⊞ HTTP/1.1 101 Switching Protocols\r\n
    Upgrade: websocket\r\n
    Connection: Upgrade\r\n
    Sec-WebSocket-Accept: Ph1U7Kt21PYbrgjZUkao4qmEGcs=\r\n
    Sec-WebSocket-Protocol: coap.v1\r\n
    Origin: http://[redacted]search.com\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.058878000 seconds]
    [Request in frame: 6]
```

3) CoAP GET



```
⊟ WebSocket
    1... .... = Fin: True
    .000 .... = Reserved: 0x00
    .... 0010 = Opcode: Binary (2)
    1... .... = Mask: True
    .000 0111 = Payload length: 7
    Masking-Key: fd4a93fd
  ⊟ Payload
      Binary: ff4b81cf4f27a3
  ⊟ Unmask Payload
      [Binary: 02011232b26d30]
```

4) CoAP Response 2.05



```
⊟ WebSocket
    1... .... = Fin: True
    .000 .... = Reserved: 0x00
    .... 0010 = Opcode: Binary (2)
    0... .... = Mask: False
    .000 0111 = Payload length: 7
  ⊟ Payload
      Binary: 02451232ff0205
```

Early prototype code by
Nadir Javed & Teemu Savolainen

ASCII Encoding for CoAP : CoAP/A

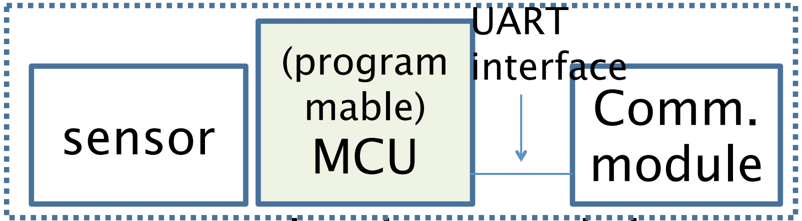
Softgear Ko and Ilkyun Park

softgear@etri.re.kr

CORE WG, IETF 87, Berlin

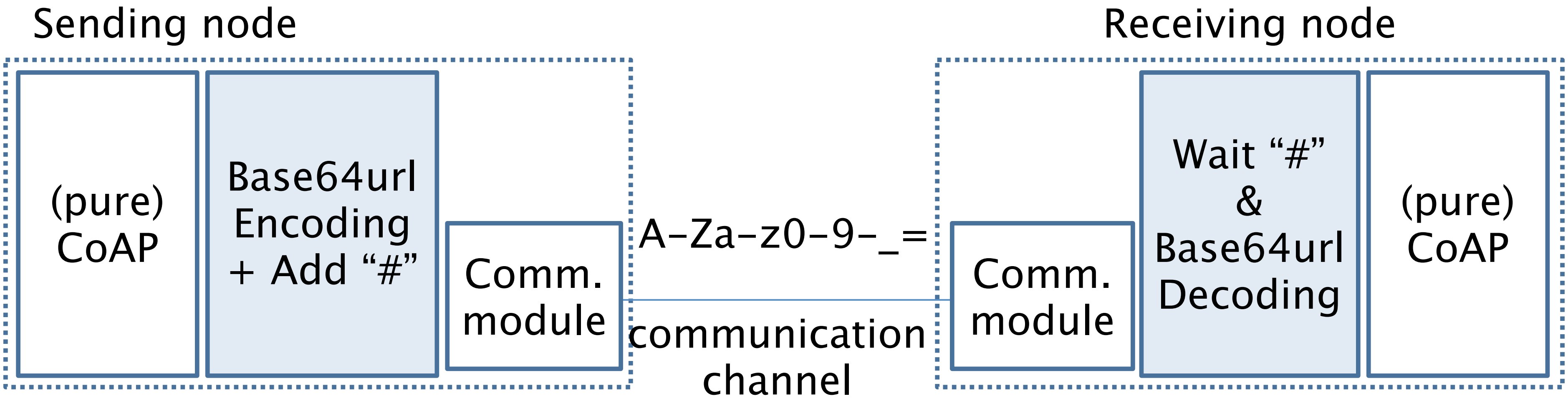
Background

- We want to use CoAP over serial communication (RS232, RS485, IEEE 802.15.3, UART over WiFi, bluetooth, ethernet) at Sensor node
  - Using commercial communication modules for fast development and cheap



- Some communication modules support transparent mode which emulate serial communication and allow ASCII character only
  - They do not allow control characters
  - They may use "+++" pattern to enter configuration mode

Processing Sequence



For ASCII only communication module,

CoAP protocol stack passes binary encoded messages to Base64 Encoder.

Base64 Encoder translates binary message to ASCII message using Base64url character set. And, add “#” delimiter to mark the end of a message.

Now, ASCII encoded messages are transferred to Receiver node

Base64 Decoder module in Receiver node collect characters from sender until “#” mark. And Decoder translates them to binary coded CoAP message.

And pass it to pure CoAP protocol stack.

Example

Original binary encoded CoAP GET Request:

40 01 7d 34 bb 74 65 6d 70 65 72 61 74 75 72 65

33% overhead

ASCII encoded CoAP Request:

QAF9NLt0ZW1wZXJhdHVyZQ==#

Original binary encoded CoAP Response:

60 45 7d 34 ff 32 32 2e 33 43

ASCII encoded CoAP Response:

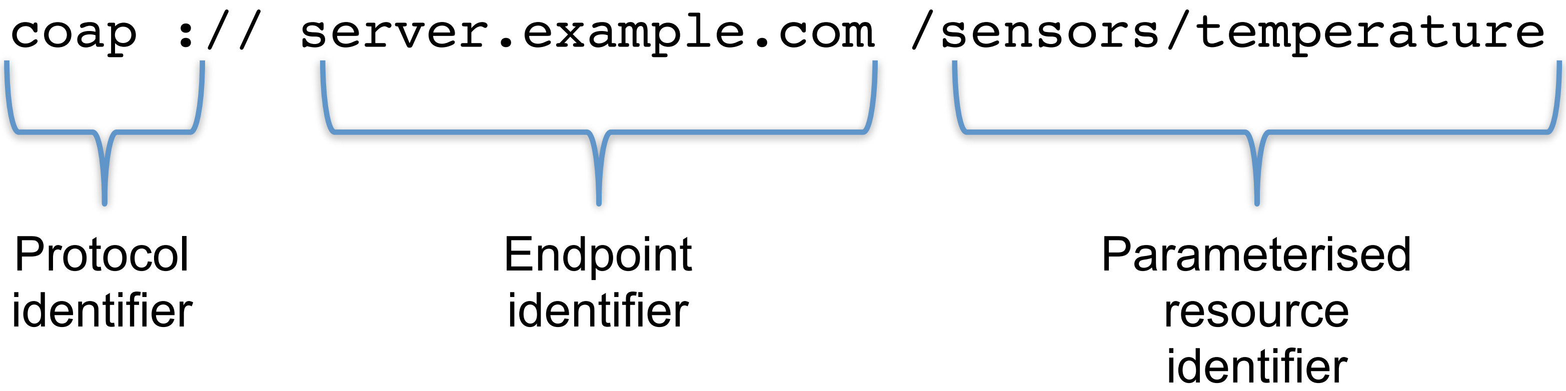
YEV9NP8yMi4zIEM=#

draft-bormann-core-coap-tcp-00.txt

- Discusses TCP alternatives
- Length-based header (16-bit or SDNV)
  - ➔ Trivial, obvious, efficient, ...
- Delimiter-based (MINION!)
  - Allows processing TCP packets out of order
- Allowing CoAP messages to be self-delimiting
  - encoding payload length into 0xFX payload marker
  - keep 0xFF payload marker as "up to total length"
  - ➔ General (not just TCP, but also e.g. SMS aggregation)

CoAP URI: Transport Representations

URI = scheme ":" "//" authority path-abempty ["?"query]



`coap :// server.example.com /sensors/temperature`

Protocol identifier

Endpoint identifier

Parameterised resource identifier

At the Orlando meeting, 3 ways proposed:

- Within the scheme name
- In the URI path
- As a query component

✓ **Some expressed preference for using the scheme name but we need to revisit this discussion now to go forward**

URI Representations for CoAP

Means of expressing transport types

- Within the scheme name
  - coap+tel://+15105550101/sensors/temperature
  - coap+tcp://example.com/sensors/temperature
- In the URI path
  - coap://host.example.com;transport=tcp/.well-known/core?rt=core-rd
  - Diameter Protocol (RFC 6733) does this already
    - aaa://host.example.com:6666;transport=tcp;protocol=diameter
    - aaa://host.example.com:1813;transport=udp;protocol=radius

URI Representations for CoAP

Means of expressing transport types

- Use a new CoAP URI

  scheme : protocol : endpointidentifier : coapresource

  Examples:

  ○ coap-alt:tcp:example.org:coap/sensors/temperature

  ○ coap-alt:tel:+123456789:coap/sensors/temperature

- This is similar to the SLP's *service:* scheme (RFC 2609)

  service: URL = "service:" service-type ":" site url-path

  Example:

  ○ service:printer:ipp://print-server.example.org/printqueue

Return-Path

Transport of CoAP over SMS, USSD and GPRS
draft-becker-core-coap-sms-gprs-03

Markus Becker, Kepeng Li,
Koojana Kuladinithi, Thomas Pötsch

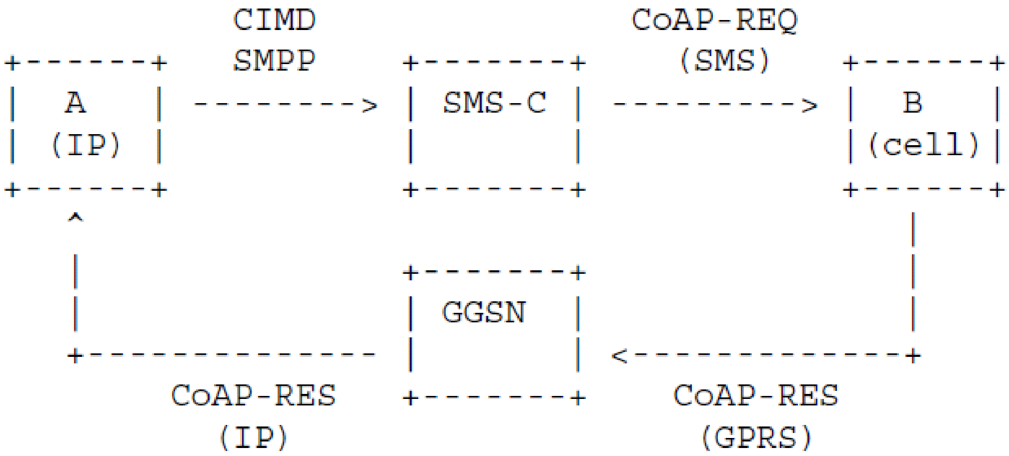
CoRE WG, IETF-87, Berlin

Motivation

- In M2M communication, IP connectivity is not **always** supported by the constrained end-points
    - Power saving
    - Coverage (GPRS, 3G, LTE)
- SMS based communication is almost **always** supported
- OMA uses SMS as an alternative transport in OMA-TS-LightweightM2M

Return Path Question

- Endpoint may have different transports
- Forward and return paths use different transports



- Use Response-To-Uri-Host and Response-To-Uri-Port options?

No	C	U	N	R	Name	Format	Length	Default
34					Response-To-Uri-Host	string	1-270 B	(none)
38					Response-To-Uri-Port	uint	0-2 B	5683

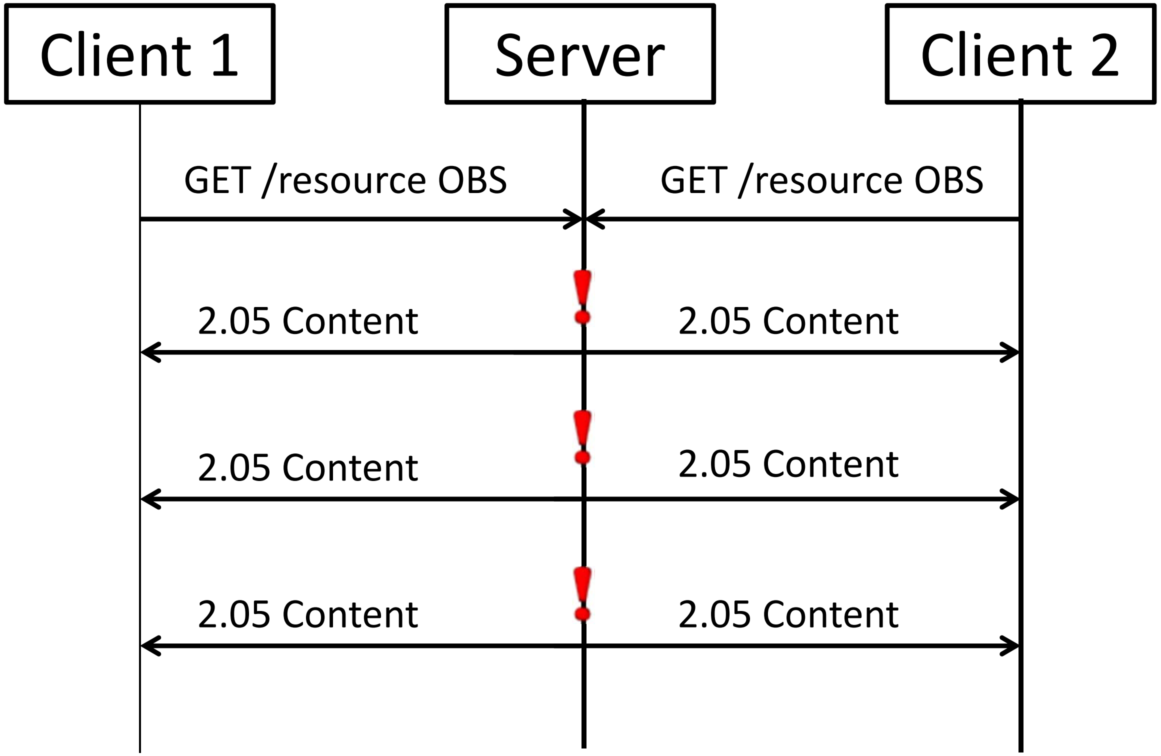
Group 3: “new work” (continued)

Conditional observe in CoAP

draft-li-core-conditional-observe-04.txt
Shitao li
Jeroen Hoebeke
Antonio J. Jara
Floris Van den Abeele

http://tools.ietf.org/html/draft-li-core-conditional-observe-04

Why?



Some observers aren’t interested in receiving ALL representations of a resource.

How to limit the set of representations that are transferred to the client?

How?

- Transfer all representations and perform client-side filtering (plain observe)
- Deploy separate resource per set of representations on server
- Clients signal set of representations in request. Two proposals thusfar:
  - Signaling through URI query parameters
  - Signaling through new CoAP option
  - Use method foo?

Our proposal

- Add Condition option to Observe request:

```
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   TYPE  |R| V |      VAL      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Condition types identified by integer
- Discovering supported condition types via web linking attribute
- ContikiOS implementation available

Discussion

- Do you feel this is actually a problem?
- If so, does this need <a standard solution>?
- Do you see problems with our proposal?
- Any other remarks?

Enhanced Sleepy Node Support for CoAP

Akbar Rahman

IETF 87, July 2013

http://www.ietf.org/id/draft-rahman-core-sleepy-03.txt

Introduction

- It is expected that in CoAP networks there will be a certain portion of devices that are "sleepy" and which may occasionally go into a sleep mode (i.e. go into a low power state to conserve power) and subsequently have reduced CoAP protocol communication ability

- This I-D proposes a minimal and efficient mechanism building on the Resource Directory concept (which can be integrated into a CoAP Proxy) to enhance sleepy node support in CoAP networks

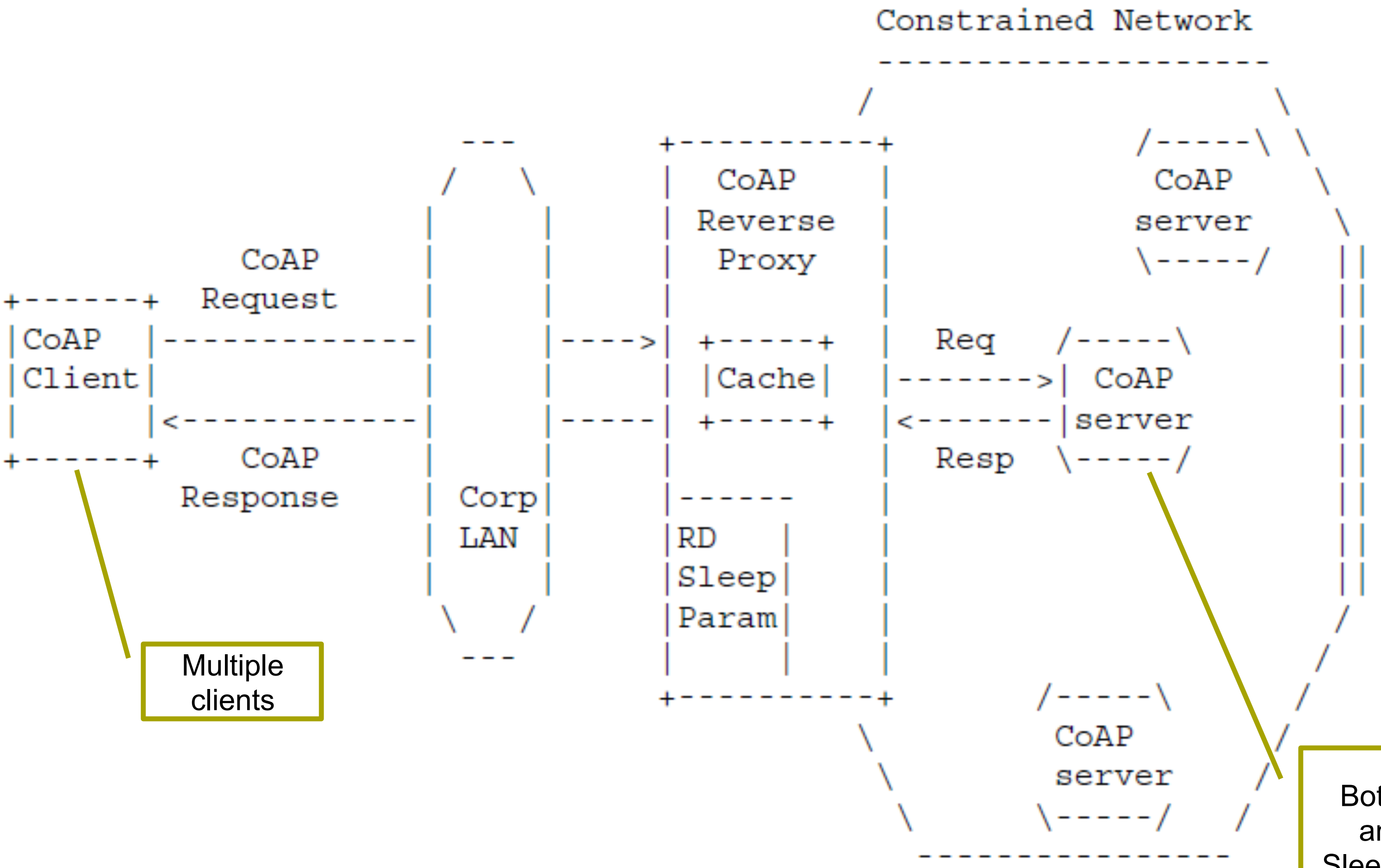
Main Question from IETF-86 (Orlando)

- (In Orlando, the WG first reviewed the performance analysis showing that the proposed Sleepy Node support solution gave consistent performance improvements, in many scenarios, over a network that just has standard CoAP caching enabled)

- How does the performance of the proposed Sleepy Node support compare with standard CoAP OBSERVE [I-D.ietf-core-observe]?

Sleepy Node Performance Analysis

Experimental Network



```
                                            Constrained Network
                                            --------------------
                                           /                    \
                       ---       +----------+            /-----\ \
                      /   \      |  CoAP    |             CoAP    \
                     |     |     | Reverse  |            server    \
            CoAP     |     |     |  Proxy   |            \-----/   ||
+------+  Request    |     |     |          |                      ||
|CoAP  |-------------|     |---->| +-----+  |  Req   /-----\       ||
|Client|             |     |     | |Cache|  |-------->| CoAP       ||
|      |<------------|     |-----| +-----+  |<--------|server      ||
+------+    CoAP     |     |     |          |  Resp  \-----/       ||
         Response    | Corp|     |------    |                      ||
                     | LAN |     |RD   |    |                      ||
                     |     |     |Sleep|    |                      ||
                      \   /      |Param|    |                      /
                       ---       |     |    |                     /
                                 +----------+       /-----\      /
                                           \         CoAP       /
                                            \        server    /
                                             \      \-----/   /
                                              ----------------
```

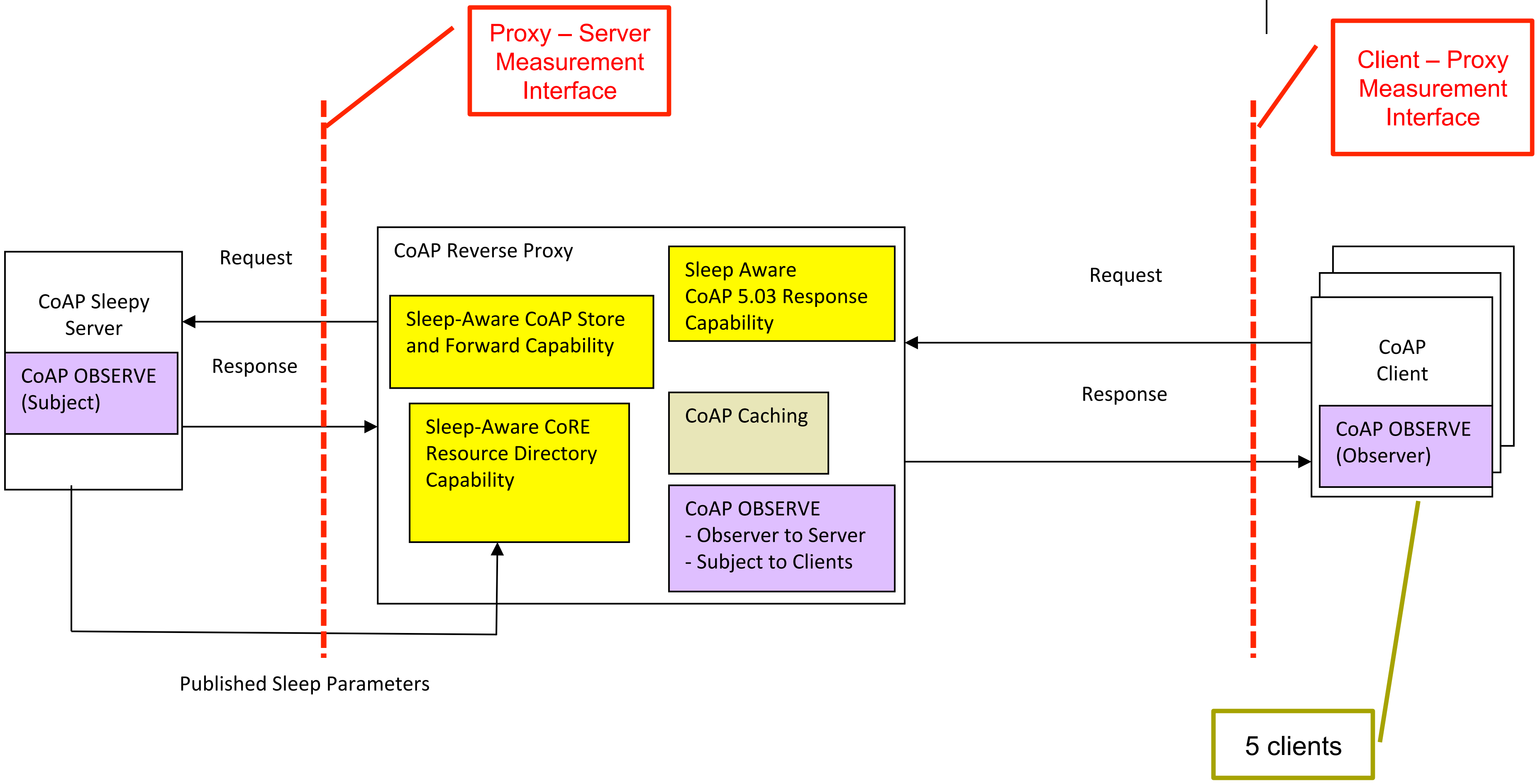
Multiple clients

Both Sleepy and Non-Sleepy servers

Sleepy Node Performance Analysis Network Setup (1/2)

- Sleepy Node support:
  - Server supports publishing sleep parameters to (RD based) proxy
    - E.g. I’m going to sleep, I’m waking up, I’m going to be asleep for *X* seconds, etc
  - Proxy supports sleep-awareness capabilities
  - Protocol flow follows approach of Fig. 1 (Synchronous RD Based Sleep Tracking) of I-D
- Proxy supports standard CoAP caching capability (based on maxAge)
- Proxy supports standard CoAP OBSERVING resources (on the Server) on behalf of the Clients

Sleepy Node Performance Analysis Network Setup (2/2)


Proxy – Server Measurement Interface
Client – Proxy Measurement Interface
CoAP Sleepy Server
CoAP OBSERVE (Subject)
Request
Response
CoAP Reverse Proxy
Sleep-Aware CoAP Store and Forward Capability
Sleep Aware CoAP 5.03 Response Capability
Sleep-Aware CoRE Resource Directory Capability
CoAP Caching
CoAP OBSERVE
- Observer to Server
- Subject to Clients
Request
Response
CoAP Client
CoAP OBSERVE (Observer)
Published Sleep Parameters
5 clients

** Proxy capabilities can be selectively enabled/disabled*

Reverse Proxy Features for Sleepy Node Support

- CoRE Sleep-aware Resource Directory
  - Support storing published sleep parameters from CoAP servers
    - sleepState (AWAKE, ASLEEP)
    - sleepDuration

- Sleep-aware CoAP 5.03 Response Capability
  - If CoAP request to a sleeping server is received (and there is no valid cache for that request), proxy returns a '5.03 Retry-After' response to client
    - 5.03 contains a timestamp indicating when the sever will wake back up (timestamp delivered in CoAP maxAge option)

- Sleep-aware CoAP Store-and-Forward Capability
  - If CoAP request to a sleeping server is received (and there is no valid cache for that request), proxy stores request until server wakes up and then forwards it

Other Reverse Proxy Features

- Caching capability
  - Cache GET responses from server (if maxAge option is present)
- OBSERVE capability
  - Proxy will act as Observer in relation to Server
  - Proxy will act as Subject in relation to Client
    - The Proxy will establish only one OBSERVE to Server for a given resource (even if it is requested to do so by multiple clients)
    - As per section 5 (Intermediaries) of [I-D.ietf-core-observe]

Goals of Performance Analysis

- For networks having sleepy servers, provide measurements to quantify the impact that CoAP sleep-awareness capabilities can have

- See if sleep-awareness capabilities can provide additional benefits with respect to:
  - CoAP caching (based on maxAge)
  - CoAP OBSERVE

Test Settings

Settings Applicable to All Test Scenarios	
# of Servers	1
# of Proxies	1
# of Clients	5
# Requests issued per Client	100
Time period between Client Requests	60 seconds
Sleep Pattern of Sleepy Server	Goto sleep for 5 minutes / Wake up for 30 seconds
Max-Age of Server Responses	120 seconds
Time period between OBSERVE Notifications due to state change in Server	5 minutes

Test Scenarios

Test Scenario	Type of Requests Issued by Client	Server Sleeps?	Proxy Sleep Aware Features Enabled?
1	GET	No	No
2	GET	Yes	No
3	GET	Yes	Yes
4	OBSERVE	No	No
5	OBSERVE	Yes	No
6	OBSERVE	Yes	Yes
7	PUT	No	No
8	PUT	Yes	No
9	PUT	Yes	Yes

Format of Results

The following results were collected for each test scenario

- Breakdown of the # and types of transactions on each of the proxy's interfaces:
  - Interface between clients and proxy
  - Interface between proxy and sleepy server
- Results are shown in tabular and bar chart formats

Summary of Results – Client/Proxy Interface

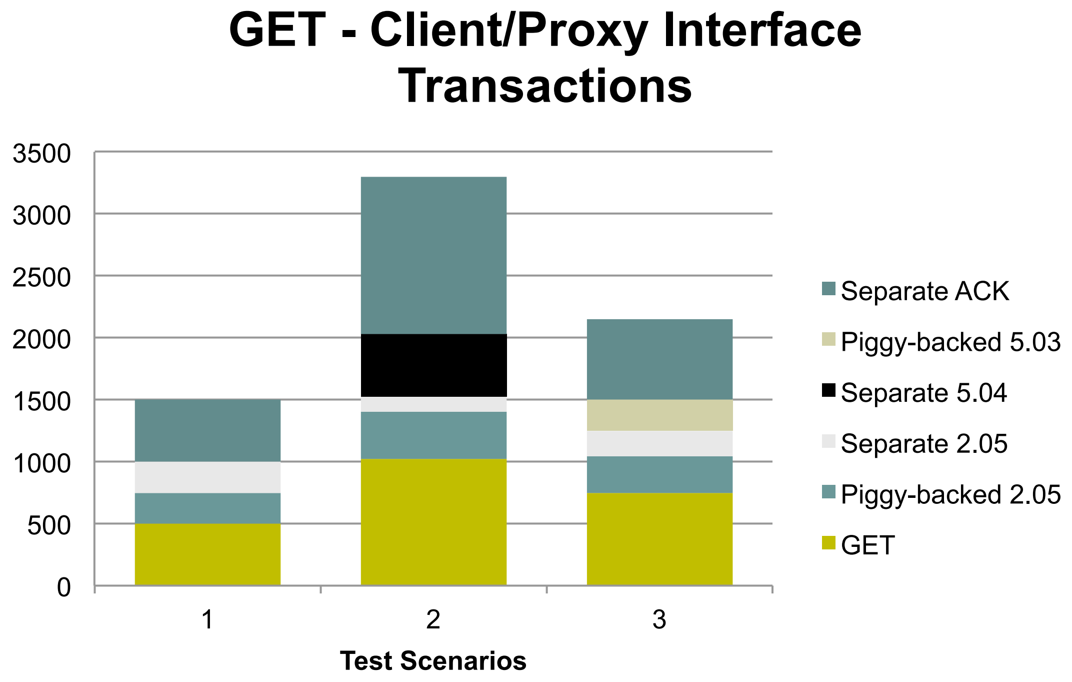
Client-Proxy Interface Transaction Counts									
Abbrev.	Test Scenarios								
	GET			OBSERVE (GET)			PUT		
	1	2	3	4	5	6	7	8	9
GET	500	1020	750	5	125	10	0	0	0
Piggy-backed 2.05	250	380	290	0	0	0	0	0	0
Separate 2.05	250	120	210	500	500	500	0	0	0
PUT	0	0	0	0	0	0	500	2330	1000
Separate 2.04	0	0	0	0	0	0	500	500	500
Separate 5.04	0	510	0	0	100	0	0	1770	0
Piggy-backed 5.03	0	0	250	0	0	25	0	0	500
Separate ACK	500	1270	650	505	725	535	1000	4600	1000
Totals:	1500	3300	2150	1010	1450	1070	2000	9200	3000

Summary of Results – Proxy/Server Interface

Proxy-Server Interface Transaction Counts									
Abbrev.	Test Scenarios								
	GET			OBSERVE (GET)			PUT		
	1	2	3	4	5	6	7	8	9
GET	60	3540	220	1	25	1	0	0	0
Piggy-backed 2.05	60	120	210	1	1	1	0	0	0
Separate 2.05	0	0	0	95	95	95	0	0	0
PUT	0	0	0	0	0	0	500	12395	500
Piggy-backed 2.04	0	0	0	0	0	0	500	500	500
POST	0	0	1	0	0	1	0	0	1
Piggybacked 2.01	0	0	1	0	0	1	0	0	1
Separate ACK	0	0	0	19	19	19	0	0	0
Totals:	120	3660	432	116	140	118	1000	12895	1002

GET – Test Scenario Results

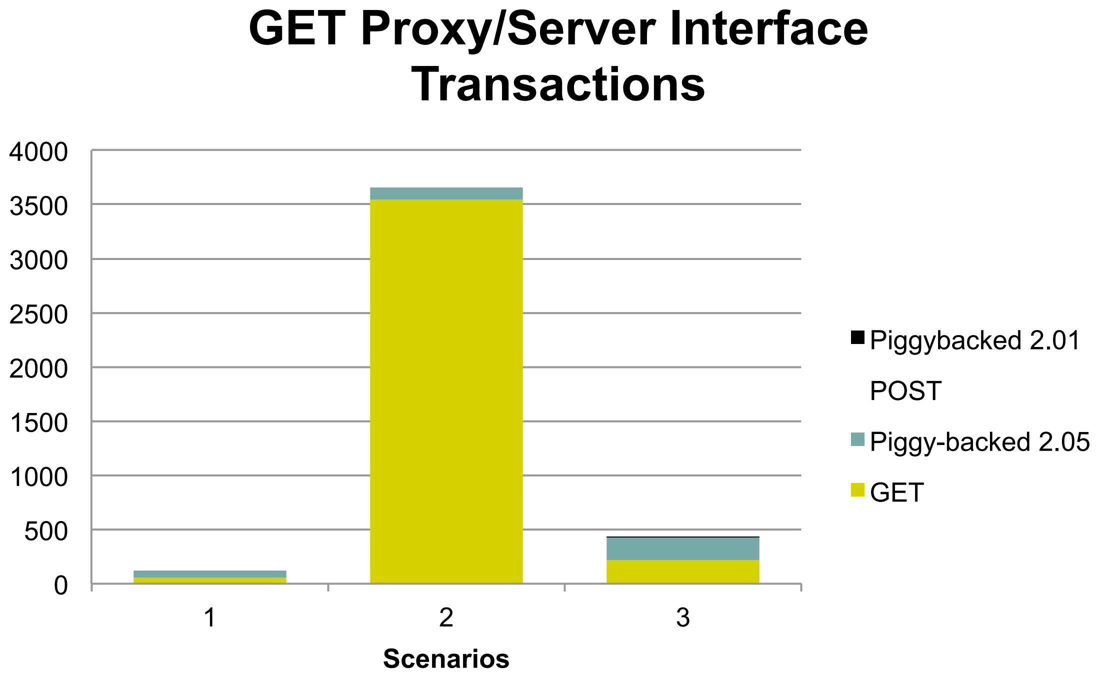
GET – Client/Proxy Interface Transaction Mix



Test Scenarios:
1 - Non-sleepy server, non-sleep aware proxy, 5 clients performing GETs
2- Sleepy server, non sleep aware proxy, 5 clients performing GETs
3 - Sleepy server, sleep aware proxy, 5 clients performing GETs

→ Sleep Aware Proxy can help reduce number of GETs, 5.04s, ACKs between client and proxy by leveraging 5.03 (Retry After time X)

GET – Proxy/Server Interface Transaction Mix



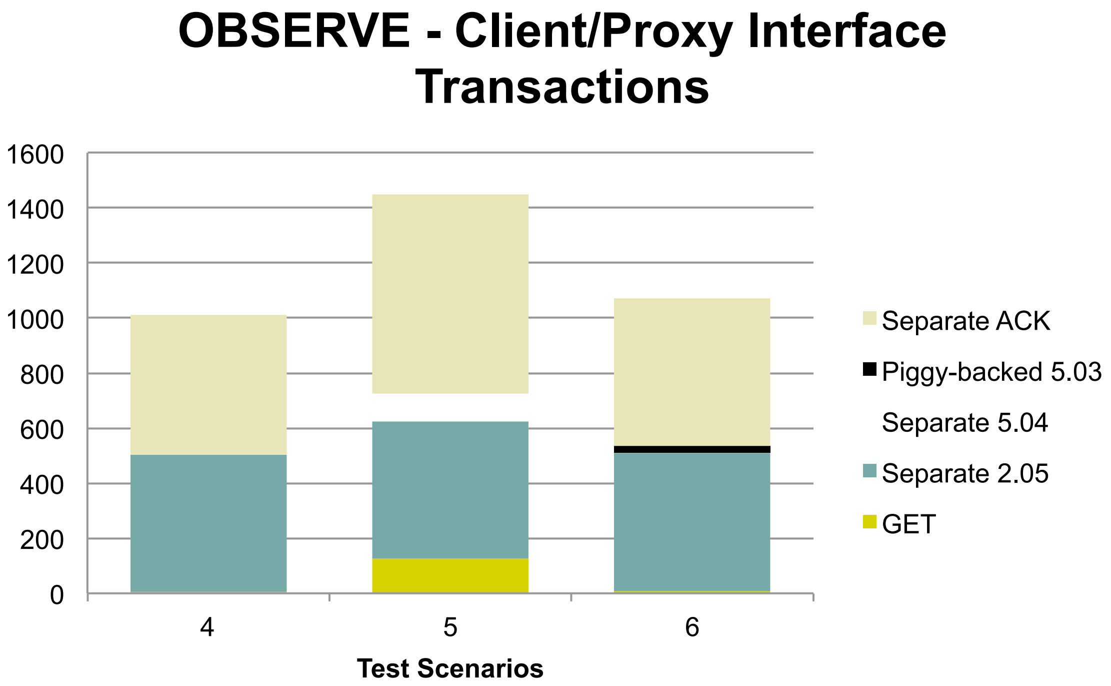
Test Scenarios:

1 - Non-sleepy server, non-sleep aware proxy, 5 clients performing GETs
2- Sleepy server, non sleep aware proxy, 5 clients performing GETs
3 - Sleepy server, sleep aware proxy, 5 clients performing GETs

→ Sleep Aware Proxy can greatly reduce number of GET transactions issued from proxy to sleepy server

OBSERVE – Test Scenario Results

OBSERVE– Client/Proxy Interface Transaction Mix



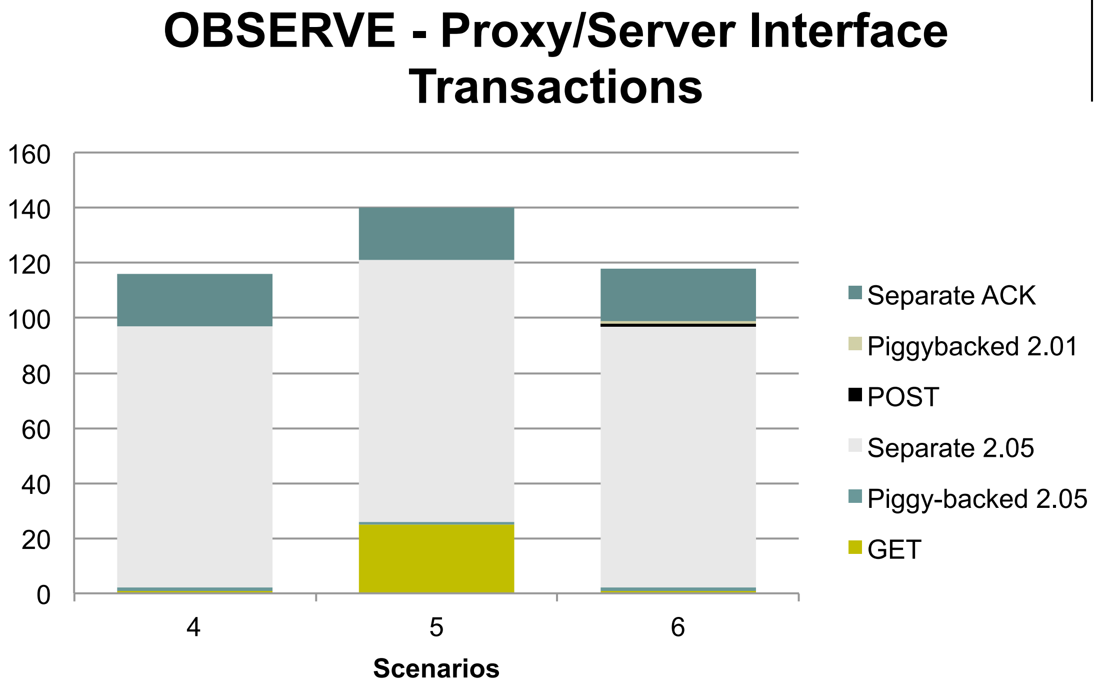
Test Scenarios:

4 - Non-sleepy server, non-sleep aware proxy, 5 clients performing Observes
5- Sleepy server, non sleep aware proxy, 5 clients performing Observes
6 - Sleepy server, sleep aware proxy, 5 clients performing Observes

→ OBSERVE (alone) will obviously have better performance then Sleep Aware Proxy (alone) (i.e. Scenario 3 vs. Scenario 5)

→ But OBSERVE in combination with Sleep Aware Proxy can help minimize number of transactions

OBSERVE- Proxy/Server Interface Transaction Mix



Test Scenarios:

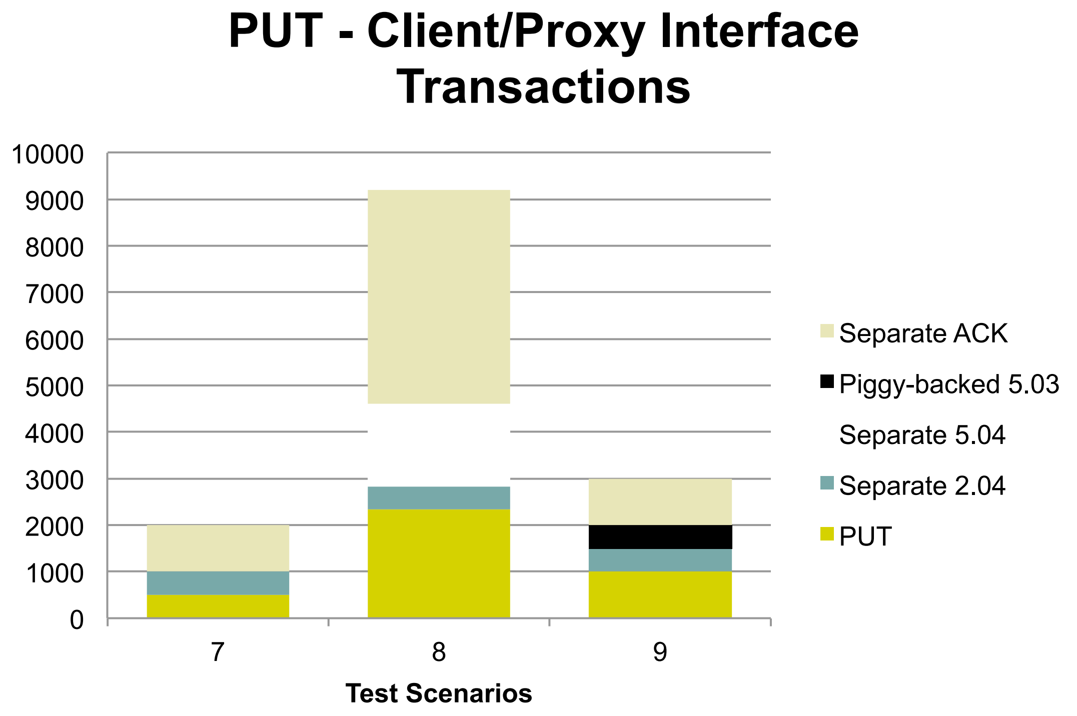
4 - Non-sleepy server, non-sleep aware proxy, 5 clients performing Observes
5- Sleepy server, non sleep aware proxy, 5 clients performing Observes
6 - Sleepy server, sleep aware proxy, 5 clients performing Observes

→ OBSERVE (alone) will obviously have better performance then Sleep Aware Proxy (alone) (i.e. Scenario 3 vs. Scenario 5)

→ But OBSERVE in combination with Sleep Aware Proxy can help minimize number of transactions

“PUT” – Test Scenario Results

PUT – Client/Proxy Interface Transaction Mix

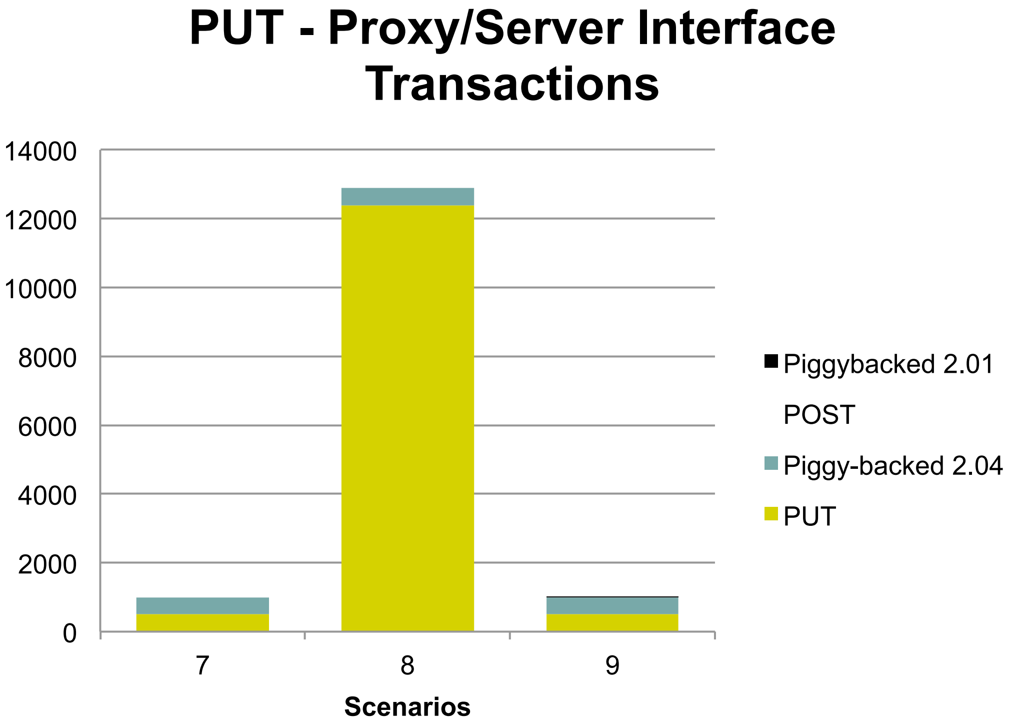


Test Scenarios:

4 - Non-sleepy server, non-sleep aware proxy, 5 clients performing PUTs
5- Sleepy server, non sleep aware proxy, 5 clients performing PUTs
6 - Sleepy server, sleep aware proxy, 5 clients performing PUTs

→ Sleep Aware Proxy reduces number of PUTs, 5.04s, and ACKs between client and proxy

PUT – Proxy/Server Interface Transaction Mix



Test Scenarios:

4 - Non-sleepy server, non-sleep aware proxy, 5 clients performing PUTs
5- Sleepy server, non sleep aware proxy, 5 clients performing PUTs
6 - Sleepy server, sleep aware proxy, 5 clients performing PUTs

→ Sleep Aware Proxy reduces number of PUTs between proxy and sleepy server

Conclusions

Conclusions

- These results show that sleep-aware CoAP proxy features can significantly optimize communication with sleepy servers in most scenarios

- These results also show that sleep-awareness capabilities can provide additional benefits when used in conjunction with proxy based CoAP caching & OBSERVE

Backup

Current CoAP Support of Sleepy Node (1/2)

- CoAP proxies can use a previously cached response to service a new GET request for a sleepy origin server (as in HTTP)
  - But if no valid cache then proxy has to attempt to retrieve and may fail if origin server is sleeping
  - [I-D.ietf-core-coap]

- Clients can discover list of resources from RD (GET /rd-lookup/…) for sleepy servers
  - But attempt to GET resource from sleepy origin server may fail if origin server is sleeping
  - [I.D.ietf-core-link-format & I.D.ietf-core-resource-directory]

Current CoAP Support of Sleepy Node (2/2)

- Lower layer support for sleepy nodes in most wireless technologies (e.g. WiFi, ZigBee).
  - But limited to MAC packet scheduling for sleepy nodes and not aware of specific needs of IP applications (like CoAP)

Proposal – RD Based Sleep Tracking (1/4)

- The current CoAP approach to support sleepy nodes can be significantly improved by introducing RD based mechanisms for a CoAP client to determine whether:
  - A targeted resource is located on a sleepy server
  - A sleepy server is currently in sleep mode or not
  - There is any associated caching Proxy (possibly the RD itself) for a sleepy server

Proposal – RD Based Sleep Tracking (2/4)

- We define the following new RD attributes to characterize the properties of a sleepy node:
  - SleepState - Indicates whether the node is currently in sleep mode or not (i.e. Sleeping or Awake)
  - SleepDuration - Indicates the maximum duration of time that the node stays in sleep mode
  - TimeSleeping - Indicates the length of time the node has been sleeping (i.e. if Sleep State = Sleeping)
  - NextSleep - Indicates the next time the node will go to sleep (i.e. if Sleep State = Awake)

  - CachingProxy – Indicates the caching proxy of the sleepy node (i.e. the RD itself or another node)

Proposal – RD Based Sleep Tracking (3/4)

- These attributes are all server (node) level and are new parameters added to the RD URI Template Variables

- Finally, we also define a new lookup-type ("ss") for the RD lookup interface specified in [I-D.ietf-core-resource-directory].
  - This new lookup-type supports looking up the “SleepState” (ss) of a specified end-point

Proposal – RD Based Sleep Tracking (4/4)

- The three time based parameters (SleepDuration, TimeSleeping, NextSleep) can be based on either an absolute network time (for a time synchronized network) or a raw number of seconds (measured at the local node)

- Following the approach of [I-D.ietf-core-link-format] and [I-D.ietf-core-resource-directory], sleep parameters for sleepy servers can be stored by the server in the RD and accessed by all interested clients

- Examples of using these parameters in a synchronous or asynchronous manner are shown in the I-D

Group 3:“new work” (continued)

CoAP Entities

draft-ishaq-core-entities-00

Isam Ishaq

Jeroen Hoebeke

Floris Van den Abeele

http://tools.ietf.org/id/draft-ishaq-core-entities-00.txt

Why?

- Supporting multicast can be expensive/hard in constrained node networks.
- Usually some form of group communication is still wanted.
- This ID tries to solve this by providing an uni-cast based group communication solution.

How?

- Clients can create ‘entities’ with a central broker through a RESTful interface (“core.em”)

```
Req: POST coap://em.example.com/e (application/link-format)
       Body: <coap://sen5.example.com/tmp>,
             <coap://sen8.example.com/tmp>

Res: 2.05 Content (text/plain)
      Body: /1 created
```

- Broker acts as a message (de)multiplexer, but it can also provide extra functionality

How?

- After creation, clients can use the new resource to interact with the entity.

  Req: GET coap://em.example.com/1

  2.05 Content (application/senml+json)

  ```
     Payload: {"e":[
           {"n": "Sen5/tmp", "v": "26.6", u="degC"},
           {"n": "Sen8/tmp", "v": "23.5", u="degC"}]}
  ```

Entity manager

- Performs validation during entity creation
  - Check whether the resources exist
  - Check supported methods and CoAP options
- Describes every entity in a profile using draft-greevenbosch-core-profile-description

```
{   "profile":[                    "entity":[
     { "path":"1",                     {"r":["coap://sen5.example.com/tmp",
       "op":[3,4,7,11,12],                  "coap://sen8.example.com/tmp"]
       "cf":[55],                      }]
       "m":[1]}],                  }
```

- Can also define operations (avg, min)

Conclusion

- Lightweight group communication alternative to multicast-based solutions
- Current version of the draft is very basic, will be extended further in the future. Contact me if you're interested in working together.
- Note: IPR declaration by KPN N.V.:

  Reasonable and Non-Discriminatory License to All Implementers with Possible Royalty/Fee.
  https://datatracker.ietf.org/ipr/2134/

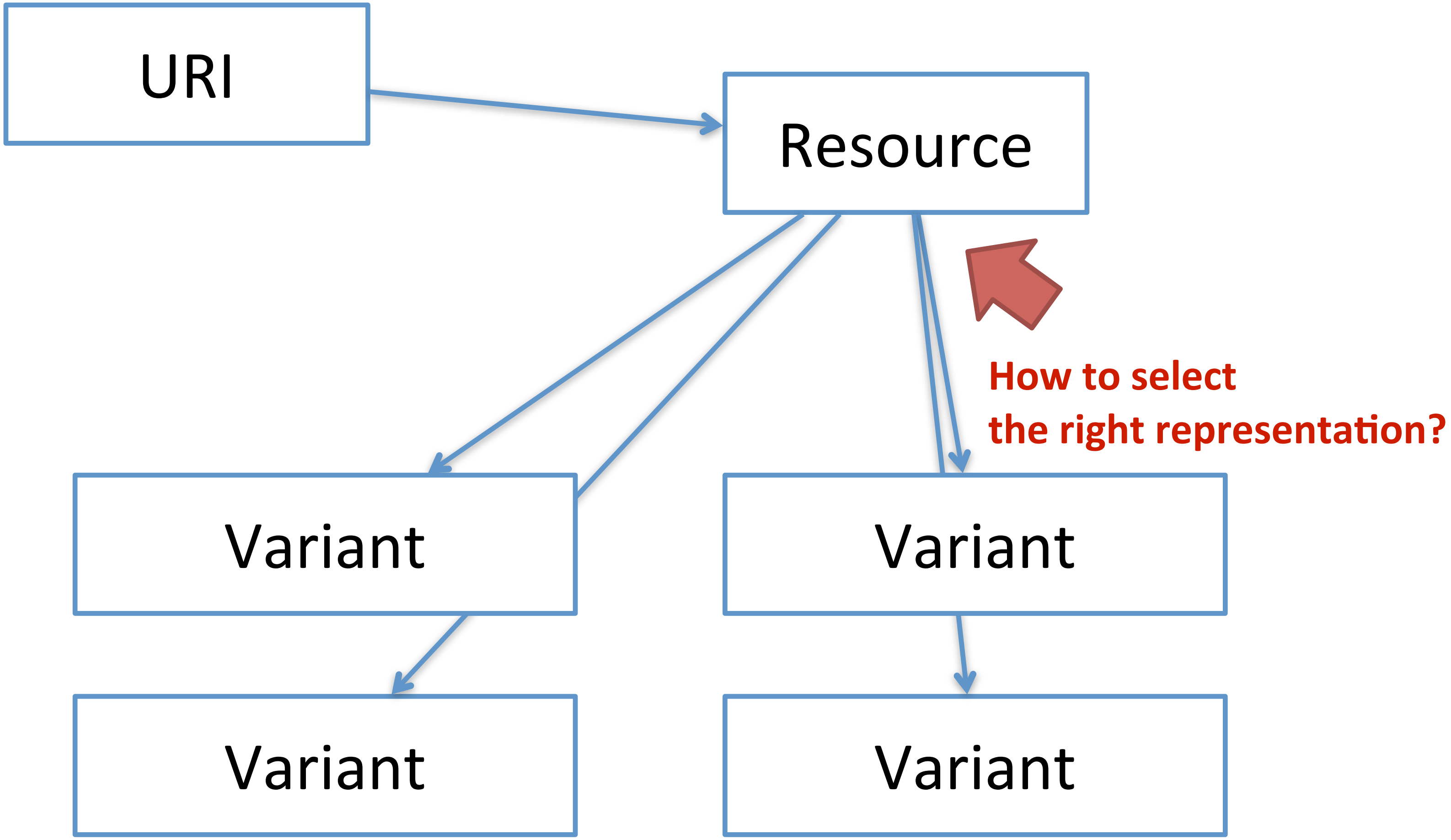
draft-doi-core-parameter-option-02

Yusuke DOI

Kerry Lynn

Problem Statement

One URI → One Resource

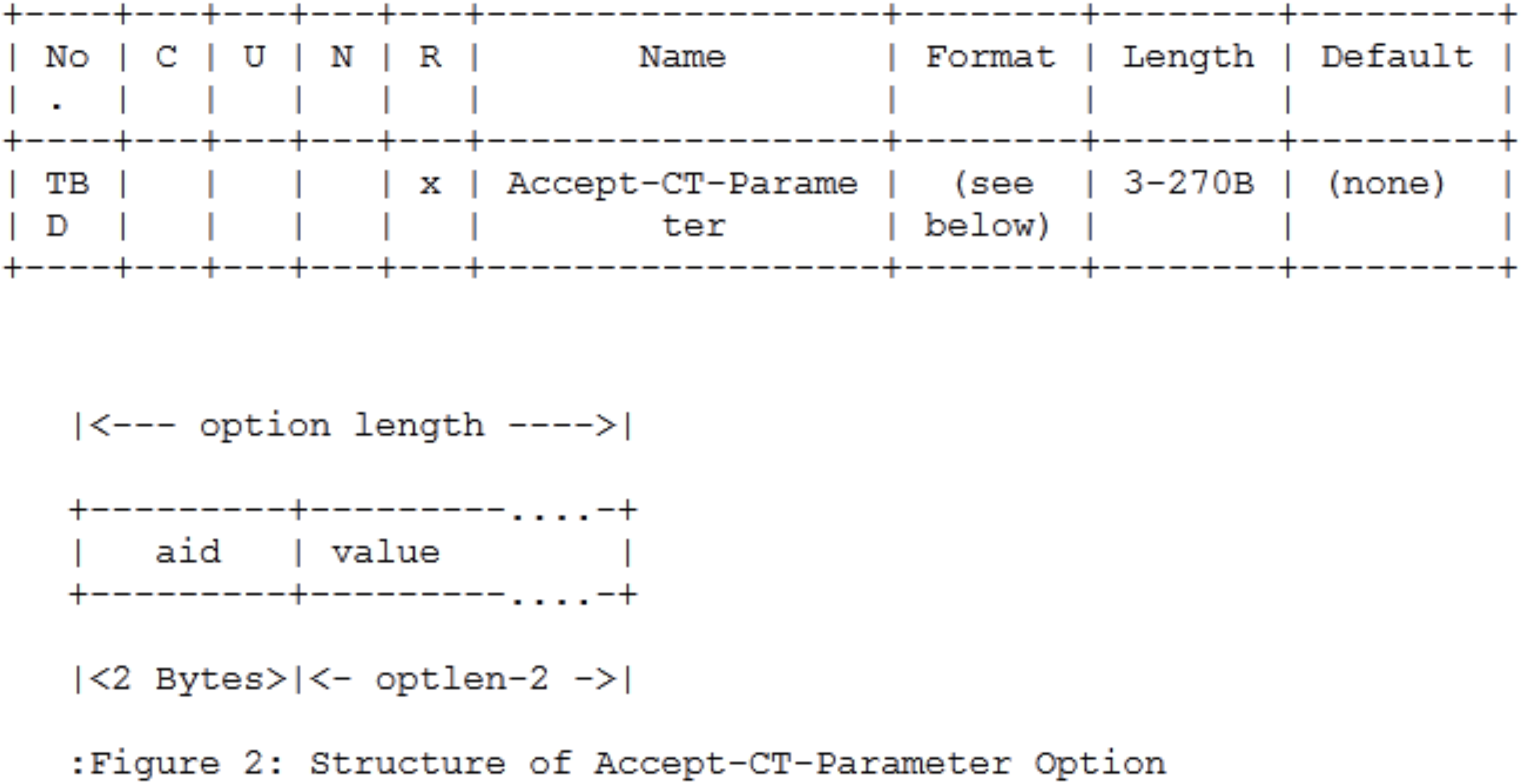


How to select the right representation?

Server-Side Content Negotiation

- Accept header from client tells which variant is requested on the resource
- Various extensions are possible
  - for example: draft-wilde-atom-profile-01
- CoAP does not have room for *parameterized* server-side content negotiation
  - Basic spec does not have use case for it, but I believe extended spec should have (at least, for EXI schema negotiation)

Accept Content-Type Parameter Option



No.	C	U	N	R	Name	Format	Length	Default
TBD				x	Accept-CT-Parameter	(see below)	3-270B	(none)

```
|<--- option length ---->|

+---------+---------....-+
|   aid   | value        |
+---------+---------....-+

|<2 Bytes>|<- optlen-2 ->|
```

:Figure 2: Structure of Accept-CT-Parameter Option

Attribute ID

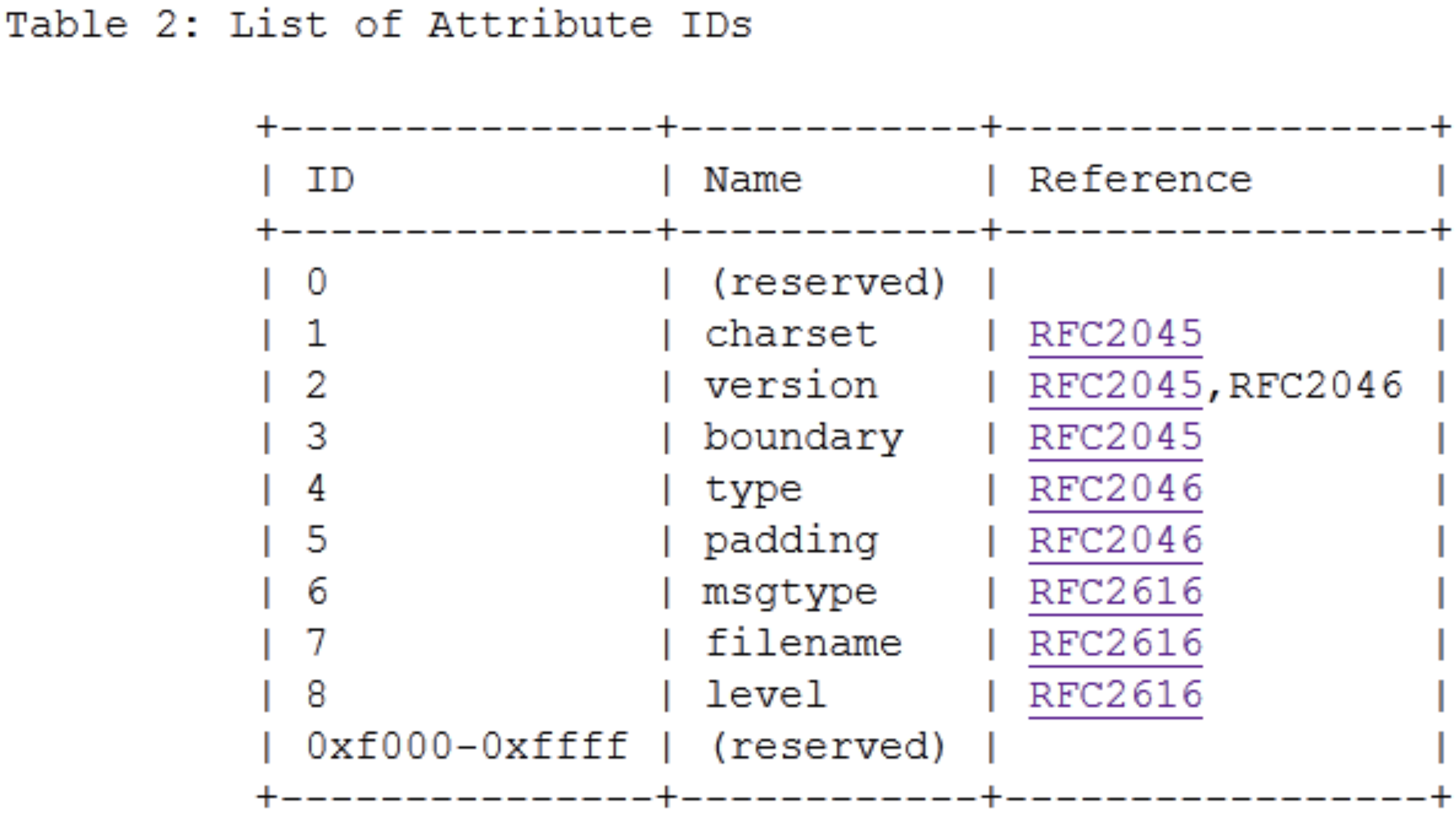


Table 2: List of Attribute IDs

ID	Name	Reference
0	(reserved)	
1	charset	RFC2045
2	version	RFC2045,RFC2046
3	boundary	RFC2045
4	type	RFC2046
5	padding	RFC2046
6	msgtype	RFC2616
7	filename	RFC2616
8	level	RFC2616
0xf000-0xffff	(reserved)	

Questions?

Group Authentication

Problem Statement

With the development of Internet of Things, the scale of IOT system become larger and larger. A large amount of smart power meter terminals are deployed in a block, thus brings more cost for communication

Example 1:

Use case：

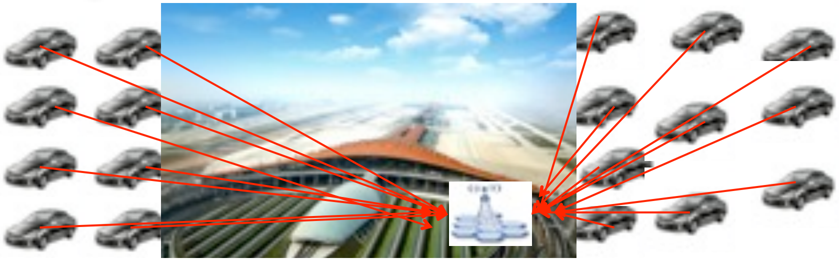
80,000 taxis in one city, 90% with IOT devices for monitoring. Authentication is needed when these IOT devices connects with network

Issues:：

A plenty of taxis stay in the same area, the communication will be frequent and cause overload for network.

Frequency for the issue happening：

- Every day in airplane
- Once per week when the taxi company call together of all taxi drivers.



Example 2：

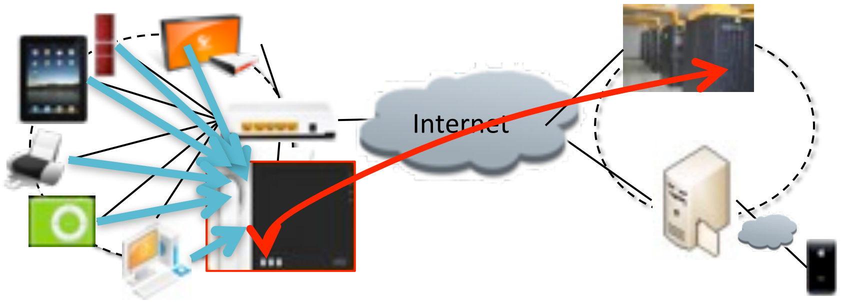
Use case：

4760 users in one block. Each owns one IOT devices for smart metering. They will report data at the same time. Authentication happens when these devices report data.

Issues：

The communication will be large and cause overload for the network when all smart metering devices report data at the same time.

Existing authentication analysis

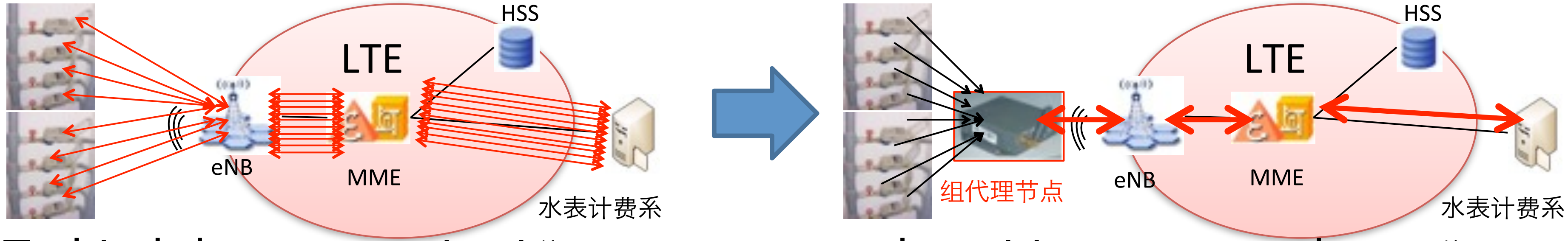


- Now some authentication can be used to solve the problems. An agent is introduced aggregate the message
  - All network entities can connect with an agent and make mutual authentication with agent independently
  - Agent makes mutual authentication with network server independently.

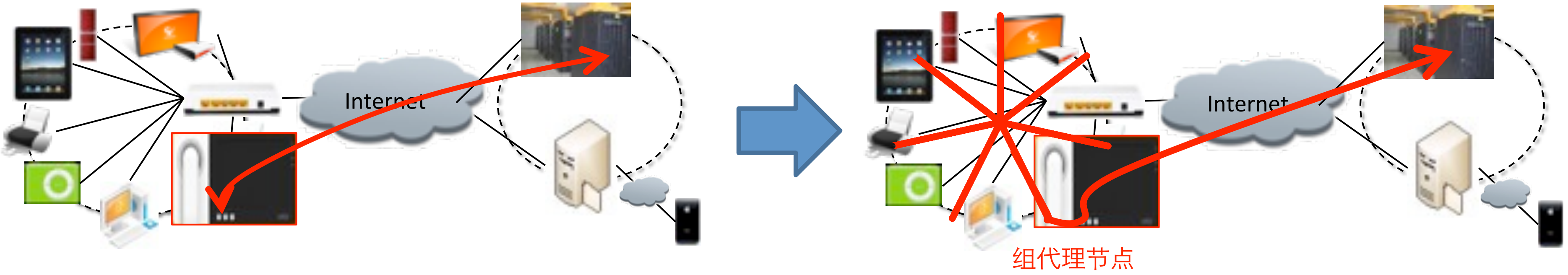
- But some problems still exists:
  - Agent becomes key point for the communication. A MITM attack will be happened if agent is compromised.
  - Agent can get all information transferred between entities and server. The communication could be broken if agent belongs to 3rd–party.

The motivation to build up group

1.To reduce communication cost with group authenticaion

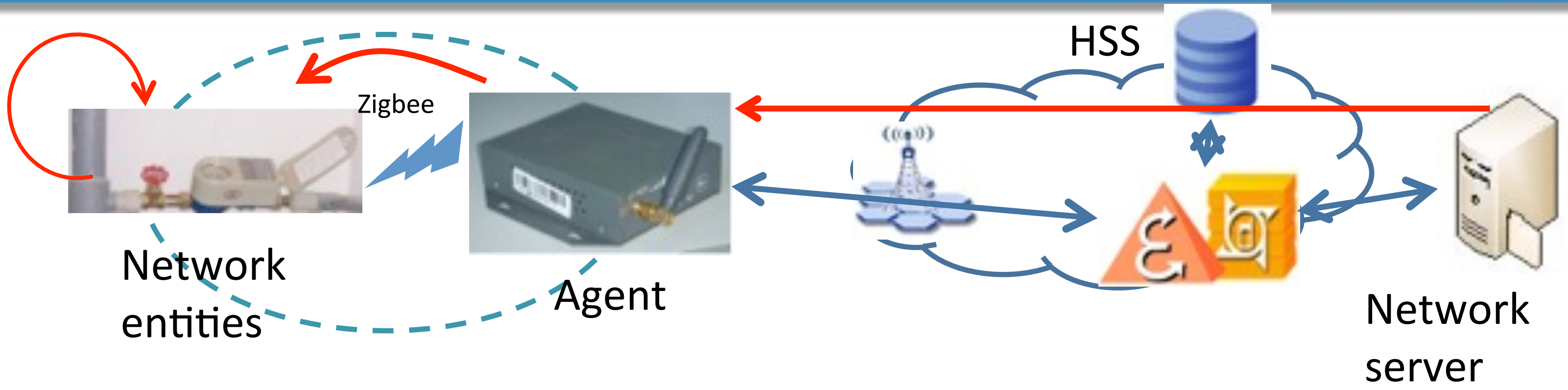


2.To bind the communication among network entities, agent and network server



3. To establish end-to-end communication for privacy protection

The principle of group authentication



0.A group is build up with an agent and several network entities.

1. Group authentication is triggered by the behavior to upload data or the behavior to re-configure all nodes by network server

2. Inner group authentication is made between all network entities and agent.

3. Agent makes mutual authentication with network server on behalf of whole group. The network server could authenticate network entities also through some pre-shared credentials.

Thanks!